ВАРИАНТ 1

Задача 1 (10 баллов)

Разведчики

Нашими разведчиками на командный пункт были переданы места сосредоточения войск противника в виде горизонтальных и вертикальных координат, расположенных в ячейках B2:F2 и A3:A7. Определите их по формулам закодированной электронной таблицы, представленной в двух режимах, на рисунках расположенных ниже.

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9	1	=СУММ(B2:F2)/(A4-A6)		a=	=ABS(ОКРУГЛВНИЗ(-ПИ();0))	
10	2	=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))		b=	=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)	
11	3	=A3*(A4+A6)		c=	=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))	
12	4	=A3-A4-A6		d=	=ЕСЛИ(ЕОШ(B9);E9*E10;E10-E9)	
13	5	=ЕСЛИ(СУММ(A3:A7)>45;A3+A4;A6+A5)		e=	=B10-E9	
14	6	=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))		f=	=B10^2+E15	
15	7	=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10		g=	1	
16	8	=НАЙТИ(C2;"ABCDEFGHIJKLMNOPQ")-E11+E12				
17	9	=НАЙТИ(D2;"ABCDEFGHIJKLMNOPQ")-E13+E14				
18	10	=НАЙТИ(E2;"ABCDEFGHIJKLMNOPQ")-E14+E15				
19	11	=НАЙТИ(F2;"ABCDEFGHIJKLMNOPQ")				

	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							
9	1	#ДЕЛ/0!		a=			
10	2	5		b=			
11	3	112		c=			
12	4	-6		d=			
13	5	16		e=			
14	6	абвгде		f=			
15	7	8		g=			
16	8	12					
17	9	28					
18	10	-19					
19	11	3					

Ответ оформите в виде двух строк, в верхней строке запишите через запятую значения ячеек B2:F2, а в нижней строке также через запятую значения ячеек A3:A7. В случае нескольких вариантов ответов запишите их через союз «или».

Решение

Начнем с определения коэффициентов a, b, c, d, e, f и g.

Значение коэффициента a (ячейка E9) в соответствии с формулой «=ABS(ОКРУГЛВНИЗ(-ПИ();0))» равно 3.

Значение коэффициента b (ячейка E10) в соответствии с формулой «=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)» равно 4.

Значение коэффициента c (ячейка E11) в соответствии с формулой «=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))» равно 2.

Значение коэффициента d (ячейка E12) в соответствии с формулой «=ЕСЛИ(ЕОШ(B9); E9*E10;E10-E9)» равно 12.

Значение коэффициента e (ячейка E13) в соответствии с формулой «=B10-E9» равно 2.

Значение коэффициента f (ячейка E14) в соответствии с формулой «=B10^2+E15» равно 26.

Значение коэффициента g (ячейка E15) равно 1.

Получаем:

	D	E
8		
9	a=	3
10	b=	4
11	c=	2
12	d=	12
13	e=	2
14	f=	26
15	g=	1

1) Так как значение формулы в ячейке B9 «=СУММ(B2:F2)/(A4-A6)» определяется как ошибка деления на 0, то значения ячеек A4 и A6 одинаковы.

2) Значение формулы в ячейке B10 «=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))» равно 5, следовательно значения координат в ячейках A3:A7 состоят из одного символа.

3) Формулы в ячейках B11 «=A3*(A4+A6)» и B12 «=A3-A4-A6» рассмотрим, как систему уравнений, учитывая, что значения ячеек A4 и A6 одинаковы.

$$\begin{cases} A3 * (A4 + A6) = 112 \\ A3 - A4 - A6 = -6 \end{cases} \Rightarrow \begin{cases} A3 * 2A4 = 112 \\ A3 - 2A4 = -6 \end{cases}$$

Решая систему уравнений, получаем два набора значений для ячеек A3 и A4 (A3 = 8 и A4 = 7) либо (A3 = -14 и A4 = -4). Так как, в соответствии со вторым пунктом, значения ячеек A3:A7 состоят из одного символа, а это диапазон от 0 до 9, то A3 = 8, A4 = A6 = 7.

4) Значение формулы в ячейке B13 «=ЕСЛИ(СУММ(A3:A7)>45; A3+A4;A6+A5)» равно 16. Максимальное значение формулы СУММ(A3:A7)

из-за того, что значения в этих ячейках не превышают 9, равно 45. В соответствие с этим, условие в формуле ячейки B13 не выполнится, следовательно A6 + A5 = 16 и A5 = 16 - 7 = 9.

5) Значение формулы в ячейке B14 «=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))» равно «абвгде». Следовательно разница между ячейками A7 и A5 равна 6, откуда A7 равно 3 или 15. По условию 0<=A7<=9, A7 = 3.

6) Значение формулы в ячейке B15 «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10» равно 8. Откуда, значение формулы «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")» = 8 + E9 – E10 = 8 + 3 - 4 = 7, следовательно B2 = «G».

7) Значения ячеек C2, D2, E2 и F2 аналогичным образом находим из формул в ячейках B16, B17, B18 и B19. Получаем C2 = «B», D2 = «D», E2 = «F» и F2 = «C».

Ответ: **G, B, D, F, C**
8, 7, 9, 7, 3

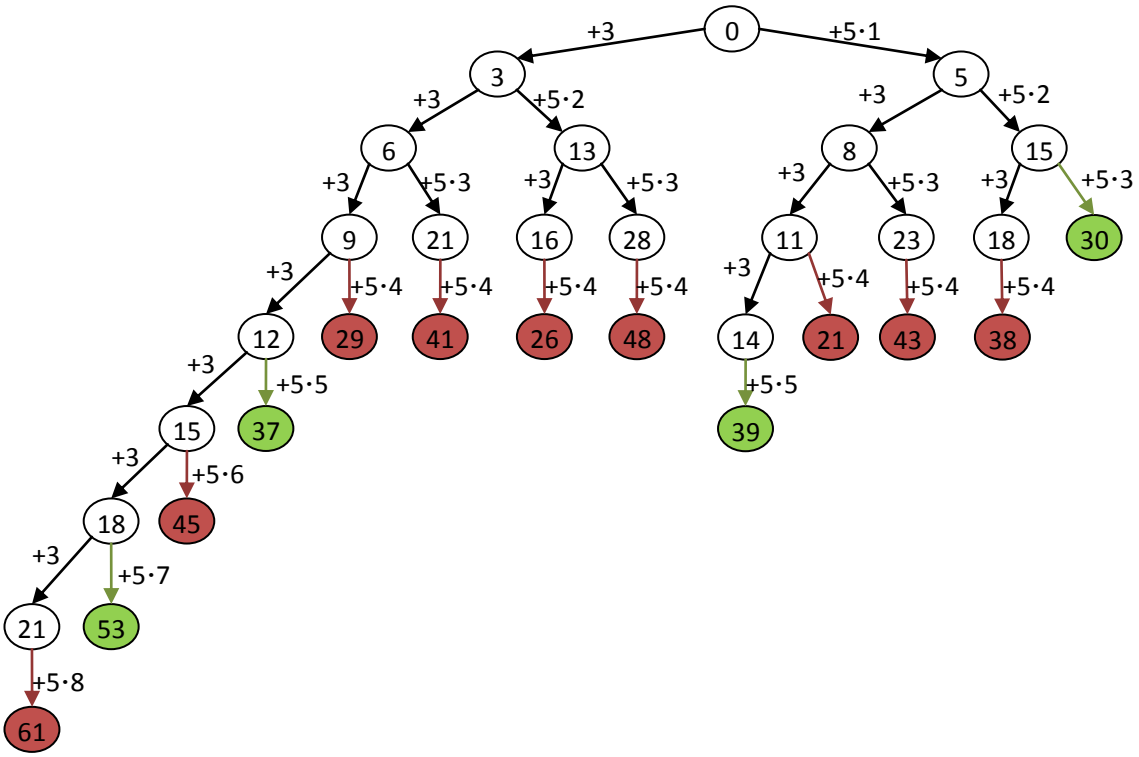
Задача 2 (10 баллов)

Гномы и драгоценные камни: стратегии и выигрыши

Два богатых гнома решили сыграть в следующую игру. По очереди они складывают драгоценные камни в кучку. За один ход можно либо положить в кучку пять камней, либо положить в кучку количество камней, определяемое формулой пять умноженное на номер хода в игре. При превышении количества 30 камней в кучке, игра заканчивается. Кто первым, сделав ход, получит в кучке 30 камней или больше – считается победителем. Кто выигрывает при безошибочной игре – гном, делающий первый ход, или гном, делающий второй ход? Каково минимальное число ходов до победы, если выигрывающий гном пытается закончить игру за меньшее число ходов, а проигрывающий пытается максимально задержать свое поражение? Какое максимальное число камней может оказаться в кучке в конце игры? Какой гном при этом победит? Поясните алгоритм графически, используя ориентированные графы в виде деревьев, с обозначением выигрышных и проигрышных позиций как вершин, а действий игроков – дугами графа с указанием численных изменений.

Решение

Построим ориентированный граф, где зеленым цветом выделены выигрышные позиции первого гнома, красным цветом – выигрышные позиции второго гнома, а номер соответствует количеству камней в кучке. Рядом со стрелкой указано изменение числа камней в кучке на этом ходу. В случае, если один из вариантов хода приводит к победе, то второй вариант не отображается, за исключением самой правой ветви, которая демонстрирует игру с максимально возможным числом камней в конце игры.



Из графа видно, что при любой игре первого гнома, гарантированно победит второй гном на 4 ходу. Максимально возможное число камней можно получить, если увеличивать количество камней на минимальное значение (+3) до последнего хода перед достижением победы, в котором берется вариант с большим значением (+5·8). При этом побеждает второй гном с количеством камней 61.

Ответ: гарантированно победит второй гном за 4 хода, максимум 61 камень при победе второго гнома.

Задача 3 (10 баллов)

Логическая головоломка: разгадка функции F

Паша решил усовершенствовать задание ЕГЭ по информатике.

Логическая функция задается выражением:

$$(x \vee \neg y) \rightarrow (z \equiv w) \vee (w \wedge \neg y).$$

Ниже приведен фрагмент таблицы истинности функции F, содержащий неповторяющиеся строки. Однако вместо значений, Паша решил вписать тоже несколько логических выражений, результат которых надо определить, если **a, b, c** – следующие высказывания:

a – «число 17 – нечетное»

b – «1107 не делится на 9 без остатка»

c – «факториал пяти 5! = 120»

Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных **w,x,y,z.**

?	?	?	?	F
		a∧b∧c	(a∨c≡b)∧b	0
	a→¬c⊕b		¬b∧c→¬a	0
a⊕c∧¬b			a≡b∧c⊕a	0
	b∨a∧¬c		(a∨b)≡c	0

В ответе напишите буквы **w,x,y,z** в том порядке, в котором идут соответствующие им столбцы.

Решение

Сперва определим значения логических высказываний. Очевидно, что

a=1
b=0
c=1

Теперь определим значения логических выражений в таблице. В результате получим следующую таблицу:

?	?	?	?	F
		0	0	0
	0		0	0
0			1	0
	0		1	0

Следующим шагом будет составление таблицы истинности для функции F. Сперва можно немного упростить функцию. Получаем

x∨¬y→(z≡w∨w∧¬y)

тогда получим следующую таблицу истинности:

w	x	y	z	¬y	x∨¬y	z≡w	w∧¬y	z≡w∨w∧¬y	F
0	0	0	0	1	1	1	0	1	1
0	0	0	1	1	1	0	0	0	0
0	0	1	0	0	0	1	0	1	1
0	0	1	1	0	0	0	0	0	1
0	1	0	0	1	1	1	0	1	1
0	1	0	1	1	1	0	0	0	0
0	1	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0	0
1	0	0	0	1	1	0	1	1	1
1	0	0	1	1	1	1	1	1	1
1	0	1	0	0	0	0	0	0	1
1	0	1	1	0	0	1	0	1	1
1	1	0	0	1	1	0	1	1	1
1	1	0	1	1	1	1	1	1	1
1	1	1	0	0	1	0	0	0	0
1	1	1	1	0	1	1	0	1	1

Выберем из нее строки, где функция F = 0.

w	x	y	z	F
0	0	0	1	0
0	1	0	1	0
0	1	1	1	0
1	1	1	0	0

Сравнивая эту таблицу с заданной в задании, и учитывая, что строки в этой таблице выписаны в случайном порядке, легко установить с помощью несложных рассуждений искомые наименования столбцов

z	w	x	y	F
1	0	0	0	0
1	0	1	0	0
0	1	1	1	0
1	0	1	1	0

Ответ: z,w,x,y

Задача 4 (20 баллов)

Зашифрованный Рудник

В горнодобывающей компании «Рудник-Х» используется система шифрования для передачи информации о наименовании полезного ископаемого и параметрах пласта с данным ценным компонентом (длина, ширина), которая была получена при проведении геологической разведки месторождений полезных ископаемых. Передача информации осуществляется геологами с места проведения полевых работ в административное управление компании.

Процесс шифрования в горнодобывающей компании заключается в следующем:

1) Генерируется супервозрастающая последовательность чисел $A = \{a_1, a_2, \dots a_k\}$, каждый член которой больше суммы всех предыдущих членов, т.е. при $j = 2,3, \dots k: a_j > \sum_{i=1}^{j-1} a_i$, которая является секретным ключом.

Например, секретный ключ: $A = \{a_1, a_2, a_3, a_4\} = \{2, 3, 9, 17\}$,
где $a_2 > a_1 \rightarrow 3 > 2$;
$a_3 > (a_1 + a_2) \rightarrow 9 > 5$;
$a_4 > (a_1 + a_2 + a_3) \rightarrow 17 > 14$.

2) После этого выбирается число m большее, чем сумма чисел последовательности, т.е. $m > \sum_{i=1}^{k} a_i$. Также выбирается число n взаимно простое с m, т.е. наибольший общий делитель m и n равен 1.

3) Далее создается открытый ключ, т.е. новая последовательность чисел $B = \{b_1, b_2, \dots b_k\}$ путем умножения каждого числа a_i на n и взятия остатка от деления данного произведения на m.

Например, для открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ каждый член последовательности вычисляется по формуле $b_i = (a_i \cdot n) \bmod m$,
где a_i — i-й элемент последовательности секретного ключа A;
n — число взаимно простое с m;
m — модуль операции взятия остатка, иными словами, делитель;
mod — операция взятия остатка от деления.
При $n = 91$, $m = 320$ и $A = \{2, 3, 9, 17\}$:
$b_1 = (a_1 \cdot n)\ mod\ m = (2 \cdot 91)\ mod\ 320 = 182$ или, иными словами, $(2 \cdot 91)$ делится на 320 с остатком 182.
Сделав аналогичные вычисления для остальных элементов, получается последовательность $B = \{182,\ 273, 179, 267\}$

4) Полученная последовательность B используется для шифрования отправляемого сообщения. Отправляемое сообщение, представленное в виде двоичной строки, определяет, какие элементы этой последовательности суммируются для получения последовательности шифротекста $C = \{c_1, c_2, \dots c_p\}$, которая передается получателю.

Например, при шифровании числа 10 последовательность шифротекста C будет вычисляться следующим образом:

1) Число 10 представляется в двоичной системе счисления, т.е. $10 = 1010_2$;

2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1 и b_3 суммируются для получения шифротекста, b_2 и b_4 – опускаются.

3) Шифротекст $C = \{c_1\} = \{b_1 + b_3\} = 182 + 179 = 361$.

Например, при шифровании числа E_{16} последовательность шифротекста C будет вычисляться следующим образом:

1) Число E_{16} представляется в двоичной системе счисления, т.е. $E_{16} = 1110_2$;

2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1, b_2 и b_3 суммируются для получения шифротекста, b_4 – опускается.

3) Шифротекст $C = \{c_1\} = \{b_1 + b_2 + b_3\} = 182 + 273 + 179 = 634$.

Для дешифрирования сообщения получатель, исходя из известного ему n, определяет n^{-1} – мультипликативное обратное к значению n по модулю m (остаток от деления $n \cdot n^{-1}$ на m равен 1).

Иными словами, $(n \cdot n^{-1}) \bmod m = 1$,

где n — заданное число взаимно простое с m;

m — модуль операции взятия остатка;

mod — операция взятия остатка от деления.

Далее вычисляются промежуточные значения последовательности C' путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

Например, для шифротекста $C = \{c_1\} = 361$, каждый член последовательности C' вычисляется по формуле $c'_i = (c_i \cdot n^{-1}) \bmod m$,

где c_i — i-й элемент последовательности шифротекста C;

n^{-1} — мультипликативное обратное;

m — модуль операции взятия остатка;

mod — операция взятия остатка от деления.

В данном случае, последовательность

$C' = \{c'_1\} = (361 \cdot 211) \bmod 320 = 11$, где 211 – мультипликативное обратное n^{-1}, рассчитанное получателем.

Далее получатель определяет двоичную строку M путем последовательного сравнения каждого элемента последовательности C' с элементами секретного ключа A справа налево: если $c'_i \geq a_i$, то $M_i = 1$ и осуществляется вычитание a_i из c'_i, в противном случае $M_i = 0$. Таким образом восстанавливается зашифрованное сообщение.

Например, для последовательности $C' = \{c'_1\}$ и секретного ключа $A = \{a_1, a_2, a_3, a_4\}$ осуществляется последовательное сравнение элементов справа налево:

1) Если $c'_1 \geq a_4$, то $M_4 = 1$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число $c'_1 - a_4$;

2) Если $c'_1 < a_4$, то $M_4 = 0$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число c'_1.

3) В ответе получается двоичная строка $M = \{M_1, M_2, M_3, M_4\}$.

Например, при $A = \{2, 3, 9, 17\}$ и $C' = \{11\}$ двоичная строка $M = \{1, 0, 1, 0\}$, что согласуется с двоичным представлением числа 10.

Для последовательности C', состоящей из нескольких элементов, например, $C' = \{c'_1, c'_2, \dots c'_k\}$ осуществляется последовательное сравнение каждого элемента $c'_1, c'_2, \dots c'_k$ с элементами секретного ключа A с получением количества двоичных строк, равного количеству элементов последовательности C'.

1) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4\}$ количество двоичных строк M равно 4.

2) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7, c'_8\}$ количество двоичных строк M равно 8.

Геологи передали четыре последовательных сообщения в административное управление о двух месторождениях полезных ископаемых. Первое сообщение в каждой паре содержало наименование полезного ископаемого. Второе сообщение содержало информацию о параметрах пласта, при этом первое число соответствовало длине пласта, второе – ширине пласта. Для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы согласно таблице ASCII для Windows-1251, далее полученному шестнадцатеричному числовому значению присваивалась соответствующая двоичная тетрада. Для шифрования числа осуществлялся его перевод в двоичную систему. Таблица ASCII для Windows-1251:

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Были получены следующие зашифрованные сообщения:

1 сообщение: 1449 1283 341 1059 758 1320 341

2 сообщение: 875 1225

3 сообщение: 924 903 693 1513

4 сообщение: 1735 1790

Секретный ключ, хранящийся в организации, представляет собой следующую последовательность {2, 3, 6, 13, 27, 52, 105, 210}, значения n и m равны 31 и 420 соответственно.

Расшифруйте полученные сообщения. В ответе укажите последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» – «Длина», «Ширина» (числа в десятичной системе). Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Решение

Сначала необходимо вычислить промежуточные значения последовательности C' для осуществления дешифровки. В исходных данных сказано, что данные значения вычисляются путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

В условии дано только значение n, однако мы знаем, что n^{-1} – мультипликативное обратное к значению n по модулю m, то есть остаток от деления $n \times n^{-1}$ на m равен 1, поэтому для нахождения значения n^{-1} можно применить расширенный алгоритм Евклида.

Расширенный алгоритм Евклида является модификацией алгоритма Евклида, вычисляющая кроме наибольшего общего делителя (НОД) целых чисел a и b, еще и коэффициенты соотношения Безу, то есть такие x и y, что $ax + by = \text{НОД}(a, b)$. В свою очередь, алгоритм Евклида заключается в последовательном делении с остатком для нахождения наибольшего общего делителя двух чисел. На каждом этапе производится деление большего из пары чисел на меньшее, а остаток записывается и используется при дальнейших вычислениях в качестве нового делителя для предыдущего. Вычисление остатка производят до тех пор, пока он не будет равен нулю. Тогда последний использованный делитель и будет являться наибольшим общим делителем, полученным через алгоритм Евклида. В нашем случае наибольший общий делитель чисел n и m (31 и 420) должен равняться 1.

Применяем алгоритм Евклида:

$$420 = 31 \times 13 + 17 \Rightarrow \quad 31 = 17 \times 1 + 14 \Rightarrow \quad 17 = 14 \times 1 + 3 \Rightarrow$$
$$14 = 3 \times 4 + 2 \Rightarrow \qquad 3 = 2 \times 1 + 1 \qquad 2 = 2 \times 1 + 0$$

Применяем расширенный алгоритм Евклида. Нам нужно выразить 1 как линейную комбинацию 31 и 420, т. е. найти целые числа x и y, такие, что $31x + 420y = 1$

Из алгоритма Евклида выше:

$$1 = 3 - 2 \times 1 \Leftarrow \qquad 2 = 14 - 3 \times 4 \Leftarrow \quad 3 = 17 - 14 \times 1 \Leftarrow$$
$$14 = 31 - 17 \times 1 \Leftarrow \quad 17 = 420 - 31 \times 13$$

Подставим обратно:

$$1 = 3 - (14 - 3 \times 4) \times 1 = 3 \times 5 - 14 \times 1$$
$$1 = (17 - 14 \times 1) \times 5 - 14 \times 1 = 17 \times 5 - 14 \times 6$$
$$1 = 17 \times 5 - (31 - 17 \times 1) \times 6 = 17 \times 11 - 31 \times 6$$
$$1 = (420 - 31 \times 13) \times 11 - 31 \times 6 = 420 \times 11 - 31 \times 149$$
$$1 = 420 \times 11 - 31 \times 149$$

Мы получили, что $1 = 420 \times 11 - 31 \times 149$, следовательно, $310 \times (-149) - 420 \times 11 = 1$. Это означает, что -149 является обратным числом 31 по модулю 420. Поскольку нам нужно положительное целое число, мы прибавляем 420 к -149: $-149 + 420 = 271$. Получаем, что $n^{-1} = 271$.

Зная значение n^{-1} находим промежуточные значения последовательности C' для каждого полученного сообщения. Для первого

сообщения – 1449 1283 341 1059 758 1320 341:

1) (1449×271) делится на 420 с остатком 399

2) (1283×271) делится на 420 с остатком 353

3) (341×271) делится на 420 с остатком 11

4) (1059×271) делится на 420 с остатком 129

5) (758×271) делится на 420 с остатком 38

6) (1320×271) делится на 420 с остатком 300

7) (341×271) делится на 420 с остатком 11

Промежуточные значения последовательности {399, 353, 11, 129, 38, 300, 11}.

Для второго сообщения – 875 1225:

1) (875×271) делится на 420 с остатком 245

2) (1225×271) делится на 420 с остатком 175

Промежуточные значения последовательности {245, 175}.

Для третьего сообщения – 924 903 693 1513:

1) (924×271) делится на 420 с остатком 84

2) (903×271) делится на 420 с остатком 273

3) (693×271) делится на 420 с остатком 63

4) (1513×271) делится на 420 с остатком 103

Промежуточные значения последовательности {84, 273, 63, 103}.

Для четвертого сообщения – 1735 1790:

1) (1735×271) делится на 420 с остатком 205

2) (1790×271) делится на 420 с остатком 410

Промежуточные значения последовательности {205, 410}.

Далее используя секретный ключ A, получатель определяет битовую строку M путем последовательного сравнения C' с элементами a_i. Секретный ключ, исходя из исходных условий, представляет собой последовательность {2, 3, 6, 13, 27, 52, 105, 210}.

Для первого сообщения:

1) $399 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{matrix}$

2) $353 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{matrix}$

3) $11 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

4) $129 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{matrix}$

5) $38 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{matrix}$

6) $300 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{matrix}$

7) $11 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

Для второго сообщения:

1) $245 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{matrix}$

2) $175 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{matrix}$

Для третьего сообщения:

1) $84 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{matrix}$

2) $273 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{matrix}$

3) $63 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{matrix}$

4) $103 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{matrix}$

Для четвертого сообщения:

1) $205 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{matrix}$

2) $410 = \begin{matrix} 2 & 3 & 6 & 13 & 27 & 52 & 105 & 210 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{matrix}$

Из условий задачи известно, что для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы, далее каждой шестнадцатеричной цифре соответствующая двоичная тетрада; для шифрования числа осуществлялся его перевод в двоичную систему. Для дешифровки необходимо произвести данные действия в обратном порядке.

Сначала дешифруем текстовые сообщения. Для этого воспользуемся таблицей тетрад и далее таблицей ASCII для Windows-1251 из условий задачи.

16-ричная цифра	Двоичная тетрада	16-ричная цифра	Двоичная тетрада
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Таким образом, произведем дешифровку текстовых сообщений:

Первое сообщение:

1) $1449 = 11001111 = \begin{matrix}1100\\1111\end{matrix}\ \begin{matrix}C\\F\end{matrix} = П$

2) $1283 = 11101011 = \begin{matrix}1110\\1011\end{matrix}\ \begin{matrix}E\\B\end{matrix} = л$

3) $1283 = 11100000 = \begin{matrix}1110\\0000\end{matrix}\ \begin{matrix}E\\0\end{matrix} = а$

4) $1283 = 11110010 = \begin{matrix}1111\\0010\end{matrix}\ \begin{matrix}F\\2\end{matrix} = т$

5) $1283 = 11101000 = \begin{matrix}1110\\1000\end{matrix}\ \begin{matrix}E\\8\end{matrix} = и$

6) $1283 = 11101101 = \begin{matrix}1110\\1101\end{matrix}\ \begin{matrix}E\\D\end{matrix} = н$

7) $1283 = 11100000 = \begin{matrix}1110\\0000\end{matrix}\ \begin{matrix}E\\0\end{matrix} = а$

Третье сообщение:

1) $924 = 11001100 = \begin{matrix}1100\\1100\end{matrix}\ \begin{matrix}C\\C\end{matrix} = М$

2) $903 = 11100101 = \begin{matrix}1110\\0101\end{matrix}\ \begin{matrix}E\\5\end{matrix} = е$

3) $693 = 11100100 = \begin{matrix}1110\\0100\end{matrix}\ \begin{matrix}E\\4\end{matrix} = д$

4) $1513 = 11111100 = \begin{matrix}1111\\1100\end{matrix}\ \begin{matrix}F\\C\end{matrix} = ь$

Дешифруем числовые сообщения: для этого осуществим перевод найденной битовой последовательности в десятичную систему счисления. Чтобы перевести число из двоичной системы счисления в десятичную, нужно: справа налево пронумеровать цифры в двоичном числе, начиная с 0, далее слева направо умножить каждую цифру двоичного числа на 2 в степени номера этой цифры и сложить полученные значения.

Число в 2-ой системе счисления	Перевод в 10-ую систему счисления	Число в 10-ой системе счисления
Второе сообщение		
10101001_2	$1 \times 2^7 + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^0 = 128 + 32 + 8 + 1 = 169$	169_{10}
11010110_2	$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 = 128 + 64 + 16 + 4 + 2 = 214$	214_{10}
Четвертое сообщение		
10111110_2	$1 \times 2^7 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 = 128 + 32 + 16 + 8 + 4 + 2 = 190$	190_{10}
1011111_2	$1 \times 2^6 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 64 + 16 + 8 + 4 + 2 + 1 = 95$	95_{10}

В ответе необходимо указать последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» - «Длина», «Ширина» (числа в десятичной системе). Следовательно, ответ задачи выглядит следующим образом: Платина – 169, 214, Медь – 190, 95.

Ответ: Платина – 169, 214, Медь – 190, 95

Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Python

```
def find_pairs(m):
    pairs = []
    for n in range(m):
        for nn in range(m):
            if (n * nn) % m == 1:
                pairs.append((n, nn))
    return pairs

m = int(input("Enter the value of m: "))
pairs = find_pairs(m)
for n, nn in pairs:
    print(f"({n}, {nn})")
```

Задача 5 (25 баллов)

Буквенные прямоугольники

Вениамин играет в новую интеллектуальную игру с ИИ. У них есть прямоугольная таблица размера **n** × **m**, в клетки которой они по очереди выбирают и ставят по одной из букв 'a', 'b', 'c', 'd' или 'e'. После того, как все ячейки будут заполнены, ИИ честно подсчитает количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами. Если оно чётно, выиграет Вениамин, иначе выиграет ИИ.

Осталось сделать последний ход, и этот ход делает Вениамин. Подскажите ему, какие из этих букв он может поставить, чтобы выиграть.

Формат входных данных

В первой строке содержатся два натуральных числа **n** и **m** через пробел – размеры игрового поля. **1 ≤ n, m ≤ 500.**

В следующих **n** строках содержится описание текущего заполнения таблицы. Каждая строка состоит из **m** символов, каждый из них – это буква от 'a' до 'e'. Помимо этого, в таблице содержится ровно один символ '.', который обозначает пустую клетку, в которую нужно сделать заключительный ход.

Формат выходных данных

Вывести в одну строку без пробелов список букв, которые может поставить в последнюю свободную клетку Вениамин и выиграть. Буквы не нужно разделять пробелом и требуется выводить в алфавитном порядке. Буквы должны быть из интервала от 'a' до 'e'. Если нет ни одной выигрывающей буквы, вывести сообщение «AI win» без кавычек.

Примеры

стандартный ввод	стандартный вывод
3 4 aadd a.bb eebd	bcd
1 1 .	AI win

Комментарий к примерам

Исходно, из букв 'a' можно составить 5 прямоугольников, из букв 'b' – 5 прямоугольников, из букв 'c' – ни одного, из букв 'd' – 4 прямоугольника и из букв 'e' – 3 прямоугольника.

Если поставить букву 'a' получим 9 прямоугольников из этой буквы и 12 остальных, то есть нечётное количество, выиграет ИИ.

Если поставить букву 'b' получим 8 прямоугольников из этой буквы и 12 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'c' получим 1 прямоугольник из этой буквы и 17 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'd' получим 5 прямоугольников из этой буквы и 13 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'e' получим 5 прямоугольников из этой буквы и 14 остальных, то есть нечётное количество, тогда выиграет ИИ.

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных. Результат выполнения программы для указанных исходных данных должен содержать: итоговый вывод программы и количество различных прямоугольников при подстановке каждой буквы вместо точки.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
3 3 .bb aaa acc	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Решение

Прежде чем перейти к решению, важно понимать структуру данных и подходы, которые можно использовать. У нас имеется двумерный массив (матрица) символов, и мы хотим подсчитать количество прямоугольных областей, состоящих из одного и того же символа. Важно отметить, что прямоугольник определяется как область, в которой все символы одинаковы и находятся на любых позициях в матрице, которые могут образовать прямоугольник.

Элементарное решение требует шестерного цикла: перебираем левый верхний угол двумя циклами, внутри правый нижний еще двумя и далее перебираем всю внутреннюю часть выбранного прямоугольника чтобы проверить, что она состоит из одной и той же буквы. Получим правильное, но не эффективное решение.

Основная идея эффективного решения заключается в применении алгоритма, основанного на динамическом программировании. Вот шаги, которые мы будем выполнять:

Создание вспомогательной матрицы

Мы создаем переменную d, которая будет хранить высоту столбцов для каждого символа в матрице. Эта матрица позволяет нам отслеживать, сколько последовательных символов одного и того же типа находится над каждой позицией в матрице. Если текущий элемент такой же, как элемент над ним, мы увеличиваем счетчик.

Инициализация

Мы заполняем первую строку матрицы d единицами, поскольку в ней нет элементов, находящихся выше.

Обработка матрицы

Затем мы проходим по всей матрице, заполняя d в зависимости от значений над текущей ячейкой. Если символ текущей ячейки такой же, как символ над ней, мы увеличиваем значение в d, иначе устанавливаем его в 1.

Подсчет прямоугольников

После того как матрица d заполнена, мы начинаем считывать высоту каждого столбца (начиная с нижней строки). Для каждого символа мы находим максимальную высоту, начиная с нижней строки и проводя подсчет, количества символов. Если символы отличаются, мы обнуляем счетчик. Если символ того же типа, то увеличиваем счетчик и добавляем его в общий результат.

Получение результата

После того как мы обработали всю матрицу, результатом будет общее количество найденных прямоугольников, которое мы возвращаем.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Вспомогательная матрица для хранения высот символов
vector<vector<int>> d;

// Функция для подсчета прямоугольников для символа c, позиция (posx, posy)
int F(int n, int m, vector<string> &v, char c, int posx, int posy) {
    d.resize(n, vector<int>(m, 0)); // Инициализация матрицы высот
```

```
    v[posx][posy] = c; // Установка буквы в пустую ячейку

    // Заполнение матрицы высот
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (i == 0) {
                d[i][j] = 1; // Первая строка всегда имеет высоту 1
            } else {
                d[i][j] = (v[i - 1][j] == v[i][j]) ? d[i - 1][j] + 1 : 1;
              // Высота для текущей ячейки
            }
        }
    }

    // Подсчет прямоугольников
    int res = 0;
    for (int i = 0; i < n; i++) {
        // Проход по строкам
        for (int j = i; j >= 0; j--) {
            // Проход по подстрокам для каждой строки
            int t = 0; // Количество текущих прямоугольников
            if (d[i][0] >= i - j + 1) { // Проверка первой колонки
                t = 1; // Находим хотя бы один прямоугольник
            }
            res += t; // Добавляем в общий счетчик

            for (int k = 1; k < m; k++) { // Проход по столбцам
                if (d[i][k] < i - j + 1) {
                    t = 0;
                    // Сбрасываем счетчик, если высота меньше текущей
                    continue;
                }
                if (v[j][k] == v[j][k - 1]) {
                   // Если текущий символ такой же, как предыдущий
                    t++; // Увеличиваем количество прямоугольников
                } else {
                    t = 1; // Обнуляем счетчик
                }
                res += t; // Добавляем к общему результату
            }
        }
    }

    return res; // Возвращаем общее количество прямоугольников
}

signed main() {
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    // Оптимизация ввода-вывода

    int n, m;
    cin >> n >> m; // Чтение размеров массива
    int posx, posy;
    vector<string> v(n);

    // Чтение поля и нахождение позиции пустой ячейки
    for (int i = 0; i < n; i++) {
        cin >> v[i];
        for (int j = 0; j < m; j++) {
            if (v[i][j] == '.') {
                posx = i; // Запоминаем координаты пустой ячейки
```

```
                posy = j;
            }
        }
    }

    bool ok = false; // Флаг для отслеживания возможности выигрыша
    for (char c = 'a'; c <= 'e'; c++) { // Перебор букв от 'a' до 'e'
        int z = F(n, m, v, c, posx, posy);
// Подсчет прямоугольников для каждой буквы

        if (z % 2 == 0) { // Если количество четное
            ok = true; // Вениамин может выиграть
            cout << c; // Вывод выигрывающей буквы
        }
    }

    if (!ok) { // Если ни одна буква не подходит
        cout << "AI win"; // ИИ выигрывает
    }
}
```

Реализация на Python

```
def count_rectangles(n, m, grid, char, posx, posy):
    # Создаем массив d для хранения высот
    d = [[0] * m for _ in range(n)]

    # Заполняем пустую позицию выбранным символом
    grid[posx][posy] = char

    # Вычисляем высоты столбцов для каждого символа
    for i in range(n):
        for j in range(m):
            if i == 0:
                d[i][j] = 1
            else:
                if grid[i - 1][j] == grid[i][j]:
                    d[i][j] = d[i - 1][j] + 1
                else:
                    d[i][j] = 1

    # Подсчет прямоугольников
    res = 0

    for i in range(n):
        for j in range(i, -1, -1):
            t = 0
            if d[i][0] >= i - j + 1:
                t = 1

            res += t

            for k in range(1, m):
                if d[i][k] < i - j + 1:
                    t = 0
                    continue

                if grid[j][k] == grid[j][k - 1]:
                    t += 1
                    res += t
                    continue
```

```
                t = 1
                res += t

    return res

def main():
    # Чтение входных данных
    n, m = map(int, input().split())
    grid = [input().strip() for _ in range(n)]

    # Нахождение позиции пустой клетки
    posx, posy = -1, -1
    for i in range(n):
        for j in range(m):
            if grid[i][j] == '.':
                posx, posy = i, j
                break
        if posx != -1:
            break

    winning_chars = []

    # Проверка символов от 'a' до 'e'
    for char in 'abcde':
        rectangle_count = count_rectangles(n, m, \
          [list(row) for row in grid], char, posx, posy)
        if rectangle_count % 2 == 0:  # Проверка четности
            winning_chars.append(char)

    # Вывод результата
    if winning_chars:
        print(''.join(winning_chars))
    else:
        print("AI win")

if __name__ == "__main__":
    main()
```

Основной шаг алгоритма включает обход всех ячеек матрицы, чтобы определить высоту каждого столбца для каждого символа. Вы должны пройти по всем элементам матрицы и обновить вспомогательную матрицу d. Этот процесс также занимает O(m×n) времени, так как каждый элемент матрицы посещается один раз. После того, как высоты столбцов определены, необходимо подсчитать количество возможных прямоугольников. Для этого вам нужно будет обработать каждую строку матрицы, а затем для каждой строки пройтись по ее высотам, чтобы вычислить возможные прямоугольники, основываясь на текущих высотах. Общее время на этом этапе также останется в пределах O(m×n).

Таким образом, учитывая все шаги алгоритма, общая временная сложность будет составлять O(m×n), где m — количество строк, а n — количество столбцов. Динамическое программирование является ключом к оптимизации решений для задач, связанных с подсчетом и комбинаторикой, что позволяет избежать избыточного перебора всех возможных комбинаций.

В решении используется вспомогательная матрица d для хранения высоты столбцов для каждого символа. Если исходная матрица имеет

размер m×n, то для хранения этой вспомогательной матрицы нам потребуется O(m×n) памяти. Это означает, что по памяти сложность равна также O(m×n). В дополнение к матрице d, могут потребоваться несколько дополнительных переменных для хранения промежуточных результатов, таких как счетчики или индексы. Однако количество этих переменных не зависит от размера входных данных и является константным, что не влияет существенно на общую сложность по памяти.

Результат выполнения алгоритма для указанных исходных данных

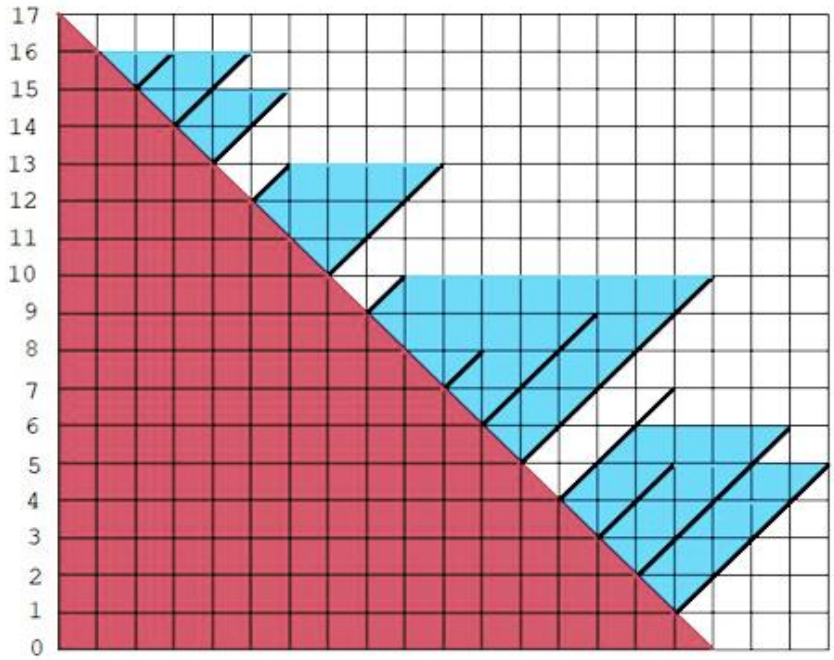
стандартный ввод	стандартный вывод
3 3 .bb aaa acc	AI win

Количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами, при подстановке вместо точки для каждой буквы: a – 17, b – 17, c – 15, d – 15, e – 15.

Задача 6 (25 баллов)

Противолавинные заграждения

На одном горном курорте на склоне построили систему противолавинных заграждений. Склон имеет угол 45° к горизонтали. На склоне в некоторых целых по высоте от подножия горы точках расположены пояса заграждений. Каждый пояс расположен строго перпендикулярно склону. Рассмотрим двумерное сечение этого сооружения (на рисунке).



Склон представлен в виде темного треугольника, а каждое заграждение – в виде отрезка, перпендикулярного склону. Снег, скатываясь сверху, попадает в некоторые отсеки, образуемые заграждениями, и застревает в них, постепенно заполняя эти отсеки. Так как снега очень много, каждый отсек заполняется до того момента, пока снег не забьет его до самого верха (образуется горизонтальный уровень заполнения). После этого снег начинает высыпаться из отсека и попадать в нижние отсеки. Некоторые заграждения настолько велики, что, при их заполнении оказываются заполненными и некоторые более вышерасположенные отсеки, у которых граница расположена ниже горизонтали заполнения. В тоже время в некоторые заграждения снег наоборот может не попасть из-за того, что при переполнении верхнего отсека, он вертикально падает вниз, минуя текущий отсек, так как вертикаль падения проходит мимо него.

Будем считать, что если граница заграждения находится ровно на вертикали падения либо горизонтали заполнения других отсеков, то из них снег в этот отсек не попадает. Например, заграждение на высоте 12 из примера пусто, так как его окончание и по вертикали и по горизонтали

находится на соответствующих прямых. В отсек, образуемый заграждением на высоте 3, снег из верхнего отсека непосредственно не попадает, но так как его высота меньше горизонтали заполнения отсека заграждения на высоте 2, то снег его всё-таки заполнит.

По заданным высотам и длинам заграждений требуется оценить максимальную задерживающую способность этих заграждений. Так как мы работаем с двумерной моделью, то требуется найти площадь сечения, заполненную снегом при полном заполнении всей системы заграждений.

Формат входных данных

В первой строке находится число **n** – количество заграждений на склоне $\mathbf{1 \le n \le 3*10^5}$.

В следующих **n** строках находятся по два числа $\mathbf{h_i}$ и $\mathbf{t_i}$ – высота расположения **i**-го заграждения и его длина. Длина любого заграждения равна целому числу диагоналей единичного квадратика со стороной 1 метр. В этих диагоналях она и указывается $\mathbf{0 \le h_i \le 2 * 10^9}$, $\mathbf{1 \le t_i \le 10^6}$. Заграждения перечислены во входных данных в порядке возрастания высоты их нахождения на склоне.

Формат выходных данных

Вывести одно число – максимальную площадь сечения, занятую снегом при полном заполнении всех отсеков, в которые снег может попасть. Можно показать, что она всегда будет целым числом. Толщиной заграждений можно пренебречь. Считаем, что высота склона достаточна, чтобы любое заграждение наполнилось.

Примеры

стандартный ввод	стандартный вывод
13 1 4 2 4 3 2 4 3 5 5 6 3 7 1 9 1 10 3 12 1 13 2 14 2 15 1	58

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения

программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
5 1 3 3 2 6 5 7 3 10 2	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Решение

Для начала заметим, что если отсек высотой h ничем слева не ограничен, то площадь его сечения равна h^2. Далее, если отсек высоты h ограничен другим отсеком слева, то нужно провести горизонталь от отсека высоты h на этот ограничивающий и узнать высоту точки пересечения. Допустим, высота этой точки равна z. Тогда площадь заполнения нашего отсека будет равна $h^2 – z^2$. Это доказывает целочисленность ответа.

Основной операцией для этой задачи рассмотрим вертикальное и горизонтальное смещение заграждения. Например, вертикальное смещение: берем наше заграждение, фиксируем вертикальные линии сетки, которые его ограничивают и двигаем это заграждение по этим линиям вниз. Очевидно, в каждой целой точке склона это заграждение будут иметь все меньшую высоту, равную t.len - (t.h - v[i].h), (где t.len – исходная высота заграждения, t.h - v[i].h – смещение) пока не уйдет со склона в минус. Аналогично горизонтальное движение влево.

Теперь эффективно найдем те отсеки, в которые снег попадает (или не попадает) сверху. Добавим фиктивный отсек на высоте $2 * 10^9$ длиной 0, объявим его текущим. Далее перебираем заграждения в обратном порядке и смещаем текущее заграждение вертикально так, чтобы оно стало на диагональ рассматриваемого. Если текущее стало строго меньше

рассматриваемого, то рассматриваемое становится текущим, в него снег попадает (отмечаем это 1 в соответствующей ячейке массива from_up), те отсеки, в которые снег сверху не попадает, остаются отмеченными 0.

Далее найдем ответ. Для этого добавим снизу фиктивный отсек на высоте -1 длиной 0, объявим его текущим. Перебираем заграждения снизу вверх (в порядке возрастания высоты крепления) и горизонтально сдвигаем текущее в точку рассматриваемого. Если текущее стало меньше либо равно 0, это означает, что его ничто не ограничивает слева и если в текущее попадает снег сверху, то добавим квадрат его высоты к ответу.

Если текущее больше 0, но меньше либо равно рассматриваемому, то (если в текущее попадает снег), добавим к ответу разницу соответствующих квадратов, текущим сделаем рассматриваемое.

Если текущее строго больше рассматриваемого в точке, то рассматриваемое никак не влияет на заполнение и будет заполнено снегом, если в текущем этот снег есть. Такой случай обрабатывать никак не нужно.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Структура, представляющая заграждение
struct bord {
    int h;    // высота заграждения
    int len;  // длина заграждения
};

signed main() {
    // Оптимизация ввода-вывода
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n;
    cin >> n; // Считываем количество заграждений
    vector<bord> v(n + 1); // Вектор для хранения заграждений
    v[0] = {-1, 0}; // Фиктивное заграждение с высотой -1 и длиной 0
    for (int i = 1; i <= n; i++) {
        cin >> v[i].h >> v[i].len; // Считываем высоту и длину заграждений
    }
    v.push_back({(int)2e9, 0}); // Фиктивное заграждение

    bord t = v.back(); // Последнее заграждение
    int last; // Поддерживаем индекс последнего заграждения
    vector<int> from_up(n + 1, 0); // Массив о попадании снега
    // Идем по заграждениям снизу вверх, определяя может ли быть заполнено
    for (int i = n; i >= 1; i--) {
        int tlen = t.len - (t.h - v[i].h);
        if (tlen < v[i].len) {
            from_up[i] = 1; // Текущее заграждение может быть заполнено
            t = v[i]; // Обновляем текущее заграждение
        }
    }
    t = v[0]; // Сбрасываем текущее заграждение на первое
    last = 0; // Обнуляем индекс последнего заграждения
```

```
    int ans = 0; // Переменная для хранения площади, занятой снегом
    //Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
     for (int i = 1; i <= n + 1; i++) {
        int tlen = t.len - (v[i].h - t.h);
        if (tlen <= 0) {
            if (from_up[last]) {
                ans += v[last].len * v[last].len;
            }
            t = v[i];
            last = i;
            continue;
        }
        if (tlen <= v[i].len) {
            if (from_up[last]) {
                ans += (v[last].len * v[last].len - tlen * tlen);
            }
            t = v[i];
            last = i;
        }
    }
    cout << ans << endl; // Выводим результирующую площадь, занятую снегом
}
```

Реализация на Python

```
class Bord:
    def __init__(self, h, length):
        self.h = h
        self.len = length

def main():
    n = int(input().split())
    v = [Bord(-1, 0)]  # Добавляем фиктивный отсек на высоте -1
    for i in range(1, n + 1):
        h, t = map(int, input().split())
        v.append(Bord(h, t))
    v.append(Bord(2 * 10**9, 0))  # Добавляем отсек на высоте 2 * 10^9
    from_up = [0] * (n + 1) # Массив о попадании снега сверху
    t = v[-1]
# Идем по заграждениям снизу вверх
    for i in range(n, 0, -1):
        tlen = t.len - (t.h - v[i].h)
        if tlen < v[i].len:
            from_up[i] = 1
            t = v[i]

    t = v[0]
    last = 0
    ans = 0
    # Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
    for i in range(1, n + 2):  # +2, из-за фиктивного отсека
        tlen = t.len - (v[i].h - t.h)
        if tlen <= 0:
            if from_up[last]:
                ans += v[last].len * v[last].len
            t = v[i]
            last = i
            continue

        if tlen <= v[i].len:
```

```
                if from_up[last]:
                    ans += (v[last].len * v[last].len - tlen * tlen)
                t = v[i]
                last = i

        print(ans)

    if __name__ == "__main__":
        main()
```

Программа в основном состоит из нескольких проходов по массиву v, который содержит заграждения. Первый проход — мы считываем данные из файла и заполняем массив v объектами класса Bord. Этот проход происходит в одном цикле от 1 до n, где n — количество заграждений. Таким образом, временная сложность этого шага составляет O(n). Второй проход — во время этого прохода мы идем в обратном направлении по массиву v, чтобы определить, может ли снег попадать сверху. Это также O(n), поскольку мы проходим по всем элементам массива по одному разу. Третий проход — здесь мы снова проходим по массиву v от начала до конца, анализируя пространство между заграждениями и высоту заполнения. Это также O(n).

Таким образом, суммируя все проходы, мы получаем, что общая временная сложность программы составляет O(n).

Мы создаем массив v, длиной n+2, который хранит объекты класса Bord. Таким образом, память, занимаемая этим массивом, составляет O(n). Массив from_up также имеет размер n+1, что добавляет еще O(n) к пространственной сложности. Другие переменные, такие как t, last, и ans, занимают постоянное количество памяти, которое не зависит от размера входных данных. Таким образом, их вклад в общую пространственную сложность можно считать константным: O(1).

Итак, собрав все элементы вместе, можно заключить, что общая пространственная сложность программы также составляет O(n).

Результат выполнения алгоритма для указанных исходных данных

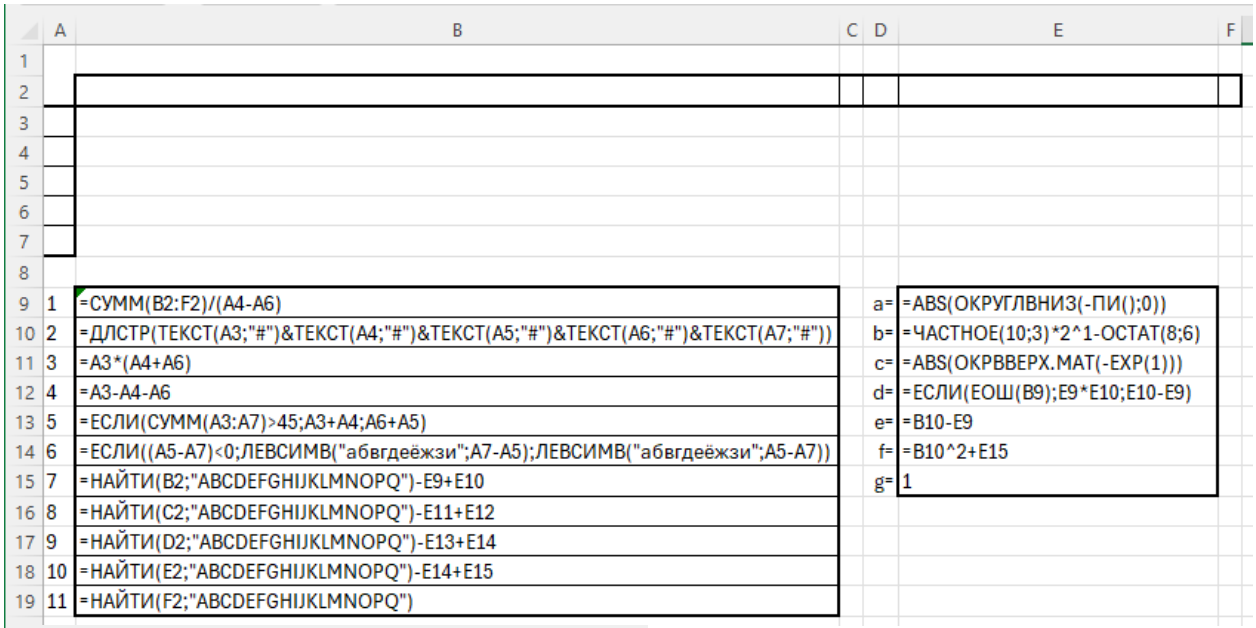
стандартный ввод	стандартный вывод
5 1 3 3 2 6 5 7 3 10 2	**36**

ВАРИАНТ 2

Задача 1 (10 баллов)

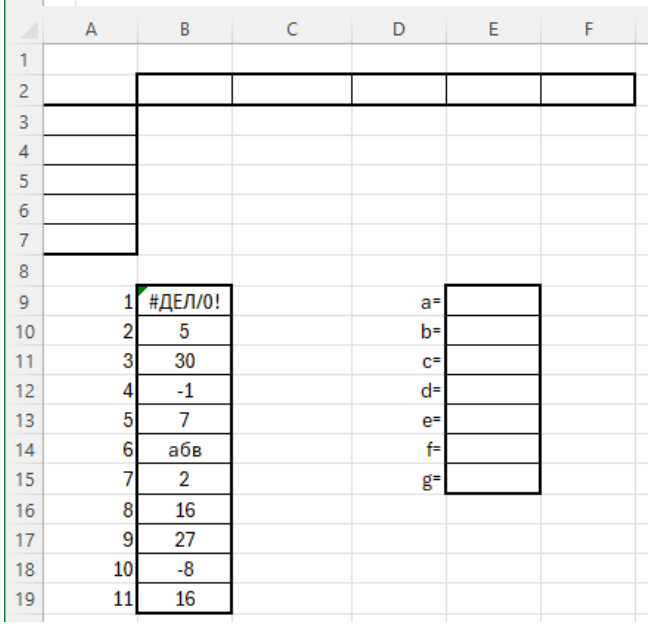
Разведчики

Нашими разведчиками на командный пункт были переданы места сосредоточения войск противника в виде горизонтальных и вертикальных координат, расположенных в ячейках B2:F2 и A3:A7. Определите их по формулам закодированной электронной таблицы, представленной в двух режимах, на рисунках расположенных ниже.



1	=СУММ(B2:F2)/(A4-A6)	
2	=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))	
3	=A3*(A4+A6)	
4	=A3-A4-A6	
5	=ЕСЛИ(СУММ(A3:A7)>45;A3+A4;A6+A5)	
6	=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))	
7	=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10	
8	=НАЙТИ(C2;"ABCDEFGHIJKLMNOPQ")-E11+E12	
9	=НАЙТИ(D2;"ABCDEFGHIJKLMNOPQ")-E13+E14	
10	=НАЙТИ(E2;"ABCDEFGHIJKLMNOPQ")-E14+E15	
11	=НАЙТИ(F2;"ABCDEFGHIJKLMNOPQ")	

a=	=ABS(ОКРУГЛВНИЗ(-ПИ();0))	
b=	=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)	
c=	=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))	
d=	=ЕСЛИ(ЕОШ(B9);E9*E10;E10-E9)	
e=	=B10-E9	
f=	=B10^2+E15	
g=	1	



1	#ДЕЛ/0!	
2	5	
3	30	
4	-1	
5	7	
6	абв	
7	2	
8	16	
9	27	
10	-8	
11	16	

a=	
b=	
c=	
d=	
e=	
f=	
g=	

Ответ оформите в виде двух строк, в верхней строке запишите через запятую значения ячеек B2:F2, а в нижней строке также через запятую значения ячеек A3:A7. В случае нескольких вариантов ответов запишите их через союз «или».

Решение

Начнем с определения коэффициентов a, b, c, d, e, f и g.

Значение коэффициента a (ячейка E9) в соответствии с формулой «=ABS(ОКРУГЛВНИЗ(-ПИ();0))» равно 3.

Значение коэффициента b (ячейка E10) в соответствии с формулой «=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)» равно 4.

Значение коэффициента c (ячейка E11) в соответствии с формулой «=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))» равно 2.

Значение коэффициента d (ячейка E12) в соответствии с формулой «=ЕСЛИ(ЕОШ(B9);E9*E10;E10-E9)» равно 12.

Значение коэффициента e (ячейка E13) в соответствии с формулой «=B10-E9» равно 2.

Значение коэффициента f (ячейка E14) в соответствии с формулой «=B10^2+E15» равно 26.

Значение коэффициента g (ячейка E15) равно 1.

Получаем:

	D	E
8		
9	a=	3
10	b=	4
11	c=	2
12	d=	12
13	e=	2
14	f=	26
15	g=	1

1) Так как значение формулы в ячейке B9 «=СУММ(B2:F2)/(A4-A6)» определяется как ошибка деления на 0, то значения ячеек A4 и A6 одинаковы.

2) Значение формулы в ячейке B10 «=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))» равно 5, следовательно значения координат в ячейках A3:A7 состоят из одного символа.

3) Формулы в ячейках B11 «=A3*(A4+A6)» и B12 «=A3-A4-A6» рассмотрим, как систему уравнений, учитывая, что значения ячеек A4 и A6 одинаковы.

$$\begin{cases} A3 * (A4 + A6) = 30 \\ A3 - A4 - A6 = -1 \end{cases} \Rightarrow \begin{cases} A3 * 2A4 = 30 \\ A3 - 2A4 = -1 \end{cases}$$

Решая систему уравнений, получаем два набора значений для ячеек A3 и A4 (A3 = 5 и A4 = 3) либо (A3 = -6 и A4 = -2,5). Так как, в соответствии со вторым пунктом, значения ячеек A3:A7 состоят из одного символа, а это диапазон от 0 до 9, то A3 = 5, A4 = A6 = 3.

4) Значение формулы в ячейке B13 «=ЕСЛИ(СУММ(A3:A7)>45; A3+A4;A6+A5)» равно 7. Максимальное значение формулы СУММ(A3:A7) из-за того, что значения в этих ячейках не превышают 9, равно 45. В соответствие с этим, условие в формуле ячейки B13 не выполнится, следовательно A6 + A5 = 7 и A5 = 7 - 3 = 4.

5) Значение формулы в ячейке B14 «=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))» равно «абв». Следовательно разница между ячейками A7 и A5 равна 3, откуда A7 равно 1 или 7.

6) Значение формулы в ячейке B15 «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10» равно 2. Откуда, значение формулы «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")» = 2 + E9 – E10 = 2 + 3 - 4 = 1, следовательно B2 = «A».

7) Значения ячеек C2, D2, E2 и F2 аналогичным образом находим из формул в ячейках B16, B17, B18 и B19. Получаем C2 = «F», D2 = «C», E2 = «Q» и F2 = «P».

Ответ: **G, B, D, F, C**
5, 3, 4, 3, 1 или 7

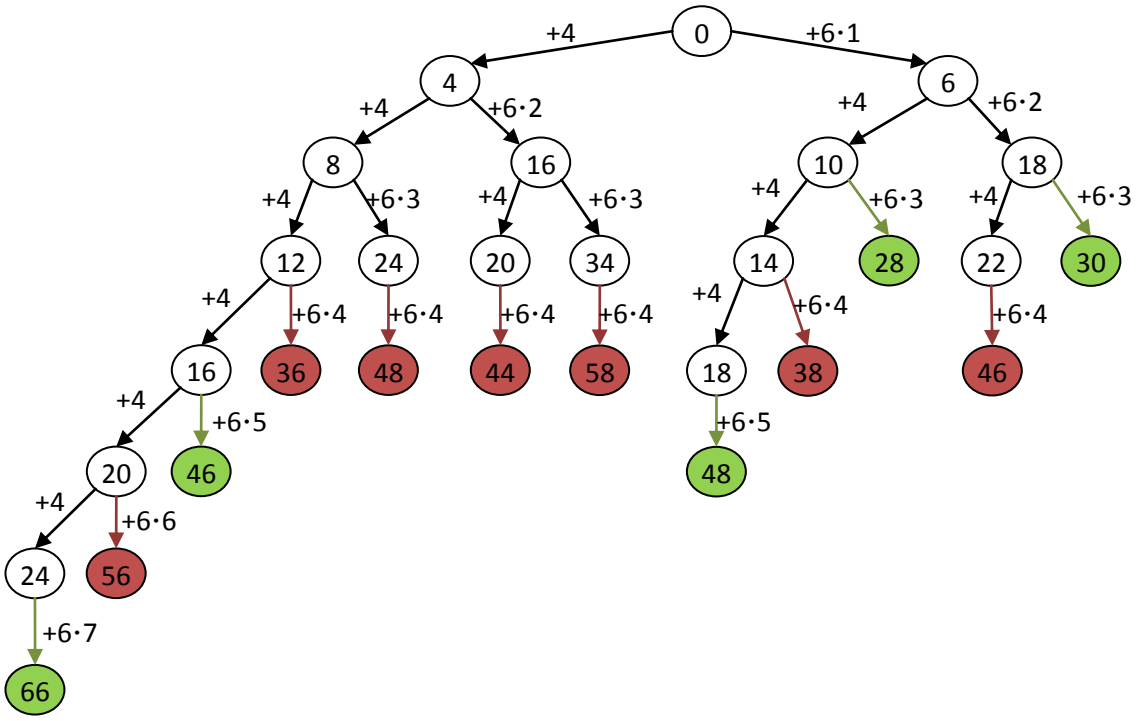
Задача 2 (10 баллов)

Бельчата и орехи: борьба за зимний запас

Два бельчонка накопили орехов на зиму и решили сыграть в следующую игру. По очереди они складывают орехи в кучку. За один ход можно либо положить в кучку четыре ореха, либо положить количество орехов, равное шесть умноженное на номер хода в игре. При превышении количества 25 орехов в кучке, игра заканчивается. Кто первым, сделав ход, получит в кучке 25 орехов или больше – считается победителем. Кто выигрывает при безошибочной игре – бельчонок, делающий первый ход, или бельчонок, делающий второй ход? Каково минимальное число ходов до победы, если выигрывающий бельчонок пытается закончить игру за меньшее число ходов, а проигрывающий пытается максимально задержать свое поражение? Какое максимальное число орехов может оказаться в кучке в конце игры? Какой бельчонок при этом победит? Поясните алгоритм графически, используя ориентированные графы в виде деревьев, с обозначением выигрышных и проигрышных позиций как вершин, а действий игроков – дугами графа с указанием численных изменений.

Решение

Построим ориентированный граф, где зеленым цветом выделены выигрышные позиции первого бельчонка, красным цветом – выигрышные позиции второго бельчонка, а номер соответствует количеству орехов в кучке. Рядом со стрелкой указано изменение числа орехов в кучке на этом ходу. В случае, если один из вариантов хода приводит к победе, то второй вариант не отображается, за исключением самой правой ветви, которая демонстрирует игру с максимально возможным числом орехов в конце игры.



Из графа видно, что при любой игре второго бельчонка, гарантированно победит первый бельчонок на 3 ходу, если первым ходом добавит 6 орехов. Максимально возможное число орехов можно получить, если увеличивать количество орехов на минимальное значение (+4) до последнего хода перед достижением победы, в котором берется вариант с большим значением (+6·7). При этом побеждает первый бельчонок с количеством орехов 66.

Ответ: гарантированно победит первый бельчонок за 3 хода, максимум 66 орехов при победе первого бельчонка.

Задача 3 (10 баллов)

Логическая головоломка: разгадка функции F

Паша решил усовершенствовать задание ЕГЭ по информатике.

Логическая функция задается выражением:

$$(x \vee \neg z \vee \neg w) \equiv (\neg x \rightarrow y)$$

Ниже приведен фрагмент таблицы истинности функции F, содержащий неповторяющиеся строки. Однако вместо значений, Паша решил вписать тоже несколько логических выражений, результат которых надо определить, если **a, b, c** – следующие высказывания:

a – «Лондон - столица Германии»

b – «H_2O - химическая формула воды»

c – «2024 делится на 3 без остатка»

Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных **w,x,y,z.**

w	?	?	?	F
$b \oplus a \vee c$		$a \wedge b \rightarrow c$		0
$b \vee a \vee c$				0
	$\neg c \wedge \neg a \wedge \neg b$	$a \wedge b \oplus c$		0
	$c \equiv a \wedge b$			0

В ответе напишите буквы **w,x,y,z** в том порядке, в котором идут соответствующие им столбцы.

Решение

Сперва определим значения логических высказываний. Очевидно, что

a=0

b=1

c=0

Теперь определим значения логических выражений в таблице. В результате получим следующую таблицу:

w	**?**	**?**	**?**	**F**
1		1		0
1				0
	0	0		0
	1			0

Следующим шагом будет составление таблицы истинности для функции F

w	x	y	z	¬z	x∨¬z	¬w	x∨¬z∨¬w	¬x	¬x→y	F
0	0	0	0	1	1	1	1	1	0	0
0	0	0	1	0	0	1	1	1	0	0
0	0	1	0	1	1	1	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1
0	1	0	1	0	1	1	1	0	1	1
0	1	1	0	1	1	1	1	0	1	1
0	1	1	1	0	1	1	1	0	1	1
1	0	0	0	1	1	0	1	1	0	0
1	0	0	1	0	0	0	0	1	0	1
1	0	1	0	1	1	0	1	1	1	1
1	0	1	1	0	0	0	0	1	1	0
1	1	0	0	1	1	0	1	0	1	1
1	1	0	1	0	1	0	1	0	1	1
1	1	1	0	1	1	0	1	0	1	1
1	1	1	1	0	1	0	1	0	1	1

Выберем из нее строки, где функция F = 0.

w	x	y	z	F
0	0	0	0	0
0	0	0	1	0
1	0	0	0	0
1	0	1	1	0

Сравнивая эту таблицу с заданной в задании, и учитывая, что строки в этой таблице выписаны в случайном порядке, легко установить с помощью несложных рассуждений искомые наименования столбцов

w	z	y	x	F
1	1	1	0	0
1	0	0	0	0
0	0	0	0	0
0	1	0	0	0

Ответ: w,z,y,x

Задача 4 (20 баллов)

Зашифрованный Рудник

В горнодобывающей компании «Рудник-Х» используется система шифрования для передачи информации о наименовании полезного ископаемого и параметрах пласта с данным ценным компонентом (длина, ширина), которая была получена при проведении геологической разведки месторождений полезных ископаемых. Передача информации осуществляется геологами с места проведения полевых работ в административное управление компании.

Процесс шифрования в горнодобывающей компании заключается в следующем:

1) Генерируется супервозрастающая последовательность чисел $A = \{a_1, a_2, \dots a_k\}$, каждый член которой больше суммы всех предыдущих членов, т.е. при $j = 2,3, \dots k: a_j > \sum_{i=1}^{j-1} a_i$, которая является секретным ключом.

Например, секретный ключ: $A = \{a_1, a_2, a_3, a_4\} = \{2, 3, 9, 17\}$,
где $a_2 > a_1 \rightarrow 3 > 2$;
$a_3 > (a_1 + a_2) \rightarrow 9 > 5$;
$a_4 > (a_1 + a_2 + a_3) \rightarrow 17 > 14$.

2) После этого выбирается число m большее, чем сумма чисел последовательности, т.е. $m > \sum_{i=1}^{k} a_i$. Также выбирается число n взаимно простое с m, т.е. наибольший общий делитель m и n равен 1.

3) Далее создается открытый ключ, т.е. новая последовательность чисел $B = \{b_1, b_2, \dots b_k\}$ путем умножения каждого числа a_i на n и взятия остатка от деления данного произведения на m.

Например, для открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ каждый член последовательности вычисляется по формуле $b_i = (a_i \cdot n) \bmod m$,
где a_i — i-й элемент последовательности секретного ключа A;
n — число взаимно простое с m;
m — модуль операции взятия остатка, иными словами, делитель;
mod — операция взятия остатка от деления.
При $n = 91$, $m = 320$ и $A = \{2, 3, 9, 17\}$:
$b_1 = (a_1 \cdot n)\ mod\ m = (2 \cdot 91)\ mod\ 320 = 182$ или, иными словами, $(2 \cdot 91)$ делится на 320 с остатком 182.
Сделав аналогичные вычисления для остальных элементов, получается последовательность $B = \{182,\ 273, 179, 267\}$

4) Полученная последовательность B используется для шифрования отправляемого сообщения. Отправляемое сообщение, представленное в виде двоичной строки, определяет, какие элементы этой последовательности суммируются для получения последовательности шифротекста $C = \{c_1, c_2, \dots c_p\}$, которая передается получателю.

Например, при шифровании числа 10 последовательность шифротекста C будет вычисляться следующим образом:
1) Число 10 представляется в двоичной системе счисления, т.е. $10 = 1010_2$;
2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1 и b_3 суммируются для получения шифротекста, b_2 и b_4 – опускаются.
3) Шифротекст $C = \{c_1\} = \{b_1 + b_3\} = 182 + 179 = 361$.
Например, при шифровании числа E_{16} последовательность шифротекста C будет вычисляться следующим образом:
1) Число E_{16} представляется в двоичной системе счисления, т.е. $E_{16} = 1110_2$;
2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1, b_2 и b_3 суммируются для получения шифротекста, b_4 – опускается.
3) Шифротекст $C = \{c_1\} = \{b_1 + b_2 + b_3\} = 182 + 273 + 179 = 634$.

Для дешифрирования сообщения получатель, исходя из известного ему n, определяет n^{-1} – мультипликативное обратное к значению n по модулю m (остаток от деления $n \cdot n^{-1}$ на m равен 1).

Иными словами, $(n \cdot n^{-1}) \bmod m = 1$,
где n — заданное число взаимно простое с m;
m — модуль операции взятия остатка;
mod — операция взятия остатка от деления.

Далее вычисляются промежуточные значения последовательности C' путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

Например, для шифротекста $C = \{c_1\} = 361$, каждый член последовательности C' вычисляется по формуле $c'_i = (c_i \cdot n^{-1}) \bmod m$,
где c_i — i-й элемент последовательности шифротекста C;
n^{-1} — мультипликативное обратное;
m — модуль операции взятия остатка;
mod — операция взятия остатка от деления.
В данном случае, последовательность
$C' = \{c'_1\} = (361 \cdot 211) \bmod 320 = 11$, где 211 – мультипликативное обратное n^{-1}, рассчитанное получателем.

Далее получатель определяет двоичную строку M путем последовательного сравнения каждого элемента последовательности C' с элементами секретного ключа A справа налево: если $c'_i \geq a_i$, то $M_i = 1$ и осуществляется вычитание a_i из c'_i, в противном случае $M_i = 0$. Таким образом восстанавливается зашифрованное сообщение.

Например, для последовательности $C' = \{c'_1\}$ и секретного ключа $A = \{a_1, a_2, a_3, a_4\}$ осуществляется последовательное сравнение элементов справа налево:
1) Если $c'_1 \geq a_4$, то $M_4 = 1$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число $c'_1 - a_4$;
2) Если $c'_1 < a_4$, то $M_4 = 0$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число c'_1.
3) В ответе получается двоичная строка $M = \{M_1, M_2, M_3, M_4\}$.

Например, при $A = \{2, 3, 9, 17\}$ и $C' = \{11\}$ двоичная строка $M = \{1, 0, 1, 0\}$, что согласуется с двоичным представлением числа 10.

Для последовательности C', состоящей из нескольких элементов, например, $C' = \{c'_1, c'_2, \dots c'_k\}$ осуществляется последовательное сравнение каждого элемента $c'_1, c'_2, \dots c'_k$ с элементами секретного ключа A с получением количества двоичных строк, равного количеству элементов последовательности C'.

1) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4\}$ количество двоичных строк M равно 4.

2) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7, c'_8\}$ количество двоичных строк M равно 8.

Геологи передали четыре последовательных сообщения в административное управление о двух месторождениях полезных ископаемых. Первое сообщение в каждой паре содержало наименование полезного ископаемого. Второе сообщение содержало информацию о параметрах пласта, при этом первое число соответствовало длине пласта, второе – ширине пласта. Для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы согласно таблице ASCII для Windows-1251, далее полученному шестнадцатеричному числовому значению присваивалась соответствующая двоичная тетрада. Для шифрования числа осуществлялся его перевод в двоичную систему. Таблица ASCII для Windows-1251:

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Были получены следующие зашифрованные сообщения:

1 сообщение: 679 2011 1546 905 1556

2 сообщение: 1707 1332

3 сообщение: 1697 2012 1454 2095 1546 905

4 сообщение: 1703 2047

Секретный ключ, хранящийся в организации, представляет собой следующую последовательность {3, 4, 8, 17, 34, 69, 137, 277}, значения n и m равны 97 и 550 соответственно.

Расшифруйте полученные сообщения. В ответе укажите последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» – «Длина», «Ширина» (в десятичной системе). Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Решение

Сначала необходимо вычислить промежуточные значения последовательности C' для осуществления дешифровки. В исходных данных сказано, что данные значения вычисляются путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

В условии дано только значение n, однако мы знаем, что n^{-1} – мультипликативное обратное к значению n по модулю m, то есть остаток от деления $n \times n^{-1}$ на m равен 1, поэтому для нахождения значения n^{-1} можно применить расширенный алгоритм Евклида.

Расширенный алгоритм Евклида является модификацией алгоритма Евклида, вычисляющая кроме наибольшего общего делителя (НОД) целых чисел a и b, еще и коэффициенты соотношения Безу, то есть такие x и y, что $ax + by = \text{НОД}(a, b)$. В свою очередь, алгоритм Евклида заключается в последовательном делении с остатком для нахождения наибольшего общего делителя двух чисел. На каждом этапе производится деление большего из пары чисел на меньшее, а остаток записывается и используется при дальнейших вычислениях в качестве нового делителя для предыдущего. Вычисление остатка производят до тех пор, пока он не будет равен нулю. Тогда последний использованный делитель и будет являться наибольшим общим делителем, полученным через алгоритм Евклида. В нашем случае наибольший общий делитель чисел n и m (97 и 550) должен равняться 1.

Применяем алгоритм Евклида:

$$550 = 97 \times 5 + 65 \Longrightarrow \quad 97 = 65 \times 1 + 32 \Longrightarrow \quad 65 = 32 \times 2 + 1 \Longrightarrow \quad 32 = 32 \times 1 + 0$$

Применяем расширенный алгоритм Евклида. Нам нужно выразить 1 как линейную комбинацию 97 и 550, т. е. найти целые числа x и y, такие, что $97x + 550y = 1$

Из алгоритма Евклида выше:

$$1 = 65 - 32 \times 2 \Longleftarrow \quad 32 = 97 - 65 \times 1 \Longleftarrow \quad 65 = 550 - 97 \times 5$$

Подставим обратно:

$$1 = 65 - (97 - 65 \times 1) \times 2 = 65 \times 3 - 97 \times 2$$
$$1 = (550 - 97 \times 5) \times 3 - 97 \times 2 = 550 \times 3 - 97 \times 17$$
$$1 = 550 \times 3 - 97 \times 17$$

Мы получили, что $1 = 550 \times 3 - 97 \times 17$, следовательно, $97 \times (-17) - 550 \times 3 = 1$. Это означает, что -17 является обратным числом 97 по модулю 550. Поскольку нам нужно положительное целое число, мы прибавляем 550 к -17: $-17 + 550 = 533$. Получаем, что $n^{-1} = 533$.

Зная значение n^{-1} находим промежуточные значения последовательности C' для каждого полученного сообщения. Для первого сообщения – 679 2011 1546 905 1556:

1) (679×533) делится на 550 с остатком 7

2) (2011×533) делится на 550 с остатком 463

3) (1546×533) делится на 550 с остатком 118

4) (905×533) делится на 550 с остатком 15

5) (1556×533) делится на 550 с остатком 498

Промежуточные значения последовательности $\{7, 463, 118, 15, 498\}$.

Для второго сообщения – 1707 1332:

1) (1707×533) делится на 550 с остатком 131

2) (1332×533) делится на 550 с остатком 456

Промежуточные значения последовательности $\{131, 456\}$.

Для третьего сообщения – 1697 2012 1454 2095 1546 905:

1) (1697×533) делится на 550 с остатком 301

2) (2012×533) делится на 550 с остатком 446

3) (1454×533) делится на 550 с остатком 32

4) (2095×533) делится на 550 с остатком 135

5) (1546×533) делится на 550 с остатком 118

6) (905×533) делится на 550 с остатком 15

Промежуточные значения последовательности $\{301, 446, 32, 135, 118, 15\}$.

Для четвертого сообщения – 1703 2047:

1) (1703×533) делится на 550 с остатком 199

2) (2047×533) делится на 550 с остатком 401

Промежуточные значения последовательности $\{199, 401\}$.

Далее используя секретный ключ A, получатель определяет битовую строку M путем последовательного сравнения C' с элементами a_i. Секретный ключ, исходя из исходных условий, представляет собой последовательность $\{3, 4, 8, 17, 34, 69, 137, 277\}$.

Для первого сообщения:

1) $7 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

2) $463 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{matrix}$

3) $118 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{matrix}$

4) $15 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

5) $498 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{matrix}$

Для второго сообщения:

1) $131 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{matrix}$

2) $456 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{matrix}$

Для третьего сообщения:

1) $301 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{matrix}$

2) $446 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{matrix}$

3) $32 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$

4) $135 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{matrix}$

5) $118 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{matrix}$

6) $15 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

Для четвертого сообщения:

1) $199 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{matrix}$

2) $401 = \begin{matrix} 3 & 4 & 8 & 17 & 34 & 69 & 137 & 277 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{matrix}$

Из условий задачи известно, что для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы, далее каждой шестнадцатеричной цифре соответствующая двоичная тетрада; для шифрования числа осуществлялся его перевод в двоичную систему. Для дешифровки необходимо произвести данные действия в обратном порядке.

Сначала дешифруем текстовые сообщения. Для этого воспользуемся таблицей тетрад и далее таблицей ASCII для Windows-1251 из условий задачи.

16-ричная цифра	Двоичная тетрада	16-ричная цифра	Двоичная тетрада
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Таким образом, произведем дешифровку текстовых сообщений:
Первое сообщение:

1) $679 = 11000000 = \begin{matrix}1100 & \text{C}\\ 0000 & 0\end{matrix} = \text{А}$

2) $2011 = 11101011 = \begin{matrix}1110 & \text{E}\\ 1011 & \text{B}\end{matrix} = \text{л}$

3) $1546 = 11101100 = \begin{matrix}1110 & \text{E}\\ 1100 & \text{C}\end{matrix} = \text{м}$

4) $905 = 11100000 = \begin{matrix}1110 & \text{E}\\ 0000 & 0\end{matrix} = \text{а}$

5) $1556 = 11100111 = \begin{matrix}1110 & \text{E}\\ 0111 & 7\end{matrix} = \text{з}$

Третье сообщение:

1) $1697 = 11010001 = \begin{matrix}1101 & \text{D}\\ 0001 & 1\end{matrix} = \text{С}$

2) $2012 = 11110011 = \begin{matrix}1111 & \text{F}\\ 0011 & 3\end{matrix} = \text{у}$

3) $1454 = 11110000 = \begin{matrix}1111 & \text{F}\\ 0000 & 0\end{matrix} = \text{р}$

4) $2095 = 11111100 = \begin{matrix}1111 & \text{F}\\ 1100 & \text{C}\end{matrix} = \text{ь}$

5) $1546 = 11101100 = \begin{matrix}1111 & \text{F}\\ 1100 & \text{C}\end{matrix} = \text{м}$

6) $905 = 11100000 = \begin{matrix}1110 & \text{E}\\ 0000 & 0\end{matrix} = \text{а}$

Дешифруем числовые сообщения: для этого осуществим перевод найденной битовой последовательности в десятичную систему счисления. Чтобы перевести число из двоичной системы счисления в десятичную, нужно: справа налево пронумеровать цифры в двоичном числе, начиная с 0, далее слева направо умножить каждую цифру двоичного числа на 2 в степени номера этой цифры и сложить полученные значения.

Число в 2-ой системе счисления	Перевод в 10-ую систему счисления	Число в 10-ой системе счисления
Второе сообщение		
10111100_2	$1\times2^7+1\times2^5+1\times2^4+1\times2^3+1\times2^2 = 128+32+16+8+4 = 188$	188_{10}
00101011_2	$1\times2^5+1\times2^3+1\times2^1+1\times2^0 = 32+8+2+1=43$	43_{10}
Четвертое сообщение		
10111010_2	$1\times2^7+1\times2^5+1\times2^4+1\times2^3+1+1\times2^1 = 128+32+16+8+2 = 186$	186_{10}
01011101_2	$1\times2^6+1\times2^4+1\times2^3+1\times2^2+1\times2^0 = 64+16+8+4+1=93$	93_{10}

В ответе необходимо указать последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» – «Длина»,

«Ширина» (числа в десятичной системе). Следовательно, ответ задачи выглядит следующим образом: Алмаз – 188, 43, Сурьма – 186, 93.

Ответ: Алмаз – 188, 43, Сурьма – 186, 93

Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Python

```
def find_pairs(m):
    pairs = []
    for n in range(m):
        for nn in range(m):
            if (n * nn) % m == 1:
                pairs.append((n, nn))
    return pairs

m = int(input("Enter the value of m: "))
pairs = find pairs(m)
for n, nn in pairs:
    print(f"({n}, {nn})")
```

Задача 5 (25 баллов)

Буквенные прямоугольники

Вениамин играет в новую интеллектуальную игру с ИИ. У них есть прямоугольная таблица размера **n** × **m**, в клетки которой они по очереди выбирают и ставят по одной из букв 'a', 'b', 'c', 'd' или 'e'. После того, как все ячейки будут заполнены, ИИ честно подсчитает количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами. Если оно чётно, выиграет Вениамин, иначе выиграет ИИ.

Осталось сделать последний ход, и этот ход делает Вениамин. Подскажите ему, какие из этих букв он может поставить, чтобы выиграть.

Формат входных данных

В первой строке содержатся два натуральных числа **n** и **m** через пробел – размеры игрового поля. **1 ≤ n, m ≤ 500.**

В следующих **n** строках содержится описание текущего заполнения таблицы. Каждая строка состоит из **m** символов, каждый из них – это буква от 'a' до 'e'. Помимо этого, в таблице содержится ровно один символ '.', который обозначает пустую клетку, в которую нужно сделать заключительный ход.

Формат выходных данных

Вывести в одну строку без пробелов список букв, которые может поставить в последнюю свободную клетку Вениамин и выиграть. Буквы не нужно разделять пробелом и требуется выводить в алфавитном порядке. Буквы должны быть из интервала от 'a' до 'e'. Если нет ни одной выигрывающей буквы, вывести сообщение «AI win» без кавычек.

Примеры

стандартный ввод	стандартный вывод
3 4 aadd a.bb eebd	bcd
1 1 .	AI win

Комментарий к примерам

Исходно, из букв 'a' можно составить 5 прямоугольников, из букв 'b' – 5 прямоугольников, из букв 'c' – ни одного, из букв 'd' – 4 прямоугольника и из букв 'e' – 3 прямоугольника.

Если поставить букву 'a' получим 9 прямоугольников из этой буквы и 12 остальных, то есть нечётное количество, выиграет ИИ.

Если поставить букву 'b' получим 8 прямоугольников из этой буквы и 12 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'c' получим 1 прямоугольник из этой буквы и 17 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'd' получим 5 прямоугольников из этой буквы и 13 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'e' получим 5 прямоугольников из этой буквы и 14 остальных, то есть нечётное количество, тогда выиграет ИИ.

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных. Результат выполнения программы для указанных исходных данных должен содержать: итоговый вывод программы и количество различных прямоугольников при подстановке каждой буквы вместо точки.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
3 3 aaa b.c bbc	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Решение

Прежде чем перейти к решению, важно понимать структуру данных и подходы, которые можно использовать. У нас имеется двумерный массив (матрица) символов, и мы хотим подсчитать количество прямоугольных областей, состоящих из одного и того же символа. Важно отметить, что прямоугольник определяется как область, в которой все символы одинаковы и находятся на любых позициях в матрице, которые могут образовать прямоугольник.

Элементарное решение требует шестерного цикла: перебираем левый верхний угол двумя циклами, внутри правый нижний еще двумя и далее перебираем всю внутреннюю часть выбранного прямоугольника чтобы проверить, что она состоит из одной и той же буквы. Получим правильное, но не эффективное решение.

Основная идея эффективного решения заключается в применении алгоритма, основанного на динамическом программировании. Вот шаги, которые мы будем выполнять:

Создание вспомогательной матрицы

Мы создаем переменную d, которая будет хранить высоту столбцов для каждого символа в матрице. Эта матрица позволяет нам отслеживать, сколько последовательных символов одного и того же типа находится над каждой позицией в матрице. Если текущий элемент такой же, как элемент над ним, мы увеличиваем счетчик.

Инициализация

Мы заполняем первую строку матрицы d единицами, поскольку в ней нет элементов, находящихся выше.

Обработка матрицы

Затем мы проходим по всей матрице, заполняя d в зависимости от значений над текущей ячейкой. Если символ текущей ячейки такой же, как символ над ней, мы увеличиваем значение в d, иначе устанавливаем его в 1.

Подсчет прямоугольников

После того как матрица d заполнена, мы начинаем считывать высоту каждого столбца (начиная с нижней строки). Для каждого символа мы находим максимальную высоту, начиная с нижней строки и проводя подсчет, количества символов. Если символы отличаются, мы обнуляем счетчик. Если символ того же типа, то увеличиваем счетчик и добавляем его в общий результат.

Получение результата

После того как мы обработали всю матрицу, результатом будет общее количество найденных прямоугольников, которое мы возвращаем.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Вспомогательная матрица для хранения высот символов
vector<vector<int>> d;

// Функция для подсчета прямоугольников для символа c, позиция (posx, posy)
int F(int n, int m, vector<string> &v, char c, int posx, int posy) {
    d.resize(n, vector<int>(m, 0)); // Инициализация матрицы высот
```

```
    v[posx][posy] = c; // Установка буквы в пустую ячейку

    // Заполнение матрицы высот
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (i == 0) {
                d[i][j] = 1; // Первая строка всегда имеет высоту 1
            } else {
                d[i][j] = (v[i - 1][j] == v[i][j]) ? d[i - 1][j] + 1 : 1;
              // Высота для текущей ячейки
            }
        }
    }

    // Подсчет прямоугольников
    int res = 0;
    for (int i = 0; i < n; i++) {
        // Проход по строкам
        for (int j = i; j >= 0; j--) {
            // Проход по подстрокам для каждой строки
            int t = 0; // Количество текущих прямоугольников
            if (d[i][0] >= i - j + 1) { // Проверка первой колонки
                t = 1; // Находим хотя бы один прямоугольник
            }
            res += t; // Добавляем в общий счетчик

            for (int k = 1; k < m; k++) { // Проход по столбцам
                if (d[i][k] < i - j + 1) {
                    t = 0;
                    // Сбрасываем счетчик, если высота меньше текущей
                    continue;
                }
                if (v[j][k] == v[j][k - 1]) {
                   // Если текущий символ такой же, как предыдущий
                    t++; // Увеличиваем количество прямоугольников
                } else {
                    t = 1; // Обнуляем счетчик
                }
                res += t; // Добавляем к общему результату
            }
        }
    }

    return res; // Возвращаем общее количество прямоугольников
}

signed main() {
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    // Оптимизация ввода-вывода

    int n, m;
    cin >> n >> m; // Чтение размеров массива
    int posx, posy;
    vector<string> v(n);

    // Чтение поля и нахождение позиции пустой ячейки
    for (int i = 0; i < n; i++) {
        cin >> v[i];
        for (int j = 0; j < m; j++) {
            if (v[i][j] == '.') {
                posx = i; // Запоминаем координаты пустой ячейки
```

```
                posy = j;
            }
        }
    }

    bool ok = false; // Флаг для отслеживания возможности выигрыша
    for (char c = 'a'; c <= 'e'; c++) { // Перебор букв от 'a' до 'e'
        int z = F(n, m, v, c, posx, posy);
// Подсчет прямоугольников для каждой буквы

        if (z % 2 == 0) { // Если количество четное
            ok = true; // Вениамин может выиграть
            cout << c; // Вывод выигрывающей буквы
        }
    }

    if (!ok) { // Если ни одна буква не подходит
        cout << "AI win"; // ИИ выигрывает
    }
}
```

Реализация на Python

```
def count_rectangles(n, m, grid, char, posx, posy):
    # Создаем массив d для хранения высот
    d = [[0] * m for _ in range(n)]

    # Заполняем пустую позицию выбранным символом
    grid[posx][posy] = char

    # Вычисляем высоты столбцов для каждого символа
    for i in range(n):
        for j in range(m):
            if i == 0:
                d[i][j] = 1
            else:
                if grid[i - 1][j] == grid[i][j]:
                    d[i][j] = d[i - 1][j] + 1
                else:
                    d[i][j] = 1

    # Подсчет прямоугольников
    res = 0

    for i in range(n):
        for j in range(i, -1, -1):
            t = 0
            if d[i][0] >= i - j + 1:
                t = 1

            res += t

            for k in range(1, m):
                if d[i][k] < i - j + 1:
                    t = 0
                    continue

                if grid[j][k] == grid[j][k - 1]:
                    t += 1
                    res += t
                    continue
```

```
                t = 1
                res += t

    return res

def main():
    # Чтение входных данных
    n, m = map(int, input().split())
    grid = [input().strip() for _ in range(n)]

    # Нахождение позиции пустой клетки
    posx, posy = -1, -1
    for i in range(n):
        for j in range(m):
            if grid[i][j] == '.':
                posx, posy = i, j
                break
        if posx != -1:
            break

    winning_chars = []

    # Проверка символов от 'a' до 'e'
    for char in 'abcde':
        rectangle_count = count_rectangles(n, m, \
          [list(row) for row in grid], char, posx, posy)
        if rectangle_count % 2 == 0:  # Проверка четности
            winning_chars.append(char)

    # Вывод результата
    if winning_chars:
        print(''.join(winning_chars))
    else:
        print("AI win")

if __name__ == "__main__":
    main()
```

Основной шаг алгоритма включает обход всех ячеек матрицы, чтобы определить высоту каждого столбца для каждого символа. Вы должны пройти по всем элементам матрицы и обновить вспомогательную матрицу d. Этот процесс также занимает O(m×n) времени, так как каждый элемент матрицы посещается один раз. После того, как высоты столбцов определены, необходимо подсчитать количество возможных прямоугольников. Для этого вам нужно будет обработать каждую строку матрицы, а затем для каждой строки пройтись по ее высотам, чтобы вычислить возможные прямоугольники, основываясь на текущих высотах. Общее время на этом этапе также останется в пределах O(m×n).

Таким образом, учитывая все шаги алгоритма, общая временная сложность будет составлять O(m×n), где m — количество строк, а n — количество столбцов. Динамическое программирование является ключом к оптимизации решений для задач, связанных с подсчетом и комбинаторикой, что позволяет избежать избыточного перебора всех возможных комбинаций.

В решении используется вспомогательная матрица d для хранения высоты столбцов для каждого символа. Если исходная матрица имеет

размер m×n, то для хранения этой вспомогательной матрицы нам потребуется O(m×n) памяти. Это означает, что по памяти сложность равна также O(m×n). В дополнение к матрице d, могут потребоваться несколько дополнительных переменных для хранения промежуточных результатов, таких как счетчики или индексы. Однако количество этих переменных не зависит от размера входных данных и является константным, что не влияет существенно на общую сложность по памяти.

Результат выполнения алгоритма для указанных исходных данных

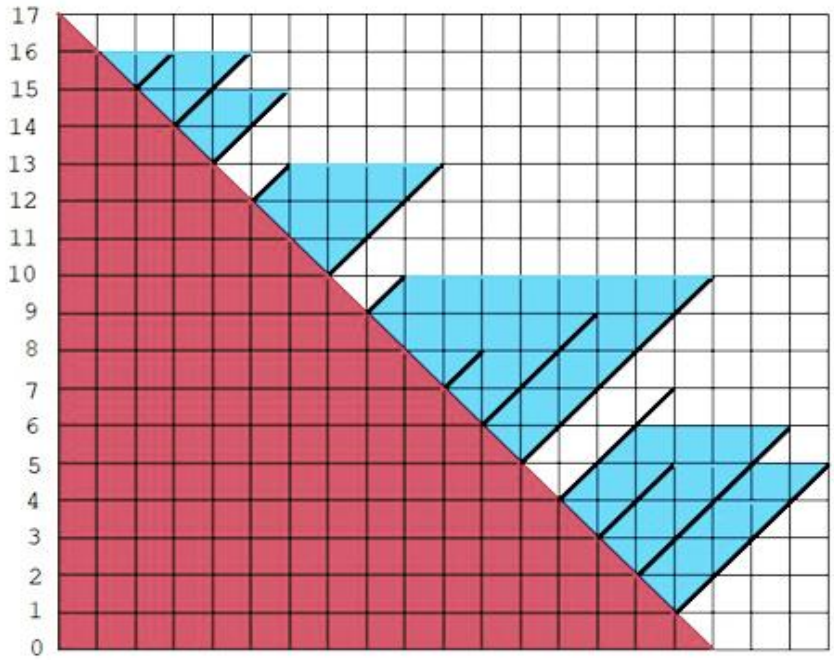
стандартный ввод	стандартный вывод
3 3 aaa b.c bbc	abc

Количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами, при подстановке вместо точки для каждой буквы: a – 16, b – 18, c – 16, d – 15, e – 15.

Задача 6 (25 баллов)

Противолавинные заграждения

На одном горном курорте на склоне построили систему противолавинных заграждений. Склон имеет угол 45° к горизонтали. На склоне в некоторых целых по высоте от подножия горы точках расположены пояса заграждений. Каждый пояс расположен строго перпендикулярно склону. Рассмотрим двумерное сечение этого сооружения (на рисунке).



Склон представлен в виде темного треугольника, а каждое заграждение – в виде отрезка, перпендикулярного склону. Снег, скатываясь сверху, попадает в некоторые отсеки, образуемые заграждениями, и застревает в них, постепенно заполняя эти отсеки. Так как снега очень много, каждый отсек заполняется до того момента, пока снег не забьет его до самого верха (образуется горизонтальный уровень заполнения). После этого снег начинает высыпаться из отсека и попадать в нижние отсеки. Некоторые заграждения настолько велики, что, при их заполнении оказываются заполненными и некоторые более вышерасположенные отсеки, у которых граница расположена ниже горизонтали заполнения. В тоже время в некоторые заграждения снег наоборот может не попасть из-за того, что при переполнении верхнего отсека, он вертикально падает вниз, минуя текущий отсек, так как вертикаль падения проходит мимо него.

Будем считать, что если граница заграждения находится ровно на вертикали падения либо горизонтали заполнения других отсеков, то из них снег в этот отсек не попадает. Например, заграждение на высоте 12 из примера пусто, так как его окончание и по вертикали и по горизонтали

находится на соответствующих прямых. В отсек, образуемый заграждением на высоте 3, снег из верхнего отсека непосредственно не попадает, но так как его высота меньше горизонтали заполнения отсека заграждения на высоте 2, то снег его всё-таки заполнит.

По заданным высотам и длинам заграждений требуется оценить максимальную задерживающую способность этих заграждений. Так как мы работаем с двумерной моделью, то требуется найти площадь сечения, заполненную снегом при полном заполнении всей системы заграждений.

Формат входных данных

В первой строке находится число **n** – количество заграждений на склоне $\mathbf{1 \le n \le 3*10^5}$.

В следующих **n** строках находятся по два числа $\mathbf{h_i}$ и $\mathbf{t_i}$ – высота расположения **i**-го заграждения и его длина. Длина любого заграждения равна целому числу диагоналей единичного квадратика со стороной 1 метр. В этих диагоналях она и указывается $\mathbf{0 \le h_i \le 2 * 10^9}$, $\mathbf{1 \le t_i \le 10^6}$. Заграждения перечислены во входных данных в порядке возрастания высоты их нахождения на склоне.

Формат выходных данных

Вывести одно число – максимальную площадь сечения, занятую снегом при полном заполнении всех отсеков, в которые снег может попасть. Можно показать, что она всегда будет целым числом. Толщиной заграждений можно пренебречь. Считаем, что высота склона достаточна, чтобы любое заграждение наполнилось.

Примеры

стандартный ввод	стандартный вывод
13 1 4 2 4 3 2 4 3 5 5 6 3 7 1 9 1 10 3 12 1 13 2 14 2 15 1	58

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения

программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
5 2 6 6 5 8 2 9 3 10 5	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Для начала заметим, что если отсек высотой h ничем слева не ограничен, то площадь его сечения равна h^2. Далее, если отсек высоты h ограничен другим отсеком слева, то нужно провести горизонталь от отсека высоты h на этот ограничивающий и узнать высоту точки пересечения. Допустим, высота этой точки равна z. Тогда площадь заполнения нашего отсека будет равна $h^2 – z^2$. Это доказывает целочисленность ответа.

Основной операцией для этой задачи рассмотрим вертикальное и горизонтальное смещение заграждения. Например, вертикальное смещение: берем наше заграждение, фиксируем вертикальные линии сетки, которые его ограничивают и двигаем это заграждение по этим линиям вниз. Очевидно, в каждой целой точке склона это заграждение будут иметь все меньшую высоту, равную t.len - (t.h - v[i].h), (где t.len – исходная высота заграждения, t.h - v[i].h – смещение) пока не уйдет со склона в минус. Аналогично горизонтальное движение влево.

Теперь эффективно найдем те отсеки, в которые снег попадает (или не попадает) сверху. Добавим фиктивный отсек на высоте $2 * 10^9$ длиной 0, объявим его текущим. Далее перебираем заграждения в обратном порядке и смещаем текущее заграждение вертикально так, чтобы оно стало на диагональ рассматриваемого. Если текущее стало строго меньше рассматриваемого, то рассматриваемое становится текущим, в него снег попадает (отмечаем это 1 в соответствующей ячейке массива from_up), те отсеки, в которые снег сверху не попадает, остаются отмеченными 0.

Далее найдем ответ. Для этого добавим снизу фиктивный отсек на высоте -1 длиной 0, объявим его текущим. Перебираем заграждения снизу вверх (в порядке возрастания высоты крепления) и горизонтально сдвигаем текущее в точку рассматриваемого. Если текущее стало меньше либо равно 0, это означает, что его ничто не ограничивает слева и если в текущее попадает снег сверху, то добавим квадрат его высоты к ответу.

Если текущее больше 0, но меньше либо равно рассматриваемому, то (если в текущее попадает снег), добавим к ответу разницу соответствующих квадратов, текущим сделаем рассматриваемое.

Если текущее строго больше рассматриваемого в точке, то рассматриваемое никак не влияет на заполнение и будет заполнено снегом, если в текущем этот снег есть. Такой случай обрабатывать никак не нужно.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Структура, представляющая заграждение
struct bord {
    int h;    // высота заграждения
    int len;  // длина заграждения
};

signed main() {
    // Оптимизация ввода-вывода
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n;
    cin >> n; // Считываем количество заграждений
    vector<bord> v(n + 1); // Вектор для хранения заграждений
    v[0] = {-1, 0}; // Фиктивное заграждение с высотой -1 и длиной 0
    for (int i = 1; i <= n; i++) {
        cin >> v[i].h >> v[i].len; // Считываем высоту и длину заграждений
    }
    v.push_back({(int)2e9, 0}); // Фиктивное заграждение

    bord t = v.back(); // Последнее заграждение
    int last; // Поддерживаем индекс последнего заграждения
    vector<int> from_up(n + 1, 0); // Массив о попадании снега
    // Идем по заграждениям снизу вверх, определяя может ли быть заполнено
    for (int i = n; i >= 1; i--) {
        int tlen = t.len - (t.h - v[i].h);
        if (tlen < v[i].len) {
            from_up[i] = 1; // Текущее заграждение может быть заполнено
            t = v[i]; // Обновляем текущее заграждение
        }
    }
    t = v[0]; // Сбрасываем текущее заграждение на первое
    last = 0; // Обнуляем индекс последнего заграждения

    int ans = 0; // Переменная для хранения площади, занятой снегом
    //Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
     for (int i = 1; i <= n + 1; i++) {
        int tlen = t.len - (v[i].h - t.h);
```

```
        if (tlen <= 0) {
            if (from_up[last]) {
                ans += v[last].len * v[last].len;
            }
            t = v[i];
            last = i;
            continue;
        }
        if (tlen <= v[i].len) {
            if (from_up[last]) {
                ans += (v[last].len * v[last].len - tlen * tlen);
            }
            t = v[i];
            last = i;
        }
    }
    cout << ans << endl; // Выводим результирующую площадь, занятую снегом
}
```

Реализация на Python

```
class Bord:
    def __init__(self, h, length):
        self.h = h
        self.len = length

def main():
    n = int(input().split())
    v = [Bord(-1, 0)]  # Добавляем фиктивный отсек на высоте -1
    for i in range(1, n + 1):
        h, t = map(int, input().split())
        v.append(Bord(h, t))
    v.append(Bord(2 * 10**9, 0))  # Добавляем отсек на высоте 2 * 10^9
    from_up = [0] * (n + 1) # Массив о попадании снега сверху
    t = v[-1]
# Идем по заграждениям снизу вверх
    for i in range(n, 0, -1):
        tlen = t.len - (t.h - v[i].h)
        if tlen < v[i].len:
            from_up[i] = 1
            t = v[i]

    t = v[0]
    last = 0
    ans = 0
    # Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
    for i in range(1, n + 2):  # +2, из-за фиктивного отсека
        tlen = t.len - (v[i].h - t.h)
        if tlen <= 0:
            if from_up[last]:
                ans += v[last].len * v[last].len
            t = v[i]
            last = i
            continue

        if tlen <= v[i].len:
            if from_up[last]:
                ans += (v[last].len * v[last].len - tlen * tlen)
            t = v[i]
            last = i
```

```
    print(ans)

if __name__ == "__main__":
    main()
```

Программа в основном состоит из нескольких проходов по массиву v, который содержит заграждения. Первый проход — мы считываем данные из файла и заполняем массив v объектами класса Bord. Этот проход происходит в одном цикле от 1 до n, где n — количество заграждений. Таким образом, временная сложность этого шага составляет O(n). Второй проход — во время этого прохода мы идем в обратном направлении по массиву v, чтобы определить, может ли снег попадать сверху. Это также O(n), поскольку мы проходим по всем элементам массива по одному разу. Третий проход — здесь мы снова проходим по массиву v от начала до конца, анализируя пространство между заграждениями и высоту заполнения. Это также O(n).

Таким образом, суммируя все проходы, мы получаем, что общая временная сложность программы составляет O(n).

Мы создаем массив v, длиной n+2, который хранит объекты класса Bord. Таким образом, память, занимаемая этим массивом, составляет O(n). Массив from_up также имеет размер n+1, что добавляет еще O(n) к пространственной сложности. Другие переменные, такие как t, last, и ans, занимают постоянное количество памяти, которое не зависит от размера входных данных. Таким образом, их вклад в общую пространственную сложность можно считать константным: O(1).

Итак, собрав все элементы вместе, можно заключить, что общая пространственная сложность программы также составляет O(n).

Результат выполнения алгоритма для указанных исходных данных

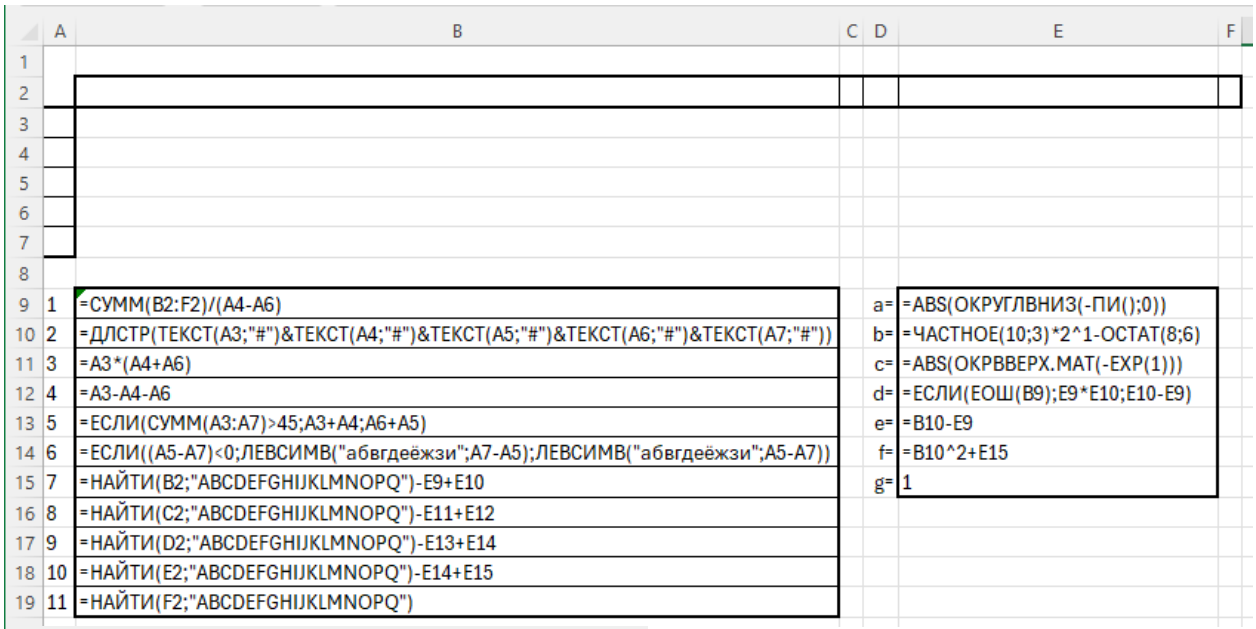
стандартный ввод	стандартный вывод
5 2 6 6 5 8 2 9 3 10 5	**78**

ВАРИАНТ 3

Задача 1 (10 баллов)

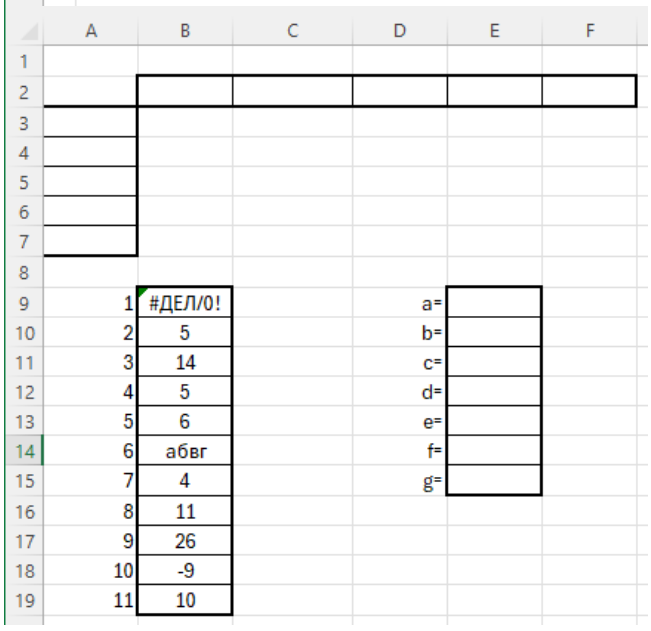
Разведчики

Нашими разведчиками на командный пункт были переданы места сосредоточения войск противника в виде горизонтальных и вертикальных координат, расположенных в ячейках B2:F2 и A3:A7. Определите их по формулам закодированной электронной таблицы, представленной в двух режимах, на рисунках расположенных ниже.



1	=СУММ(B2:F2)/(A4-A6)	
2	=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))	
3	=A3*(A4+A6)	
4	=A3-A4-A6	
5	=ЕСЛИ(СУММ(A3:A7)>45;A3+A4;A6+A5)	
6	=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))	
7	=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10	
8	=НАЙТИ(C2;"ABCDEFGHIJKLMNOPQ")-E11+E12	
9	=НАЙТИ(D2;"ABCDEFGHIJKLMNOPQ")-E13+E14	
10	=НАЙТИ(E2;"ABCDEFGHIJKLMNOPQ")-E14+E15	
11	=НАЙТИ(F2;"ABCDEFGHIJKLMNOPQ")	

a=	=ABS(ОКРУГЛВНИЗ(-ПИ();0))	
b=	=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)	
c=	=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))	
d=	=ЕСЛИ(ЕОШ(B9);E9*E10;E10-E9)	
e=	=B10-E9	
f=	=B10^2+E15	
g=	1	



1	#ДЕЛ/0!	
2	5	
3	14	
4	5	
5	6	
6	абвг	
7	4	
8	11	
9	26	
10	-9	
11	10	

a=	
b=	
c=	
d=	
e=	
f=	
g=	

Ответ оформите в виде двух строк, в верхней строке запишите через запятую значения ячеек B2:F2, а в нижней строке также через запятую значения ячеек A3:A7. В случае нескольких вариантов ответов запишите их через союз «или».

Решение

Начнем с определения коэффициентов a, b, c, d, e, f и g.

Значение коэффициента a (ячейка E9) в соответствии с формулой «=ABS(ОКРУГЛВНИЗ(-ПИ();0))» равно 3.

Значение коэффициента b (ячейка E10) в соответствии с формулой «=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)» равно 4.

Значение коэффициента c (ячейка E11) в соответствии с формулой «=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))» равно 2.

Значение коэффициента d (ячейка E12) в соответствии с формулой «=ЕСЛИ(ЕОШ(B9);E9*E10;E10-E9)» равно 12.

Значение коэффициента e (ячейка E13) в соответствии с формулой «=B10-E9» равно 2.

Значение коэффициента f (ячейка E14) в соответствии с формулой «=B10^2+E15» равно 26.

Значение коэффициента g (ячейка E15) равно 1.

Получаем:

	D	E
8		
9	a=	3
10	b=	4
11	c=	2
12	d=	12
13	e=	2
14	f=	26
15	g=	1

1) Так как значение формулы в ячейке B9 «=СУММ(B2:F2)/(A4-A6)» определяется как ошибка деления на 0, то значения ячеек A4 и A6 одинаковы.

2) Значение формулы в ячейке B10 «=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))» равно 5, следовательно значения координат в ячейках A3:A7 состоят из одного символа.

3) Формулы в ячейках B11 «=A3*(A4+A6)» и B12 «=A3-A4-A6» рассмотрим, как систему уравнений, учитывая, что значения ячеек A4 и A6 одинаковы.

$$\begin{cases} A3 * (A4 + A6) = 14 \\ A3 - A4 - A6 = 5 \end{cases} \Rightarrow \begin{cases} A3 * 2A4 = 14 \\ A3 - 2A4 = 5 \end{cases}$$

Решая систему уравнений, получаем два набора значений для ячеек A3 и A4 (A3 = 7 и A4 = 1) либо (A3 = -2 и A4 = -3,5). Так как, в соответствии со вторым пунктом, значения ячеек A3:A7 состоят из одного символа, а это диапазон от 0 до 9, то A3 = 7, A4 = A6 = 1.

4) Значение формулы в ячейке B13 «=ЕСЛИ(СУММ(A3:A7)>45; A3+A4;A6+A5)» равно 6. Максимальное значение формулы СУММ(A3:A7) из-за того, что значения в этих ячейках не превышают 9, равно 45. В соответствие с этим, условие в формуле ячейки B13 не выполнится, следовательно A6 + A5 = 6 и A5 = 6 - 1 = 5.

5) Значение формулы в ячейке B14 «=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))» равно «абвг». Следовательно разница между ячейками A7 и A5 равна 4, откуда A7 равно 1 или 9.

6) Значение формулы в ячейке B15 «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10» равно 4. Откуда, значение формулы «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")» = 4 + E9 – E10 = 4 + 3 - 4 = 3, следовательно B2 = «C».

7) Значения ячеек C2, D2, E2 и F2 аналогичным образом находим из формул в ячейках B16, B17, B18 и B19. Получаем C2 = «A», D2 = «B», E2 = «P» и F2 = «J».

Ответ: **C, A, B, P, J**
7, 1, 5, 1, 1 или 9

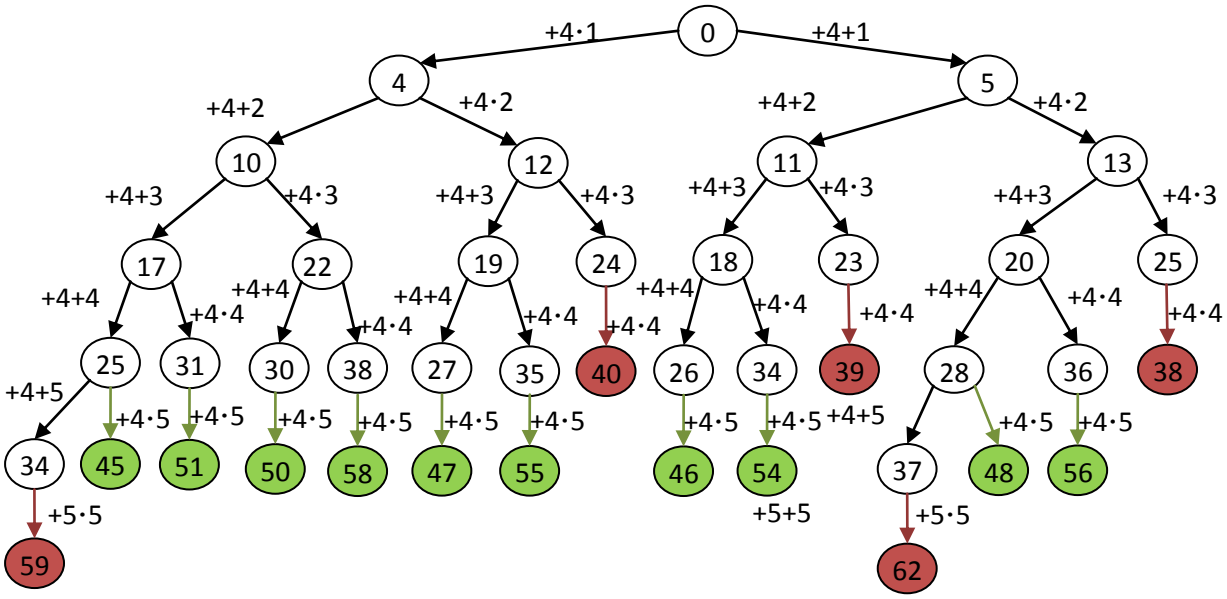
Задача 2 (10 баллов)

Вася и Петя: математическая дуэль

Вася и Петя решили сыграть в следующую игру. По очереди они складывают числа в общую сумму. За один ход можно либо прибавить четыре плюс номер хода в игре, либо прибавить четыре умноженное на номер хода в игре. При превышении значения 39 в сумме, игра заканчивается. Кто первым, сделав ход, получит в сумме 39 или больше – считается победителем. Кто выигрывает при безошибочной игре – Вася, делающий первый ход, или Петя, делающий второй ход? Каково минимальное число ходов до победы, если выигрывающий игрок пытается закончить игру за меньшее число ходов, а проигрывающий пытается максимально задержать свое поражение? Какое максимальное число может получится в сумме в конце игры? Какой игрок при этом победит? Поясните алгоритм графически, используя ориентированные графы в виде деревьев, с обозначением выигрышных и проигрышных позиций как вершин, а действий игроков – дугами графа с указанием численных изменений.

Решение

Построим ориентированный граф, где зеленым цветом выделены выигрышные позиции Васи, красным цветом – выигрышные позиции Пети, а номер соответствует числу в сумме. Рядом со стрелкой указано изменение значения в сумме на этом ходу. В случае, если один из вариантов хода приводит к победе, то второй вариант не отображается, за исключением самой ветвей, которые демонстрируют игру с максимально возможным числом орехов в конце игры.



Из графа видно, что при любой игре Пети, гарантированно победит Вася на 5 ходу. Максимально возможное число в сумме можно получить, если увеличивать количество сумму таким образом, чтобы получить как можно более высокий номер хода (5) и значения в сумме перед последним ходом (37), в котором берется вариант с большим значением (+5·5). При этом побеждает Петя с итоговым значением 62.

Ответ: гарантированно победит первый игрок (Вася) за 5 ходов, максимальная сумма 59 при победе второго игрока (Пети).

Задача 3 (10 баллов)

Логическая головоломка: разгадка функции F

Паша решил усовершенствовать задание ЕГЭ по информатике

Логическая функция задается выражением

$$(x \to z \wedge \neg z \equiv y) \vee (y \equiv w)$$

Ниже приведен фрагмент таблицы истинности функции F, содержащий неповторяющиеся строки. Однако вместо значений, Паша решил вписать тоже несколько логических выражений, результат которых надо определить, если ***a, b, c*** – следующие высказывания:

a – «В одном часе 3600 секунд»

b – «Эверест - высочайшая горная вершина на Земле»

c – «7x8=56»

Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных **w,x,y,z.**

y	?	?	?	F
		a∨b∧c		0
			c∧¬a→b	0
¬b≡c∧a	b⊕a→c			0
¬b∧¬a⊕¬c	a∨¬b→c		b∧¬c⊕a	0

В ответе напишите буквы **w,x,y,z** в том порядке, в котором идут соответствующие им столбцы.

Решение

Сперва определим значения логических высказываний. Очевидно, что

a=1
b=1
c=1

Теперь определим значения логических выражений в таблице. В результате получим следующую таблицу:

y	**?**		**?**	**?**	**F**
			1		0
				1	0
0	1				0
0	1			1	0

Следующим шагом будет составление таблицы истинности для функции F

w	x	y	z	¬z	z∧¬z	x→z∧¬z	x→z∧¬z≡y	y≡w	F
0	0	0	0	1	0	1	0	1	1
0	0	0	1	0	0	1	0	1	1
0	0	1	0	1	0	1	1	0	1
0	0	1	1	0	0	1	1	0	1
0	1	0	0	1	0	0	1	1	1
0	1	0	1	0	0	0	1	1	1
0	1	1	0	1	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0
1	0	0	1	0	0	1	0	0	0
1	0	1	0	1	0	1	1	1	1
1	0	1	1	0	0	1	1	1	1
1	1	0	0	1	0	0	1	0	1
1	1	0	1	0	0	0	1	0	1
1	1	1	0	1	0	0	0	1	1
1	1	1	1	0	0	0	0	1	1

Выберем из нее строки, где функция F = 0.

w	x	y	z	F
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0

Сравнивая эту таблицу с заданной в задании, и учитывая, что строки в этой таблице выписаны в случайном порядке, легко установить с помощью несложных рассуждений искомые наименования столбцов.

y	w	x	z	F
1	0	1	0	0
1	0	1	1	0
0	1	0	0	0
0	1	0	1	0

Ответ: y,w,x,z

Задача 4 (20 баллов)

Зашифрованный Рудник

В горнодобывающей компании «Рудник-Х» используется система шифрования для передачи информации о наименовании полезного ископаемого и параметрах пласта с данным ценным компонентом (длина, ширина), которая была получена при проведении геологической разведки месторождений полезных ископаемых. Передача информации осуществляется геологами с места проведения полевых работ в административное управление компании.

Процесс шифрования в горнодобывающей компании заключается в следующем:

1) Генерируется супервозрастающая последовательность чисел $A = \{a_1, a_2, \dots a_k\}$, каждый член которой больше суммы всех предыдущих членов, т.е. при $j = 2,3, \dots k: a_j > \sum_{i=1}^{j-1} a_i$, которая является секретным ключом.

Например, секретный ключ: $A = \{a_1, a_2, a_3, a_4\} = \{2, 3, 9, 17\}$,
где $a_2 > a_1 \rightarrow 3 > 2$;
$a_3 > (a_1 + a_2) \rightarrow 9 > 5$;
$a_4 > (a_1 + a_2 + a_3) \rightarrow 17 > 14$.

2) После этого выбирается число m большее, чем сумма чисел последовательности, т.е. $m > \sum_{i=1}^{k} a_i$. Также выбирается число n взаимно простое с m, т.е. наибольший общий делитель m и n равен 1.

3) Далее создается открытый ключ, т.е. новая последовательность чисел $B = \{b_1, b_2, \dots b_k\}$ путем умножения каждого числа a_i на n и взятия остатка от деления данного произведения на m.

Например, для открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ каждый член последовательности вычисляется по формуле $b_i = (a_i \cdot n) \bmod m$,
где a_i — i-й элемент последовательности секретного ключа A;
n — число взаимно простое с m;
m — модуль операции взятия остатка, иными словами, делитель;
mod — операция взятия остатка от деления.
При $n = 91$, $m = 320$ и $A = \{2, 3, 9, 17\}$:
$b_1 = (a_1 \cdot n)\ mod\ m = (2 \cdot 91)\ mod\ 320 = 182$ или, иными словами, $(2 \cdot 91)$ делится на 320 с остатком 182.
Сделав аналогичные вычисления для остальных элементов, получается последовательность $B = \{182,\ 273, 179, 267\}$

4) Полученная последовательность B используется для шифрования отправляемого сообщения. Отправляемое сообщение, представленное в виде двоичной строки, определяет, какие элементы этой последовательности суммируются для получения последовательности шифротекста $C = \{c_1, c_2, \dots c_p\}$, которая передается получателю.

Например, при шифровании числа 10 последовательность шифротекста C будет вычисляться следующим образом:
1) Число 10 представляется в двоичной системе счисления, т.е. $10 = 1010_2$;
2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1 и b_3 суммируются для получения шифротекста, b_2 и b_4 – опускаются.
3) Шифротекст $C = \{c_1\} = \{b_1 + b_3\} = 182 + 179 = 361$.
Например, при шифровании числа E_{16} последовательность шифротекста C будет вычисляться следующим образом:
1) Число E_{16} представляется в двоичной системе счисления, т.е. $E_{16} = 1110_2$;
2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1, b_2 и b_3 суммируются для получения шифротекста, b_4 – опускается.
3) Шифротекст $C = \{c_1\} = \{b_1 + b_2 + b_3\} = 182 + 273 + 179 = 634$.

Для дешифрирования сообщения получатель, исходя из известного ему n, определяет n^{-1} – мультипликативное обратное к значению n по модулю m (остаток от деления $n \cdot n^{-1}$ на m равен 1).

Иными словами, $(n \cdot n^{-1}) \bmod m = 1$,
где n — заданное число взаимно простое с m;
m — модуль операции взятия остатка;
mod — операция взятия остатка от деления.

Далее вычисляются промежуточные значения последовательности C' путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

Например, для шифротекста $C = \{c_1\} = 361$, каждый член последовательности C' вычисляется по формуле $c'_i = (c_i \cdot n^{-1}) \bmod m$,
где c_i — i-й элемент последовательности шифротекста C;
n^{-1} — мультипликативное обратное;
m — модуль операции взятия остатка;
mod — операция взятия остатка от деления.
В данном случае, последовательность
$C' = \{c'_1\} = (361 \cdot 211) \bmod 320 = 11$, где 211 – мультипликативное обратное n^{-1}, рассчитанное получателем.

Далее получатель определяет двоичную строку M путем последовательного сравнения каждого элемента последовательности C' с элементами секретного ключа A справа налево: если $c'_i \geq a_i$, то $M_i = 1$ и осуществляется вычитание a_i из c'_i, в противном случае $M_i = 0$. Таким образом восстанавливается зашифрованное сообщение.

Например, для последовательности $C' = \{c'_1\}$ и секретного ключа $A = \{a_1, a_2, a_3, a_4\}$ осуществляется последовательное сравнение элементов справа налево:
1) Если $c'_1 \geq a_4$, то $M_4 = 1$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число $c'_1 - a_4$;
2) Если $c'_1 < a_4$, то $M_4 = 0$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число c'_1.
3) В ответе получается двоичная строка $M = \{M_1, M_2, M_3, M_4\}$.

Например, при $A = \{2, 3, 9, 17\}$ и $C' = \{11\}$ двоичная строка $M = \{1, 0, 1, 0\}$, что согласуется с двоичным представлением числа 10.

Для последовательности C', состоящей из нескольких элементов, например, $C' = \{c'_1, c'_2, \dots c'_k\}$ осуществляется последовательное сравнение каждого элемента $c'_1, c'_2, \dots c'_k$ с элементами секретного ключа A с получением количества двоичных строк, равного количеству элементов последовательности C'.

1) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4\}$ количество двоичных строк M равно 4.

2) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7, c'_8\}$ количество двоичных строк M равно 8.

Геологи передали четыре последовательных сообщения в административное управление о двух месторождениях полезных ископаемых. Первое сообщение в каждой паре содержало наименование полезного ископаемого. Второе сообщение содержало информацию о параметрах пласта, при этом первое число соответствовало длине пласта, второе – ширине пласта. Для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы согласно таблице ASCII для Windows-1251, далее полученному шестнадцатеричному числовому значению присваивалась соответствующая двоичная тетрада. Для шифрования числа осуществлялся его перевод в двоичную систему. Таблица ASCII для Windows-1251:

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Были получены следующие зашифрованные сообщения:

1 сообщение: 411 1532 1247 1175 1072 1244

2 сообщение: 1052 1513

3 сообщение: 642 732 1017 1072 1178

4 сообщение: 637 1028

Секретный ключ, хранящийся в организации, представляет собой следующую последовательность {2, 3, 7, 13, 28, 57, 115, 226}, значения n и m равны 61 и 456 соответственно.

Расшифруйте полученные сообщения. В ответе укажите последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» – «Длина», «Ширина» (в десятичной системе). Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Решение

Сначала необходимо вычислить промежуточные значения последовательности C' для осуществления дешифровки. В исходных данных сказано, что данные значения вычисляются путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

В условии дано только значение n, однако мы знаем, что n^{-1} – мультипликативное обратное к значению n по модулю m, то есть остаток от деления $n \times n^{-1}$ на m равен 1, поэтому для нахождения значения n^{-1} можно применить расширенный алгоритм Евклида.

Расширенный алгоритм Евклида является модификацией алгоритма Евклида, вычисляющая кроме наибольшего общего делителя (НОД) целых чисел a и b, еще и коэффициенты соотношения Безу, то есть такие x и y, что $ax + by = \text{НОД}(a, b)$. В свою очередь, алгоритм Евклида заключается в последовательном делении с остатком для нахождения наибольшего общего делителя двух чисел. На каждом этапе производится деление большего из пары чисел на меньшее, а остаток записывается и используется при дальнейших вычислениях в качестве нового делителя для предыдущего. Вычисление остатка производят до тех пор, пока он не будет равен нулю. Тогда последний использованный делитель и будет являться наибольшим общим делителем, полученным через алгоритм Евклида. В нашем случае наибольший общий делитель чисел n и m (61 и 456) должен равняться 1.

Применяем алгоритм Евклида:

$$456 = 61 \times 7 + 29 \Longrightarrow \quad 61 = 29 \times 2 + 3 \Longrightarrow \quad 29 = 3 \times 9 + 2 \Longrightarrow \quad 3 = 2 \times 1 + 1$$
$$2 = 1 \times 2 + 0$$

Применяем расширенный алгоритм Евклида. Нам нужно выразить 1 как линейную комбинацию 61 и 456, т. е. найти целые числа x и y, такие, что $61x + 456y = 1$

Из алгоритма Евклида выше:

$$1 = 3 - 2 \times 1 \Leftarrow \quad 2 = 29 - 3 \times 9 \Leftarrow \quad 3 = 61 - 29 \times 2 \Leftarrow \quad 29 = 456 - 61 \times 7$$

Подставим обратно:

$$1 = 3 - (29 - 3 \times 9) \times 1 = 3 \times 10 - 29 \times 1$$
$$1 = (61 - 29 \times 2) \times 10 - 29 \times 1 = 61 \times 10 - 29 \times 21$$
$$1 = 61 \times 10 - (456 - 61 \times 7) \times 21 = 61 \times 157 - 456 \times 21$$
$$1 = 61 \times 157 - 456 \times 21$$

Мы получили, что $1 = 61 \times 157 - 456 \times 21$ Это означает, что 157 является обратным числом 96 по модулю 456. Получаем, что $n^{-1} = 157$.

Зная значение n^{-1} находим промежуточные значения последовательности C' для каждого полученного сообщения. Для первого сообщения – 411 1532 1247 1175 1072 1244:

1) (411×157) делится на 456 с остатком 231

2) (1532×157) делится на 456 с остатком 212

3) (1247×157) делится на 456 с остатком 155

4) (1175×157) делится на 456 с остатком 251

5) (1072×157) делится на 456 с остатком 40

6) (1244×157) делится на 456 с остатком 140

Промежуточные значения последовательности {231, 212, 155, 251, 40, 140}.

Для второго сообщения – 1052 1513:

1) (1052×157) делится на 456 с остатком 92

2) (1513×157) делится на 456 с остатком 421

Промежуточные значения последовательности {92, 421}.

Для третьего сообщения – 642, 732, 1017, 1072, 1178:

1) (642×157) делится на 456 с остатком 18

2) (732×157) делится на 456 с остатком 12

3) (1017×157) делится на 456 с остатком 69

4) (1072×157) делится на 456 с остатком 40

5) (1178×157) делится на 456 с остатком 256

Промежуточные значения последовательности {18, 12, 69, 40, 256}.

Для четвертого сообщения – 637 1028:

1) (637×157) делится на 456 с остатком 145

2) (1028×157) делится на 456 с остатком 428

Промежуточные значения последовательности {145, 428}.

Далее используя секретный ключ A, получатель определяет битовую строку M путем последовательного сравнения C' с элементами a_i. Секретный ключ, исходя из исходных условий, представляет собой последовательность {2, 3, 7, 13, 28, 57, 115, 226}.

Для первого сообщения:

1) $231 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$

2) $212 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{matrix}$

3) $115 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{matrix}$

4) $251 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{matrix}$

5) $40 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{matrix}$

6) $140 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{matrix}$

Для второго сообщения:

1) $92 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{matrix}$

2) $421 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{matrix}$

Для третьего сообщения:

1) $18 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{matrix}$

2) $12 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

3) $69 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{matrix}$

4) $40 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{matrix}$

5) $256 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{matrix}$

Для четвертого сообщения:

1) $145 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{matrix}$

2) $428 = \begin{matrix} 2 & 3 & 7 & 13 & 28 & 57 & 115 & 226 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{matrix}$

Из условий задачи известно, что для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы, далее каждой шестнадцатеричной цифре соответствующая двоичная тетрада; для шифрования числа осуществлялся его перевод в двоичную систему. Для дешифровки необходимо произвести данные действия в обратном порядке.

Сначала дешифруем текстовые сообщения. Для этого воспользуемся таблицей тетрад и далее таблицей ASCII для Windows-1251 из условий задачи.

16-ричная цифра	Двоичная тетрада	16-ричная цифра	Двоичная тетрада
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Таким образом, произведем дешифровку текстовых сообщений:

Первое сообщение:

1) $411 = 11000001 = \begin{matrix}1100 & \text{C}\\ 0001 & 1\end{matrix} = \text{Б}$

2) $1532 = 11101110 = \begin{matrix}1110 & \text{E}\\ 1100 & \text{E}\end{matrix} = \text{о}$

3) $1247 = 11101010 = \begin{matrix}1110 & \text{E}\\ 1010 & \text{A}\end{matrix} = \text{к}$

4) $1175 = 11110001 = \begin{matrix}1111 & \text{F}\\ 0001 & 1\end{matrix} = \text{с}$

5) $1072 = 11101000 = \begin{matrix}1110 & \text{E}\\ 1000 & 8\end{matrix} = \text{и}$

6) $1244 = 11110010 = \begin{matrix}1111 & \text{F}\\ 0010 & 2\end{matrix} = \text{т}$

Третье сообщение:

1) $642 = 11010000 = \begin{matrix}1101 & \text{D}\\ 0000 & 0\end{matrix} = \text{Р}$

2) $732 = 11100000 = \begin{matrix}1110 & \text{E}\\ 0000 & 0\end{matrix} = \text{а}$

3) $1017 = 11100100 = \begin{matrix}1110 & \text{E}\\ 0100 & 4\end{matrix} = \text{д}$

4) $1072 = 11101000 = \begin{matrix}1110 & \text{E}\\ 1000 & 8\end{matrix} = \text{и}$

5) $1178 = 11101001 = \begin{matrix}1110 & \text{E}\\ 1001 & 9\end{matrix} = \text{й}$

Дешифруем числовые сообщения: для этого осуществим перевод найденной битовой последовательности в десятичную систему счисления. Чтобы перевести число из двоичной системы счисления в десятичную, нужно: справа налево пронумеровать цифры в двоичном числе, начиная с 0, далее слева направо умножить каждую цифру двоичного числа на 2 в степени номера этой цифры и сложить полученные значения.

Число в 2-ой системе счисления	Перевод в 10-ую систему счисления	Число в 10-ой системе счисления
Второе сообщение		
00101100_2	$1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 = 32 + 8 + 4 = 44$	44_{10}
01110111_2	$1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 64 + 32 + 16 + 4 + 2 + 1 = 119$	119_{10}
Четвертое сообщение		
10001010_2	$1 \times 2^7 + 1 \times 2^3 + 1 \times 2^1 = 128 + 8 + 2 = 138$	138_{10}
10001111_2	$1 \times 2^7 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 128 + 8 + 4 + 2 + 1 = 143$	143_{10}

В ответе необходимо указать последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» - «Длина», «Ширина» (числа в десятичной системе). Следовательно, ответ задачи выглядит следующим образом: Боксит – 44, 119, Радий – 138, 143.

Ответ: Боксит – 44, 119, Радий – 138, 143

Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Python

```
def find_pairs(m):
    pairs = []
    for n in range(m):
        for nn in range(m):
            if (n * nn) % m == 1:
                pairs.append((n, nn))
    return pairs

m = int(input("Enter the value of m: "))
pairs = find_pairs(m)
for n, nn in pairs:
    print(f"({n}, {nn})")
```

Задача 5 (25 баллов)

Буквенные прямоугольники

Вениамин играет в новую интеллектуальную игру с ИИ. У них есть прямоугольная таблица размера **n** × **m**, в клетки которой они по очереди выбирают и ставят по одной из букв 'a', 'b', 'c', 'd' или 'e'. После того, как все ячейки будут заполнены, ИИ честно подсчитает количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами. Если оно чётно, выиграет Вениамин, иначе выиграет ИИ.

Осталось сделать последний ход, и этот ход делает Вениамин. Подскажите ему, какие из этих букв он может поставить, чтобы выиграть.

Формат входных данных

В первой строке содержатся два натуральных числа **n** и **m** через пробел – размеры игрового поля. **1 ≤ n, m ≤ 500.**

В следующих **n** строках содержится описание текущего заполнения таблицы. Каждая строка состоит из **m** символов, каждый из них – это буква от 'a' до 'e'. Помимо этого, в таблице содержится ровно один символ '.', который обозначает пустую клетку, в которую нужно сделать заключительный ход.

Формат выходных данных

Вывести в одну строку без пробелов список букв, которые может поставить в последнюю свободную клетку Вениамин и выиграть. Буквы не нужно разделять пробелом и требуется выводить в алфавитном порядке. Буквы должны быть из интервала от 'a' до 'e'. Если нет ни одной выигрывающей буквы, вывести сообщение «AI win» без кавычек.

Примеры

стандартный ввод	стандартный вывод
3 4 aadd a.bb eebd	bcd
1 1 .	AI win

Комментарий к примерам

Исходно, из букв 'a' можно составить 5 прямоугольников, из букв 'b' – 5 прямоугольников, из букв 'c' – ни одного, из букв 'd' – 4 прямоугольника и из букв 'e' – 3 прямоугольника.

Если поставить букву 'a' получим 9 прямоугольников из этой буквы и 12 остальных, то есть нечётное количество, выиграет ИИ.

Если поставить букву 'b' получим 8 прямоугольников из этой буквы и 12 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'c' получим 1 прямоугольник из этой буквы и 17 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'd' получим 5 прямоугольников из этой буквы и 13 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'e' получим 5 прямоугольников из этой буквы и 14 остальных, то есть нечётное количество, тогда выиграет ИИ.

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных. Результат выполнения программы для указанных исходных данных должен содержать: итоговый вывод программы и количество различных прямоугольников при подстановке каждой буквы вместо точки.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
3 3 aaa cbb cc.	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Решение

Прежде чем перейти к решению, важно понимать структуру данных и подходы, которые можно использовать. У нас имеется двумерный массив (матрица) символов, и мы хотим подсчитать количество прямоугольных областей, состоящих из одного и того же символа. Важно отметить, что прямоугольник определяется как область, в которой все символы одинаковы и находятся на любых позициях в матрице, которые могут образовать прямоугольник.

Элементарное решение требует шестерного цикла: перебираем левый

верхний угол двумя циклами, внутри правый нижний еще двумя и далее перебираем всю внутреннюю часть выбранного прямоугольника чтобы проверить, что она состоит из одной и той же буквы. Получим правильное, но не эффективное решение.

Основная идея эффективного решения заключается в применении алгоритма, основанного на динамическом программировании. Вот шаги, которые мы будем выполнять:

Создание вспомогательной матрицы

Мы создаем переменную d, которая будет хранить высоту столбцов для каждого символа в матрице. Эта матрица позволяет нам отслеживать, сколько последовательных символов одного и того же типа находится над каждой позицией в матрице. Если текущий элемент такой же, как элемент над ним, мы увеличиваем счетчик.

Инициализация

Мы заполняем первую строку матрицы d единицами, поскольку в ней нет элементов, находящихся выше.

Обработка матрицы

Затем мы проходим по всей матрице, заполняя d в зависимости от значений над текущей ячейкой. Если символ текущей ячейки такой же, как символ над ней, мы увеличиваем значение в d, иначе устанавливаем его в 1.

Подсчет прямоугольников

После того как матрица d заполнена, мы начинаем считывать высоту каждого столбца (начиная с нижней строки). Для каждого символа мы находим максимальную высоту, начиная с нижней строки и проводя подсчет, количества символов. Если символы отличаются, мы обнуляем счетчик. Если символ того же типа, то увеличиваем счетчик и добавляем его в общий результат.

Получение результата

После того как мы обработали всю матрицу, результатом будет общее количество найденных прямоугольников, которое мы возвращаем.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Вспомогательная матрица для хранения высот символов
vector<vector<int>> d;

// Функция для подсчета прямоугольников для символа c, позиция (posx, posy)
int F(int n, int m, vector<string> &v, char c, int posx, int posy) {
    d.resize(n, vector<int>(m, 0)); // Инициализация матрицы высот

    v[posx][posy] = c; // Установка буквы в пустую ячейку
```

```
    // Заполнение матрицы высот
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (i == 0) {
                d[i][j] = 1; // Первая строка всегда имеет высоту 1
            } else {
                d[i][j] = (v[i - 1][j] == v[i][j]) ? d[i - 1][j] + 1 : 1;
              // Высота для текущей ячейки
            }
        }
    }

    // Подсчет прямоугольников
    int res = 0;
    for (int i = 0; i < n; i++) {
        // Проход по строкам
        for (int j = i; j >= 0; j--) {
            // Проход по подстрокам для каждой строки
            int t = 0; // Количество текущих прямоугольников
            if (d[i][0] >= i - j + 1) { // Проверка первой колонки
                t = 1; // Находим хотя бы один прямоугольник
            }
            res += t; // Добавляем в общий счетчик

            for (int k = 1; k < m; k++) { // Проход по столбцам
                if (d[i][k] < i - j + 1) {
                    t = 0;
                    // Сбрасываем счетчик, если высота меньше текущей
                    continue;
                }
                if (v[j][k] == v[j][k - 1]) {
                   // Если текущий символ такой же, как предыдущий
                    t++; // Увеличиваем количество прямоугольников
                } else {
                    t = 1; // Обнуляем счетчик
                }
                res += t; // Добавляем к общему результату
            }
        }
    }

    return res; // Возвращаем общее количество прямоугольников
}

signed main() {
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    // Оптимизация ввода-вывода

    int n, m;
    cin >> n >> m; // Чтение размеров массива
    int posx, posy;
    vector<string> v(n);

    // Чтение поля и нахождение позиции пустой ячейки
    for (int i = 0; i < n; i++) {
        cin >> v[i];
        for (int j = 0; j < m; j++) {
            if (v[i][j] == '.') {
                posx = i; // Запоминаем координаты пустой ячейки
                posy = j;
            }
```

```
        }
    }

    bool ok = false; // Флаг для отслеживания возможности выигрыша
    for (char c = 'a'; c <= 'e'; c++) { // Перебор букв от 'a' до 'e'
        int z = F(n, m, v, c, posx, posy);
// Подсчет прямоугольников для каждой буквы

        if (z % 2 == 0) { // Если количество четное
            ok = true; // Вениамин может выиграть
            cout << c; // Вывод выигрывающей буквы
        }
    }

    if (!ok) { // Если ни одна буква не подходит
        cout << "AI win"; // ИИ выигрывает
    }
}
```

Реализация на Python

```
def count_rectangles(n, m, grid, char, posx, posy):
    # Создаем массив d для хранения высот
    d = [[0] * m for _ in range(n)]

    # Заполняем пустую позицию выбранным символом
    grid[posx][posy] = char

    # Вычисляем высоты столбцов для каждого символа
    for i in range(n):
        for j in range(m):
            if i == 0:
                d[i][j] = 1
            else:
                if grid[i - 1][j] == grid[i][j]:
                    d[i][j] = d[i - 1][j] + 1
                else:
                    d[i][j] = 1

    # Подсчет прямоугольников
    res = 0

    for i in range(n):
        for j in range(i, -1, -1):
            t = 0
            if d[i][0] >= i - j + 1:
                t = 1

            res += t

            for k in range(1, m):
                if d[i][k] < i - j + 1:
                    t = 0
                    continue

                if grid[j][k] == grid[j][k - 1]:
                    t += 1
                    res += t
                    continue

                t = 1
```

```
                res += t

    return res

def main():
    # Чтение входных данных
    n, m = map(int, input().split())
    grid = [input().strip() for _ in range(n)]

    # Нахождение позиции пустой клетки
    posx, posy = -1, -1
    for i in range(n):
        for j in range(m):
            if grid[i][j] == '.':
                posx, posy = i, j
                break
        if posx != -1:
            break

    winning_chars = []

    # Проверка символов от 'a' до 'e'
    for char in 'abcde':
        rectangle_count = count_rectangles(n, m, \
          [list(row) for row in grid], char, posx, posy)
        if rectangle_count % 2 == 0:  # Проверка четности
            winning_chars.append(char)

    # Вывод результата
    if winning_chars:
        print(''.join(winning_chars))
    else:
        print("AI win")

if __name__ == "__main__":
    main()
```

Основной шаг алгоритма включает обход всех ячеек матрицы, чтобы определить высоту каждого столбца для каждого символа. Вы должны пройти по всем элементам матрицы и обновить вспомогательную матрицу d. Этот процесс также занимает O(m×n) времени, так как каждый элемент матрицы посещается один раз. После того, как высоты столбцов определены, необходимо подсчитать количество возможных прямоугольников. Для этого вам нужно будет обработать каждую строку матрицы, а затем для каждой строки пройтись по ее высотам, чтобы вычислить возможные прямоугольники, основываясь на текущих высотах. Общее время на этом этапе также останется в пределах O(m×n).

Таким образом, учитывая все шаги алгоритма, общая временная сложность будет составлять O(m×n), где m — количество строк, а n — количество столбцов. Динамическое программирование является ключом к оптимизации решений для задач, связанных с подсчетом и комбинаторикой, что позволяет избежать избыточного перебора всех возможных комбинаций.

В решении используется вспомогательная матрица d для хранения высоты столбцов для каждого символа. Если исходная матрица имеет размер m×n, то для хранения этой вспомогательной матрицы нам

потребуется O(m×n) памяти. Это означает, что по памяти сложность равна также O(m×n). В дополнение к матрице d, могут потребоваться несколько дополнительных переменных для хранения промежуточных результатов, таких как счетчики или индексы. Однако количество этих переменных не зависит от размера входных данных и является константным, что не влияет существенно на общую сложность по памяти.

Результат выполнения алгоритма для указанных исходных данных

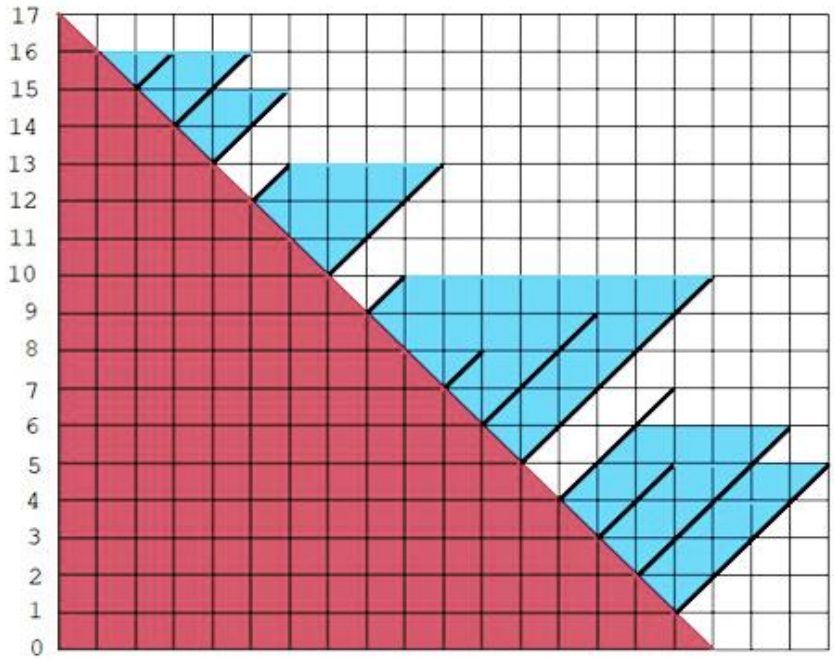
стандартный ввод	стандартный вывод
3 3 aaa cbb cc.	b

Количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами, при подстановке вместо точки для каждой буквы: a – 15, b – 16, c – 17, d – 15, e – 15.

Задача 6 (25 баллов)

Противолавинные заграждения

На одном горном курорте на склоне построили систему противолавинных заграждений. Склон имеет угол 45° к горизонтали. На склоне в некоторых целых по высоте от подножия горы точках расположены пояса заграждений. Каждый пояс расположен строго перпендикулярно склону. Рассмотрим двумерное сечение этого сооружения (на рисунке).



Склон представлен в виде темного треугольника, а каждое заграждение – в виде отрезка, перпендикулярного склону. Снег, скатываясь сверху, попадает в некоторые отсеки, образуемые заграждениями, и застревает в них, постепенно заполняя эти отсеки. Так как снега очень много, каждый отсек заполняется до того момента, пока снег не забьет его до самого верха (образуется горизонтальный уровень заполнения). После этого снег начинает высыпаться из отсека и попадать в нижние отсеки. Некоторые заграждения настолько велики, что, при их заполнении оказываются заполненными и некоторые более вышерасположенные отсеки, у которых граница расположена ниже горизонтали заполнения. В тоже время в некоторые заграждения снег наоборот может не попасть из-за того, что при переполнении верхнего отсека, он вертикально падает вниз, минуя текущий отсек, так как вертикаль падения проходит мимо него.

Будем считать, что если граница заграждения находится ровно на вертикали падения либо горизонтали заполнения других отсеков, то из них снег в этот отсек не попадает. Например, заграждение на высоте 12 из примера пусто, так как его окончание и по вертикали и по горизонтали

находится на соответствующих прямых. В отсек, образуемый заграждением на высоте 3, снег из верхнего отсека непосредственно не попадает, но так как его высота меньше горизонтали заполнения отсека заграждения на высоте 2, то снег его всё-таки заполнит.

По заданным высотам и длинам заграждений требуется оценить максимальную задерживающую способность этих заграждений. Так как мы работаем с двумерной моделью, то требуется найти площадь сечения, заполненную снегом при полном заполнении всей системы заграждений.

Формат входных данных

В первой строке находится число **n** – количество заграждений на склоне $\mathbf{1 \le n \le 3*10^5}$.

В следующих **n** строках находятся по два числа $\mathbf{h_i}$ и $\mathbf{t_i}$ – высота расположения **i**-го заграждения и его длина. Длина любого заграждения равна целому числу диагоналей единичного квадратика со стороной 1 метр. В этих диагоналях она и указывается $\mathbf{0 \le h_i \le 2 * 10^9}$, $\mathbf{1 \le t_i \le 10^6}$. Заграждения перечислены во входных данных в порядке возрастания высоты их нахождения на склоне.

Формат выходных данных

Вывести одно число – максимальную площадь сечения, занятую снегом при полном заполнении всех отсеков, в которые снег может попасть. Можно показать, что она всегда будет целым числом. Толщиной заграждений можно пренебречь. Считаем, что высота склона достаточна, чтобы любое заграждение наполнилось.

Примеры

стандартный ввод	стандартный вывод
13 1 4 2 4 3 2 4 3 5 5 6 3 7 1 9 1 10 3 12 1 13 2 14 2 15 1	58

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения

программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
5 3 3 4 3 8 7 11 3 13 3	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Для начала заметим, что если отсек высотой h ничем слева не ограничен, то площадь его сечения равна h^2. Далее, если отсек высоты h ограничен другим отсеком слева, то нужно провести горизонталь от отсека высоты h на этот ограничивающий и узнать высоту точки пересечения. Допустим, высота этой точки равна z. Тогда площадь заполнения нашего отсека будет равна $h^2 – z^2$. Это доказывает целочисленность ответа.

Основной операцией для этой задачи рассмотрим вертикальное и горизонтальное смещение заграждения. Например, вертикальное смещение: берем наше заграждение, фиксируем вертикальные линии сетки, которые его ограничивают и двигаем это заграждение по этим линиям вниз. Очевидно, в каждой целой точке склона это заграждение будут иметь все меньшую высоту, равную t.len - (t.h - v[i].h), (где t.len – исходная высота заграждения, t.h - v[i].h – смещение) пока не уйдет со склона в минус. Аналогично горизонтальное движение влево.

Теперь эффективно найдем те отсеки, в которые снег попадает (или не попадает) сверху. Добавим фиктивный отсек на высоте $2 * 10^9$ длиной 0, объявим его текущим. Далее перебираем заграждения в обратном порядке и смещаем текущее заграждение вертикально так, чтобы оно стало на диагональ рассматриваемого. Если текущее стало строго меньше рассматриваемого, то рассматриваемое становится текущим, в него снег попадает (отмечаем это 1 в соответствующей ячейке массива from_up), те отсеки, в которые снег сверху не попадает, остаются отмеченными 0.

Далее найдем ответ. Для этого добавим снизу фиктивный отсек на высоте -1 длиной 0, объявим его текущим. Перебираем заграждения снизу вверх (в порядке возрастания высоты крепления) и горизонтально сдвигаем текущее в точку рассматриваемого. Если текущее стало меньше либо равно 0, это означает, что его ничто не ограничивает слева и если в текущее попадает снег сверху, то добавим квадрат его высоты к ответу.

Если текущее больше 0, но меньше либо равно рассматриваемому, то (если в текущее попадает снег), добавим к ответу разницу соответствующих квадратов, текущим сделаем рассматриваемое.

Если текущее строго больше рассматриваемого в точке, то рассматриваемое никак не влияет на заполнение и будет заполнено снегом, если в текущем этот снег есть. Такой случай обрабатывать никак не нужно.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Структура, представляющая заграждение
struct bord {
    int h;    // высота заграждения
    int len;  // длина заграждения
};

signed main() {
    // Оптимизация ввода-вывода
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n;
    cin >> n; // Считываем количество заграждений
    vector<bord> v(n + 1); // Вектор для хранения заграждений
    v[0] = {-1, 0}; // Фиктивное заграждение с высотой -1 и длиной 0
    for (int i = 1; i <= n; i++) {
        cin >> v[i].h >> v[i].len; // Считываем высоту и длину заграждений
    }
    v.push_back({(int)2e9, 0}); // Фиктивное заграждение

    bord t = v.back(); // Последнее заграждение
    int last; // Поддерживаем индекс последнего заграждения
    vector<int> from_up(n + 1, 0); // Массив о попадании снега
    // Идем по заграждениям снизу вверх, определяя может ли быть заполнено
    for (int i = n; i >= 1; i--) {
        int tlen = t.len - (t.h - v[i].h);
        if (tlen < v[i].len) {
            from_up[i] = 1; // Текущее заграждение может быть заполнено
            t = v[i]; // Обновляем текущее заграждение
        }
    }
    t = v[0]; // Сбрасываем текущее заграждение на первое
    last = 0; // Обнуляем индекс последнего заграждения

    int ans = 0; // Переменная для хранения площади, занятой снегом
    //Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
     for (int i = 1; i <= n + 1; i++) {
        int tlen = t.len - (v[i].h - t.h);
```

```
        if (tlen <= 0) {
            if (from_up[last]) {
                ans += v[last].len * v[last].len;
            }
            t = v[i];
            last = i;
            continue;
        }
        if (tlen <= v[i].len) {
            if (from_up[last]) {
                ans += (v[last].len * v[last].len - tlen * tlen);
            }
            t = v[i];
            last = i;
        }
    }
    cout << ans << endl; // Выводим результирующую площадь, занятую снегом
}
```

Реализация на Python

```
class Bord:
    def __init__(self, h, length):
        self.h = h
        self.len = length

def main():
    n = int(input().split())
    v = [Bord(-1, 0)]  # Добавляем фиктивный отсек на высоте -1
    for i in range(1, n + 1):
        h, t = map(int, input().split())
        v.append(Bord(h, t))
    v.append(Bord(2 * 10**9, 0))  # Добавляем отсек на высоте 2 * 10^9
    from_up = [0] * (n + 1) # Массив о попадании снега сверху
    t = v[-1]
# Идем по заграждениям снизу вверх
    for i in range(n, 0, -1):
        tlen = t.len - (t.h - v[i].h)
        if tlen < v[i].len:
            from_up[i] = 1
            t = v[i]

    t = v[0]
    last = 0
    ans = 0
    # Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
    for i in range(1, n + 2):  # +2, из-за фиктивного отсека
        tlen = t.len - (v[i].h - t.h)
        if tlen <= 0:
            if from_up[last]:
                ans += v[last].len * v[last].len
            t = v[i]
            last = i
            continue

        if tlen <= v[i].len:
            if from_up[last]:
                ans += (v[last].len * v[last].len - tlen * tlen)
            t = v[i]
            last = i
```

```
    print(ans)

if __name__ == "__main__":
    main()
```

Программа в основном состоит из нескольких проходов по массиву v, который содержит заграждения. Первый проход — мы считываем данные из файла и заполняем массив v объектами класса Bord. Этот проход происходит в одном цикле от 1 до n, где n — количество заграждений. Таким образом, временная сложность этого шага составляет O(n). Второй проход — во время этого прохода мы идем в обратном направлении по массиву v, чтобы определить, может ли снег попадать сверху. Это также O(n), поскольку мы проходим по всем элементам массива по одному разу. Третий проход — здесь мы снова проходим по массиву v от начала до конца, анализируя пространство между заграждениями и высоту заполнения. Это также O(n).

Таким образом, суммируя все проходы, мы получаем, что общая временная сложность программы составляет O(n).

Мы создаем массив v, длиной n+2, который хранит объекты класса Bord. Таким образом, память, занимаемая этим массивом, составляет O(n). Массив from_up также имеет размер n+1, что добавляет еще O(n) к пространственной сложности. Другие переменные, такие как t, last, и ans, занимают постоянное количество памяти, которое не зависит от размера входных данных. Таким образом, их вклад в общую пространственную сложность можно считать константным: O(1).

Итак, собрав все элементы вместе, можно заключить, что общая пространственная сложность программы также составляет O(n).

Результат выполнения алгоритма для указанных исходных данных

стандартный ввод	стандартный вывод
5 3 3 4 3 8 7 11 3 13 3	**59**

ВАРИАНТ 4

Задача 1 (10 баллов)

Нашими разведчиками на командный пункт были переданы места сосредоточения войск противника в виде горизонтальных и вертикальных координат, расположенных в ячейках B2:F2 и A3:A7. Определите их по формулам закодированной электронной таблицы, представленной в двух режимах, на рисунках расположенных ниже.

1	=СУММ(B2:F2)/(A4-A6)		a=	=ABS(ОКРУГЛВНИЗ(-ПИ();0))
2	=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))		b=	=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)
3	=A3*(A4+A6)		c=	=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))
4	=A3-A4-A6		d=	=ЕСЛИ(ЕОШ(B9);E9*E10;E10-E9)
5	=ЕСЛИ(СУММ(A3:A7)>45;A3+A4;A6+A5)		e=	=B10-E9
6	=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))		f=	=B10^2+E15
7	=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10		g=	1
8	=НАЙТИ(C2;"ABCDEFGHIJKLMNOPQ")-E11+E12			
9	=НАЙТИ(D2;"ABCDEFGHIJKLMNOPQ")-E13+E14			
10	=НАЙТИ(E2;"ABCDEFGHIJKLMNOPQ")-E14+E15			
11	=НАЙТИ(F2;"ABCDEFGHIJKLMNOPQ")			

1	#ДЕЛ/0!		a=	
2	5		b=	
3	4		c=	
4	-3		d=	
5	10		e=	
6	a		f=	
7	5		g=	
8	12			
9	25			
10	-8			
11	11			

Ответ оформите в виде двух строк, в верхней строке запишите через запятую значения ячеек B2:F2, а в нижней строке также через запятую значения ячеек A3:A7. В случае нескольких вариантов ответов запишите их через союз «или».

Решение

Начнем с определения коэффициентов a, b, c, d, e, f и g.

Значение коэффициента a (ячейка E9) в соответствии с формулой «=ABS(ОКРУГЛВНИЗ(-ПИ();0))» равно 3.

Значение коэффициента b (ячейка E10) в соответствии с формулой «=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)» равно 4.

Значение коэффициента c (ячейка E11) в соответствии с формулой «=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))» равно 2.

Значение коэффициента d (ячейка E12) в соответствии с формулой «=ЕСЛИ(ЕОШ(B9);E9*E10;E10-E9)» равно 12.

Значение коэффициента e (ячейка E13) в соответствии с формулой «=B10-E9» равно 2.

Значение коэффициента f (ячейка E14) в соответствии с формулой «=B10^2+E15» равно 26.

Значение коэффициента g (ячейка E15) равно 1.

Получаем:

	D	E
8		
9	a=	3
10	b=	4
11	c=	2
12	d=	12
13	e=	2
14	f=	26
15	g=	1

1) Так как значение формулы в ячейке B9 «=СУММ(B2:F2)/(A4-A6)» определяется как ошибка деления на 0, то значения ячеек A4 и A6 одинаковы.

2) Значение формулы в ячейке B10 «=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))» равно 5, следовательно значения координат в ячейках A3:A7 состоят из одного символа.

3) Формулы в ячейках B11 «=A3*(A4+A6)» и B12 «=A3-A4-A6» рассмотрим, как систему уравнений, учитывая, что значения ячеек A4 и A6 одинаковы.

$$\begin{cases} A3 * (A4 + A6) = 4 \\ A3 - A4 - A6 = -3 \end{cases} => \begin{cases} A3 * 2A4 = 4 \\ A3 - 2A4 = -3 \end{cases}$$

Решая систему уравнений, получаем два набора значений для ячеек A3 и A4 (A3 = 1 и A4 = 2) либо (A3 = -4 и A4 = -0.5). Так как, в соответствии со вторым пунктом, значения ячеек A3:A7 состоят из одного символа, а это диапазон от 0 до 9, то A3 = 1, A4 = A6 = 2.

4) Значение формулы в ячейке B13 «=ЕСЛИ(СУММ(A3:A7)>45; A3+A4;A6+A5)» равно 10. Максимальное значение формулы СУММ(A3:A7) из-за того, что значения в этих ячейках не превышают 9, равно 45. В соответствие с этим, условие в формуле ячейки B13 не выполнится, следовательно A6 + A5 = 10 и A5 = 10 - 2 = 8.

5) Значение формулы в ячейке B14 «=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))» равно «а». Следовательно разница между ячейками A7 и A5 равна 1, откуда A7 равно 7 или 9.

6) Значение формулы в ячейке B15 «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10» равно 5. Откуда, значение формулы «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")» = 5 + E9 – E10 = 5 + 3 - 4 = 4, следовательно B2 = «D».

7) Значения ячеек C2, D2, E2 и F2 аналогичным образом находим из формул в ячейках B16, B17, B18 и B19. Получаем C2 = «B», D2 = «A», E2 = «Q» и F2 = «K».

Ответ: **D, B, A, Q, K**
1, 2, 8, 2, 7 или 9

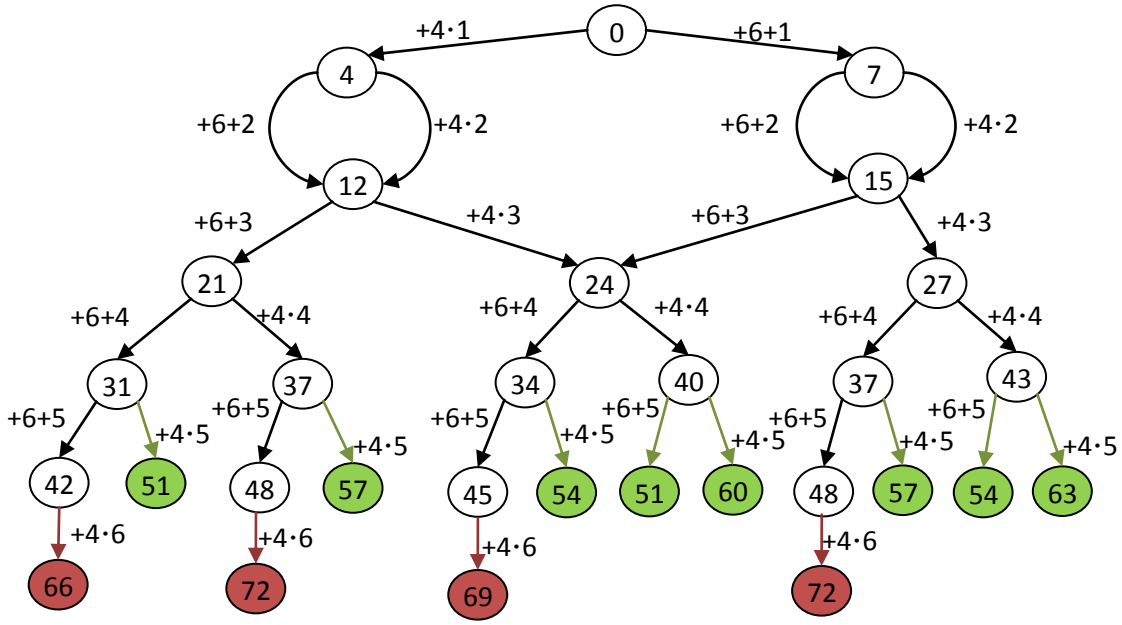
Задача 2 (10 баллов)

Сокровища пиратов: дуэль за 50 монет

Два пирата захотели закопать клад. Для этого они должны сначала заполнить сундук золотыми монетами, а чтобы дело шло веселее, они решили сыграть при этом в следующую игру. По очереди они складывают монеты в сундук. За один ход можно положить в сундук либо шесть монет плюс номер хода в игре, либо четыре умноженные на номер хода в игре. При превышении количества 50 монет в сундуке, игра заканчивается. Кто первым, сделав ход, получит в сундуке 50 монет или больше – считается победителем. Кто выигрывает при безошибочной игре – пират, делающий первый ход, или пират, делающий второй ход? Каково минимальное число ходов до победы, если выигрывающий пират пытается закончить игру за меньшее число ходов, а проигрывающий пытается максимально задержать свое поражение? Какое максимальное число монет может оказаться в сундуке в конце игры? Какой пират при этом победит? Поясните алгоритм графически, используя ориентированные графы в виде деревьев, с обозначением выигрышных и проигрышных позиций как вершин, а действий игроков – дугами графа с указанием численных изменений.

Решение

Построим ориентированный граф, где зеленым цветом выделены выигрышные позиции первого пирата, красным цветом – выигрышные позиции второго пирата, а номер соответствует количеству монет в сундуке. Рядом со стрелкой указано изменение числа монет в сундуке на этом ходу. В случае, если один из вариантов хода приводит к победе, то второй вариант не отображается, за исключением ветвей, которые демонстрирует игру с максимально возможным числом монет в конце игры.



Из графа видно, что при любой игре второго пирата, гарантированно победит первый пират на 5 ходу. Максимально возможное число монет можно получить, если увеличивать количество монет в сундуке таким образом, чтобы получить как можно более высокий номер хода (6) и количества монет в сундуке (48) перед последним ходом, в котором берется вариант с большим значением (+4·6). При этом побеждает второй пират с количеством монет 72.

Ответ: гарантированно победит первый пират за 5 ходов, максимум 72 монеты при победе второго пирата.

Задача 3 (10 баллов)

Логическая головоломка: разгадка функции F

Паша решил усовершенствовать задание ЕГЭ по информатике.

Логическая функция задается выражением:

$$\neg(y \to x) \wedge (z \equiv w) \vee (z \to y)$$

Ниже приведен фрагмент таблицы истинности функции F, содержащий неповторяющиеся строки. Однако вместо значений, Паша решил вписать тоже несколько логических выражений, результат которых надо определить, если **a, b, c** – следующие высказывания:

a – «2024 - високосный год»

b – «Монблан - высочайшая горная вершина на Земле»

c – «Самый большой океан на Земле – Индийский»

Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных **w,x,y,z.**

w	?	?	?	F
		$b \oplus c \to a$		0
		$c \wedge a \to b$	$a \to \neg b \wedge c$	0
			$c \wedge a \wedge b$	0
$a \vee b \wedge \neg c$	$\neg b \oplus a \wedge c$	$\neg b \equiv c \vee a$		0

В ответе напишите буквы **w,x,y,z** в том порядке, в котором идут соответствующие им столбцы.

Решение

Сперва определим значения логических высказываний. Очевидно, что

a=1
b=0
c=0

Теперь определим значения логических выражений в таблице. В результате получим следующую таблицу:

w	**?**	**?**	**?**	**F**
		1		0
		1	0	0
			0	0
1	1	1		0

Следующим шагом будет составление таблицы истинности для функции F

w	x	y	z	y→x	¬(y→x)	z≡w	¬(y→x)∧(z≡w)	z→y	F
0	0	0	0	1	0	1	0	1	1
0	0	0	1	1	0	0	0	0	0
0	0	1	0	0	1	1	1	1	1
0	0	1	1	0	1	0	0	1	1
0	1	0	0	1	0	1	0	1	1
0	1	0	1	1	0	0	0	0	0
0	1	1	0	1	0	1	0	1	1
0	1	1	1	1	0	0	0	1	1
1	0	0	0	1	0	0	0	1	1
1	0	0	1	1	0	1	0	0	0
1	0	1	0	0	1	0	0	1	1
1	0	1	1	0	1	1	1	1	1
1	1	0	0	1	0	0	0	1	1
1	1	0	1	1	0	1	0	0	0
1	1	1	0	1	0	0	0	1	1
1	1	1	1	1	0	1	0	1	1

Выберем из нее строки, где функция F = 0.

w	x	y	z	F
0	0	0	1	0
0	1	0	1	0
1	0	0	1	0
1	1	0	1	0

Сравнивая эту таблицу с заданной в задании, и учитывая, что строки в этой таблице выписаны в случайном порядке, легко установить с помощью несложных рассуждений искомые наименования столбцов

w	x	z	y	F
		1	0	0
		1	0	0
		1	0	0
1	1	1	0	0

Ответ: w,x,z,y

Задача 4 (20 баллов)

Зашифрованный Рудник

В горнодобывающей компании «Рудник-Х» используется система шифрования для передачи информации о наименовании полезного ископаемого и параметрах пласта с данным ценным компонентом (длина, ширина), которая была получена при проведении геологической разведки месторождений полезных ископаемых. Передача информации осуществляется геологами с места проведения полевых работ в административное управление компании.

Процесс шифрования в горнодобывающей компании заключается в следующем:

1) Генерируется супервозрастающая последовательность чисел $A = \{a_1, a_2, \dots a_k\}$, каждый член которой больше суммы всех предыдущих членов, т.е. при $j = 2,3, \dots k: a_j > \sum_{i=1}^{j-1} a_i$, которая является секретным ключом.

Например, секретный ключ: $A = \{a_1, a_2, a_3, a_4\} = \{2, 3, 9, 17\}$,
где $a_2 > a_1 \rightarrow 3 > 2$;
$a_3 > (a_1 + a_2) \rightarrow 9 > 5$;
$a_4 > (a_1 + a_2 + a_3) \rightarrow 17 > 14$.

2) После этого выбирается число m большее, чем сумма чисел последовательности, т.е. $m > \sum_{i=1}^{k} a_i$. Также выбирается число n взаимно простое с m, т.е. наибольший общий делитель m и n равен 1.

3) Далее создается открытый ключ, т.е. новая последовательность чисел $B = \{b_1, b_2, \dots b_k\}$ путем умножения каждого числа a_i на n и взятия остатка от деления данного произведения на m.

Например, для открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ каждый член последовательности вычисляется по формуле $b_i = (a_i \cdot n) \bmod m$,
где a_i — i-й элемент последовательности секретного ключа A;
n — число взаимно простое с m;
m — модуль операции взятия остатка, иными словами, делитель;
mod — операция взятия остатка от деления.
При $n = 91$, $m = 320$ и $A = \{2, 3, 9, 17\}$:
$b_1 = (a_1 \cdot n)\ mod\ m = (2 \cdot 91)\ mod\ 320 = 182$ или, иными словами, $(2 \cdot 91)$ делится на 320 с остатком 182.
Сделав аналогичные вычисления для остальных элементов, получается последовательность $B = \{182,\ 273, 179, 267\}$

4) Полученная последовательность B используется для шифрования отправляемого сообщения. Отправляемое сообщение, представленное в виде двоичной строки, определяет, какие элементы этой последовательности суммируются для получения последовательности шифротекста $C = \{c_1, c_2, \dots c_p\}$, которая передается получателю.

Например, при шифровании числа 10 последовательность шифротекста C будет вычисляться следующим образом:

1) Число 10 представляется в двоичной системе счисления, т.е. $10 = 1010_2$;

2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1 и b_3 суммируются для получения шифротекста, b_2 и b_4 – опускаются.

3) Шифротекст $C = \{c_1\} = \{b_1 + b_3\} = 182 + 179 = 361$.

Например, при шифровании числа E_{16} последовательность шифротекста C будет вычисляться следующим образом:

1) Число E_{16} представляется в двоичной системе счисления, т.е. $E_{16} = 1110_2$;

2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1, b_2 и b_3 суммируются для получения шифротекста, b_4 – опускается.

3) Шифротекст $C = \{c_1\} = \{b_1 + b_2 + b_3\} = 182 + 273 + 179 = 634$.

Для дешифрирования сообщения получатель, исходя из известного ему n, определяет n^{-1} – мультипликативное обратное к значению n по модулю m (остаток от деления $n \cdot n^{-1}$ на m равен 1).

Иными словами, $(n \cdot n^{-1}) \bmod m = 1$,

где n — заданное число взаимно простое с m;

m — модуль операции взятия остатка;

mod — операция взятия остатка от деления.

Далее вычисляются промежуточные значения последовательности C' путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

Например, для шифротекста $C = \{c_1\} = 361$, каждый член последовательности C' вычисляется по формуле $c'_i = (c_i \cdot n^{-1}) \bmod m$,

где c_i — i-й элемент последовательности шифротекста C;

n^{-1} — мультипликативное обратное;

m — модуль операции взятия остатка;

mod — операция взятия остатка от деления.

В данном случае, последовательность

$C' = \{c'_1\} = (361 \cdot 211) \bmod 320 = 11$, где 211 – мультипликативное обратное n^{-1}, рассчитанное получателем.

Далее получатель определяет двоичную строку M путем последовательного сравнения каждого элемента последовательности C' с элементами секретного ключа A справа налево: если $c'_i \geq a_i$, то $M_i = 1$ и осуществляется вычитание a_i из c'_i, в противном случае $M_i = 0$. Таким образом восстанавливается зашифрованное сообщение.

Например, для последовательности $C' = \{c'_1\}$ и секретного ключа $A = \{a_1, a_2, a_3, a_4\}$ осуществляется последовательное сравнение элементов справа налево:

1) Если $c'_1 \geq a_4$, то $M_4 = 1$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число $c'_1 - a_4$;

2) Если $c'_1 < a_4$, то $M_4 = 0$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число c'_1.

3) В ответе получается двоичная строка $M = \{M_1, M_2, M_3, M_4\}$.

Например, при $A = \{2, 3, 9, 17\}$ и $C' = \{11\}$ двоичная строка $M = \{1, 0, 1, 0\}$, что согласуется с двоичным представлением числа 10.

Для последовательности C', состоящей из нескольких элементов, например, $C' = \{c'_1, c'_2, \dots c'_k\}$ осуществляется последовательное сравнение каждого элемента $c'_1, c'_2, \dots c'_k$ с элементами секретного ключа A с получением количества двоичных строк, равного количеству элементов последовательности C'.

1) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4\}$ количество двоичных строк M равно 4.

2) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7, c'_8\}$ количество двоичных строк M равно 8.

Геологи передали четыре последовательных сообщения в административное управление о двух месторождениях полезных ископаемых. Первое сообщение в каждой паре содержало наименование полезного ископаемого. Второе сообщение содержало информацию о параметрах пласта, при этом первое число соответствовало длине пласта, второе – ширине пласта. Для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы согласно таблице ASCII для Windows-1251, далее полученному шестнадцатеричному числовому значению присваивалась соответствующая двоичная тетрада. Для шифрования числа осуществлялся его перевод в двоичную систему. Таблица ASCII для Windows-1251:

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Были получены следующие зашифрованные сообщения:

1 сообщение: 596 739 1537 739 1154 893 1122

2 сообщение: 628 1088

3 сообщение: 1009 923 739 1537

4 сообщение: 803 1043

Секретный ключ, хранящийся в организации, представляет собой следующую последовательность {1, 4, 6, 14, 26, 53, 107, 215}, значения n и m равны 107 и 438 соответственно.

Расшифруйте полученные сообщения. В ответе укажите последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» – «Длина», «Ширина» (в десятичной системе). Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Решение

Сначала необходимо вычислить промежуточные значения последовательности C' для осуществления дешифровки. В исходных данных сказано, что данные значения вычисляются путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

В условии дано только значение n, однако мы знаем, что n^{-1} – мультипликативное обратное к значению n по модулю m, то есть остаток от деления $n \times n^{-1}$ на m равен 1, поэтому для нахождения значения n^{-1} можно применить расширенный алгоритм Евклида.

Расширенный алгоритм Евклида является модификацией алгоритма Евклида, вычисляющая кроме наибольшего общего делителя (НОД) целых чисел a и b, еще и коэффициенты соотношения Безу, то есть такие x и y, что $ax + by = \text{НОД}(a, b)$. В свою очередь, алгоритм Евклида заключается в последовательном делении с остатком для нахождения наибольшего общего делителя двух чисел. На каждом этапе производится деление большего из пары чисел на меньшее, а остаток записывается и используется при дальнейших вычислениях в качестве нового делителя для предыдущего. Вычисление остатка производят до тех пор, пока он не будет равен нулю. Тогда последний использованный делитель и будет являться наибольшим общим делителем, полученным через алгоритм Евклида. В нашем случае наибольший общий делитель чисел n и m (107 и 438) должен равняться 1.

Применяем алгоритм Евклида:

$$438 = 107 \times 4 + 10 \Longrightarrow \quad 107 = 10 \times 10 + 7 \Longrightarrow \quad 10 = 7 \times 1 + 3 \Longrightarrow \quad 7 = 3 \times 2 + 1$$
$$2 = 1 \times 2 + 0$$

Применяем расширенный алгоритм Евклида. Нам нужно выразить 1 как линейную комбинацию 107 и 438, т. е. найти целые числа x и y, такие, что $107x + 438y = 1$

Из алгоритма Евклида выше:

$$1 = 7 - 3 \times 2 \Leftarrow \quad 3 = 10 - 7 \times 1 \Leftarrow \quad 7 = 107 - 10 \times 10 \Leftarrow \quad 10 = 438 - 107 \times 4$$

Подставим обратно:

$$1 = 7 - (10 - 7 \times 1) \times 2 = 7 \times 3 - 10 \times 2$$
$$1 = (107 - 10 \times 10) \times 3 - 10 \times 2 = 107 \times 3 - 10 \times 32$$
$$1 = 107 \times 3 - (438 - 107 \times 4) \times 32 = 107 \times 131 - 438 \times 32$$
$$1 = 107 \times 131 - 438 \times 32$$

Мы получили, что $1 = 107 \times 131 - 438 \times 32$ Это означает, что 131 является обратным числом 107 по модулю 438. Получаем, что $n^{-1} = 131$.

Зная значение n^{-1} находим промежуточные значения последовательности C' для каждого полученного сообщения. Для первого сообщения – 596 739 1537 739 1154 893 1122:

1) (596×131) делится на 438 с остатком 112

2) (739×131) делится на 438 с остатком 11

3) (1537×131) делится на 438 с остатком 305

4) (739×131) делится на 438 с остатком 11

5) (1154×131) делится на 438 с остатком 64

6) (893×131) делится на 438 с остатком 37

7) (1122×131) делится на 438 с остатком 252

Промежуточные значения последовательности $\{112, 11, 305, 11, 64, 37, 252\}$.

Для второго сообщения – 628 1088:

1) (628×131) делится на 438 с остатком 362

2) (1088×131) делится на 438 с остатком 178

Промежуточные значения последовательности $\{362, 178\}$.

Для третьего сообщения – 1009 923 739 1537:

1) (1009×131) делится на 438 с остатком 341

2) (923×131) делится на 438 с остатком 25

3) (739×131) делится на 438 с остатком 11

4) (1537×131) делится на 438 с остатком 305

Промежуточные значения последовательности $\{341, 25, 11, 305\}$.

Для четвертого сообщения – 803 1043:

1) (803×131) делится на 438 с остатком 73

2) (1043×131) делится на 438 с остатком 415

Промежуточные значения последовательности $\{73, 415\}$.

Далее используя секретный ключ A, получатель определяет битовую строку M путем последовательного сравнения C' с элементами a_i. Секретный ключ, исходя из исходных условий, представляет собой последовательность $\{1, 4, 6, 14, 26, 53, 107, 215\}$.

Для первого сообщения:

1) $112 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix}$

2) $11 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

3) $305 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{matrix}$

4) $11 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

5) $64 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{matrix}$

6) $37 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{matrix}$

7) $252 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{matrix}$

Для второго сообщения:

1) $362 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{matrix}$

2) $178 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{matrix}$

Для третьего сообщения:

1) $341 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{matrix}$

2) $25 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$

3) $11 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

4) $305 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{matrix}$

Для четвертого сообщения:

1) $73 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{matrix}$

2) $415 = \begin{matrix} 1 & 4 & 6 & 14 & 26 & 53 & 107 & 215 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{matrix}$

Из условий задачи известно, что для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы, далее каждой шестнадцатеричной цифре соответствующая двоичная тетрада; для шифрования числа осуществлялся его перевод в двоичную систему. Для дешифровки необходимо произвести данные действия в обратном порядке.

Сначала дешифруем текстовые сообщения. Для этого воспользуемся таблицей тетрад и далее таблицей ASCII для Windows-1251 из условий задачи.

16-ричная цифра	Двоичная тетрада	16-ричная цифра	Двоичная тетрада
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Таким образом, произведем дешифровку текстовых сообщений:

Первое сообщение:

1) $596 = 11000010 = \begin{matrix}1100 & \text{C}\\ 0010 & 2\end{matrix} = \text{В}$

2) $739 = 11100000 = \begin{matrix}1110 & \text{E}\\ 0000 & 0\end{matrix} = \text{а}$

3) $1537 = 11101101 = \begin{matrix}1110 & \text{E}\\ 1101 & \text{D}\end{matrix} = \text{н}$

4) $739 = 11100000 = \begin{matrix}1110 & \text{E}\\ 0000 & 0\end{matrix} = \text{а}$

5) $1154 = 11100100 = \begin{matrix}1110 & \text{E}\\ 0100 & 4\end{matrix} = \text{д}$

6) $893 = 11101000 = \begin{matrix}1110 & \text{E}\\ 1000 & 8\end{matrix} = \text{и}$

7) $1122 = 11101001 = \begin{matrix}1110 & \text{E}\\ 1001 & 9\end{matrix} = \text{й}$

Третье сообщение:

1) $1009 = 11010011 = \begin{matrix}1101 & \text{D}\\ 0011 & 3\end{matrix} = \text{У}$

2) $923 = 11110000 = \begin{matrix}1111 & \text{F}\\ 0000 & 0\end{matrix} = \text{р}$

3) $739 = 11100000 = \begin{matrix}1110 & \text{E}\\ 0000 & 0\end{matrix} = \text{а}$

4) $1537 = 11101101 = \begin{matrix}1110 & \text{E}\\ 1100 & \text{D}\end{matrix} = \text{н}$

Дешифруем числовые сообщения: для этого осуществим перевод найденной битовой последовательности в десятичную систему счисления. Чтобы перевести число из двоичной системы счисления в десятичную, нужно: справа налево пронумеровать цифры в двоичном числе, начиная с 0, далее слева направо умножить каждую цифру двоичного числа на 2 в степени номера этой цифры и сложить полученные значения.

Число в 2-ой системе счисления	Перевод в 10-ую систему счисления	Число в 10-ой системе счисления
Второе сообщение		
00011011_2	$1 \times 2^4 + 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0 = 16 + 8 + 2 + 1 = 27$	27_{10}
01010110_2	$1 \times 2^6 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 = 64 + 16 + 4 + 2 = 86$	86_{10}
Четвертое сообщение		
00110100_2	$1 \times 2^5 + 1 \times 2^4 + 1 \times 2^2 = 32 + 16 + 4 = 52$	52_{10}
00011111_2	$1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 16 + 8 + 4 + 2 + 1 = 31$	31_{10}

В ответе необходимо указать последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» – «Длина», «Ширина» (числа в десятичной системе). Следовательно, ответ задачи выглядит следующим образом: Ванадий – 27, 86, Уран – 52, 31.

Ответ: Ванадий – 27, 86, Уран – 52, 31

Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Python

```
def find_pairs(m):
    pairs = []
    for n in range(m):
        for nn in range(m):
            if (n * nn) % m == 1:
                pairs.append((n, nn))
    return pairs

m = int(input("Enter the value of m: "))
pairs = find_pairs(m)
for n, nn in pairs:
    print(f"({n}, {nn})")
```

Задача 5 (25 баллов)

Буквенные прямоугольники

Вениамин играет в новую интеллектуальную игру с ИИ. У них есть прямоугольная таблица размера **n** × **m**, в клетки которой они по очереди выбирают и ставят по одной из букв 'a', 'b', 'c', 'd' или 'e'. После того, как все ячейки будут заполнены, ИИ честно подсчитает количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами. Если оно чётно, выиграет Вениамин, иначе выиграет ИИ.

Осталось сделать последний ход, и этот ход делает Вениамин. Подскажите ему, какие из этих букв он может поставить, чтобы выиграть.

Формат входных данных

В первой строке содержатся два натуральных числа **n** и **m** через пробел – размеры игрового поля. **1 ≤ n, m ≤ 500.**

В следующих **n** строках содержится описание текущего заполнения таблицы. Каждая строка состоит из **m** символов, каждый из них – это буква от 'a' до 'e'. Помимо этого, в таблице содержится ровно один символ '.', который обозначает пустую клетку, в которую нужно сделать заключительный ход.

Формат выходных данных

Вывести в одну строку без пробелов список букв, которые может поставить в последнюю свободную клетку Вениамин и выиграть. Буквы не нужно разделять пробелом и требуется выводить в алфавитном порядке. Буквы должны быть из интервала от 'a' до 'e'. Если нет ни одной выигрывающей буквы, вывести сообщение «AI win» без кавычек.

Примеры

стандартный ввод	стандартный вывод
3 4 aadd a.bb eebd	bcd
1 1 .	AI win

Комментарий к примерам

Исходно, из букв 'a' можно составить 5 прямоугольников, из букв 'b' – 5 прямоугольников, из букв 'c' – ни одного, из букв 'd' – 4 прямоугольника и из букв 'e' – 3 прямоугольника.

Если поставить букву 'a' получим 9 прямоугольников из этой буквы и 12 остальных, то есть нечётное количество, выиграет ИИ.

Если поставить букву 'b' получим 8 прямоугольников из этой буквы и 12 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'c' получим 1 прямоугольник из этой буквы и 17 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'd' получим 5 прямоугольников из этой буквы и 13 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'e' получим 5 прямоугольников из этой буквы и 14 остальных, то есть нечётное количество, тогда выиграет ИИ.

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных. Результат выполнения программы для указанных исходных данных должен содержать: итоговый вывод программы и количество различных прямоугольников при подстановке каждой буквы вместо точки.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
3 3 eee d.b dbb	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Решение

Прежде чем перейти к решению, важно понимать структуру данных и подходы, которые можно использовать. У нас имеется двумерный массив (матрица) символов, и мы хотим подсчитать количество прямоугольных областей, состоящих из одного и того же символа. Важно отметить, что прямоугольник определяется как область, в которой все символы одинаковы и находятся на любых позициях в матрице, которые могут образовать прямоугольник.

Элементарное решение требует шестерного цикла: перебираем левый

верхний угол двумя циклами, внутри правый нижний еще двумя и далее перебираем всю внутреннюю часть выбранного прямоугольника чтобы проверить, что она состоит из одной и той же буквы. Получим правильное, но не эффективное решение.

Основная идея эффективного решения заключается в применении алгоритма, основанного на динамическом программировании. Вот шаги, которые мы будем выполнять:

Создание вспомогательной матрицы

Мы создаем переменную d, которая будет хранить высоту столбцов для каждого символа в матрице. Эта матрица позволяет нам отслеживать, сколько последовательных символов одного и того же типа находится над каждой позицией в матрице. Если текущий элемент такой же, как элемент над ним, мы увеличиваем счетчик.

Инициализация

Мы заполняем первую строку матрицы d единицами, поскольку в ней нет элементов, находящихся выше.

Обработка матрицы

Затем мы проходим по всей матрице, заполняя d в зависимости от значений над текущей ячейкой. Если символ текущей ячейки такой же, как символ над ней, мы увеличиваем значение в d, иначе устанавливаем его в 1.

Подсчет прямоугольников

После того как матрица d заполнена, мы начинаем считывать высоту каждого столбца (начиная с нижней строки). Для каждого символа мы находим максимальную высоту, начиная с нижней строки и проводя подсчет, количества символов. Если символы отличаются, мы обнуляем счетчик. Если символ того же типа, то увеличиваем счетчик и добавляем его в общий результат.

Получение результата

После того как мы обработали всю матрицу, результатом будет общее количество найденных прямоугольников, которое мы возвращаем.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Вспомогательная матрица для хранения высот символов
vector<vector<int>> d;

// Функция для подсчета прямоугольников для символа c, позиция (posx, posy)
int F(int n, int m, vector<string> &v, char c, int posx, int posy) {
    d.resize(n, vector<int>(m, 0)); // Инициализация матрицы высот

    v[posx][posy] = c; // Установка буквы в пустую ячейку
```

```
    // Заполнение матрицы высот
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (i == 0) {
                d[i][j] = 1; // Первая строка всегда имеет высоту 1
            } else {
                d[i][j] = (v[i - 1][j] == v[i][j]) ? d[i - 1][j] + 1 : 1;
              // Высота для текущей ячейки
            }
        }
    }

    // Подсчет прямоугольников
    int res = 0;
    for (int i = 0; i < n; i++) {
        // Проход по строкам
        for (int j = i; j >= 0; j--) {
            // Проход по подстрокам для каждой строки
            int t = 0; // Количество текущих прямоугольников
            if (d[i][0] >= i - j + 1) { // Проверка первой колонки
                t = 1; // Находим хотя бы один прямоугольник
            }
            res += t; // Добавляем в общий счетчик

            for (int k = 1; k < m; k++) { // Проход по столбцам
                if (d[i][k] < i - j + 1) {
                    t = 0;
                    // Сбрасываем счетчик, если высота меньше текущей
                    continue;
                }
                if (v[j][k] == v[j][k - 1]) {
                   // Если текущий символ такой же, как предыдущий
                    t++; // Увеличиваем количество прямоугольников
                } else {
                    t = 1; // Обнуляем счетчик
                }
                res += t; // Добавляем к общему результату
            }
        }
    }

    return res; // Возвращаем общее количество прямоугольников
}

signed main() {
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    // Оптимизация ввода-вывода

    int n, m;
    cin >> n >> m; // Чтение размеров массива
    int posx, posy;
    vector<string> v(n);

    // Чтение поля и нахождение позиции пустой ячейки
    for (int i = 0; i < n; i++) {
        cin >> v[i];
        for (int j = 0; j < m; j++) {
            if (v[i][j] == '.') {
                posx = i; // Запоминаем координаты пустой ячейки
                posy = j;
            }
```

```
        }
    }

    bool ok = false; // Флаг для отслеживания возможности выигрыша
    for (char c = 'a'; c <= 'e'; c++) { // Перебор букв от 'a' до 'e'
        int z = F(n, m, v, c, posx, posy);
// Подсчет прямоугольников для каждой буквы

        if (z % 2 == 0) { // Если количество четное
            ok = true; // Вениамин может выиграть
            cout << c; // Вывод выигрывающей буквы
        }
    }

    if (!ok) { // Если ни одна буква не подходит
        cout << "AI win"; // ИИ выигрывает
    }
}
```

Реализация на Python

```
def count_rectangles(n, m, grid, char, posx, posy):
    # Создаем массив d для хранения высот
    d = [[0] * m for _ in range(n)]

    # Заполняем пустую позицию выбранным символом
    grid[posx][posy] = char

    # Вычисляем высоты столбцов для каждого символа
    for i in range(n):
        for j in range(m):
            if i == 0:
                d[i][j] = 1
            else:
                if grid[i - 1][j] == grid[i][j]:
                    d[i][j] = d[i - 1][j] + 1
                else:
                    d[i][j] = 1

    # Подсчет прямоугольников
    res = 0

    for i in range(n):
        for j in range(i, -1, -1):
            t = 0
            if d[i][0] >= i - j + 1:
                t = 1

            res += t

            for k in range(1, m):
                if d[i][k] < i - j + 1:
                    t = 0
                    continue

                if grid[j][k] == grid[j][k - 1]:
                    t += 1
                    res += t
                    continue

                t = 1
```

```
                res += t

    return res

def main():
    # Чтение входных данных
    n, m = map(int, input().split())
    grid = [input().strip() for _ in range(n)]

    # Нахождение позиции пустой клетки
    posx, posy = -1, -1
    for i in range(n):
        for j in range(m):
            if grid[i][j] == '.':
                posx, posy = i, j
                break
        if posx != -1:
            break

    winning_chars = []

    # Проверка символов от 'a' до 'e'
    for char in 'abcde':
        rectangle_count = count_rectangles(n, m, \
          [list(row) for row in grid], char, posx, posy)
        if rectangle_count % 2 == 0:  # Проверка четности
            winning_chars.append(char)

    # Вывод результата
    if winning_chars:
        print(''.join(winning_chars))
    else:
        print("AI win")

if __name__ == "__main__":
    main()
```

Основной шаг алгоритма включает обход всех ячеек матрицы, чтобы определить высоту каждого столбца для каждого символа. Вы должны пройти по всем элементам матрицы и обновить вспомогательную матрицу d. Этот процесс также занимает O(m×n) времени, так как каждый элемент матрицы посещается один раз. После того, как высоты столбцов определены, необходимо подсчитать количество возможных прямоугольников. Для этого вам нужно будет обработать каждую строку матрицы, а затем для каждой строки пройтись по ее высотам, чтобы вычислить возможные прямоугольники, основываясь на текущих высотах. Общее время на этом этапе также останется в пределах O(m×n).

Таким образом, учитывая все шаги алгоритма, общая временная сложность будет составлять O(m×n), где m — количество строк, а n — количество столбцов. Динамическое программирование является ключом к оптимизации решений для задач, связанных с подсчетом и комбинаторикой, что позволяет избежать избыточного перебора всех возможных комбинаций.

В решении используется вспомогательная матрица d для хранения высоты столбцов для каждого символа. Если исходная матрица имеет размер m×n, то для хранения этой вспомогательной матрицы нам

потребуется O(m×n) памяти. Это означает, что по памяти сложность равна также O(m×n). В дополнение к матрице d, могут потребоваться несколько дополнительных переменных для хранения промежуточных результатов, таких как счетчики или индексы. Однако количество этих переменных не зависит от размера входных данных и является константным, что не влияет существенно на общую сложность по памяти.

Результат выполнения алгоритма для указанных исходных данных

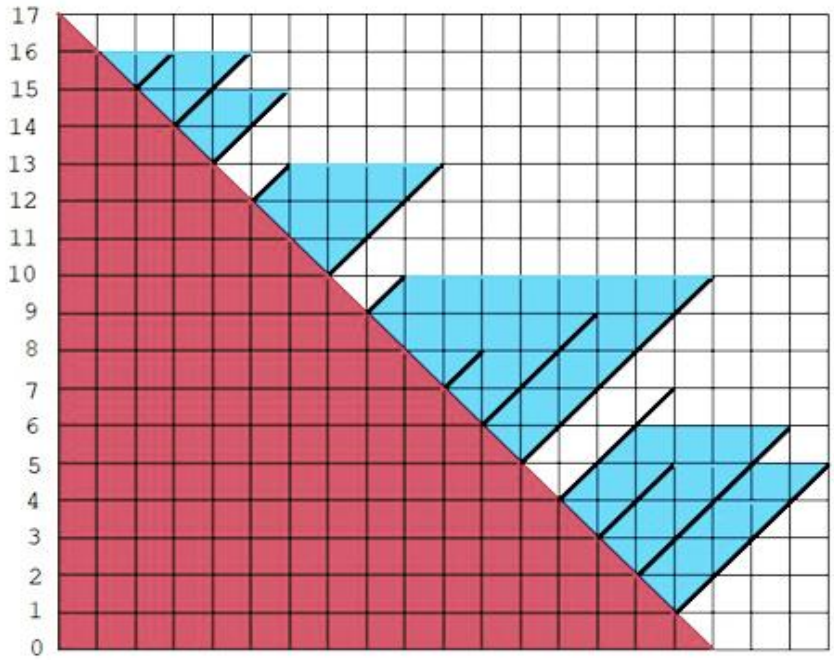
стандартный ввод	стандартный вывод
3 3 eee d.b dbb	bde

Количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами, при подстановке вместо точки для каждой буквы: a – 15, b – 18, c – 15, d – 16, e – 16.

Задача 6 (25 баллов)

Противолавинные заграждения

На одном горном курорте на склоне построили систему противолавинных заграждений. Склон имеет угол 45° к горизонтали. На склоне в некоторых целых по высоте от подножия горы точках расположены пояса заграждений. Каждый пояс расположен строго перпендикулярно склону. Рассмотрим двумерное сечение этого сооружения (на рисунке).



Склон представлен в виде темного треугольника, а каждое заграждение – в виде отрезка, перпендикулярного склону. Снег, скатываясь сверху, попадает в некоторые отсеки, образуемые заграждениями, и застревает в них, постепенно заполняя эти отсеки. Так как снега очень много, каждый отсек заполняется до того момента, пока снег не забьет его до самого верха (образуется горизонтальный уровень заполнения). После этого снег начинает высыпаться из отсека и попадать в нижние отсеки. Некоторые заграждения настолько велики, что, при их заполнении оказываются заполненными и некоторые более вышерасположенные отсеки, у которых граница расположена ниже горизонтали заполнения. В тоже время в некоторые заграждения снег наоборот может не попасть из-за того, что при переполнении верхнего отсека, он вертикально падает вниз, минуя текущий отсек, так как вертикаль падения проходит мимо него.

Будем считать, что если граница заграждения находится ровно на вертикали падения либо горизонтали заполнения других отсеков, то из них снег в этот отсек не попадает. Например, заграждение на высоте 12 из примера пусто, так как его окончание и по вертикали и по горизонтали

находится на соответствующих прямых. В отсек, образуемый заграждением на высоте 3, снег из верхнего отсека непосредственно не попадает, но так как его высота меньше горизонтали заполнения отсека заграждения на высоте 2, то снег его всё-таки заполнит.

По заданным высотам и длинам заграждений требуется оценить максимальную задерживающую способность этих заграждений. Так как мы работаем с двумерной моделью, то требуется найти площадь сечения, заполненную снегом при полном заполнении всей системы заграждений.

Формат входных данных

В первой строке находится число **n** – количество заграждений на склоне $\mathbf{1 \le n \le 3*10^5}$.

В следующих **n** строках находятся по два числа $\mathbf{h_i}$ и $\mathbf{t_i}$ – высота расположения **i**-го заграждения и его длина. Длина любого заграждения равна целому числу диагоналей единичного квадратика со стороной 1 метр. В этих диагоналях она и указывается $\mathbf{0 \le h_i \le 2 * 10^9}$, $\mathbf{1 \le t_i \le 10^6}$. Заграждения перечислены во входных данных в порядке возрастания высоты их нахождения на склоне.

Формат выходных данных

Вывести одно число – максимальную площадь сечения, занятую снегом при полном заполнении всех отсеков, в которые снег может попасть. Можно показать, что она всегда будет целым числом. Толщиной заграждений можно пренебречь. Считаем, что высота склона достаточна, чтобы любое заграждение наполнилось.

Примеры

стандартный ввод	стандартный вывод
13 1 4 2 4 3 2 4 3 5 5 6 3 7 1 9 1 10 3 12 1 13 2 14 2 15 1	58

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения

программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
5 2 5 3 2 5 3 8 3 10 7	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Для начала заметим, что если отсек высотой h ничем слева не ограничен, то площадь его сечения равна h^2. Далее, если отсек высоты h ограничен другим отсеком слева, то нужно провести горизонталь от отсека высоты h на этот ограничивающий и узнать высоту точки пересечения. Допустим, высота этой точки равна z. Тогда площадь заполнения нашего отсека будет равна $h^2 – z^2$. Это доказывает целочисленность ответа.

Основной операцией для этой задачи рассмотрим вертикальное и горизонтальное смещение заграждения. Например, вертикальное смещение: берем наше заграждение, фиксируем вертикальные линии сетки, которые его ограничивают и двигаем это заграждение по этим линиям вниз. Очевидно, в каждой целой точке склона это заграждение будут иметь все меньшую высоту, равную t.len - (t.h - v[i].h), (где t.len – исходная высота заграждения, t.h - v[i].h – смещение) пока не уйдет со склона в минус. Аналогично горизонтальное движение влево.

Теперь эффективно найдем те отсеки, в которые снег попадает (или не попадает) сверху. Добавим фиктивный отсек на высоте $2 * 10^9$ длиной 0, объявим его текущим. Далее перебираем заграждения в обратном порядке и смещаем текущее заграждение вертикально так, чтобы оно стало на диагональ рассматриваемого. Если текущее стало строго меньше рассматриваемого, то рассматриваемое становится текущим, в него снег попадает (отмечаем это 1 в соответствующей ячейке массива from_up), те отсеки, в которые снег сверху не попадает, остаются отмеченными 0.

Далее найдем ответ. Для этого добавим снизу фиктивный отсек на высоте -1 длиной 0, объявим его текущим. Перебираем заграждения снизу вверх (в порядке возрастания высоты крепления) и горизонтально сдвигаем текущее в точку рассматриваемого. Если текущее стало меньше либо равно 0, это означает, что его ничто не ограничивает слева и если в текущее попадает снег сверху, то добавим квадрат его высоты к ответу.

Если текущее больше 0, но меньше либо равно рассматриваемому, то (если в текущее попадает снег), добавим к ответу разницу соответствующих квадратов, текущим сделаем рассматриваемое.

Если текущее строго больше рассматриваемого в точке, то рассматриваемое никак не влияет на заполнение и будет заполнено снегом, если в текущем этот снег есть. Такой случай обрабатывать никак не нужно.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Структура, представляющая заграждение
struct bord {
    int h;    // высота заграждения
    int len;  // длина заграждения
};

signed main() {
    // Оптимизация ввода-вывода
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n;
    cin >> n; // Считываем количество заграждений
    vector<bord> v(n + 1); // Вектор для хранения заграждений
    v[0] = {-1, 0}; // Фиктивное заграждение с высотой -1 и длиной 0
    for (int i = 1; i <= n; i++) {
        cin >> v[i].h >> v[i].len; // Считываем высоту и длину заграждений
    }
    v.push_back({(int)2e9, 0}); // Фиктивное заграждение

    bord t = v.back(); // Последнее заграждение
    int last; // Поддерживаем индекс последнего заграждения
    vector<int> from_up(n + 1, 0); // Массив о попадании снега
    // Идем по заграждениям снизу вверх, определяя может ли быть заполнено
    for (int i = n; i >= 1; i--) {
        int tlen = t.len - (t.h - v[i].h);
        if (tlen < v[i].len) {
            from_up[i] = 1; // Текущее заграждение может быть заполнено
            t = v[i]; // Обновляем текущее заграждение
        }
    }
    t = v[0]; // Сбрасываем текущее заграждение на первое
    last = 0; // Обнуляем индекс последнего заграждения

    int ans = 0; // Переменная для хранения площади, занятой снегом
    //Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
     for (int i = 1; i <= n + 1; i++) {
        int tlen = t.len - (v[i].h - t.h);
```

```
        if (tlen <= 0) {
            if (from_up[last]) {
                ans += v[last].len * v[last].len;
            }
            t = v[i];
            last = i;
            continue;
        }
        if (tlen <= v[i].len) {
            if (from_up[last]) {
                ans += (v[last].len * v[last].len - tlen * tlen);
            }
            t = v[i];
            last = i;
        }
    }
    cout << ans << endl; // Выводим результирующую площадь, занятую снегом
}
```

Реализация на Python

```
class Bord:
    def __init__(self, h, length):
        self.h = h
        self.len = length

def main():
    n = int(input().split())
    v = [Bord(-1, 0)]  # Добавляем фиктивный отсек на высоте -1
    for i in range(1, n + 1):
        h, t = map(int, input().split())
        v.append(Bord(h, t))
    v.append(Bord(2 * 10**9, 0))  # Добавляем отсек на высоте 2 * 10^9
    from_up = [0] * (n + 1) # Массив о попадании снега сверху
    t = v[-1]
# Идем по заграждениям снизу вверх
    for i in range(n, 0, -1):
        tlen = t.len - (t.h - v[i].h)
        if tlen < v[i].len:
            from_up[i] = 1
            t = v[i]

    t = v[0]
    last = 0
    ans = 0
    # Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
    for i in range(1, n + 2):  # +2, из-за фиктивного отсека
        tlen = t.len - (v[i].h - t.h)
        if tlen <= 0:
            if from_up[last]:
                ans += v[last].len * v[last].len
            t = v[i]
            last = i
            continue

        if tlen <= v[i].len:
            if from_up[last]:
                ans += (v[last].len * v[last].len - tlen * tlen)
            t = v[i]
            last = i
```

```
    print(ans)

if __name__ == "__main__":
    main()
```

Программа в основном состоит из нескольких проходов по массиву v, который содержит заграждения. Первый проход — мы считываем данные из файла и заполняем массив v объектами класса Bord. Этот проход происходит в одном цикле от 1 до n, где n — количество заграждений. Таким образом, временная сложность этого шага составляет O(n). Второй проход — во время этого прохода мы идем в обратном направлении по массиву v, чтобы определить, может ли снег попадать сверху. Это также O(n), поскольку мы проходим по всем элементам массива по одному разу. Третий проход — здесь мы снова проходим по массиву v от начала до конца, анализируя пространство между заграждениями и высоту заполнения. Это также O(n).

Таким образом, суммируя все проходы, мы получаем, что общая временная сложность программы составляет O(n).

Мы создаем массив v, длиной n+2, который хранит объекты класса Bord. Таким образом, память, занимаемая этим массивом, составляет O(n). Массив from_up также имеет размер n+1, что добавляет еще O(n) к пространственной сложности. Другие переменные, такие как t, last, и ans, занимают постоянное количество памяти, которое не зависит от размера входных данных. Таким образом, их вклад в общую пространственную сложность можно считать константным: O(1).

Итак, собрав все элементы вместе, можно заключить, что общая пространственная сложность программы также составляет O(n).

Результат выполнения алгоритма для указанных исходных данных

стандартный ввод	стандартный вывод
5 2 5 3 2 5 3 8 3 10 7	**79**

ВАРИАНТ 5

Задача 1 (10 баллов)

Нашими разведчиками на командный пункт были переданы места сосредоточения войск противника в виде горизонтальных и вертикальных координат, расположенных в ячейках B2:F2 и A3:A7. Определите их по формулам закодированной электронной таблицы, представленной в двух режимах, на рисунках расположенных ниже.

1	=СУММ(B2:F2)/(A4-A6)	
2	=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))	
3	=A3*(A4+A6)	
4	=A3-A4-A6	
5	=ЕСЛИ(СУММ(A3:A7)>45;A3+A4;A6+A5)	
6	=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))	
7	=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10	
8	=НАЙТИ(C2;"ABCDEFGHIJKLMNOPQ")-E11+E12	
9	=НАЙТИ(D2;"ABCDEFGHIJKLMNOPQ")-E13+E14	
10	=НАЙТИ(E2;"ABCDEFGHIJKLMNOPQ")-E14+E15	
11	=НАЙТИ(F2;"ABCDEFGHIJKLMNOPQ")	

a=	=ABS(ОКРУГЛВНИЗ(-ПИ();0))	
b=	=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)	
c=	=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))	
d=	=ЕСЛИ(ЕОШ(B9);E9*E10;E10-E9)	
e=	=B10-E9	
f=	=B10^2+E15	
g=	1	

1	#ДЕЛ/0!	
2	5	
3	30	
4	-7	
5	12	
6	абв	
7	6	
8	20	
9	36	
10	-24	
11	3	

a=	
b=	
c=	
d=	
e=	
f=	
g=	

Ответ оформите в виде двух строк, в верхней строке запишите через запятую значения ячеек B2:F2, а в нижней строке также через запятую значения ячеек A3:A7. В случае нескольких вариантов ответов запишите их через союз «или».

Решение

Начнем с определения коэффициентов a, b, c, d, e, f и g.

Значение коэффициента a (ячейка E9) в соответствии с формулой «=ABS(ОКРУГЛВНИЗ(-ПИ();0))» равно 3.

Значение коэффициента b (ячейка E10) в соответствии с формулой «=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)» равно 4.

Значение коэффициента c (ячейка E11) в соответствии с формулой «=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))» равно 2.

Значение коэффициента d (ячейка E12) в соответствии с формулой «=ЕСЛИ(ЕОШ(B9);E9*E10;E10-E9)» равно 12.

Значение коэффициента e (ячейка E13) в соответствии с формулой «=B10-E9» равно 2.

Значение коэффициента f (ячейка E14) в соответствии с формулой «=B10^2+E15» равно 26.

Значение коэффициента g (ячейка E15) равно 1.

Получаем:

	D	E
8		
9	a=	3
10	b=	4
11	c=	2
12	d=	12
13	e=	2
14	f=	26
15	g=	1

1) Так как значение формулы в ячейке B9 «=СУММ(B2:F2)/(A4-A6)» определяется как ошибка деления на 0, то значения ячеек A4 и A6 одинаковы.

2) Значение формулы в ячейке B10 «=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))» равно 5, следовательно значения координат в ячейках A3:A7 состоят из одного символа.

3) Формулы в ячейках B11 «=A3*(A4+A6)» и B12 «=A3-A4-A6» рассмотрим, как систему уравнений, учитывая, что значения ячеек A4 и A6 одинаковы.

$$\begin{cases} A3 * (A4 + A6) = 30 \\ A3 - A4 - A6 = -7 \end{cases} \Rightarrow \begin{cases} A3 * 2A4 = 30 \\ A3 - 2A4 = -7 \end{cases}$$

Решая систему уравнений, получаем два набора значений для ячеек A3 и A4 (A3 = 3 и A4 = 5) либо (A3 = -10 и A4 = -1,5). Так как, в соответствии со вторым пунктом, значения ячеек A3:A7 состоят из одного символа, а это диапазон от 0 до 9, то A3 = 3, A4 = A6 = 5.

4) Значение формулы в ячейке B13 «=ЕСЛИ(СУММ(A3:A7)>45; A3+A4;A6+A5)» равно 12. Максимальное значение формулы СУММ(A3:A7) из-за того, что значения в этих ячейках не превышают 9, равно 45. В соответствие с этим, условие в формуле ячейки B13 не выполнится, следовательно A6 + A5 = 12 и A5 = 12 - 5 = 7.

5) Значение формулы в ячейке B14 «=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))» равно «абв». Следовательно разница между ячейками A7 и A5 равна 3, откуда A7 равно 4 или 10. По условию 0<=A7<=9, A7 = 4.

6) Значение формулы в ячейке B15 «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10» равно 6. Откуда, значение формулы «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")» = 6 + E9 – E10 = 6 + 3 - 4 = 5, следовательно B2 = «E».

7) Значения ячеек C2, D2, E2 и F2 аналогичным образом находим из формул в ячейках B16, B17, B18 и B19. Получаем C2 = «J», D2 = «L», E2 = «A» и F2 = «C».

Ответ: **E, J, L, A, C**
3, 5, 7, 5, 4

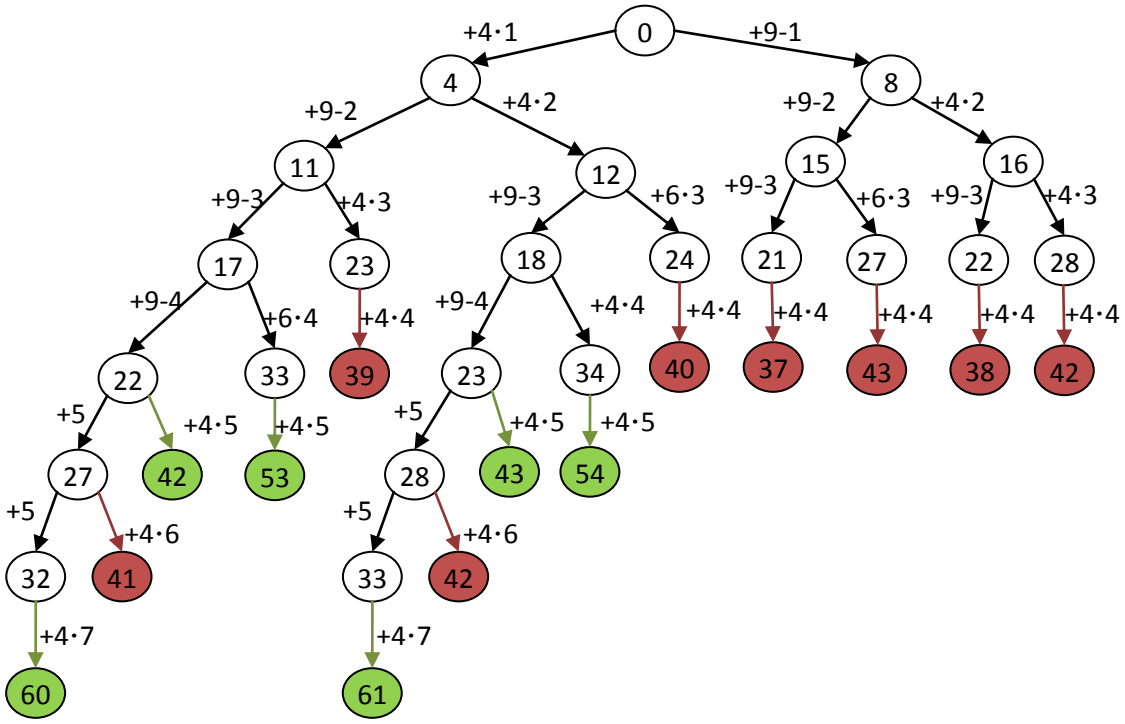
Задача 2 (10 баллов)

Пакеты соли: стратегическая дуэль на складе

Два работника ночной смены на складе выполнили задачу на смену и решили сыграть в следующую игру. По очереди они выкладывают пачки с солью на полку. За один ход можно либо положить на полку количество пачек, определяемое формулой девять минус номер хода в игре, или пять, если результат менее пяти, либо положить на полку количество пачек, определяемое формулой четыре умноженное на номер хода в игре. При превышении количества 35 пачек на полке, игра заканчивается. Кто первым, сделав ход, получит на полке 35 пачек или больше – считается победителем. Кто выигрывает при безошибочной игре – работник, делающий первый ход, или работник, делающий второй ход? Каково минимальное число ходов до победы, если выигрывающий работник пытается закончить игру за меньшее число ходов, а проигрывающий пытается максимально задержать свое поражение? Какое максимальное число пачек может оказаться на полке в конце игры? Какой работник при этом победит? Поясните алгоритм графически, используя ориентированные графы в виде деревьев, с обозначением выигрышных и проигрышных позиций как вершин, а действий игроков – дугами графа с указанием численных изменений.

Решение

Построим ориентированный граф, где зеленым цветом выделены выигрышные позиции первого работника, красным цветом – выигрышные позиции второго работника, а номер соответствует количеству пачек на полке. Рядом со стрелкой указано изменение числа пачек на полке на этом ходу. В случае, если один из вариантов хода приводит к победе, то второй вариант не отображается, за исключением ветвей, которые демонстрирует игру с максимально возможным числом пачек на полке в конце игры.



Из графа видно, что при любой игре второго работника, гарантированно победит первый работник на 5 ходу, если первым ходом положит 4, а третьим 17 или 18 пачек. Максимально возможное число пачек можно получить, если увеличивать количество пачек на полке таким образом, чтобы получить как можно более высокий номер хода (7) и количества пачек на полке (33) перед последним ходом, в котором берется вариант с большим значением (+4·7). При этом побеждает первый работник с количеством пачек 61.

Ответ: гарантированно победит первый работник за 5 ходов, максимум 46 пачек при победе первого работника.

Задача 3 (10 баллов)

Логическая головоломка: разгадка функции F

Паша решил усовершенствовать задание ЕГЭ по информатике
Логическая функция задается выражением

$$(z \vee x) \equiv (y \vee w) \rightarrow (\neg z \rightarrow w)$$

Ниже приведен фрагмент таблицы истинности функции F, содержащий неповторяющиеся строки. Однако вместо значений, Паша решил вписать тоже несколько логических выражений, результат которых надо определить, если **a, b, c** – следующие высказывания:

a – «NaCl - химическая формула воды»

b – «Париж - столица Франции»

c – «В 1 тонне 1000 килограмм»

Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных **w,x,y,z.**

?	?	?	?	F
				0
	a→b→c	b→c∨a	b∧¬c∧a	0
			a→(b≡c)	0
		c⊕b∧a		0

В ответе напишите буквы **w,x,y,z** в том порядке, в котором идут соответствующие им столбцы.

Решение

Сперва определим значения логических высказываний. Очевидно, что

a=0
b=1
c=1

Теперь определим значения логических выражений в таблице. В результате получим следующую таблицу:

?	**?**	**?**	**?**	**F**
				0
	1	1	0	0
			1	0
		1		0

Следующим шагом будет составление таблицы истинности для функции F

w	x	y	z	zVx	yVw	¬z	¬z→w	(yVw)→(¬z→w)	F
0	0	0	0	0	0	1	0	1	0
0	0	0	1	1	0	0	1	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	1	0	1	1	1
0	1	0	0	1	0	1	0	1	1
0	1	0	1	1	0	0	1	1	1
0	1	1	0	1	1	1	0	0	0
0	1	1	1	1	1	0	1	1	1
1	0	0	0	0	1	1	1	1	0
1	0	0	1	1	1	0	1	1	1
1	0	1	0	0	1	1	1	1	0
1	0	1	1	1	1	0	1	1	1
1	1	0	0	1	1	1	1	1	1
1	1	0	1	1	1	0	1	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	0	1	1	1

Выберем из нее строки, где функция F = 0.

w	x	y	z	F
0	0	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0

Сравнивая эту таблицу с заданной в задании, и учитывая, что строки в этой таблице выписаны в случайном порядке, легко установить с помощью несложных рассуждений искомые наименования столбцов

z	x	y	w	F
0	0	0	0	0
0	1	1	0	0
0	0	0	1	0
0	0	1	1	0

Ответ: z,x,y,w

Задача 4 (20 баллов)

Зашифрованный Рудник

В горнодобывающей компании «Рудник-Х» используется система шифрования для передачи информации о наименовании полезного ископаемого и параметрах пласта с данным ценным компонентом (длина, ширина), которая была получена при проведении геологической разведки месторождений полезных ископаемых. Передача информации осуществляется геологами с места проведения полевых работ в административное управление компании.

Процесс шифрования в горнодобывающей компании заключается в следующем:

1) Генерируется супервозрастающая последовательность чисел $A = \{a_1, a_2, \dots a_k\}$, каждый член которой больше суммы всех предыдущих членов, т.е. при $j = 2,3, \dots k: a_j > \sum_{i=1}^{j-1} a_i$, которая является секретным ключом.

Например, секретный ключ: $A = \{a_1, a_2, a_3, a_4\} = \{2, 3, 9, 17\}$,
где $a_2 > a_1 \rightarrow 3 > 2$;
$a_3 > (a_1 + a_2) \rightarrow 9 > 5$;
$a_4 > (a_1 + a_2 + a_3) \rightarrow 17 > 14$.

2) После этого выбирается число m большее, чем сумма чисел последовательности, т.е. $m > \sum_{i=1}^{k} a_i$. Также выбирается число n взаимно простое с m, т.е. наибольший общий делитель m и n равен 1.

3) Далее создается открытый ключ, т.е. новая последовательность чисел $B = \{b_1, b_2, \dots b_k\}$ путем умножения каждого числа a_i на n и взятия остатка от деления данного произведения на m.

Например, для открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ каждый член последовательности вычисляется по формуле $b_i = (a_i \cdot n) \bmod m$,
где a_i — i-й элемент последовательности секретного ключа A;
n — число взаимно простое с m;
m — модуль операции взятия остатка, иными словами, делитель;
mod — операция взятия остатка от деления.
При $n = 91$, $m = 320$ и $A = \{2, 3, 9, 17\}$:
$b_1 = (a_1 \cdot n)\ mod\ m = (2 \cdot 91)\ mod\ 320 = 182$ или, иными словами, $(2 \cdot 91)$ делится на 320 с остатком 182.
Сделав аналогичные вычисления для остальных элементов, получается последовательность $B = \{182,\ 273, 179, 267\}$

4) Полученная последовательность B используется для шифрования отправляемого сообщения. Отправляемое сообщение, представленное в виде двоичной строки, определяет, какие элементы этой последовательности суммируются для получения последовательности шифротекста $C = \{c_1, c_2, \dots c_p\}$, которая передается получателю.

Например, при шифровании числа 10 последовательность шифротекста C будет вычисляться следующим образом:
1) Число 10 представляется в двоичной системе счисления, т.е. $10 = 1010_2$;
2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1 и b_3 суммируются для получения шифротекста, b_2 и b_4 – опускаются.
3) Шифротекст $C = \{c_1\} = \{b_1 + b_3\} = 182 + 179 = 361$.
Например, при шифровании числа E_{16} последовательность шифротекста C будет вычисляться следующим образом:
1) Число E_{16} представляется в двоичной системе счисления, т.е. $E_{16} = 1110_2$;
2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1, b_2 и b_3 суммируются для получения шифротекста, b_4 – опускается.
3) Шифротекст $C = \{c_1\} = \{b_1 + b_2 + b_3\} = 182 + 273 + 179 = 634$.

Для дешифрирования сообщения получатель, исходя из известного ему n, определяет n^{-1} – мультипликативное обратное к значению n по модулю m (остаток от деления $n \cdot n^{-1}$ на m равен 1).

Иными словами, $(n \cdot n^{-1}) \bmod m = 1$,
где n — заданное число взаимно простое с m;
m — модуль операции взятия остатка;
mod — операция взятия остатка от деления.

Далее вычисляются промежуточные значения последовательности C' путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

Например, для шифротекста $C = \{c_1\} = 361$, каждый член последовательности C' вычисляется по формуле $c'_i = (c_i \cdot n^{-1}) \bmod m$,
где c_i — i-й элемент последовательности шифротекста C;
n^{-1} — мультипликативное обратное;
m — модуль операции взятия остатка;
mod — операция взятия остатка от деления.
В данном случае, последовательность
$C' = \{c'_1\} = (361 \cdot 211) \bmod 320 = 11$, где 211 – мультипликативное обратное n^{-1}, рассчитанное получателем.

Далее получатель определяет двоичную строку M путем последовательного сравнения каждого элемента последовательности C' с элементами секретного ключа A справа налево: если $c'_i \geq a_i$, то $M_i = 1$ и осуществляется вычитание a_i из c'_i, в противном случае $M_i = 0$. Таким образом восстанавливается зашифрованное сообщение.

Например, для последовательности $C' = \{c'_1\}$ и секретного ключа $A = \{a_1, a_2, a_3, a_4\}$ осуществляется последовательное сравнение элементов справа налево:
1) Если $c'_1 \geq a_4$, то $M_4 = 1$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число $c'_1 - a_4$;
2) Если $c'_1 < a_4$, то $M_4 = 0$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число c'_1.
3) В ответе получается двоичная строка $M = \{M_1, M_2, M_3, M_4\}$.

Например, при $A = \{2, 3, 9, 17\}$ и $C' = \{11\}$ двоичная строка $M = \{1, 0, 1, 0\}$, что согласуется с двоичным представлением числа 10.

Для последовательности C', состоящей из нескольких элементов, например, $C' = \{c'_1, c'_2, \dots c'_k\}$ осуществляется последовательное сравнение каждого элемента $c'_1, c'_2, \dots c'_k$ с элементами секретного ключа A с получением количества двоичных строк, равного количеству элементов последовательности C'.

1) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4\}$ количество двоичных строк M равно 4.

2) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7, c'_8\}$ количество двоичных строк M равно 8.

Геологи передали четыре последовательных сообщения в административное управление о двух месторождениях полезных ископаемых. Первое сообщение в каждой паре содержало наименование полезного ископаемого. Второе сообщение содержало информацию о параметрах пласта, при этом первое число соответствовало длине пласта, второе – ширине пласта. Для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы согласно таблице ASCII для Windows-1251, далее полученному шестнадцатеричному числовому значению присваивалась соответствующая двоичная тетрада. Для шифрования числа осуществлялся его перевод в двоичную систему. Таблица ASCII для Windows-1251:

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Были получены следующие зашифрованные сообщения:

1 сообщение: 754 1198 835 1203 1266 772

2 сообщение: 950 1060

3 сообщение: 616 1198 1416 697 1140

4 сообщение: 556 875

Секретный ключ, хранящийся в организации, представляет собой следующую последовательность {3, 5, 9, 18, 36, 75, 150, 298}, значения n и m равны 41 и 600 соответственно.

Расшифруйте полученные сообщения. В ответе укажите последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» – «Длина», «Ширина» (в десятичной системе). Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Решение

Сначала необходимо вычислить промежуточные значения последовательности C' для осуществления дешифровки. В исходных данных сказано, что данные значения вычисляются путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

В условии дано только значение n, однако мы знаем, что n^{-1} – мультипликативное обратное к значению n по модулю m, то есть остаток от деления $n \times n^{-1}$ на m равен 1, поэтому для нахождения значения n^{-1} можно применить расширенный алгоритм Евклида.

Расширенный алгоритм Евклида является модификацией алгоритма Евклида, вычисляющая кроме наибольшего общего делителя (НОД) целых чисел a и b, еще и коэффициенты соотношения Безу, то есть такие x и y, что $ax + by = \text{НОД}(a, b)$. В свою очередь, алгоритм Евклида заключается в последовательном делении с остатком для нахождения наибольшего общего делителя двух чисел. На каждом этапе производится деление большего из пары чисел на меньшее, а остаток записывается и используется при дальнейших вычислениях в качестве нового делителя для предыдущего. Вычисление остатка производят до тех пор, пока он не будет равен нулю. Тогда последний использованный делитель и будет являться наибольшим общим делителем, полученным через алгоритм Евклида. В нашем случае наибольший общий делитель чисел n и m (41 и 600) должен равняться 1.

Применяем алгоритм Евклида:

$600 = 41 \times 14 + 26 \Longrightarrow \quad 41 = 26 \times 1 + 15 \Longrightarrow \quad 26 = 15 \times 1 + 11 \Longrightarrow \quad 15 = 11 \times 1 + 4$

$11 = 4 \times 2 + 3 \qquad 4 = 3 \times 1 + 1 \qquad 3 = 3 \times 1 + 0$

Применяем расширенный алгоритм Евклида. Нам нужно выразить 1 как линейную комбинацию 41 и 600, т. е. найти целые числа x и y, такие, что $41x + 600y = 1$

Из алгоритма Евклида выше:

$1 = 4 - 3 \times 1 \Leftarrow \quad 3 = 11 - 4 \times 2 \Leftarrow \quad 4 = 15 - 11 \times 1 \Leftarrow \quad 11 = 26 - 15 \times 1 \Leftarrow$

$15 = 41 - 26 \times 1 \Leftarrow \quad 26 = 600 - 41 \times 14$

Подставим обратно:

$$1 = 4 - (11 - 4 \times 2) \times 1 = 4 \times 3 - 11 \times 1$$
$$1 = (15 - 11 \times 1) \times 3 - 11 \times 1 = 15 \times 3 - 11 \times 4$$
$$1 = 15 \times 3 - (26 - 15 \times 1) \times 4 = 15 \times 7 - 26 \times 4$$
$$1 = (41 - 26 \times 1) \times 7 - 26 \times 4 = 41 \times 7 - 26 \times 11$$
$$1 = 41 \times 7 - (600 - 41 \times 14) \times 11 = 41 \times 161 - 600 \times 41$$
$$1 = 41 \times 161 - 600 \times 41$$

Мы получили, что $1 = 41 \times 161 - 600 \times 41$ Это означает, что 161 является обратным числом 41 по модулю 600. Получаем, что $n^{-1} = 161$.

Зная значение n^{-1} находим промежуточные значения последовательности C' для каждого полученного сообщения. Для первого сообщения – 754 1198 835 1203 1266 772:

1) (754×161) делится на 600 с остатком 194

2) (1198×161) делится на 600 с остатком 278

3) (835×161) делится на 600 с остатком 35

4) (1203×161) делится на 600 с остатком 483

5) (1266×161) делится на 600 с остатком 426

6) (772×161) делится на 600 с остатком 92

Промежуточные значения последовательности $\{194, 278, 35, 483, 426, 92\}$.

Для второго сообщения – 950 1060:

1) (950×161) делится на 600 с остатком 550

2) (1060×161) делится на 600 с остатком 260

Промежуточные значения последовательности $\{550, 260\}$.

Для третьего сообщения – 616 1198 1416 697 1140:

1) (616×161) делится на 600 с остатком 176

2) (1198×161) делится на 600 с остатком 278

3) (1416×161) делится на 600 с остатком 576

4) (697×161) делится на 600 с остатком 17

5) (1140×161) делится на 600 с остатком 540

Промежуточные значения последовательности $\{176, 278, 576, 17, 540\}$.

Для четвертого сообщения – 556 875:

1) (556×161) делится на 600 с остатком 116

2) (875×161) делится на 600 с остатком 475

Промежуточные значения последовательности $\{116, 475\}$.

Далее используя секретный ключ A, получатель определяет битовую строку M путем последовательного сравнения C' с элементами a_i. Секретный ключ, исходя из исходных условий, представляет собой последовательность $\{3, 5, 9, 18, 36, 75, 150, 298\}$.

Для первого сообщения:

1) $194 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{matrix}$

2) $278 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{matrix}$

3) $35 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$

4) $483 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{matrix}$

5) $426 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{matrix}$

6) $92 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{matrix}$

Для второго сообщения:

1) $550 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{matrix}$

2) $260 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{matrix}$

Для третьего сообщения:

1) $176 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{matrix}$

2) $278 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{matrix}$

3) $576 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{matrix}$

4) $17 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

5) $540 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{matrix}$

Для четвертого сообщения:

1) $116 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{matrix}$

2) $475 = \begin{matrix} 3 & 5 & 9 & 18 & 36 & 75 & 150 & 298 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{matrix}$

Из условий задачи известно, что для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы, далее каждой шестнадцатеричной цифре соответствующая двоичная тетрада; для шифрования числа осуществлялся его перевод в двоичную систему. Для дешифровки необходимо произвести данные действия в обратном порядке.

Сначала дешифруем текстовые сообщения. Для этого воспользуемся таблицей тетрад и далее таблицей ASCII для Windows-1251 из условий задачи.

16-ричная цифра	Двоичная тетрада	16-ричная цифра	Двоичная тетрада
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Таким образом, произведем дешифровку текстовых сообщений:

Первое сообщение:

1) $754 = 11001010 = \begin{matrix}1100 & \text{C}\\ 1010 & \text{A}\end{matrix} = \text{К}$

2) $1198 = 11101110 = \begin{matrix}1110 & \text{E}\\ 1110 & \text{E}\end{matrix} = \text{о}$

3) $835 = 11110000 = \begin{matrix}1111 & \text{F}\\ 0000 & 0\end{matrix} = \text{р}$

4) $1203 = 11110011 = \begin{matrix}1111 & \text{F}\\ 0011 & 3\end{matrix} = \text{у}$

5) $1266 = 11101101 = \begin{matrix}1110 & \text{E}\\ 1101 & \text{D}\end{matrix} = \text{н}$

6) $772 = 11100100 = \begin{matrix}1110 & \text{E}\\ 0100 & 4\end{matrix} = \text{д}$

Третье сообщение:

1) $616 = 11010010 = \begin{matrix}1101 & \text{D}\\ 0010 & 2\end{matrix} = \text{Т}$

2) $1198 = 11101110 = \begin{matrix}1110 & \text{E}\\ 1110 & \text{E}\end{matrix} = \text{о}$

3) $1416 = 11101111 = \begin{matrix}1110 & \text{E}\\ 1111 & \text{F}\end{matrix} = \text{п}$

4) $697 = 11100000 = \begin{matrix}1110 & \text{E}\\ 0000 & 0\end{matrix} = \text{а}$

5) $1140 = 11100111 = \begin{matrix}1110 & \text{E}\\ 0111 & 7\end{matrix} = \text{з}$

Дешифруем числовые сообщения: для этого осуществим перевод найденной битовой последовательности в десятичную систему счисления. Чтобы перевести число из двоичной системы счисления в десятичную, нужно: справа налево пронумеровать цифры в двоичном числе, начиная с 0, далее слева направо умножить каждую цифру двоичного числа на 2 в степени номера этой цифры и сложить полученные значения.

Число в 2-ой системе счисления	Перевод в 10-ую систему счисления	Число в 10-ой системе счисления
Второе сообщение		
00110111_2	$1 \times 2^5 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 32 + 16 + 4 + 2 + 1 = 55$	55_{10}
11110110_2	$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 = 128 + 64 + 32 + 16 + 4 + 2 = 246$	246_{10}
Четвертое сообщение		
01001100_2	$1 \times 2^6 + 1 \times 2^3 + 1 \times 2^2 = 64 + 8 + 4 = 76$	76_{10}
00110011_2	$1 \times 2^5 + 1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 = 32 + 16 + 2 + 1 = 51$	51_{10}

В ответе необходимо указать последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» - «Длина», «Ширина» (числа в десятичной системе). Следовательно, ответ задачи выглядит следующим образом: Корунд – 55, 246, Топаз – 76, 51.

Ответ: Корунд – 55, 246, Топаз – 76, 51

Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Python

```
def find_pairs(m):
    pairs = []
    for n in range(m):
        for nn in range(m):
            if (n * nn) % m == 1:
                pairs.append((n, nn))
    return pairs

m = int(input("Enter the value of m: "))
pairs = find_pairs(m)
for n, nn in pairs:
    print(f"({n}, {nn})")
```

Задача 5 (25 баллов)

Буквенные прямоугольники

Вениамин играет в новую интеллектуальную игру с ИИ. У них есть прямоугольная таблица размера **n** × **m**, в клетки которой они по очереди выбирают и ставят по одной из букв 'a', 'b', 'c', 'd' или 'e'. После того, как все ячейки будут заполнены, ИИ честно подсчитает количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами. Если оно чётно, выиграет Вениамин, иначе выиграет ИИ.

Осталось сделать последний ход, и этот ход делает Вениамин. Подскажите ему, какие из этих букв он может поставить, чтобы выиграть.

Формат входных данных

В первой строке содержатся два натуральных числа **n** и **m** через пробел – размеры игрового поля. **1 ≤ n, m ≤ 500.**

В следующих **n** строках содержится описание текущего заполнения таблицы. Каждая строка состоит из **m** символов, каждый из них – это буква от 'a' до 'e'. Помимо этого, в таблице содержится ровно один символ '.', который обозначает пустую клетку, в которую нужно сделать заключительный ход.

Формат выходных данных

Вывести в одну строку без пробелов список букв, которые может поставить в последнюю свободную клетку Вениамин и выиграть. Буквы не нужно разделять пробелом и требуется выводить в алфавитном порядке. Буквы должны быть из интервала от 'a' до 'e'. Если нет ни одной выигрывающей буквы, вывести сообщение «AI win» без кавычек.

Примеры

стандартный ввод	стандартный вывод
3 4 aadd a.bb eebd	bcd
1 1 .	AI win

Комментарий к примерам

Исходно, из букв 'a' можно составить 5 прямоугольников, из букв 'b' – 5 прямоугольников, из букв 'c' – ни одного, из букв 'd' – 4 прямоугольника и из букв 'e' – 3 прямоугольника.

Если поставить букву 'a' получим 9 прямоугольников из этой буквы и 12 остальных, то есть нечётное количество, выиграет ИИ.

Если поставить букву 'b' получим 8 прямоугольников из этой буквы и 12 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'c' получим 1 прямоугольник из этой буквы и 17 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'd' получим 5 прямоугольников из этой буквы и 13 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'e' получим 5 прямоугольников из этой буквы и 14 остальных, то есть нечётное количество, тогда выиграет ИИ.

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных. Результат выполнения программы для указанных исходных данных должен содержать: итоговый вывод программы и количество различных прямоугольников при подстановке каждой буквы вместо точки.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
3 3 e.e eaa baa	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Решение

Прежде чем перейти к решению, важно понимать структуру данных и подходы, которые можно использовать. У нас имеется двумерный массив (матрица) символов, и мы хотим подсчитать количество прямоугольных областей, состоящих из одного и того же символа. Важно отметить, что прямоугольник определяется как область, в которой все символы одинаковы и находятся на любых позициях в матрице, которые могут образовать прямоугольник.

Элементарное решение требует шестерного цикла: перебираем левый

верхний угол двумя циклами, внутри правый нижний еще двумя и далее перебираем всю внутреннюю часть выбранного прямоугольника чтобы проверить, что она состоит из одной и той же буквы. Получим правильное, но не эффективное решение.

Основная идея эффективного решения заключается в применении алгоритма, основанного на динамическом программировании. Вот шаги, которые мы будем выполнять:

Создание вспомогательной матрицы

Мы создаем переменную d, которая будет хранить высоту столбцов для каждого символа в матрице. Эта матрица позволяет нам отслеживать, сколько последовательных символов одного и того же типа находится над каждой позицией в матрице. Если текущий элемент такой же, как элемент над ним, мы увеличиваем счетчик.

Инициализация

Мы заполняем первую строку матрицы d единицами, поскольку в ней нет элементов, находящихся выше.

Обработка матрицы

Затем мы проходим по всей матрице, заполняя d в зависимости от значений над текущей ячейкой. Если символ текущей ячейки такой же, как символ над ней, мы увеличиваем значение в d, иначе устанавливаем его в 1.

Подсчет прямоугольников

После того как матрица d заполнена, мы начинаем считывать высоту каждого столбца (начиная с нижней строки). Для каждого символа мы находим максимальную высоту, начиная с нижней строки и проводя подсчет, количества символов. Если символы отличаются, мы обнуляем счетчик. Если символ того же типа, то увеличиваем счетчик и добавляем его в общий результат.

Получение результата

После того как мы обработали всю матрицу, результатом будет общее количество найденных прямоугольников, которое мы возвращаем.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Вспомогательная матрица для хранения высот символов
vector<vector<int>> d;

// Функция для подсчета прямоугольников для символа c, позиция (posx, posy)
int F(int n, int m, vector<string> &v, char c, int posx, int posy) {
    d.resize(n, vector<int>(m, 0)); // Инициализация матрицы высот

    v[posx][posy] = c; // Установка буквы в пустую ячейку
```

```
    // Заполнение матрицы высот
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (i == 0) {
                d[i][j] = 1; // Первая строка всегда имеет высоту 1
            } else {
                d[i][j] = (v[i - 1][j] == v[i][j]) ? d[i - 1][j] + 1 : 1;
              // Высота для текущей ячейки
            }
        }
    }

    // Подсчет прямоугольников
    int res = 0;
    for (int i = 0; i < n; i++) {
        // Проход по строкам
        for (int j = i; j >= 0; j--) {
            // Проход по подстрокам для каждой строки
            int t = 0; // Количество текущих прямоугольников
            if (d[i][0] >= i - j + 1) { // Проверка первой колонки
                t = 1; // Находим хотя бы один прямоугольник
            }
            res += t; // Добавляем в общий счетчик

            for (int k = 1; k < m; k++) { // Проход по столбцам
                if (d[i][k] < i - j + 1) {
                    t = 0;
                    // Сбрасываем счетчик, если высота меньше текущей
                    continue;
                }
                if (v[j][k] == v[j][k - 1]) {
                   // Если текущий символ такой же, как предыдущий
                    t++; // Увеличиваем количество прямоугольников
                } else {
                    t = 1; // Обнуляем счетчик
                }
                res += t; // Добавляем к общему результату
            }
        }
    }

    return res; // Возвращаем общее количество прямоугольников
}

signed main() {
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    // Оптимизация ввода-вывода

    int n, m;
    cin >> n >> m; // Чтение размеров массива
    int posx, posy;
    vector<string> v(n);

    // Чтение поля и нахождение позиции пустой ячейки
    for (int i = 0; i < n; i++) {
        cin >> v[i];
        for (int j = 0; j < m; j++) {
            if (v[i][j] == '.') {
                posx = i; // Запоминаем координаты пустой ячейки
                posy = j;
            }
```

```
        }
    }

    bool ok = false; // Флаг для отслеживания возможности выигрыша
    for (char c = 'a'; c <= 'e'; c++) { // Перебор букв от 'a' до 'e'
        int z = F(n, m, v, c, posx, posy);
// Подсчет прямоугольников для каждой буквы

        if (z % 2 == 0) { // Если количество четное
            ok = true; // Вениамин может выиграть
            cout << c; // Вывод выигрывающей буквы
        }
    }

    if (!ok) { // Если ни одна буква не подходит
        cout << "AI win"; // ИИ выигрывает
    }
}
```

Реализация на Python

```
def count_rectangles(n, m, grid, char, posx, posy):
    # Создаем массив d для хранения высот
    d = [[0] * m for _ in range(n)]

    # Заполняем пустую позицию выбранным символом
    grid[posx][posy] = char

    # Вычисляем высоты столбцов для каждого символа
    for i in range(n):
        for j in range(m):
            if i == 0:
                d[i][j] = 1
            else:
                if grid[i - 1][j] == grid[i][j]:
                    d[i][j] = d[i - 1][j] + 1
                else:
                    d[i][j] = 1

    # Подсчет прямоугольников
    res = 0

    for i in range(n):
        for j in range(i, -1, -1):
            t = 0
            if d[i][0] >= i - j + 1:
                t = 1

            res += t

            for k in range(1, m):
                if d[i][k] < i - j + 1:
                    t = 0
                    continue

                if grid[j][k] == grid[j][k - 1]:
                    t += 1
                    res += t
                    continue

                t = 1
```

```
                res += t

    return res

def main():
    # Чтение входных данных
    n, m = map(int, input().split())
    grid = [input().strip() for _ in range(n)]

    # Нахождение позиции пустой клетки
    posx, posy = -1, -1
    for i in range(n):
        for j in range(m):
            if grid[i][j] == '.':
                posx, posy = i, j
                break
        if posx != -1:
            break

    winning_chars = []

    # Проверка символов от 'a' до 'e'
    for char in 'abcde':
        rectangle_count = count_rectangles(n, m, \
          [list(row) for row in grid], char, posx, posy)
        if rectangle_count % 2 == 0:  # Проверка четности
            winning_chars.append(char)

    # Вывод результата
    if winning_chars:
        print(''.join(winning_chars))
    else:
        print("AI win")

if __name__ == "__main__":
    main()
```

Основной шаг алгоритма включает обход всех ячеек матрицы, чтобы определить высоту каждого столбца для каждого символа. Вы должны пройти по всем элементам матрицы и обновить вспомогательную матрицу d. Этот процесс также занимает O(m×n) времени, так как каждый элемент матрицы посещается один раз. После того, как высоты столбцов определены, необходимо подсчитать количество возможных прямоугольников. Для этого вам нужно будет обработать каждую строку матрицы, а затем для каждой строки пройтись по ее высотам, чтобы вычислить возможные прямоугольники, основываясь на текущих высотах. Общее время на этом этапе также останется в пределах O(m×n).

Таким образом, учитывая все шаги алгоритма, общая временная сложность будет составлять O(m×n), где m — количество строк, а n — количество столбцов. Динамическое программирование является ключом к оптимизации решений для задач, связанных с подсчетом и комбинаторикой, что позволяет избежать избыточного перебора всех возможных комбинаций.

В решении используется вспомогательная матрица d для хранения высоты столбцов для каждого символа. Если исходная матрица имеет размер m×n, то для хранения этой вспомогательной матрицы нам

потребуется O(m×n) памяти. Это означает, что по памяти сложность равна также O(m×n). В дополнение к матрице d, могут потребоваться несколько дополнительных переменных для хранения промежуточных результатов, таких как счетчики или индексы. Однако количество этих переменных не зависит от размера входных данных и является константным, что не влияет существенно на общую сложность по памяти.

Результат выполнения алгоритма для указанных исходных данных

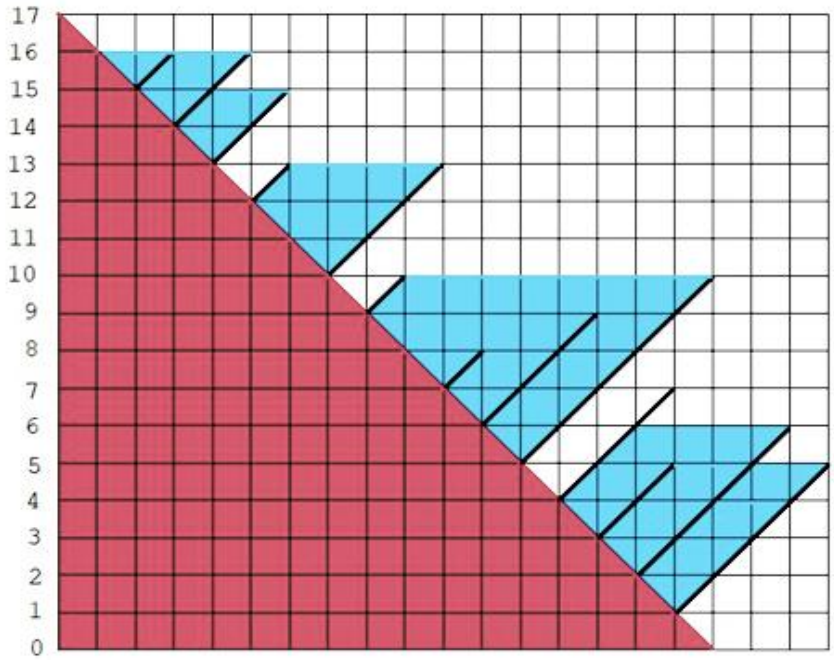
стандартный ввод	стандартный вывод
3 3 e.e eaa baa	e

Количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами, при подстановке вместо точки для каждой буквы: a – 17, b – 15, c – 15, d – 15, e – 18.

Задача 6 (25 баллов)

Противолавинные заграждения

На одном горном курорте на склоне построили систему противолавинных заграждений. Склон имеет угол 45° к горизонтали. На склоне в некоторых целых по высоте от подножия горы точках расположены пояса заграждений. Каждый пояс расположен строго перпендикулярно склону. Рассмотрим двумерное сечение этого сооружения (на рисунке).



Склон представлен в виде темного треугольника, а каждое заграждение – в виде отрезка, перпендикулярного склону. Снег, скатываясь сверху, попадает в некоторые отсеки, образуемые заграждениями, и застревает в них, постепенно заполняя эти отсеки. Так как снега очень много, каждый отсек заполняется до того момента, пока снег не забьет его до самого верха (образуется горизонтальный уровень заполнения). После этого снег начинает высыпаться из отсека и попадать в нижние отсеки. Некоторые заграждения настолько велики, что, при их заполнении оказываются заполненными и некоторые более вышерасположенные отсеки, у которых граница расположена ниже горизонтали заполнения. В тоже время в некоторые заграждения снег наоборот может не попасть из-за того, что при переполнении верхнего отсека, он вертикально падает вниз, минуя текущий отсек, так как вертикаль падения проходит мимо него.

Будем считать, что если граница заграждения находится ровно на вертикали падения либо горизонтали заполнения других отсеков, то из них снег в этот отсек не попадает. Например, заграждение на высоте 12 из примера пусто, так как его окончание и по вертикали и по горизонтали

находится на соответствующих прямых. В отсек, образуемый заграждением на высоте 3, снег из верхнего отсека непосредственно не попадает, но так как его высота меньше горизонтали заполнения отсека заграждения на высоте 2, то снег его всё-таки заполнит.

По заданным высотам и длинам заграждений требуется оценить максимальную задерживающую способность этих заграждений. Так как мы работаем с двумерной моделью, то требуется найти площадь сечения, заполненную снегом при полном заполнении всей системы заграждений.

Формат входных данных

В первой строке находится число **n** – количество заграждений на склоне $\mathbf{1 \le n \le 3*10^5}$.

В следующих **n** строках находятся по два числа $\mathbf{h_i}$ и $\mathbf{t_i}$ – высота расположения **i**-го заграждения и его длина. Длина любого заграждения равна целому числу диагоналей единичного квадратика со стороной 1 метр. В этих диагоналях она и указывается $\mathbf{0 \le h_i \le 2 * 10^9}$, $\mathbf{1 \le t_i \le 10^6}$. Заграждения перечислены во входных данных в порядке возрастания высоты их нахождения на склоне.

Формат выходных данных

Вывести одно число – максимальную площадь сечения, занятую снегом при полном заполнении всех отсеков, в которые снег может попасть. Можно показать, что она всегда будет целым числом. Толщиной заграждений можно пренебречь. Считаем, что высота склона достаточна, чтобы любое заграждение наполнилось.

Примеры

стандартный ввод	стандартный вывод
13 1 4 2 4 3 2 4 3 5 5 6 3 7 1 9 1 10 3 12 1 13 2 14 2 15 1	58

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения

программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
5 2 5 4 4 6 6 8 3 10 4	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Для начала заметим, что если отсек высотой h ничем слева не ограничен, то площадь его сечения равна h^2. Далее, если отсек высоты h ограничен другим отсеком слева, то нужно провести горизонталь от отсека высоты h на этот ограничивающий и узнать высоту точки пересечения. Допустим, высота этой точки равна z. Тогда площадь заполнения нашего отсека будет равна $h^2 – z^2$. Это доказывает целочисленность ответа.

Основной операцией для этой задачи рассмотрим вертикальное и горизонтальное смещение заграждения. Например, вертикальное смещение: берем наше заграждение, фиксируем вертикальные линии сетки, которые его ограничивают и двигаем это заграждение по этим линиям вниз. Очевидно, в каждой целой точке склона это заграждение будут иметь все меньшую высоту, равную t.len - (t.h - v[i].h), (где t.len – исходная высота заграждения, t.h - v[i].h – смещение) пока не уйдет со склона в минус. Аналогично горизонтальное движение влево.

Теперь эффективно найдем те отсеки, в которые снег попадает (или не попадает) сверху. Добавим фиктивный отсек на высоте $2 * 10^9$ длиной 0, объявим его текущим. Далее перебираем заграждения в обратном порядке и смещаем текущее заграждение вертикально так, чтобы оно стало на диагональ рассматриваемого. Если текущее стало строго меньше рассматриваемого, то рассматриваемое становится текущим, в него снег попадает (отмечаем это 1 в соответствующей ячейке массива from_up), те отсеки, в которые снег сверху не попадает, остаются отмеченными 0.

Далее найдем ответ. Для этого добавим снизу фиктивный отсек на высоте -1 длиной 0, объявим его текущим. Перебираем заграждения снизу вверх (в порядке возрастания высоты крепления) и горизонтально сдвигаем текущее в точку рассматриваемого. Если текущее стало меньше либо равно 0, это означает, что его ничто не ограничивает слева и если в текущее попадает снег сверху, то добавим квадрат его высоты к ответу.

Если текущее больше 0, но меньше либо равно рассматриваемому, то (если в текущее попадает снег), добавим к ответу разницу соответствующих квадратов, текущим сделаем рассматриваемое.

Если текущее строго больше рассматриваемого в точке, то рассматриваемое никак не влияет на заполнение и будет заполнено снегом, если в текущем этот снег есть. Такой случай обрабатывать никак не нужно.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Структура, представляющая заграждение
struct bord {
    int h;    // высота заграждения
    int len;  // длина заграждения
};

signed main() {
    // Оптимизация ввода-вывода
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n;
    cin >> n; // Считываем количество заграждений
    vector<bord> v(n + 1); // Вектор для хранения заграждений
    v[0] = {-1, 0}; // Фиктивное заграждение с высотой -1 и длиной 0
    for (int i = 1; i <= n; i++) {
        cin >> v[i].h >> v[i].len; // Считываем высоту и длину заграждений
    }
    v.push_back({(int)2e9, 0}); // Фиктивное заграждение

    bord t = v.back(); // Последнее заграждение
    int last; // Поддерживаем индекс последнего заграждения
    vector<int> from_up(n + 1, 0); // Массив о попадании снега
    // Идем по заграждениям снизу вверх, определяя может ли быть заполнено
    for (int i = n; i >= 1; i--) {
        int tlen = t.len - (t.h - v[i].h);
        if (tlen < v[i].len) {
            from_up[i] = 1; // Текущее заграждение может быть заполнено
            t = v[i]; // Обновляем текущее заграждение
        }
    }
    t = v[0]; // Сбрасываем текущее заграждение на первое
    last = 0; // Обнуляем индекс последнего заграждения

    int ans = 0; // Переменная для хранения площади, занятой снегом
    //Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
     for (int i = 1; i <= n + 1; i++) {
        int tlen = t.len - (v[i].h - t.h);
```

```
        if (tlen <= 0) {
            if (from_up[last]) {
                ans += v[last].len * v[last].len;
            }
            t = v[i];
            last = i;
            continue;
        }
        if (tlen <= v[i].len) {
            if (from_up[last]) {
                ans += (v[last].len * v[last].len - tlen * tlen);
            }
            t = v[i];
            last = i;
        }
    }
    cout << ans << endl; // Выводим результирующую площадь, занятую снегом
}
```

Реализация на Python

```
class Bord:
    def __init__(self, h, length):
        self.h = h
        self.len = length

def main():
    n = int(input().split())
    v = [Bord(-1, 0)]  # Добавляем фиктивный отсек на высоте -1
    for i in range(1, n + 1):
        h, t = map(int, input().split())
        v.append(Bord(h, t))
    v.append(Bord(2 * 10**9, 0))  # Добавляем отсек на высоте 2 * 10^9
    from_up = [0] * (n + 1) # Массив о попадании снега сверху
    t = v[-1]
# Идем по заграждениям снизу вверх
    for i in range(n, 0, -1):
        tlen = t.len - (t.h - v[i].h)
        if tlen < v[i].len:
            from_up[i] = 1
            t = v[i]

    t = v[0]
    last = 0
    ans = 0
    # Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
    for i in range(1, n + 2):  # +2, из-за фиктивного отсека
        tlen = t.len - (v[i].h - t.h)
        if tlen <= 0:
            if from_up[last]:
                ans += v[last].len * v[last].len
            t = v[i]
            last = i
            continue

        if tlen <= v[i].len:
            if from_up[last]:
                ans += (v[last].len * v[last].len - tlen * tlen)
            t = v[i]
            last = i
```

```
    print(ans)

if __name__ == "__main__":
    main()
```

Программа в основном состоит из нескольких проходов по массиву v, который содержит заграждения. Первый проход — мы считываем данные из файла и заполняем массив v объектами класса Bord. Этот проход происходит в одном цикле от 1 до n, где n — количество заграждений. Таким образом, временная сложность этого шага составляет O(n). Второй проход — во время этого прохода мы идем в обратном направлении по массиву v, чтобы определить, может ли снег попадать сверху. Это также O(n), поскольку мы проходим по всем элементам массива по одному разу. Третий проход — здесь мы снова проходим по массиву v от начала до конца, анализируя пространство между заграждениями и высоту заполнения. Это также O(n).

Таким образом, суммируя все проходы, мы получаем, что общая временная сложность программы составляет O(n).

Мы создаем массив v, длиной n+2, который хранит объекты класса Bord. Таким образом, память, занимаемая этим массивом, составляет O(n). Массив from_up также имеет размер n+1, что добавляет еще O(n) к пространственной сложности. Другие переменные, такие как t, last, и ans, занимают постоянное количество памяти, которое не зависит от размера входных данных. Таким образом, их вклад в общую пространственную сложность можно считать константным: O(1).

Итак, собрав все элементы вместе, можно заключить, что общая пространственная сложность программы также составляет O(n).

Результат выполнения алгоритма для указанных исходных данных

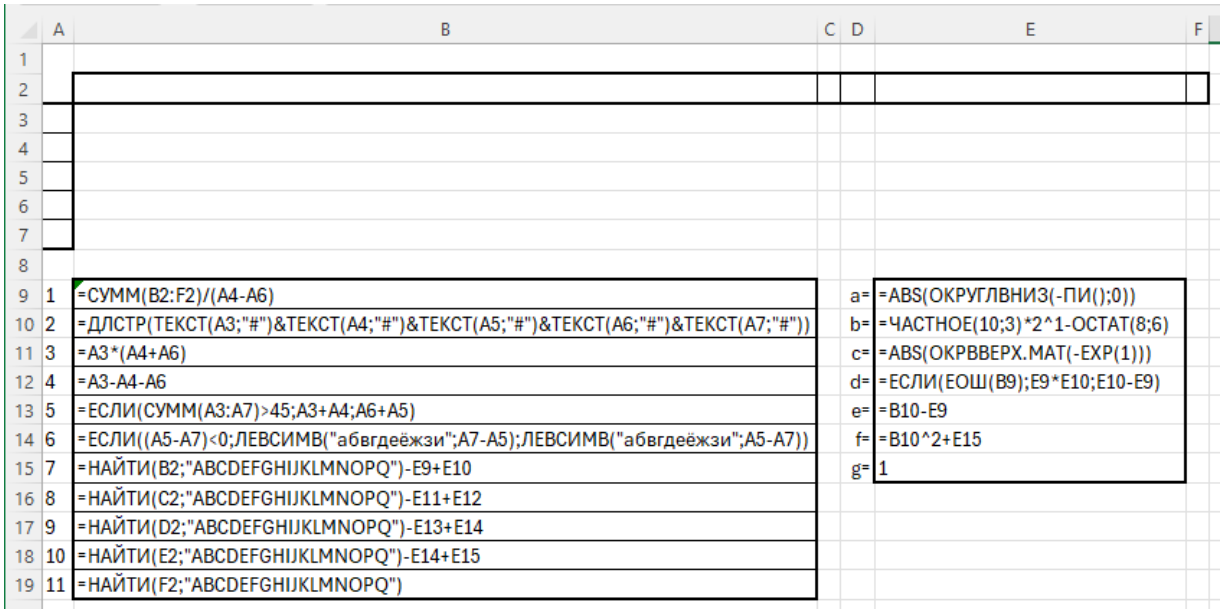
стандартный ввод	стандартный вывод
5 2 5 4 4 6 6 8 3 10 4	**64**

ВАРИАНТ 6

Задача 1 (10 баллов)

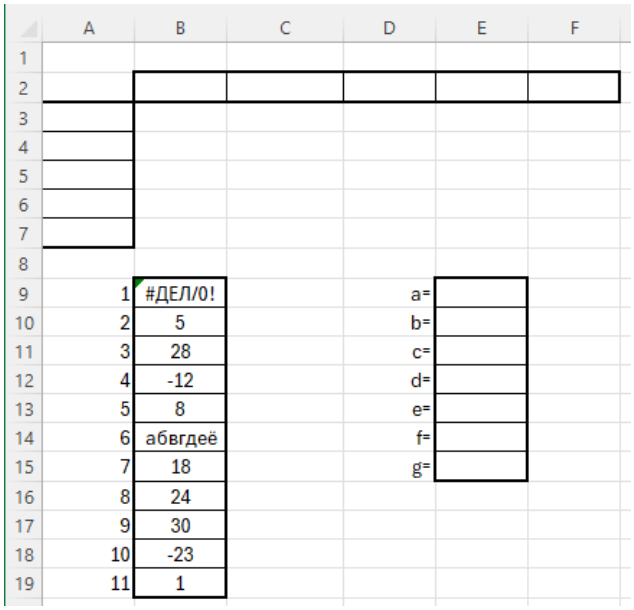
Разведчики

Нашими разведчиками на командный пункт были переданы места сосредоточения войск противника в виде горизонтальных и вертикальных координат, расположенных в ячейках B2:F2 и A3:A7. Определите их по формулам закодированной электронной таблицы, представленной в двух режимах, на рисунках расположенных ниже.



	A	B
9	1	=СУММ(B2:F2)/(A4-A6)
10	2	=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))
11	3	=A3*(A4+A6)
12	4	=A3-A4-A6
13	5	=ЕСЛИ(СУММ(A3:A7)>45;A3+A4;A6+A5)
14	6	=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))
15	7	=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10
16	8	=НАЙТИ(C2;"ABCDEFGHIJKLMNOPQ")-E11+E12
17	9	=НАЙТИ(D2;"ABCDEFGHIJKLMNOPQ")-E13+E14
18	10	=НАЙТИ(E2;"ABCDEFGHIJKLMNOPQ")-E14+E15
19	11	=НАЙТИ(F2;"ABCDEFGHIJKLMNOPQ")

	D	E
a=	=ABS(ОКРУГЛВНИЗ(-ПИ();0))	
b=	=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)	
c=	=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))	
d=	=ЕСЛИ(ЕОШ(B9);E9*E10;E10-E9)	
e=	=B10-E9	
f=	=B10^2+E15	
g=	1	



	A	B
9	1	#ДЕЛ/0!
10	2	5
11	3	28
12	4	-12
13	5	8
14	6	абвгдеё
15	7	18
16	8	24
17	9	30
18	10	-23
19	11	1

	D	E
a=		
b=		
c=		
d=		
e=		
f=		
g=		

Ответ оформите в виде двух строк, в верхней строке запишите через запятую значения ячеек B2:F2, а в нижней строке также через запятую значения ячеек A3:A7. В случае нескольких вариантов ответов запишите их через союз «или».

Решение

Начнем с определения коэффициентов a, b, c, d, e, f и g.

Значение коэффициента a (ячейка E9) в соответствии с формулой «=ABS(ОКРУГЛВНИЗ(-ПИ();0))» равно 3.

Значение коэффициента b (ячейка E10) в соответствии с формулой «=ЧАСТНОЕ(10;3)*2^1-ОСТАТ(8;6)» равно 4.

Значение коэффициента c (ячейка E11) в соответствии с формулой «=ABS(ОКРВВЕРХ.МАТ(-EXP(1)))» равно 2.

Значение коэффициента d (ячейка E12) в соответствии с формулой «=ЕСЛИ(ЕОШ(B9);E9*E10;E10-E9)» равно 12.

Значение коэффициента e (ячейка E13) в соответствии с формулой «=B10-E9» равно 2.

Значение коэффициента f (ячейка E14) в соответствии с формулой «=B10^2+E15» равно 26.

Значение коэффициента g (ячейка E15) равно 1.

Получаем:

	D	E
8		
9	a=	3
10	b=	4
11	c=	2
12	d=	12
13	e=	2
14	f=	26
15	g=	1

1) Так как значение формулы в ячейке B9 «=СУММ(B2:F2)/(A4-A6)» определяется как ошибка деления на 0, то значения ячеек A4 и A6 одинаковы.

2) Значение формулы в ячейке B10 «=ДЛСТР(ТЕКСТ(A3;"#")&ТЕКСТ(A4;"#")&ТЕКСТ(A5;"#")&ТЕКСТ(A6;"#")&ТЕКСТ(A7;"#"))» равно 5, следовательно значения координат в ячейках A3:A7 состоят из одного символа.

3) Формулы в ячейках B11 «=A3*(A4+A6)» и B12 «=A3-A4-A6» рассмотрим, как систему уравнений, учитывая, что значения ячеек A4 и A6 одинаковы.

$$\begin{cases} A3 * (A4 + A6) = 28 \\ A3 - A4 - A6 = -12 \end{cases} \Rightarrow \begin{cases} A3 * 2A4 = 28 \\ A3 - 2A4 = -12 \end{cases}$$

Решая систему уравнений, получаем два набора значений для ячеек A3 и A4 (A3 = 2 и A4 = 7) либо (A3 = -14 и A4 = -1). Так как, в соответствии со вторым пунктом, значения ячеек A3:A7 состоят из одного символа, а это диапазон от 0 до 9, то A3 = 2, A4 = A6 = 7.

4) Значение формулы в ячейке B13 «=ЕСЛИ(СУММ(A3:A7)>45; A3+A4;A6+A5)» равно 8. Максимальное значение формулы СУММ(A3:A7) из-за того, что значения в этих ячейках не превышают 9, равно 45. В соответствие с этим, условие в формуле ячейки B13 не выполнится, следовательно A6 + A5 = 8 и A5 = 8 - 7 = 1.

5) Значение формулы в ячейке B14 «=ЕСЛИ((A5-A7)<0;ЛЕВСИМВ("абвгдеёжзи";A7-A5);ЛЕВСИМВ("абвгдеёжзи";A5-A7))» равно «абвгдеё». Следовательно разница между ячейками A7 и A5 равна 7, откуда A7 равно -6 или 8. По условию 0<=A7<=9, A7 = 8.

6) Значение формулы в ячейке B15 «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")-E9+E10» равно 18. Откуда, значение формулы «=НАЙТИ(B2;"ABCDEFGHIJKLMNOPQ")» = 18 + E9 – E10 = 18 + 3 - 4 = 17, следовательно B2 = «Q».

7) Значения ячеек C2, D2, E2 и F2 аналогичным образом находим из формул в ячейках B16, B17, B18 и B19. Получаем C2 = «N», D2 = «F», E2 = «B» и F2 = «A».

Ответ: **Q, N, F, B, A**
2, 7, 1, 7, 8

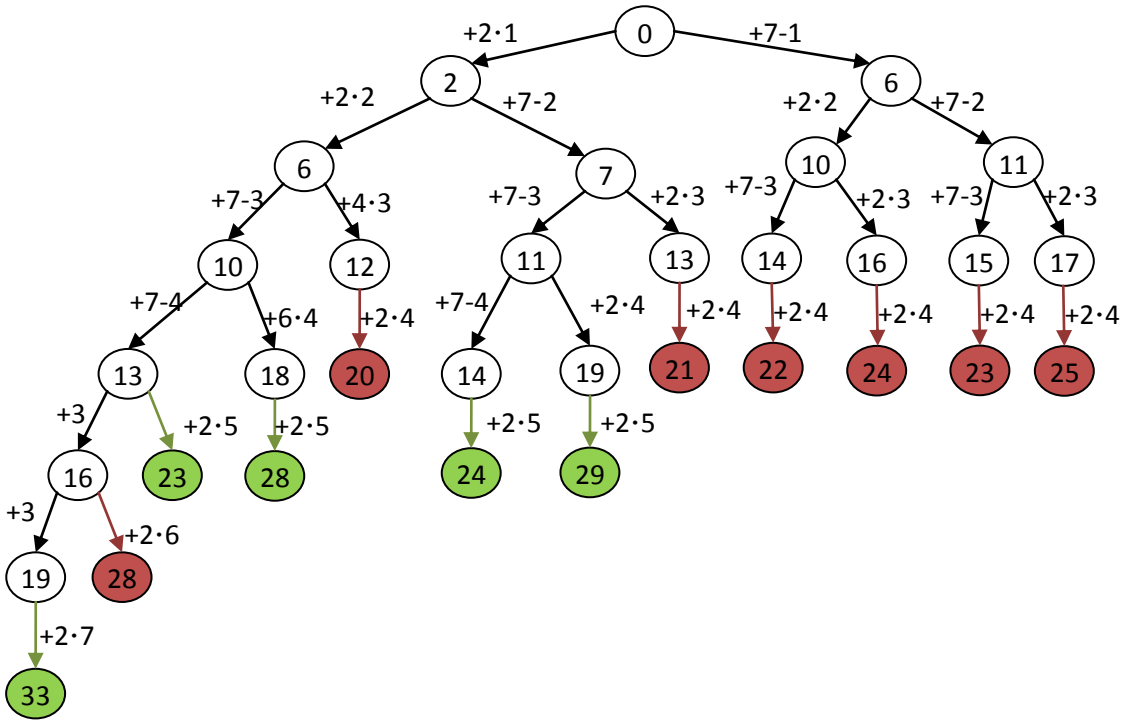
Задача 2 (10 баллов)

Карточная схватка

Два азартных игрока решили сыграть в следующую игру. По очереди они складывают игральные карты в колоду. За один ход можно положить в колоду количество карт, определяемое либо формулой семь минус номер хода в игре, но не менее трех, либо формулой две умноженные на номер хода в игре. При превышении количества 20 карт в колоде, игра заканчивается. Кто первым, сделав ход, получит в колоде 20 карт или больше – считается победителем. Кто выигрывает при безошибочной игре – игрок, делающий первый ход, или игрок, делающий второй ход? Каково минимальное число ходов до победы, если выигрывающий игрок пытается закончить игру за меньшее число ходов, а проигрывающий пытается максимально задержать свое поражение? Какое максимальное число карт может оказаться в колоде в конце игры? Какой игрок при этом победит? Поясните алгоритм графически, используя ориентированные графы в виде деревьев, с обозначением выигрышных и проигрышных позиций как вершин, а действий игроков – дугами графа с указанием численных изменений.

Решение

Построим ориентированный граф, где зеленым цветом выделены выигрышные позиции первого игрока, красным цветом – выигрышные позиции второго игрока, а номер соответствует количеству карт в колоде. Рядом со стрелкой указано изменение числа карт в колоде на этом ходу. В случае, если один из вариантов хода приводит к победе, то второй вариант не отображается, за исключением ветви, которая демонстрирует игру с максимально возможным числом пачек на полке в конце игры.



Из графа видно, что при любой игре второго игрока, гарантированно победит первый игрок на 5 ходу, если первым ходом положит 2, а третьим 10 или 11 карт. Максимально возможное число карт можно получить, если увеличивать количество пачек на полке таким образом, чтобы получить как можно более высокий номер хода (7) и количества пачек на полке (19) перед последним ходом, в котором берется вариант с большим значением (+2·7). При этом побеждает первый работник с количеством карт 33.

Ответ: гарантированно победит первый игрок за 5 ходов, максимум 33 карты при победе первого работника.

Задача 3 (10 баллов)

Логическая головоломка: разгадка функции F

Паша решил усовершенствовать задание ЕГЭ по информатике.

Логическая функция задается выражением

$$(x \vee y) \equiv (w \equiv z) \rightarrow (\neg y \rightarrow w)$$

Ниже приведен фрагмент таблицы истинности функции F, содержащий неповторяющиеся строки. Однако вместо значений, Паша решил вписать тоже несколько логических выражений, результат которых надо определить, если **a, b, c** – следующие высказывания:

a – «Земля вращается вокруг Солнца»

b – «7x9=72»

c – «В русском алфавите 33 буквы»

Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных **w,x,y,z.**

?	?	z	?	F
		a∧b∨c		0
	b→c∨a	(c≡a)→b		0
				0
		a∧(¬b∨c)	b→c∧a	0

В ответе напишите буквы **w,x,y,z** в том порядке, в котором идут соответствующие им столбцы.

Решение

Сперва определим значения логических высказываний. Очевидно, что

a=1
b=0
c=1

Теперь определим значения логических выражений в таблице. В результате получим следующую таблицу:

?	**?**	**z**	**?**	**F**
		1		0
	1	0		0
				0
		1	1	0

Следующим шагом будет составление таблицы истинности для функции F

w	x	y	z	xVy	w≡z	¬y	¬y→w	w≡z→(¬y→w)	F
0	0	0	0	0	1	1	0	0	1
0	0	0	1	0	0	1	0	1	0
0	0	1	0	1	1	0	1	1	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	1	1	0	0	0
0	1	0	1	1	0	1	0	1	1
0	1	1	0	1	1	0	1	1	1
0	1	1	1	1	0	0	1	1	1
1	0	0	0	0	0	1	1	1	0
1	0	0	1	0	1	1	1	1	0
1	0	1	0	1	0	0	1	1	1
1	0	1	1	1	1	0	1	1	1
1	1	0	0	1	0	1	1	1	1
1	1	0	1	1	1	1	1	1	1
1	1	1	0	1	0	0	1	1	1
1	1	1	1	1	1	0	1	1	1

Выберем из нее строки, где функция F = 0.

w	x	y	z	F
0	0	0	1	0
0	1	0	0	0
1	0	0	0	0
1	0	0	1	0

Сравнивая эту таблицу с заданной в задании, и учитывая, что строки в этой таблице выписаны в случайном порядке, легко установить с помощью несложных рассуждений искомые наименования столбцов

y	x	z	w	F
0	0	1	0	0
0	1	0	0	0
0	0	0	1	0
0	0	1	1	0

Ответ: y,x,z,w

Задача 4 (20 баллов)

Зашифрованный Рудник

В горнодобывающей компании «Рудник-Х» используется система шифрования для передачи информации о наименовании полезного ископаемого и параметрах пласта с данным ценным компонентом (длина, ширина), которая была получена при проведении геологической разведки месторождений полезных ископаемых. Передача информации осуществляется геологами с места проведения полевых работ в административное управление компании.

Процесс шифрования в горнодобывающей компании заключается в следующем:

1) Генерируется супервозрастающая последовательность чисел $A = \{a_1, a_2, \dots a_k\}$, каждый член которой больше суммы всех предыдущих членов, т.е. при $j = 2,3, \dots k: a_j > \sum_{i=1}^{j-1} a_i$, которая является секретным ключом.

Например, секретный ключ: $A = \{a_1, a_2, a_3, a_4\} = \{2, 3, 9, 17\}$,
где $a_2 > a_1 \rightarrow 3 > 2$;
$a_3 > (a_1 + a_2) \rightarrow 9 > 5$;
$a_4 > (a_1 + a_2 + a_3) \rightarrow 17 > 14$.

2) После этого выбирается число m большее, чем сумма чисел последовательности, т.е. $m > \sum_{i=1}^{k} a_i$. Также выбирается число n взаимно простое с m, т.е. наибольший общий делитель m и n равен 1.

3) Далее создается открытый ключ, т.е. новая последовательность чисел $B = \{b_1, b_2, \dots b_k\}$ путем умножения каждого числа a_i на n и взятия остатка от деления данного произведения на m.

Например, для открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ каждый член последовательности вычисляется по формуле $b_i = (a_i \cdot n) \bmod m$,
где a_i — i-й элемент последовательности секретного ключа A;
n — число взаимно простое с m;
m — модуль операции взятия остатка, иными словами, делитель;
mod — операция взятия остатка от деления.
При $n = 91$, $m = 320$ и $A = \{2, 3, 9, 17\}$:
$b_1 = (a_1 \cdot n)\ mod\ m = (2 \cdot 91)\ mod\ 320 = 182$ или, иными словами, $(2 \cdot 91)$ делится на 320 с остатком 182.
Сделав аналогичные вычисления для остальных элементов, получается последовательность $B = \{182,\ 273, 179, 267\}$

4) Полученная последовательность B используется для шифрования отправляемого сообщения. Отправляемое сообщение, представленное в виде двоичной строки, определяет, какие элементы этой последовательности суммируются для получения последовательности шифротекста $C = \{c_1, c_2, \dots c_p\}$, которая передается получателю.

Например, при шифровании числа 10 последовательность шифротекста C будет вычисляться следующим образом:
1) Число 10 представляется в двоичной системе счисления, т.е. $10 = 1010_2$;
2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1 и b_3 суммируются для получения шифротекста, b_2 и b_4 – опускаются.
3) Шифротекст $C = \{c_1\} = \{b_1 + b_3\} = 182 + 179 = 361$.
Например, при шифровании числа E_{16} последовательность шифротекста C будет вычисляться следующим образом:
1) Число E_{16} представляется в двоичной системе счисления, т.е. $E_{16} = 1110_2$;
2) «1» и «0» определяют, какие элементы открытого ключа $B = \{b_1, b_2, b_3, b_4\}$ суммируются, т.е. элементы b_1, b_2 и b_3 суммируются для получения шифротекста, b_4 – опускается.
3) Шифротекст $C = \{c_1\} = \{b_1 + b_2 + b_3\} = 182 + 273 + 179 = 634$.

Для дешифрирования сообщения получатель, исходя из известного ему n, определяет n^{-1} – мультипликативное обратное к значению n по модулю m (остаток от деления $n \cdot n^{-1}$ на m равен 1).

Иными словами, $(n \cdot n^{-1}) \bmod m = 1$,
где n — заданное число взаимно простое с m;
m — модуль операции взятия остатка;
mod — операция взятия остатка от деления.

Далее вычисляются промежуточные значения последовательности C' путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

Например, для шифротекста $C = \{c_1\} = 361$, каждый член последовательности C' вычисляется по формуле $c'_i = (c_i \cdot n^{-1}) \bmod m$,
где c_i — i-й элемент последовательности шифротекста C;
n^{-1} — мультипликативное обратное;
m — модуль операции взятия остатка;
mod — операция взятия остатка от деления.
В данном случае, последовательность
$C' = \{c'_1\} = (361 \cdot 211) \bmod 320 = 11$, где 211 – мультипликативное обратное n^{-1}, рассчитанное получателем.

Далее получатель определяет двоичную строку M путем последовательного сравнения каждого элемента последовательности C' с элементами секретного ключа A справа налево: если $c'_i \geq a_i$, то $M_i = 1$ и осуществляется вычитание a_i из c'_i, в противном случае $M_i = 0$. Таким образом восстанавливается зашифрованное сообщение.

Например, для последовательности $C' = \{c'_1\}$ и секретного ключа $A = \{a_1, a_2, a_3, a_4\}$ осуществляется последовательное сравнение элементов справа налево:
1) Если $c'_1 \geq a_4$, то $M_4 = 1$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число $c'_1 - a_4$;
2) Если $c'_1 < a_4$, то $M_4 = 0$ и для следующего сравнения с элементом последовательности A, т.е. с a_3, берется число c'_1.
3) В ответе получается двоичная строка $M = \{M_1, M_2, M_3, M_4\}$.

Например, при $A = \{2, 3, 9, 17\}$ и $C' = \{11\}$ двоичная строка $M = \{1, 0, 1, 0\}$, что согласуется с двоичным представлением числа 10.

Для последовательности C', состоящей из нескольких элементов, например, $C' = \{c'_1, c'_2, \dots c'_k\}$ осуществляется последовательное сравнение каждого элемента $c'_1, c'_2, \dots c'_k$ с элементами секретного ключа A с получением количества двоичных строк, равного количеству элементов последовательности C'.

1) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4\}$ количество двоичных строк M равно 4.

2) Для последовательности $C' = \{c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7, c'_8\}$ количество двоичных строк M равно 8.

Геологи передали четыре последовательных сообщения в административное управление о двух месторождениях полезных ископаемых. Первое сообщение в каждой паре содержало наименование полезного ископаемого. Второе сообщение содержало информацию о параметрах пласта, при этом первое число соответствовало длине пласта, второе – ширине пласта. Для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы согласно таблице ASCII для Windows-1251, далее полученному шестнадцатеричному числовому значению присваивалась соответствующая двоичная тетрада. Для шифрования числа осуществлялся его перевод в двоичную систему. Таблица ASCII для Windows-1251:

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Были получены следующие зашифрованные сообщения:

1 сообщение: 927 767 958 1075 1009 1505

2 сообщение: 1222 951

3 сообщение: 1092 1009 1505 767 1317

4 сообщение: 1391 1463

Секретный ключ, хранящийся в организации, представляет собой следующую последовательность {2, 4, 7, 14, 28, 59, 118, 233}, значения n и m равны 59 и 470 соответственно.

Расшифруйте полученные сообщения. В ответе укажите последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» – «Длина», «Ширина» (в десятичной системе). Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Решение

Сначала необходимо вычислить промежуточные значения последовательности C' для осуществления дешифровки. В исходных данных сказано, что данные значения вычисляются путем умножения каждого числа c_i на n^{-1} и взятия остатка от деления на m.

В условии дано только значение n, однако мы знаем, что n^{-1} – мультипликативное обратное к значению n по модулю m, то есть остаток от деления $n \times n^{-1}$ на m равен 1, поэтому для нахождения значения n^{-1} можно применить расширенный алгоритм Евклида.

Расширенный алгоритм Евклида является модификацией алгоритма Евклида, вычисляющая кроме наибольшего общего делителя (НОД) целых чисел a и b, еще и коэффициенты соотношения Безу, то есть такие x и y, что $ax + by = \text{НОД}(a, b)$. В свою очередь, алгоритм Евклида заключается в последовательном делении с остатком для нахождения наибольшего общего делителя двух чисел. На каждом этапе производится деление большего из пары чисел на меньшее, а остаток записывается и используется при дальнейших вычислениях в качестве нового делителя для предыдущего. Вычисление остатка производят до тех пор, пока он не будет равен нулю. Тогда последний использованный делитель и будет являться наибольшим общим делителем, полученным через алгоритм Евклида. В нашем случае наибольший общий делитель чисел n и m (59 и 470) должен равняться 1.

Применяем алгоритм Евклида:

$$470 = 59 \times 7 + 57 \Longrightarrow \quad 59 = 57 \times 1 + 2 \Longrightarrow \quad 57 = 2 \times 28 + 1 \Longrightarrow \quad 2 = 2 \times 1 + 0$$

Применяем расширенный алгоритм Евклида. Нам нужно выразить 1 как линейную комбинацию 59 и 470, т. е. найти целые числа x и y, такие, что $59x + 470y = 1$

Из алгоритма Евклида выше:

$$1 = 57 - 28 \times 2 \Longleftarrow \quad 2 = 59 - 57 \times 1 \Longleftarrow \quad 57 = 470 - 59 \times 7$$

Подставим обратно:

$$1 = 57 - (59 - 57 \times 1) \times 28 = 57 \times 29 - 59 \times 28$$
$$1 = (470 - 59 \times 7) \times 29 - 59 \times 28 = 470 \times 29 - 59 \times 231$$
$$1 = 470 \times 29 - 59 \times 231$$

Мы получили, что $1 = 470 \times 29 - 59 \times 231$, следовательно, $59 \times (-231) - 470 \times 29 = 1$. Это означает, что -231 является обратным числом 59 по модулю 470. Поскольку нам нужно положительное целое число, мы прибавляем 470 к -231: $-231 + 470 = 239$. Получаем, что $n^{-1} = 239$.

Зная значение n^{-1} находим промежуточные значения последовательности C' для каждого полученного сообщения. Для первого сообщения – 927 767 958 1075 1009 1505:

1) (927×239) делится на 470 с остатком 183

2) (767×239) делится на 470 с остатком 13

3) (958×239) делится на 470 с остатком 72

4) (1075×239) делится на 470 с остатком 305

5) (1009×239) делится на 470 с остатком 41

6) (1505×239) делится на 470 с остатком 145

Промежуточные значения последовательности $\{183, 13, 72, 305, 41, 145\}$.

Для второго сообщения – 1222 951:

1) (1222×239) делится на 470 с остатком 188

2) (951×239) делится на 470 с остатком 279

Промежуточные значения последовательности $\{188, 279\}$.

Для третьего сообщения – 1092 1009 1505 767 1317:

1) (1092×239) делится на 470 с остатком 138

2) (1009×239) делится на 470 с остатком 41

3) (1505×239) делится на 470 с остатком 145

4) (767×239) делится на 470 с остатком 13

5) (1317×239) делится на 470 с остатком 333

Промежуточные значения последовательности $\{138, 41, 145, 13, 333\}$.

Для четвертого сообщения – 1391 1463:

1) (1391×239) делится на 470 с остатком 159

2) (1463×239) делится на 470 с остатком 447

Промежуточные значения последовательности $\{159, 447\}$.

Далее используя секретный ключ A, получатель определяет битовую строку M путем последовательного сравнения C' с элементами a_i. Секретный ключ, исходя из исходных условий, представляет собой последовательность $\{2, 4, 7, 14, 28, 59, 118, 233\}$.

Для первого сообщения:

1) $183 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{matrix}$

2) $13 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

3) $72 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{matrix}$

4) $305 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{matrix}$

5) $41 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{matrix}$

6) $145 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{matrix}$

Для второго сообщения:

1) $188 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{matrix}$

2) $279 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{matrix}$

Для третьего сообщения:

1) $138 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{matrix}$

2) $41 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{matrix}$

3) $145 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{matrix}$

4) $13 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

5) $33 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{matrix}$

Для четвертого сообщения:

1) $159 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{matrix}$

2) $447 = \begin{matrix} 2 & 4 & 7 & 14 & 28 & 59 & 118 & 233 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{matrix}$

Из условий задачи известно, что для шифрования каждой букве текстового сообщения присваивался код шестнадцатеричной системы, далее каждой шестнадцатеричной цифре соответствующая двоичная тетрада; для шифрования числа осуществлялся его перевод в двоичную систему. Для дешифровки необходимо произвести данные действия в обратном порядке.

Сначала дешифруем текстовые сообщения. Для этого воспользуемся таблицей тетрад и далее таблицей ASCII для Windows-1251 из условий задачи.

16-ричная цифра	Двоичная тетрада	16-ричная цифра	Двоичная тетрада
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

Таким образом, произведем дешифровку текстовых сообщений:

Первое сообщение:

1) $927 = 11000110 = \begin{matrix}1100 & \text{C}\\ 0110 & 6\end{matrix} = \text{Ж}$

2) $767 = 11100000 = \begin{matrix}1110 & \text{E}\\ 0000 & 0\end{matrix} = \text{а}$

3) $958 = 11100100 = \begin{matrix}1110 & \text{E}\\ 0100 & 4\end{matrix} = \text{д}$

4) $1075 = 11100101 = \begin{matrix}1110 & \text{E}\\ 0101 & 5\end{matrix} = \text{е}$

5) $1009 = 11101000 = \begin{matrix}1110 & \text{E}\\ 1000 & 8\end{matrix} = \text{и}$

6) $1505 = 11110010 = \begin{matrix}1111 & \text{F}\\ 0010 & 2\end{matrix} = \text{т}$

Третье сообщение:

1) $1092 = 11010010 = \begin{matrix}1101 & \text{D}\\ 0010 & 2\end{matrix} = \text{Т}$

2) $1009 = 11101000 = \begin{matrix}1110 & \text{E}\\ 1000 & 8\end{matrix} = \text{и}$

3) $1505 = 11110010 = \begin{matrix}1111 & \text{F}\\ 0010 & 2\end{matrix} = \text{т}$

4) $767 = 11100000 = \begin{matrix}1110 & \text{E}\\ 0000 & 0\end{matrix} = \text{а}$

5) $1317 = 11101101 = \begin{matrix}1110 & \text{E}\\ 1101 & \text{D}\end{matrix} = \text{н}$

Дешифруем числовые сообщения: для этого осуществим перевод найденной битовой последовательности в десятичную систему счисления. Чтобы перевести число из двоичной системы счисления в десятичную, нужно: справа налево пронумеровать цифры в двоичном числе, начиная с 0, далее слева направо умножить каждую цифру двоичного числа на 2 в степени номера этой цифры и сложить полученные значения.

Число в 2-ой системе счисления	Перевод в 10-ую систему счисления	Число в 10-ой системе счисления
Второе сообщение		
01100110_2	$1 \times 2^6 + 1 \times 2^5 + 1 \times 2^2 + 1 \times 2^1 = 64 + 32 + 4 + 2 = 102$	102_{10}
01011001_2	$1 \times 2^6 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^0 = 64 + 16 + 8 + 1 = 89$	89_{10}
Четвертое сообщение		
11101010_2	$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^1 = 128 + 64 + 32 + 8 + 2 = 234$	234_{10}
10101111_2	$1 \times 2^7 + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 128 + 32 + 8 + 4 + 2 + 1 = 175$	175_{10}

В ответе необходимо указать последовательность четырех полученных сообщений в формате «Наименование полезного ископаемого» - «Длина», «Ширина» (числа в десятичной системе). Следовательно, ответ задачи выглядит следующим образом: Жадеит – 102, 89, Титан – 234, 175.

Ответ: Жадеит – 102, 89, Титан – 234, 175

Для получения максимального количества баллов за задачу необходимо написать программу для нахождения величины n^{-1}.

Python

```
def find_pairs(m):
    pairs = []
    for n in range(m):
        for nn in range(m):
            if (n * nn) % m == 1:
                pairs.append((n, nn))
    return pairs

m = int(input("Enter the value of m: "))
pairs = find_pairs(m)
for n, nn in pairs:
    print(f"({n}, {nn})")
```

Задача 5 (25 баллов)

Буквенные прямоугольники

Вениамин играет в новую интеллектуальную игру с ИИ. У них есть прямоугольная таблица размера **n** × **m**, в клетки которой они по очереди выбирают и ставят по одной из букв 'a', 'b', 'c', 'd' или 'e'. После того, как все ячейки будут заполнены, ИИ честно подсчитает количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами. Если оно чётно, выиграет Вениамин, иначе выиграет ИИ.

Осталось сделать последний ход, и этот ход делает Вениамин. Подскажите ему, какие из этих букв он может поставить, чтобы выиграть.

Формат входных данных

В первой строке содержатся два натуральных числа **n** и **m** через пробел – размеры игрового поля. **1 ≤ n, m ≤ 500.**

В следующих **n** строках содержится описание текущего заполнения таблицы. Каждая строка состоит из **m** символов, каждый из них – это буква от 'a' до 'e'. Помимо этого, в таблице содержится ровно один символ '.', который обозначает пустую клетку, в которую нужно сделать заключительный ход.

Формат выходных данных

Вывести в одну строку без пробелов список букв, которые может поставить в последнюю свободную клетку Вениамин и выиграть. Буквы не нужно разделять пробелом и требуется выводить в алфавитном порядке. Буквы должны быть из интервала от 'a' до 'e'. Если нет ни одной выигрывающей буквы, вывести сообщение «AI win» без кавычек.

Примеры

стандартный ввод	стандартный вывод
3 4 aadd a.bb eebd	Bcd
1 1 .	AI win

Комментарий к примерам

Исходно, из букв 'a' можно составить 5 прямоугольников, из букв 'b' – 5 прямоугольников, из букв 'c' – ни одного, из букв 'd' – 4 прямоугольника и из букв 'e' – 3 прямоугольника.

Если поставить букву 'a' получим 9 прямоугольников из этой буквы и 12 остальных, то есть нечётное количество, выиграет ИИ.

Если поставить букву 'b' получим 8 прямоугольников из этой буквы и 12 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'c' получим 1 прямоугольник из этой буквы и 17 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'd' получим 5 прямоугольников из этой буквы и 13 остальных, то есть чётное количество, выиграет Вениамин.

Если поставить букву 'e' получим 5 прямоугольников из этой буквы и 14 остальных, то есть нечётное количество, тогда выиграет ИИ.

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных. Результат выполнения программы для указанных исходных данных должен содержать: итоговый вывод программы и количество различных прямоугольников при подстановке каждой буквы вместо точки.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
3 3 d.d dbb ebb	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Решение

Прежде чем перейти к решению, важно понимать структуру данных и подходы, которые можно использовать. У нас имеется двумерный массив (матрица) символов, и мы хотим подсчитать количество прямоугольных областей, состоящих из одного и того же символа. Важно отметить, что прямоугольник определяется как область, в которой все символы одинаковы и находятся на любых позициях в матрице, которые могут образовать прямоугольник.

Элементарное решение требует шестерного цикла: перебираем левый

верхний угол двумя циклами, внутри правый нижний еще двумя и далее перебираем всю внутреннюю часть выбранного прямоугольника чтобы проверить, что она состоит из одной и той же буквы. Получим правильное, но не эффективное решение.

Основная идея эффективного решения заключается в применении алгоритма, основанного на динамическом программировании. Вот шаги, которые мы будем выполнять:

Создание вспомогательной матрицы

Мы создаем переменную d, которая будет хранить высоту столбцов для каждого символа в матрице. Эта матрица позволяет нам отслеживать, сколько последовательных символов одного и того же типа находится над каждой позицией в матрице. Если текущий элемент такой же, как элемент над ним, мы увеличиваем счетчик.

Инициализация

Мы заполняем первую строку матрицы d единицами, поскольку в ней нет элементов, находящихся выше.

Обработка матрицы

Затем мы проходим по всей матрице, заполняя d в зависимости от значений над текущей ячейкой. Если символ текущей ячейки такой же, как символ над ней, мы увеличиваем значение в d, иначе устанавливаем его в 1.

Подсчет прямоугольников

После того как матрица d заполнена, мы начинаем считывать высоту каждого столбца (начиная с нижней строки). Для каждого символа мы находим максимальную высоту, начиная с нижней строки и проводя подсчет, количества символов. Если символы отличаются, мы обнуляем счетчик. Если символ того же типа, то увеличиваем счетчик и добавляем его в общий результат.

Получение результата

После того как мы обработали всю матрицу, результатом будет общее количество найденных прямоугольников, которое мы возвращаем.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Вспомогательная матрица для хранения высот символов
vector<vector<int>> d;

// Функция для подсчета прямоугольников для символа c, позиция (posx, posy)
int F(int n, int m, vector<string> &v, char c, int posx, int posy) {
    d.resize(n, vector<int>(m, 0)); // Инициализация матрицы высот

    v[posx][posy] = c; // Установка буквы в пустую ячейку
```

```
    // Заполнение матрицы высот
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (i == 0) {
                d[i][j] = 1; // Первая строка всегда имеет высоту 1
            } else {
                d[i][j] = (v[i - 1][j] == v[i][j]) ? d[i - 1][j] + 1 : 1;
              // Высота для текущей ячейки
            }
        }
    }

    // Подсчет прямоугольников
    int res = 0;
    for (int i = 0; i < n; i++) {
        // Проход по строкам
        for (int j = i; j >= 0; j--) {
            // Проход по подстрокам для каждой строки
            int t = 0; // Количество текущих прямоугольников
            if (d[i][0] >= i - j + 1) { // Проверка первой колонки
                t = 1; // Находим хотя бы один прямоугольник
            }
            res += t; // Добавляем в общий счетчик

            for (int k = 1; k < m; k++) { // Проход по столбцам
                if (d[i][k] < i - j + 1) {
                    t = 0;
                    // Сбрасываем счетчик, если высота меньше текущей
                    continue;
                }
                if (v[j][k] == v[j][k - 1]) {
                   // Если текущий символ такой же, как предыдущий
                    t++; // Увеличиваем количество прямоугольников
                } else {
                    t = 1; // Обнуляем счетчик
                }
                res += t; // Добавляем к общему результату
            }
        }
    }

    return res; // Возвращаем общее количество прямоугольников
}

signed main() {
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    // Оптимизация ввода-вывода

    int n, m;
    cin >> n >> m; // Чтение размеров массива
    int posx, posy;
    vector<string> v(n);

    // Чтение поля и нахождение позиции пустой ячейки
    for (int i = 0; i < n; i++) {
        cin >> v[i];
        for (int j = 0; j < m; j++) {
            if (v[i][j] == '.') {
                posx = i; // Запоминаем координаты пустой ячейки
                posy = j;
            }
```

```
        }
    }

    bool ok = false; // Флаг для отслеживания возможности выигрыша
    for (char c = 'a'; c <= 'e'; c++) { // Перебор букв от 'a' до 'e'
        int z = F(n, m, v, c, posx, posy);
// Подсчет прямоугольников для каждой буквы

        if (z % 2 == 0) { // Если количество четное
            ok = true; // Вениамин может выиграть
            cout << c; // Вывод выигрывающей буквы
        }
    }

    if (!ok) { // Если ни одна буква не подходит
        cout << "AI win"; // ИИ выигрывает
    }
}
```

Реализация на Python

```
def count_rectangles(n, m, grid, char, posx, posy):
    # Создаем массив d для хранения высот
    d = [[0] * m for _ in range(n)]

    # Заполняем пустую позицию выбранным символом
    grid[posx][posy] = char

    # Вычисляем высоты столбцов для каждого символа
    for i in range(n):
        for j in range(m):
            if i == 0:
                d[i][j] = 1
            else:
                if grid[i - 1][j] == grid[i][j]:
                    d[i][j] = d[i - 1][j] + 1
                else:
                    d[i][j] = 1

    # Подсчет прямоугольников
    res = 0

    for i in range(n):
        for j in range(i, -1, -1):
            t = 0
            if d[i][0] >= i - j + 1:
                t = 1

            res += t

            for k in range(1, m):
                if d[i][k] < i - j + 1:
                    t = 0
                    continue

                if grid[j][k] == grid[j][k - 1]:
                    t += 1
                    res += t
                    continue

                t = 1
```

```
                res += t

    return res

def main():
    # Чтение входных данных
    n, m = map(int, input().split())
    grid = [input().strip() for _ in range(n)]

    # Нахождение позиции пустой клетки
    posx, posy = -1, -1
    for i in range(n):
        for j in range(m):
            if grid[i][j] == '.':
                posx, posy = i, j
                break
        if posx != -1:
            break

    winning_chars = []

    # Проверка символов от 'a' до 'e'
    for char in 'abcde':
        rectangle_count = count_rectangles(n, m, \
          [list(row) for row in grid], char, posx, posy)
        if rectangle_count % 2 == 0:  # Проверка четности
            winning_chars.append(char)

    # Вывод результата
    if winning_chars:
        print(''.join(winning_chars))
    else:
        print("AI win")

if __name__ == "__main__":
    main()
```

Основной шаг алгоритма включает обход всех ячеек матрицы, чтобы определить высоту каждого столбца для каждого символа. Вы должны пройти по всем элементам матрицы и обновить вспомогательную матрицу d. Этот процесс также занимает O(m×n) времени, так как каждый элемент матрицы посещается один раз. После того, как высоты столбцов определены, необходимо подсчитать количество возможных прямоугольников. Для этого вам нужно будет обработать каждую строку матрицы, а затем для каждой строки пройтись по ее высотам, чтобы вычислить возможные прямоугольники, основываясь на текущих высотах. Общее время на этом этапе также останется в пределах O(m×n).

Таким образом, учитывая все шаги алгоритма, общая временная сложность будет составлять O(m×n), где m — количество строк, а n — количество столбцов. Динамическое программирование является ключом к оптимизации решений для задач, связанных с подсчетом и комбинаторикой, что позволяет избежать избыточного перебора всех возможных комбинаций.

В решении используется вспомогательная матрица d для хранения высоты столбцов для каждого символа. Если исходная матрица имеет размер m×n, то для хранения этой вспомогательной матрицы нам

потребуется O(m×n) памяти. Это означает, что по памяти сложность равна также O(m×n). В дополнение к матрице d, могут потребоваться несколько дополнительных переменных для хранения промежуточных результатов, таких как счетчики или индексы. Однако количество этих переменных не зависит от размера входных данных и является константным, что не влияет существенно на общую сложность по памяти.

Результат выполнения алгоритма для указанных исходных данных

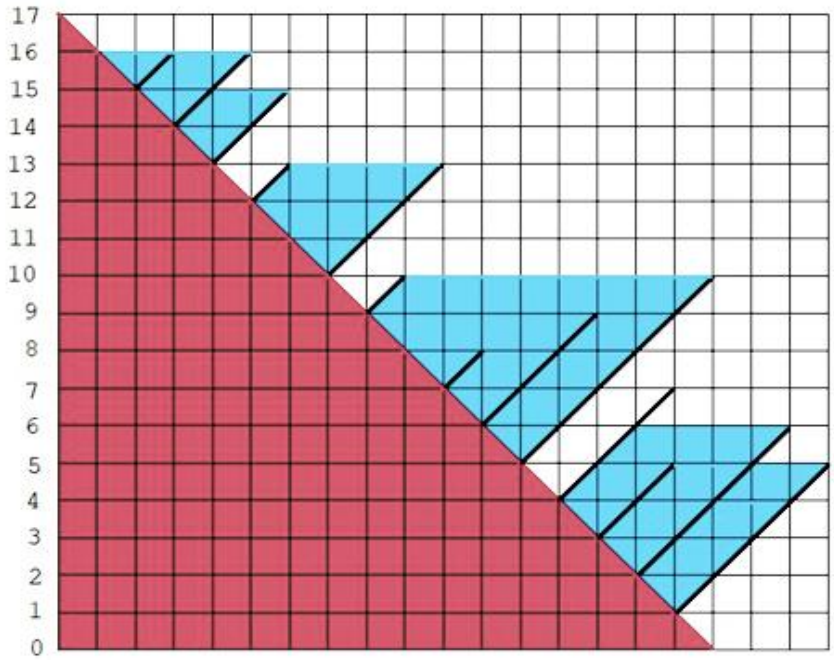
стандартный ввод	стандартный вывод
3 3 d.d dbb ebb	D

Количество различных прямоугольников из ячеек таблицы, заполненных одинаковыми буквами, при подстановке вместо точки для каждой буквы: a – 15, b – 17, c – 15, d – 18, e – 15.

Задача 6 (25 баллов)

Противолавинные заграждения

На одном горном курорте на склоне построили систему противолавинных заграждений. Склон имеет угол 45° к горизонтали. На склоне в некоторых целых по высоте от подножия горы точках расположены пояса заграждений. Каждый пояс расположен строго перпендикулярно склону. Рассмотрим двумерное сечение этого сооружения (на рисунке).



Склон представлен в виде темного треугольника, а каждое заграждение – в виде отрезка, перпендикулярного склону. Снег, скатываясь сверху, попадает в некоторые отсеки, образуемые заграждениями, и застревает в них, постепенно заполняя эти отсеки. Так как снега очень много, каждый отсек заполняется до того момента, пока снег не забьет его до самого верха (образуется горизонтальный уровень заполнения). После этого снег начинает высыпаться из отсека и попадать в нижние отсеки. Некоторые заграждения настолько велики, что, при их заполнении оказываются заполненными и некоторые более вышерасположенные отсеки, у которых граница расположена ниже горизонтали заполнения. В тоже время в некоторые заграждения снег наоборот может не попасть из-за того, что при переполнении верхнего отсека, он вертикально падает вниз, минуя текущий отсек, так как вертикаль падения проходит мимо него.

Будем считать, что если граница заграждения находится ровно на вертикали падения либо горизонтали заполнения других отсеков, то из них снег в этот отсек не попадает. Например, заграждение на высоте 12 из примера пусто, так как его окончание и по вертикали и по горизонтали

находится на соответствующих прямых. В отсек, образуемый заграждением на высоте 3, снег из верхнего отсека непосредственно не попадает, но так как его высота меньше горизонтали заполнения отсека заграждения на высоте 2, то снег его всё-таки заполнит.

По заданным высотам и длинам заграждений требуется оценить максимальную задерживающую способность этих заграждений. Так как мы работаем с двумерной моделью, то требуется найти площадь сечения, заполненную снегом при полном заполнении всей системы заграждений.

Формат входных данных

В первой строке находится число **n** – количество заграждений на склоне $\mathbf{1 \le n \le 3*10^5}$.

В следующих **n** строках находятся по два числа $\mathbf{h_i}$ и $\mathbf{t_i}$ – высота расположения **i**-го заграждения и его длина. Длина любого заграждения равна целому числу диагоналей единичного квадратика со стороной 1 метр. В этих диагоналях она и указывается $\mathbf{0 \le h_i \le 2 * 10^9}$, $\mathbf{1 \le t_i \le 10^6}$. Заграждения перечислены во входных данных в порядке возрастания высоты их нахождения на склоне.

Формат выходных данных

Вывести одно число – максимальную площадь сечения, занятую снегом при полном заполнении всех отсеков, в которые снег может попасть. Можно показать, что она всегда будет целым числом. Толщиной заграждений можно пренебречь. Считаем, что высота склона достаточна, чтобы любое заграждение наполнилось.

Примеры

стандартный ввод	стандартный вывод
13 1 4 2 4 3 2 4 3 5 5 6 3 7 1 9 1 10 3 12 1 13 2 14 2 15 1	58

Решите указанную задачу, используя один из языков программирования, и укажите, какой именно язык был выбран. Предложите эффективное решение, учитывая минимизацию времени выполнения

программы и объема памяти, необходимого для её работы. Постарайтесь использовать алгоритмы и структуры данных, которые обеспечивают оптимальную производительность.

Реализуйте решение в виде программного кода, обязательно добавив комментарии, которые поясняют ключевые шаги алгоритма и переменные, если это важно для понимания проверяющему. Допускается использование только стандартных библиотек, встроенных в язык. Подробно опишите алгоритм решения, предоставив текстовое описание или блок-схему, объяснив логику работы программы.

Приведите результат выполнения программы для заданных исходных данных.

Исходные данные для выполнения задания

стандартный ввод	стандартный вывод
5 2 4 3 4 6 3 7 5 11 2	?

Ваше решение должно быть аккуратно оформлено, читабельно и соответствовать стандартам выбранного языка программирования.

Для начала заметим, что если отсек высотой h ничем слева не ограничен, то площадь его сечения равна h^2. Далее, если отсек высоты h ограничен другим отсеком слева, то нужно провести горизонталь от отсека высоты h на этот ограничивающий и узнать высоту точки пересечения. Допустим, высота этой точки равна z. Тогда площадь заполнения нашего отсека будет равна $h^2 – z^2$. Это доказывает целочисленность ответа.

Основной операцией для этой задачи рассмотрим вертикальное и горизонтальное смещение заграждения. Например, вертикальное смещение: берем наше заграждение, фиксируем вертикальные линии сетки, которые его ограничивают и двигаем это заграждение по этим линиям вниз. Очевидно, в каждой целой точке склона это заграждение будут иметь все меньшую высоту, равную t.len - (t.h - v[i].h), (где t.len – исходная высота заграждения, t.h - v[i].h – смещение) пока не уйдет со склона в минус. Аналогично горизонтальное движение влево.

Теперь эффективно найдем те отсеки, в которые снег попадает (или не попадает) сверху. Добавим фиктивный отсек на высоте $2 * 10^9$ длиной 0, объявим его текущим. Далее перебираем заграждения в обратном порядке и смещаем текущее заграждение вертикально так, чтобы оно стало на диагональ рассматриваемого. Если текущее стало строго меньше рассматриваемого, то рассматриваемое становится текущим, в него снег попадает (отмечаем это 1 в соответствующей ячейке массива from_up), те отсеки, в которые снег сверху не попадает, остаются отмеченными 0.

Далее найдем ответ. Для этого добавим снизу фиктивный отсек на высоте -1 длиной 0, объявим его текущим. Перебираем заграждения снизу вверх (в порядке возрастания высоты крепления) и горизонтально сдвигаем текущее в точку рассматриваемого. Если текущее стало меньше либо равно 0, это означает, что его ничто не ограничивает слева и если в текущее попадает снег сверху, то добавим квадрат его высоты к ответу.

Если текущее больше 0, но меньше либо равно рассматриваемому, то (если в текущее попадает снег), добавим к ответу разницу соответствующих квадратов, текущим сделаем рассматриваемое.

Если текущее строго больше рассматриваемого в точке, то рассматриваемое никак не влияет на заполнение и будет заполнено снегом, если в текущем этот снег есть. Такой случай обрабатывать никак не нужно.

Реализация на C++

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

// Структура, представляющая заграждение
struct bord {
    int h;    // высота заграждения
    int len;  // длина заграждения
};

signed main() {
    // Оптимизация ввода-вывода
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n;
    cin >> n; // Считываем количество заграждений
    vector<bord> v(n + 1); // Вектор для хранения заграждений
    v[0] = {-1, 0}; // Фиктивное заграждение с высотой -1 и длиной 0
    for (int i = 1; i <= n; i++) {
        cin >> v[i].h >> v[i].len; // Считываем высоту и длину заграждений
    }
    v.push_back({(int)2e9, 0}); // Фиктивное заграждение

    bord t = v.back(); // Последнее заграждение
    int last; // Поддерживаем индекс последнего заграждения
    vector<int> from_up(n + 1, 0); // Массив о попадании снега
    // Идем по заграждениям снизу вверх, определяя может ли быть заполнено
    for (int i = n; i >= 1; i--) {
        int tlen = t.len - (t.h - v[i].h);
        if (tlen < v[i].len) {
            from_up[i] = 1; // Текущее заграждение может быть заполнено
            t = v[i]; // Обновляем текущее заграждение
        }
    }
    t = v[0]; // Сбрасываем текущее заграждение на первое
    last = 0; // Обнуляем индекс последнего заграждения

    int ans = 0; // Переменная для хранения площади, занятой снегом
    //Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
     for (int i = 1; i <= n + 1; i++) {
        int tlen = t.len - (v[i].h - t.h);
```

```
        if (tlen <= 0) {
            if (from_up[last]) {
                ans += v[last].len * v[last].len;
            }
            t = v[i];
            last = i;
            continue;
        }
        if (tlen <= v[i].len) {
            if (from_up[last]) {
                ans += (v[last].len * v[last].len - tlen * tlen);
            }
            t = v[i];
            last = i;
        }
    }
    cout << ans << endl; // Выводим результирующую площадь, занятую снегом
}
```

Реализация на Python

```
class Bord:
    def __init__(self, h, length):
        self.h = h
        self.len = length

def main():
    n = int(input().split())
    v = [Bord(-1, 0)]  # Добавляем фиктивный отсек на высоте -1
    for i in range(1, n + 1):
        h, t = map(int, input().split())
        v.append(Bord(h, t))
    v.append(Bord(2 * 10**9, 0))  # Добавляем отсек на высоте 2 * 10^9
    from_up = [0] * (n + 1) # Массив о попадании снега сверху
    t = v[-1]
# Идем по заграждениям снизу вверх
    for i in range(n, 0, -1):
        tlen = t.len - (t.h - v[i].h)
        if tlen < v[i].len:
            from_up[i] = 1
            t = v[i]

    t = v[0]
    last = 0
    ans = 0
    # Проход по заграждениям снизу вверх, чтобы находить площадь заполнения
    for i in range(1, n + 2):  # +2, из-за фиктивного отсека
        tlen = t.len - (v[i].h - t.h)
        if tlen <= 0:
            if from_up[last]:
                ans += v[last].len * v[last].len
            t = v[i]
            last = i
            continue

        if tlen <= v[i].len:
            if from_up[last]:
                ans += (v[last].len * v[last].len - tlen * tlen)
            t = v[i]
            last = i
```

```
    print(ans)

if __name__ == "__main__":
    main()
```

Программа в основном состоит из нескольких проходов по массиву v, который содержит заграждения. Первый проход — мы считываем данные из файла и заполняем массив v объектами класса Bord. Этот проход происходит в одном цикле от 1 до n, где n — количество заграждений. Таким образом, временная сложность этого шага составляет O(n). Второй проход — во время этого прохода мы идем в обратном направлении по массиву v, чтобы определить, может ли снег попадать сверху. Это также O(n), поскольку мы проходим по всем элементам массива по одному разу. Третий проход — здесь мы снова проходим по массиву v от начала до конца, анализируя пространство между заграждениями и высоту заполнения. Это также O(n).

Таким образом, суммируя все проходы, мы получаем, что общая временная сложность программы составляет O(n).

Мы создаем массив v, длиной n+2, который хранит объекты класса Bord. Таким образом, память, занимаемая этим массивом, составляет O(n). Массив from_up также имеет размер n+1, что добавляет еще O(n) к пространственной сложности. Другие переменные, такие как t, last, и ans, занимают постоянное количество памяти, которое не зависит от размера входных данных. Таким образом, их вклад в общую пространственную сложность можно считать константным: O(1).

Итак, собрав все элементы вместе, можно заключить, что общая пространственная сложность программы также составляет O(n).

Результат выполнения алгоритма для указанных исходных данных

стандартный ввод	стандартный вывод
5 2 4 3 4 6 3 7 5 11 2	**50**