

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**



**ОЛИМПИАДА ШКОЛЬНИКОВ
«ГРАНИТ НАУКИ»**

ИНФОРМАТИКА

**Методические указания для подготовки к заключительному туру олимпиады
в 2025/2026 учебном году
для 11 классов**

Санкт-Петербург, 2026



СОДЕРЖАНИЕ

Правила проведения и содержание олимпиадных заданий.....	3
Задания для подготовки к олимпиаде	7
Список рекомендованной литературы и других источников для подготовки	21



Правила проведения и содержание олимпиадных заданий

Характер и уровень сложности олимпиадных задач направлены на достижение целей проведения олимпиады: выявить способных участников, твердо владеющих школьной программой и наиболее подготовленных к освоению образовательных программ технических ВУЗов, обладающих логикой и творческим характером мышления, умеющих алгоритмически описать реальные ситуации из различных предметных областей и применить к ним наиболее подходящие методы информатики.

Задания дифференцированы по сложности и требуют различных временных затрат на верное и полное решение. Они охватывают все разделы школьной программы, но носят, в большинстве, комплексный характер, позволяющий варьировать оценки в зависимости от проявленных в решении творческих подходов и продемонстрированных технических навыков. Участники должны самостоятельно разбить задачу на подзадачи, грамотно выполнить решение каждой подзадачи, синтезировать решение всей задачи из решений отдельных подзадач. Успешное выполнение олимпиадной работы не предполагает знаний, выходящих за пределы школьной программы, но требует творческого подхода, логического мышления, умения увидеть и составить правильный и оптимальный план решения, четкого и технически грамотного выполнения каждой части решения.

Очный тур олимпиады проводится только в письменной форме. Каждый участник олимпиады получает билет, содержащий шесть заданий. При выполнении заданий требуется дать теоретическое обоснование и привести все этапы решения задачи:

- при определении результата перевода заданного числа из одной системы счисления в другую следует указать правило перевода в общем виде, а затем привести схему перевода заданного числа в указанную систему счисления;

- при определении значений ячеек электронной таблицы приводятся результаты, формулы, полученные при копировании, даются пояснения о типах ссылок в формулах, как типы ссылок влияют на результат и др.;

- при ответе на вопрос об определении значения переменной после выполнения программы, алгоритм которой приводится в билете, следует пояснить работу каждого оператора (блока), выписать промежуточные и окончательный результаты;

- в случае решения задач, связанных с таблицами истинности необходимо описывать все промежуточные результаты;

- при решении задачи на программирование необходимо составить блок-схему или описать алгоритм решения задачи, написать текст программы на любом из языков программирования, указав используемый язык программирования и его версию, допускается использовать только стандартную библиотеку языка. Алгоритм должен раскрывать решение на каждом шаге. Запрещено использовать уникальные особенности языка программирования (классы, методы, объекты, специальные функции и библиотеки и др.) для реализации решения. Считывание заранее рассчитанных значений не засчитывается за решение задачи. Код должен содержать достаточное для понимания логики работы количество комментариев;

- программы, составленные участниками, тестируются и всесторонне оцениваются с учетом быстродействия, рациональности, минимального использования памяти компьютера и оригинальности предложенных решений;

- при использовании методов научного перебора необходимо привести доказательства использования этого метода;

При подготовке к олимпиаде следует повторить приведенные ниже темы.



Теоретические основы информатики

Информация, данные и знания. Универсальность дискретного представления информации. Двоичное кодирование. Равномерные и неравномерные коды. Подходы к измерению информации. Основы криптографии: шифры подстановки и перестановки, симметричные и асимметричные методы. Принципы открытого и закрытого ключей. Модульная арифметика и остатки по модулю. Алгоритм RSA на базовом уровне: генерация ключей, шифрование и расшифрование, роль больших простых чисел. Практические примеры использования и вопросы безопасности на интуитивном уровне.

Системы счисления. Развёрнутая запись целых и дробных чисел в позиционных системах счисления. Свойства позиционной записи числа: количество цифр в записи, признак делимости числа на основание системы счисления. Алгоритм перевода целого числа из P -ичной системы счисления в десятичную. Алгоритм перевода конечной P -ичной дроби в десятичную. Алгоритм перевода целого числа из десятичной системы счисления в P -ичную. Двоичная, восьмеричная и шестнадцатеричная системы счисления; перевод чисел между этими системами. Арифметические операции в позиционных системах счисления.

Представление целых и вещественных чисел в памяти компьютера.

Кодирование текстов. Кодировка ASCII. Однобайтные кодировки. Стандарт UNICODE. Кодировка UTF-8. Определение информационного объёма текстовых сообщений.

Кодирование изображений. Оценка информационного объёма растрового графического изображения при заданном разрешении и глубине кодирования цвета.

Кодирование звука. Оценка информационного объёма звуковых данных при заданных частоте дискретизации и разрядности кодирования.

Алгебра логики. Высказывания. Логические операции: «и» (конъюнкция, логическое умножение, $\wedge$), «или» (дизъюнкция, логическое сложение, $\vee$), «не» (логическое отрицание, $\neg$), «сложение по модулю два» (исключающее или, $\oplus$). Логические выражения. Вычисление логического значения составного высказывания при известных значениях входящих в него элементарных высказываний. Таблицы истинности логических выражений. Логические операции и операции над множествами.

Примеры законов алгебры логики. Эквивалентные преобразования логических выражений. Решение простейших логических уравнений. Логические функции. Построение логического выражения с данной таблицей истинности. Нормальные формы: дизъюнктивная и конъюнктивная нормальные формы.

Логические элементы компьютера. Триггер. Сумматор. Построение схемы на логических элементах по логическому выражению. Запись логического выражения по логической схеме.

Модели и моделирование. Цели моделирования. Адекватность модели моделируемому объекту или процессу. Формализация прикладных задач.

Представление результатов моделирования в виде, удобном для восприятия человеком. Графическое представление данных (схемы, таблицы, графики).

Графы. Основные понятия. Виды графов. Решение алгоритмических задач, связанных с анализом графов (построение оптимального пути между вершинами графа; определение количества различных путей между вершинами ориентированного ациклического графа).

Деревья. Бинарное дерево. Дискретные игры двух игроков с полной информацией. Построение дерева перебора вариантов; описание стратегии игры в табличной форме.



Выигрышные стратегии.

Использование графов и деревьев при описании объектов и процессов окружающего мира.

Алгоритмы и программирование

Определение возможных результатов работы простейших алгоритмов управления исполнителями и вычислительных алгоритмов. Определение исходных данных, при которых алгоритм может дать требуемый результат.

Этапы решения задач на компьютере. Язык программирования (Паскаль, Python, Java, C++, C#). Основные конструкции языка программирования. Типы данных: целочисленные, вещественные, символьные, логические. Ветвления. Составные условия. Циклы с условием. Циклы по переменной. Использование таблиц трассировки.

Разработка и программная реализация алгоритмов решения типовых задач базового уровня. Примеры задач: алгоритмы обработки конечной числовой последовательности (вычисление сумм, произведений, количества элементов с заданными свойствами); алгоритмы анализа записи чисел в позиционной системе счисления; алгоритмы решения задач методом перебора (поиск наибольшего общего делителя двух натуральных чисел, проверка числа на простоту).

Обработка символьных данных. Встроенные функции языка программирования для обработки символьных строк. Алгоритмы редактирования текстов (замена символа/фрагмента, удаление и вставка символа/фрагмента, поиск вхождения заданного образца).

Табличные величины (массивы). Понятие о двумерных массивах (матрицах). Алгоритмы работы с элементами массива с однократным просмотром массива: суммирование элементов массива; подсчёт количества (суммы) элементов массива, удовлетворяющих заданному условию; нахождение наибольшего (наименьшего) значения элементов массива; нахождение второго по величине наибольшего (наименьшего) значения; линейный поиск элемента; перестановка элементов массива в обратном порядке.

Сортировка одномерного массива. Методы сортировки (метод пузырька, метод выбора, сортировка вставками, сортировка подсчетом), применение встроенных сортировок.

Подпрограммы. Рекурсивные алгоритмы.

Целочисленная арифметика. Арифметические операции (умножение, деление, остатки, сложение, вычитание). Битовые операции и работа с отдельными битами.

Теоретико-числовые алгоритмы. НОД и НОК, системы счисления, длинная арифметика, простые числа и разложение на делители, остатки, быстрое возведение в степень.

Алгоритмы поиска. Линейный поиск, двоичный поиск, поиск подстроки в строке, два указателя.

Динамическое программирование. Рекуррентные последовательности, простое динамическое программирование. Динамическое программирование с несколькими параметрами, по подстрокам, по подмножествам, по профилю, по поддеревьям, на ациклических графах.

Жадный алгоритм. Области применения и стандартные задачи, решаемые жадным алгоритмом. Доказательство применимости.

Алгоритмы на невзвешенных графах. Обход в ширину и глубину и их применение.



Топологическая сортировка, компоненты связности, поиск циклов, проверка на двудольность, мосты, точки сочленения, конденсация. Паросочетания. Эйлеров цикл.

Алгоритмы на взвешенных графах. Поиск кратчайших путей: алгоритмы Дейкстры, Беллмана-Форда, Флойда. Минимальные остовные деревья. Потоки.

Вычислительная геометрия. Скалярное и косое произведение. Площади. Взаимное расположение фигур на плоскости и в пространстве. Выпуклые оболочки.

Линейные структуры данных. Стек, дек, очередь. Решение задачи о проверки правильной скобочной последовательности, минимум в окне, обратная польская нотация.

Деревья. Бинарное дерево поиска. Сбалансированность бинарных деревьев поиска. Корневые деревья, система непересекающихся множеств. Дерево отрезков, решение задач RMQ и RSQ. Куча. Дерево Фенвика. Декартово дерево.

Сложность вычисления: количество выполненных операций, размер используемой памяти; зависимость количества операций от размера исходных данных.



Задания для подготовки к олимпиаде

Задача 1. Исследование поведения формул в Excel

С тех пор, как Иван изучил формулы в Excel, он ежедневно оттачивает свое мастерство. Сегодня он записал число 2 в ячейку A1, в соседнюю ячейку B1 он ввел формулу `=ОКРУГЛ(ОСТАТ(8;$A$1)+СТЕПЕНЬ(A1;2)-ЕСЛИ(A1>0;A1^2/3;A1/3);1)` и растянул до ячейки F1. На следующей строке, начиная с ячейки A2 Иван ввел формулу `=ОКРУГЛ(ОСТАТ(B1;A1)-КОРЕНЬ(ABS(A1-B1))+ЕСЛИ(A1+B1<2;B1/A1;A1+3);2)` и скопировал в диапазон B2:F2. На следующей строке он ввел в первую ячейку число 2,5 и повторил действия, как для первой строки. А в четвертую (диапазон A4:F4) скопировал формулу из 2 строки. В пятой строке Иван ввел в первую ячейку число 3 и снова решил повторить действия, как и для строки 1. В ячейку A6 он вставил формулу из A2 и растянул до ячейки F6. После этих действий Иван решил воспользоваться инструментом условного форматирования, выставив следующие условия:

- значения больше \$D\$1 выделить красным цветом;
- значения меньше \$C\$3 выделить желтым цветом;
- значения между 10 и 16 выделить зеленым цветом.

Укажите, какое количество ячеек оказалось выделено желтым цветом?

Ответ: 15

Решение

	A	B	C	D	E	F
1	2	2,7	4,9	16	170,7	19425,7
2	4,86	6,42	5,87	17,26	171,54	19289,32
3	2,5	4,2	11,8	92,8	5741,2	21974252
4	5,9	7,84	16	101,04	3736,34	21969567
5	3	6	24	384	98304	6,44E+09
6	4,27	4,76	8,03	74,08	18042,73	6,44E+09

Фрагмент в режиме отображения формул:

	A	B
1	2	<code>=ОКРУГЛ(ОСТАТ(8;\$A\$1)+СТЕПЕНЬ(A1;2)-ЕСЛИ(A1>0;A1^2/3;A1/3);1)</code>
2	<code>=ОКРУГЛ(ОСТАТ(B1;A1)-КОРЕНЬ(ABS(A1-B1))+ЕСЛИ(A1+B1<2;B1/A1;A1+3);2)</code>	<code>=ОКРУГЛ(ОСТАТ(C1;B1)-КОРЕНЬ(ABS(B1-C1))+ЕСЛИ(B1+C1<2;C1/B1;B1*3);2)</code>
3	2,5	<code>=ОКРУГЛ(ОСТАТ(B;\$A\$1)+СТЕПЕНЬ(A3;2)-ЕСЛИ(A3>0;A3^2/3;A3/3);1)</code>
4	<code>=ОКРУГЛ(ОСТАТ(B3;A3)-КОРЕНЬ(ABS(A3-B3))+ЕСЛИ(A3+B3<2;B3/A3;A3+3);2)</code>	<code>=ОКРУГЛ(ОСТАТ(C3;B3)-КОРЕНЬ(ABS(B3-C3))+ЕСЛИ(B3+C3<2;C3/B3;B3*3);2)</code>
5	3	<code>=ОКРУГЛ(ОСТАТ(B;\$A\$1)+СТЕПЕНЬ(A5;2)-ЕСЛИ(A5>0;A5^2/3;A5/3);1)</code>
6	<code>=ОКРУГЛ(ОСТАТ(B5;A5)-КОРЕНЬ(ABS(A5-B5))+ЕСЛИ(A5+B5<2;B5/A5;A5+3);2)</code>	<code>=ОКРУГЛ(ОСТАТ(C5;B5)-КОРЕНЬ(ABS(B5-C5))+ЕСЛИ(B5+C5<2;C5/B5;B5*3);2)</code>
7		
8		

**Задача 2. Вскрытие сейфа**

Для получения секретной информации нашим разведчикам необходимо вскрыть сейф, замок которого состоит из 5 ячеек (B2:F2). Задача осложняется тем, что код можно ввести только один раз (метод подбора не поможет), в ячейках может находиться как числовая, так и текстовая информация, и самое главное, формулы с подсказками из ячеек B4:B12 были скопированы в ячейки C5:C13, что привело к смещению ссылок.

Помогите разведчикам найти код и выиграть эту схватку. В Вашем распоряжении значения формул с подсказками в первоначальном состоянии (B4:B12) и формулы в режиме отображений формул после операции копирования (C5:C13).

	A	B	C	D	E	F	G	H	I	J
1										
2	Код:									
3										

	A	B
4	№1	0
5	№2	4
6	№3	3
7	№4	#ДЕЛ/0!
8	№5	#ЗНАЧ!
9	№6	дбор
10	№7	3
11	№8	4
12	№9	17
13		

	C
4	
5	=СЧЁТЕСЛИ(С3:G3;">=10")+СЧЁТЕСЛИ(С3:G3;"<=0")
6	=СЧЁТ(С3:G3)
7	=ДЛСТР(ЕСЛИ(ЕТЕКСТ(D3);D3;ТЕКСТ(D3;"#")))+ДЛСТР(ЕСЛИ(ЕТЕКСТ(E3);E3;ТЕКСТ(E3;"#")))
8	=СУММ(С3:E3)/(F\$2-G\$2)
9	=\$D\$2+F\$2+G\$2
10	=ПСТР("Подбор";\$F3;6)
11	=ПОИСК(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(D3);D3;"");ЕСЛИ(ЕТЕКСТ(E3);E3;"");"гипнотизер"))
12	=ДЛСТР(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(D3);D3;"");ЕСЛИ(ЕТЕКСТ(E3);E3;"")))+С3
13	=СУММ(С3:G3)

Решение

1) Находим первоначальное значение формул с учетом операции копирования.



	A	B
4	№1	=СЧЁТЕСЛИ(B2:F2;">=10")+СЧЁТЕСЛИ(B2:F2;"<=0")
5	№2	=СЧЁТ(B2:F2)
6	№3	=ДЛСТР(ЕСЛИ(ЕТЕКСТ(C2);C2;ТЕКСТ(C2;"#")))+ДЛСТР(ЕСЛИ(ЕТЕКСТ(D2);D2;ТЕКСТ(D2;"#")))
7	№4	=СУММ(B2:D2)/(E\$2-F\$2)
8	№5	=D\$2+E\$2+F\$2
9	№6	=ПСТР("Подбор";\$F2;6)
10	№7	=ПОИСК(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(C2);C2;"");ЕСЛИ(ЕТЕКСТ(D2);D2;"");"гипнотизер"))
11	№8	=ДЛСТР(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(C2);C2;"");ЕСЛИ(ЕТЕКСТ(D2);D2;"")))+B2
12	№9	=СУММ(B2:F2)

2) По формуле №1, находящейся в ячейке B4 «=СЧЁТЕСЛИ(B2:F2;">=10")+СЧЁТЕСЛИ(B2:F2;"<=0")» равной 0 определяем, что все числовые значения находятся в диапазоне от 1 до 9.

3) По формуле №2, находящейся в ячейке B5 «=СЧЁТ(B2:F2)» равной 4 определяем, что количество числовых значений равно 4, следовательно одна из ячеек содержит текстовую информацию.

4) Формула №3, находящаяся в ячейке B6 «=ДЛСТР(ЕСЛИ(ЕТЕКСТ(C2);C2;ТЕКСТ(C2;"#")))+ДЛСТР(ЕСЛИ(ЕТЕКСТ(D2);D2;ТЕКСТ(D2;"#"))))» равная 3 подсчитывает количество символов в ячейках C2 и D2, переводя числовые значения в текстовые. В соответствии с тем, что числовые значения находятся в диапазоне от 1 до 9, сумма количества символов, при условии нахождения в этих ячейках числовых значений, равнялась бы 2. Значение 3 говорит о том, что одна из ячеек имеет длину 2 символа и это – ячейка с текстовой информацией.

5) Значение формулы №4, находящейся в ячейке B7 «=СУММ(B2:D2)/(E\$2-F\$2)» является сообщением об ошибке деления на 0 «#ДЕЛ/0!», это возможно при условии, что значения в ячейках E2 и F2 одинаковы.

6) Значение формулы №5, находящейся в ячейке B8 «=D\$2+E\$2+F\$2» является сообщением об ошибке «#ЗНАЧ!» означающее, что значение используемое в формуле, имеет неправильный тип данных. Это возможно в том случае, если в этих ячейках находится текстовая информация. В соответствии с пунктом 4 данного решения мы знаем, что текст находится либо в ячейке C2 либо D2, следовательно ячейка с текстовой информацией – D2.

7) Значение формулы №6, находящейся в ячейке B9 «=ПСТР("Подбор";\$F2;6)» равно подстроке «дбор», откуда F2 равно 3, а значения в ячейках E2 и F2 одинаковы, в соответствии с пунктом 5 данного решения. Следовательно, E2=F2=3.

8) Формула №7, находящаяся в ячейке B10 «=ПОИСК(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(C2);C2;"");ЕСЛИ(ЕТЕКСТ(D2);D2;"");"гипнотизер"))» показывает, что наше текстовое поле имеется в слове «гипнотизер» начиная с третьей позиции. Зная, что его длина 2 символа, в соответствии с пунктом 4 решения, получаем значение ячейки D2 = «пн».

9) Формула №8, находящаяся в ячейке B11 «=ДЛСТР(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(C2);C2;"");ЕСЛИ(ЕТЕКСТ(D2);D2;"")))+B2» равная 4, позволяет найти значение ячейки B2, как разницы между значением формулы и длиной текстового поля ячейки D2. B2=4-2=2.

10) Формула №9, находящаяся в ячейке B12 «=СУММ(B2:F2)» позволяет найти значение последней неизвестной ячейки C2 = 17-2-3-3 = 9.



Ответ:

B2	C2	D2	E2	F2
2	9	пн	3	3

Задача 3. Каменная стратегия гномов

Два гнома решили сыграть в следующую игру. На столе кучка из 45 драгоценных камней. Гномы ходят по очереди. За один ход можно либо взять из кучки три камня, или меньше, если камней осталось меньше трех, либо уменьшить количество камней в кучке в два раза, округляя в большую сторону, если количество камней в кучке было нечетным. Кто первым на своем ходе достигнет нуля камней, побеждает. 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? Поясните алгоритм графически, используя ориентированные графы и обозначая проигрышные и выигрышные позиции.

Решение

Выигрывает тот гном, у которого в кучке останется 3 или меньше камней, следовательно, гном, который ходил предпоследним проиграет, если в кучке останется от 4 до $6=3+3$ камней. Позиции 4 – 6 камней в кучке являются проигрышными и победит тот гном, который сможет гарантированно привести своего оппонента в эти позиции. Следующие позиции 7 – 9 приведут в позиции 4 – 6 при убиении трех камней, а позиции 10 – 12 приведут в позиции 4 – 6 при уменьшении числа камней в два раза. Эти позиции являются выигрышными. Позиции 13 – 15 являются проигрышными, поскольку могут привести только в выигрышные позиции. Позиции 16 – 18 являются выигрышными поскольку могут гарантированно привести в позиции 13 – 15. Аналогичным образом позиции 19 – 21 являются проигрышными, а позиции 22 – 24, при убиении трех камней, и позиции 25 – 30, при уменьшении числа камней в два раза, являются выигрышными. Позиции 31 – 33 являются проигрышными, а позиции 34 – 36, при убиении трех камней, и позиции 37 – 42, при уменьшении числа камней в два раза, являются выигрышными. Наконец, позиции 43 – 45 являются проигрышными, следовательно, гном, который начинает ходить первым гарантированно приведет своего оппонента к выигрышу.

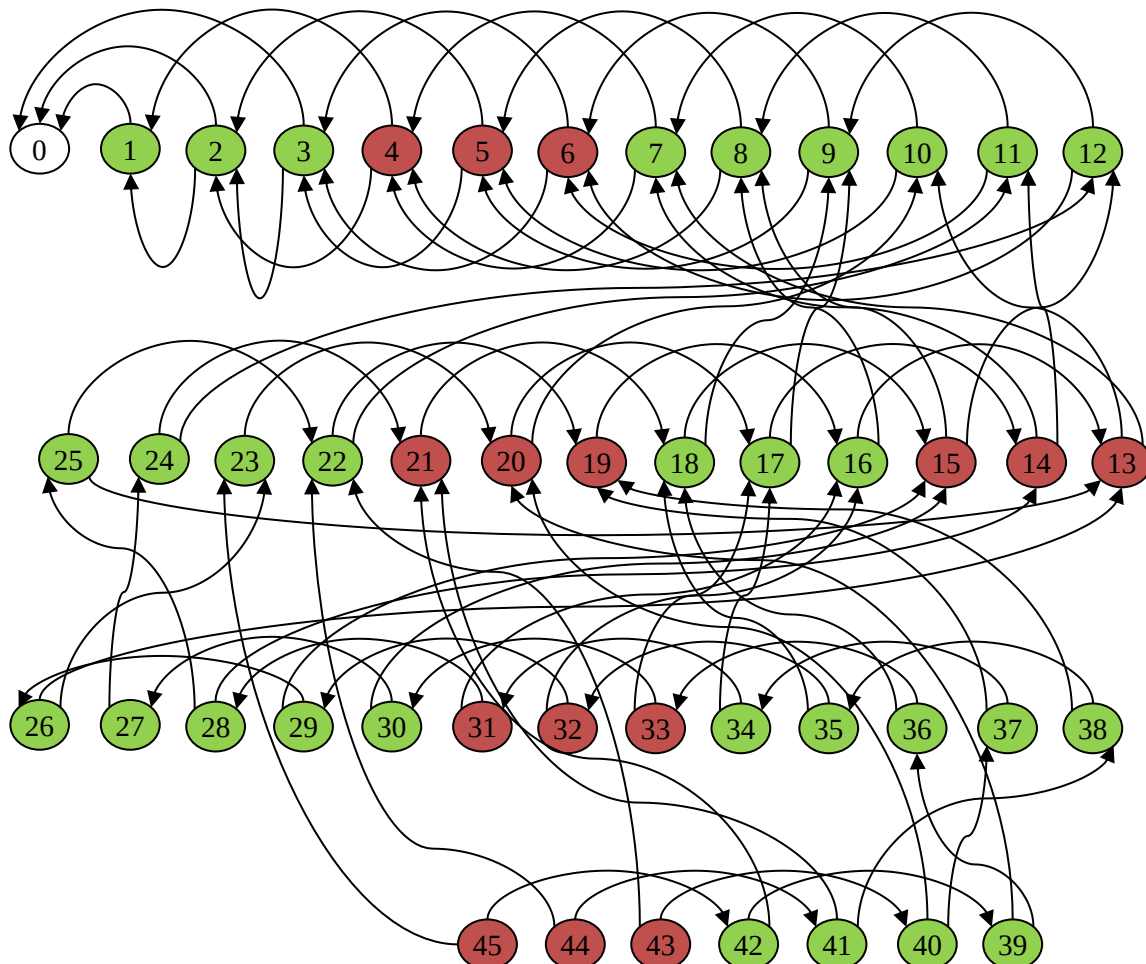
Если стратегия гнома, который ходит первым, заключается в затягивании игры, а второго гнома, в достижении скорейшей победы, то порядок ходов будет следующим:

- 1) Первый гном заберет из кучки три камня, в кучке останется $45 - 3 = 42$ камня;
- 2) Второй гном уменьшит количество камней в кучке в два раза, в кучке останется $42/2 = 21$ камень;
- 3) Первый гном заберет из кучки три камня, в кучке останется $21 - 3 = 18$ камней;
- 4) Второй гном заберет из кучки три камня, в кучке останется $18 - 3 = 15$ камней;
- 5) Первый гном заберет из кучки три камня, в кучке останется $15 - 3 = 12$ камней;
- 6) Второй гном уменьшит количество камней в кучке в два раза, в кучке останется $12/2 = 6$ камень;
- 7) Первый гном заберет из кучки три камня или уменьшит количество камней в кучке в два раза, в любом случае в кучке останется 3 камня;
- 8) Второй гном заберет из кучки три камня и станет победителем – больше камней в кучке не останется.

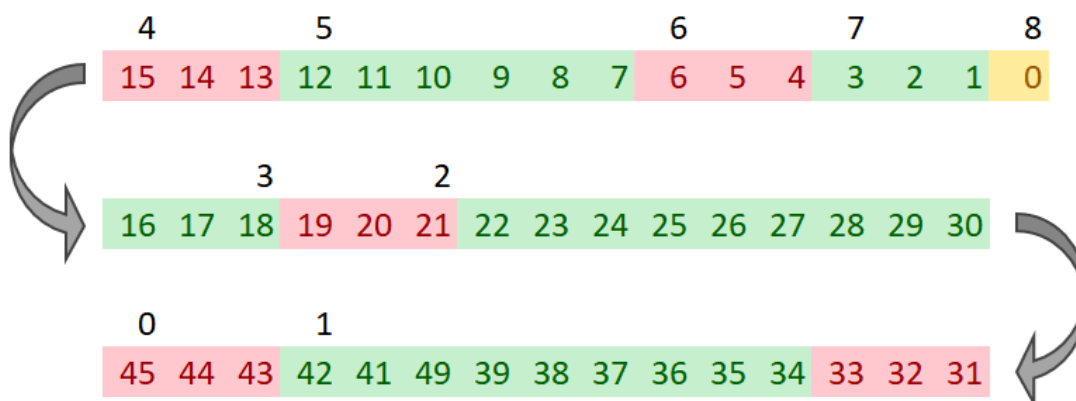


Вывод: гарантированно победит второй гном за 8 ходов.

Ориентированный граф, где зеленым цветом выделены выигрышные позиции, красным цветом проигрышные позиции, а номер соответствует количеству камней в кучке, выглядит следующим образом:



Упрощенный вид, где зеленым цветом выделены выигрышные позиции, красным цветом проигрышные позиции, номер ячейки соответствует количеству камней в кучке, а цифра над ячейкой соответствует ходу игры, выглядит следующим образом:



**Задача 4. Логическая функция**

Логическая функция F задается выражением:

$$F = x \wedge \neg z \equiv (y \wedge \neg z \vee \neg x \wedge y)$$

Ниже приведен фрагмент таблицы истинности функции F . Выписаны случайным образом строки таблицы, в которых значение функции F равняется 0. Вместо значений приведены логические выражения, которые предварительно надо вычислить. Переменные a, b, c представляют собой следующие логические высказывания:

a – «число 2025 – нечетное»

b – «2025 год – високосный»

c – «число 2025 делится на 125 без остатка»

Определите, какому столбцу таблицы истинности соответствует каждая из переменных x, y, z .

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

В ответе напишите буквы в том порядке, в котором идут соответствующие им столбцы.

Решение

Определим значения логических высказываний. Очевидно: $a=1$, $b=0$ и $c=0$.

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

?	?	?	F
0		1	0
	1		0
		1	0

Следующим шагом будет составление таблицы истинности для функции F

x	y	z	$\neg z$	$x \wedge \neg z$	$y \wedge \neg z$	$\neg x$	$\neg x \wedge y$	$y \wedge \neg z \vee \neg x \wedge y$	$x \wedge \neg z \equiv (y \wedge \neg z \vee \neg x \wedge y)$
0	0	0	1	0	0	1	0	0	1
0	0	1	0	0	0	1	0	0	1
0	1	0	1	0	1	1	1	1	0
0	1	1	0	0	0	1	1	1	0
1	0	0	1	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1
1	1	0	1	1	1	0	0	1	1
1	1	1	0	0	0	0	0	0	1



Выберем из нее строки, где функция $F = 0$.

x	y	z	$x \wedge \neg z \equiv (y \wedge \neg z \vee \neg x \wedge y)$
0	1	0	0
0	1	1	0
1	0	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	F
0	0	1	0
0	1	0	0
1	0	1	0

Ответ: zxy

Задача 5. Секретный агент

Агент посылает два сообщения пользователю А. Первое сообщение шифруется при помощи RSA-алгоритма, второе – при помощи неравномерного кода. В RSA-алгоритме открытый ключ агента $(e, n) = (5, 91)$. Для шифрования второго сообщения при помощи неравномерного кода у агента имеется таблица двоичных кодов для букв:

000 $\rightarrow$ n

001 $\rightarrow$ e

010 $\rightarrow$ d

Ответьте на вопросы:

1. Какое число получит пользователь А после шифрования агентом сообщения, содержащего значение 7?
2. Какое сообщение получится у пользователя А при декодировании полученной от агента последовательности 000001010 по указанной таблице?

Решение

1. Сообщение, шифруемое по RSA-алгоритму, вычисляется по формуле $C_i = M_i^e \bmod n$,

где C_i – i-й элемент последовательности шифротекста С;

M_i – i-й элемент последовательности шифра М;

e – открытая экспонента – параметр открытого ключа RSA – число, используемое при шифровании в качестве степени;

n – модуль операции взятия остатка;

mod – операция взятия остатка от деления.

В условии задачи шифр (отправляемое сообщение) состоит из одного элемента $M = 7$, открытый ключ агента $e = 5$, $n = 91$. Вычислим число (шифротекст), который получит пользователь А. Произведем процесс шифрования:

$$C = M^e \bmod n = 7^5 \bmod 91 = 16807 \bmod 91 = 63, \text{ т. к. } 91 \times 184 = 16744 \text{ (остаток 63).}$$



Следовательно, число, которое получит пользователь А после шифрования агентом сообщения, равно 63.

2. Дана таблица значений: $000 \rightarrow n$, $001 \rightarrow e$, $010 \rightarrow d$. Для расшифровки полученной от агента последовательности 000001010 необходимо прочитать биты последовательности слева направо и выделить получаемые коды по таблице:

1) Читаем биты: $0 \rightarrow 00 \rightarrow 000$. 000 есть в таблице? Да, это n. Результат: n.

2) Продолжаем со следующего бита: $0 \rightarrow 00 \rightarrow 001$. 001 есть в таблице? Да, это e. Результат: e.

3) Продолжаем со следующего бита: $0 \rightarrow 01 \rightarrow 010$. 010 есть в таблице? Да, это d. Результат: d.

Следовательно, сообщение, которое получится у пользователя А при декодировании полученной от агента последовательности, равно n e d.

Ответ: 1) 63, 2) n e d.

Задача 6. Шифрующая перестановка

Перестановка Р длины n – это упорядоченный набор, содержащий числа от 1 до n, каждое из которых входит в него ровно один раз. Например, перестановкой длины 13 является набор (5 11 13 12 6 1 8 4 10 9 7 2 3). При помощи перестановки букв можно зашифровать слово. Для примера возьмем приведенную выше перестановку и слово transposition, которое состоит тоже из 13 букв. Далее, следуя перестановке, на первую позицию поставим пятую букву слова, на вторую – одиннадцатую букву, ..., на последнюю позицию – третью букву слова. В итоге получим sinoptsniora. К этому слову снова применим эту же перестановку и получим poartsnoitsin. Повторив эти стадии шифрования k раз, получим зашифрованное сообщение.

t	r	a	n	s	p	o	s	i	t	i	o	n
1	2	3	4	5	6	7	8	9	10	11	12	13
5	11	13	12	6	1	8	4	10	9	7	2	3
s	i	n	o	p	t	s	n	t	i	o	r	a
1	2	3	4	5	6	7	8	9	10	11	12	13
5	11	13	12	6	1	8	4	10	9	7	2	3
p	o	a	r	t	s	n	o	i	t	s	i	n

Вам дано зашифрованное таким образом слово, шифрующая перестановка Р и число k. Необходимо восстановить слово.

Формат входных данных

Первая строка содержит два числа n и k – длину перестановки и число стадий шифрования ($1 \leq n, k \leq 1000$). Вторая строка содержит перестановку длины n, числа в ней разделены одним пробелом. Третья строка содержит зашифрованное слово, оно состоит из n прописных букв латиницы.

**Формат выходных данных**

Вывести одну строку – исходное слово.

Примеры

стандартный ввод	стандартный вывод
13 2 5 11 13 12 6 1 8 4 10 9 7 2 3 poartsnoitsin	transposition

Решение

Инициализация и ввод данных:

Сначала программа считывает количество символов в строке и количество итераций, за которыми следует вектор перестановок и сама строка.

Дешифрование:

Процесс дешифрования заключается в том, что за каждую итерацию создается новая копия строки, основанная на текущем состоянии строки *vs*. Символы из *vs* размещаются в строке *s* по индексам, указанным в векторе *p*. Обратите внимание, что индексы в векторе *p* начинаются с 1, поэтому нужно вычитать 1 при доступе к элементам.

Обновление строки:

После завершения всех перестановок на каждой итерации программа обновляет строку *vs* текущим состоянием *s*, таким образом, в следующем цикле будет использоваться уже изменённое состояние строки.

Вывод результата:

После того как все итерации завершены, финальная результирующая строка выводится на экран.

Сложность данного алгоритма можно проанализировать, исходя из структуры вложенных циклов. Внешний цикл выполняется *k* раз, где *k* – это количество повторений перестановки. Это означает, что алгоритм повторяет процесс перестановки строк *k* раз.

Внутренний цикл работает *n* раз, где *n* – длина строки. На каждой итерации внутреннего цикла происходит переопределение символов в строке *s* с учетом перестановки.

Таким образом, суммарная сложность алгоритма может быть выражена как: $O(k \times n)$.

C++

```
#include <bits/stdc++.h>
using namespace std;
// Объявление переменных для хранения длины строки, количества
перестановок, самой строки и временной строки.
int n, k;
string s, vs;
int main() {
    // Считываем длину строки (n) и количество применений перестановки (k).
    cin >> n >> k;
    // Создаем вектор p для хранения перестановки.
    vector<int> p(n);
    // Считываем саму перестановку.
    for(int i = 0; i < n; i++) {
        cin >> p[i];
    }
```




```
// Считываем строку, которую нужно дешифровать.
cin >> s;
// Изначально присваиваем vs значение входной строки s.
vs = s;
// Делаем k итераций применения перестановки.
for(int i = 0; i < k; i++) {
    // На каждой итерации создаем циклом новое зашифрованное слово.
    for(int j = 0; j < n; j++) {
        // Применяем перестановку: p[j] - 1 дает индекс для s.
        /* То есть берем символ из временной строки vs и помещаем его на
        нужное место в s.*/
        s[p[j] - 1] = vs[j];
    }
    /* После завершения внутреннего цикла обновляем vs значением
    зашифрованного слова s.*/
    vs = s;
}
// Выводим окончательную строку после всех применений перестановки.
cout << s;
return 0;
}
```

Python

```
def encrypt_with_permutation(n, k, permutation, s):
    vs = s # Сохраняем исходное слово
    for _ in range(k):
        # Создаем новый список символов для зашифрованного слова
        s_list = [''] * n
        for j in range(n):
            # Применяем перестановку: индексы учитываются с 0 (в Python)
            s_list[permutation[j] - 1] = vs[j]
        vs = ''.join(s_list) # Обновляем vs с зашифрованным словом
    return vs

if __name__ == "__main__":
    n, k = map(int, input().split())
    permutation = list(map(int, input().split()))
    s = input("")
    encrypted_message = encrypt_with_permutation(n, k, permutation, s)
    print(encrypted_message)
```

Задача 8. Фермер

Фермер хочет построить на своем участке как можно больший по площади сарай прямоугольной формы. Но на его участке есть деревья и хозяйственные постройки, которые он не хочет никуда переносить. Для простоты, зададим участок прямоугольной сеткой размера $M \times N$. Каждое из деревьев и построек размещается в одном или нескольких узлах сетки. Сарай должен быть построен на свободных узлах сетки.

Помогите фермеру определить максимально возможную площадь сарая.

Формат входных данных

В первой строке вводятся два натуральных числа N и M ($1 \leq N, M \leq 1000$) – размеры участка. Далее, следует N строк, в каждой из которых находится последовательность (без пробелов) из M нулей и единиц, описывающих ферму. Единицы соответствуют свободным для постройки участкам.

**Формат выходных данных**

Необходимо вывести максимально возможную площадь сарая, который может построить фермер на своем участке.

Примеры

стандартный ввод	стандартный вывод
5 10 1011011111 0111111110 1111111111 1011111111 1101110111	21
3 4 1011 1101 1111	4

Решение

Инициализация и ввод данных:

Сначала программа считывает размеры матрицы n и m , а затем каждую ячейку матрицы, которая может принимать значения '0' или '1'.

Подсчет высот:

Используется вспомогательный вектор `heights`, чтобы отслеживать высоты «столбцов» в текущей строке. При проходе по каждой строке матрицы, если элемент равен '1', увеличивается соответствующая высота столбца. Если элемент равен '0', высота столбца сбрасывается до нуля. Таким образом, на каждой итерации мы получаем высоты прямоугольников, которые заканчиваются на текущей строке.

Поиск максимальной площади:

Для нахождения максимальной площади на основе полученных высот используется подход, аналогичный решению задачи о самом большом прямоугольнике в гистограмме, рассмотренном в задаче 7.

Вывод результата:

После проверки всех строк код выводит максимальную найденную площадь на экран.

Хотя в приведенном коде традиционных методов динамического программирования, как, например, в методах с помощью сохранения промежуточных результатов, не встречается, аналогичное поведение присутствует. Применение `heights` фактически использует идею сохранения промежуточных состояний высот для строк, что важно для решения задачи эффективно.

Временная сложность алгоритма $O(n * m)$.

C++:

```
#include<bits/stdc++.h>

using namespace std;

int n, k;
int maxArea = 0;
```



```
int main() {
    int n, m;
    cin >> n >> m;
    vector<vector<int>> farm(n, vector<int>(m));

    // Чтение входных данных
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            char c;
            cin >> c; // Чтение символа '0' или '1'
            farm[i][j] = c - '0'; // Преобразование в целое число
        }
    }
    // Вспомогательный массив для хранения высот
    vector<int> heights(m, 0);

    // Проходим построчно по матрице
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (farm[i][j] == 1) {
                heights[j]++;
            } else {
                heights[j] = 0;
            }
        }
        // Находим максимальную площадь для данной линии
        stack<int> indices;
        for (int j = 0; j <= m; j++) {
            int h = (j == m) ? 0 : heights[j];
            while (!indices.empty() && heights[indices.top()] > h) {
                int height = heights[indices.top()];
                indices.pop();
                int width = indices.empty() ? j : j - indices.top() - 1;
                maxArea = max(maxArea, height * width);
            }
            indices.push(j);
        }
    }
    cout << maxArea << endl; // Вывод максимальной площади
    return 0;
}
```

Python:

```
n, m = map(int, input().split()) # Ввод размеров матрицы
farm = [list(input()) for _ in range(n)] # Ввод самой матрицы
max_area = 0 # Переменная для хранения максимальной площади
heights = [0] * m # Массив для хранения высот, инициализируем нулями
heights.append(0)
# Добавляем нуль, чтобы обработать все оставшиеся столбцы

# Проходим построчно по матрице
for i in range(n):
    for j in range(m):
        # Обновляем высоту для текущего столбца
        if farm[i][j] == '1':
            heights[j] += 1
        else:
            heights[j] = 0
```



```
# Находим максимальную площадь для данной линии

stack = [] # Стек для хранения индексов высот
for j in range(len(heights)):
    # Пока высота текущего столбца меньше высоты на вершине стека
    while stack and heights[stack[-1]] > heights[j]:
        h = heights[stack.pop()] # Высота прямоугольника
        width = j if not stack else j - stack[-1] - 1 # Ширина
        max_area = max(max_area, h * width)
    # Обновляем максимальную площадь
    stack.append(j) # Добавляем текущий индекс в стек

print(max_area)
```

Задача 8. Бинарный поиск единственного вхождения

Дано число $N \leq 10^5$, далее следуют N чисел A_i . Числа A_i приведены в порядке неубывания. Нумерация начинается с 1. Все числа A_i различны. Затем дано число $M \leq 10^5$, далее следуют M чисел B_j . Для каждого B_j бинарным поиском найти и выдать номер вхождения этого числа в последовательности A_i , либо определить, что его там нет.

Формат входных данных

В первой строке число $N \leq 10^5$. Во второй строке N чисел A_i в порядке неубывания. Все числа A_i различны. В третьей строке число $M \leq 10^5$. В четвертой строке M чисел B_j . Все числа A_i и B_j в пределах от 1 до $2 \cdot 10^9$.

Формат выходных данных

Вывести строку из M чисел через пробел — для каждого из M чисел B_j вывести либо номер его вхождения в список A_i , либо 0, если его там нет.

Примеры

стандартный ввод	стандартный вывод
6 1 5 6 7 10 22 6 1 2 5 6 15 22	1 0 2 3 0 6

Решение

Дано упорядоченное по неубыванию (в данном случае все элементы различны, значит строго возрастающее) множество A размера N и последовательность запросов B размера M . Для каждого запроса B_j требуется определить, встречается ли B_j в массиве A , и если встречается — вывести индекс (нумерация с 1), иначе вывести 0.

Идея решения: для каждого запроса B_j выполнить бинарный поиск по массиву A . Поскольку A отсортирован, бинарный поиск позволяет за $O(\log N)$ времени определить, есть ли элемент в массиве, и если есть — найти его позицию.

Python

```
n = int(input())
A = list(map(int, input().split()))
d = int(input())
B = list(map(int, input().split()))
def binarysearch(a, key):
```



```
L = 0
R = len(a) - 1
while R >= L:
    mid = (L + R) // 2
    if a[mid] == key:
        return mid + 1
    elif a[mid] > key:
        R = mid - 1
    else:
        L = mid + 1
return 0
k = []
for j in range(d):
    k.append(binarysearch(A, B[j]))
print(" ".join(map(str, k)))
```

C++:

```
#include<bits/stdc++.h>

using namespace std;

int n, m, x;

int main(){

    cin >> n;
    vector<int> a(n+1);
    for(int i = 1; i <= n; i++) cin >> a[i];
    cin >> m;
    for(int i = 0; i < m; i++){
        cin >> x;
        int L = 0, R = n;
        while(R-L > 1){
            int M = (R+L)/2;
            if(a[M] < x) L = M; else R = M;
        }
        if(a[R] == x) cout<<R<<' ';
        else cout<<0<<' ';
    }
}
```

Временная сложность $O(M \log N)$ и памятью $O(N)$ (плюс $O(M)$ если хранить все запросы).

**Список рекомендованной литературы и других источников для подготовки****Литературные источники**

1. Анеликова Л. А. Лабораторные работы по Excel. – Салон-Пресс, 2022. – 112 с.
2. Долинский М.С. Решение сложных и олимпиадных задач по программированию. – СПб.: Питер, 2006. – 366 с.
3. Кирюхин В.М., Цветкова М.С. Информатика. Программы внеурочной деятельности учащихся по подготовке к Всероссийской олимпиаде школьников: 5–11 классы. – М.: БИНОМ. Лаборатория знаний, 2014. – 224 с.
4. Левитин А.В. Алгоритмы: введение в разработку и анализ. – М.: Издательский дом «Вильямс», 2006.
5. Род Стивенс Алгоритмы: теория и практическое применение. – М.: Издательство «Э», 2016.
6. Антти Лааксонен Олимпиадное программирование. – М.: ДМК Пресс, 2018.
7. Меньшиков, Ф. В. Олимпиадные задачи по программированию. – Москва: Питер, 2006. – 315 с.
8. Московские олимпиады по информатике / Под ред. Е.В. Андреевой, В. М. Гуровица и В. А. Матюхина – М.: МЦНМО, 2006. – 256 с
9. Волченков С.Г., Корнилов П.А., Белов Ю.А. и др. Ярославские олимпиады по информатике. Сборник задач с решениями. – М.: БИНОМ. Лаборатория знаний. 2010. – 405 с.
10. Просветов Г.И. Дискретная математика: задачи и решения: учебное пособие. – М.: БИНОМ. Лаборатория знаний. 2008. – 222 с
11. Задачи по программированию /С.М. Окулов, Т.В. Ашихмина, Н.А. Бушмелева и др.; Под ред. С.М. Окулова. – М.: БИНОМ. Лаборатория знаний, 2006. – 820 с.

Интернет-источники

1. Онлайн-курс «Введение в алгоритмы: реализация на языке C++»
[Электронный ресурс] – URL: <https://edu.sirius.online/#/course/2155>
2. Онлайн-курс «Олимпиадное программирование. Базовый уровень»
[Электронный ресурс] – URL: <https://stepik.org/course/74336/>
3. Тренировки по алгоритмам от Яндекса [Электронный ресурс] – URL: https://yandex.ru/yaintern/algorithm-training_2021

Составители: доцент каф. ИиКТ *Иванов И.В.*, доцент каф. ИСиВТ *Иванченко Д.И.*, доцент каф. ИиКТ *Перязева Ю.В.*, ассистент каф. ИиКТ *Ботян Е.Ю.*, ассистент каф. ИиКТ *Коротаева А.Э.*

Научный редактор: зав. каф. ИиКТ, доцент *Маховиков А.Б.*