Олимпиада школьников «Гранит науки» Шифр работы Из8-080

БЛАНК ОЛИМПИАДНОЙ РАБОТЫ (не заполнять)

ВАРИАНТ № 8

Оценка выполнения олимпиадной работы
(заполняется проверяющим)

	№ задания	1	2	3	4	5	6	7	8	9	10	Σ
Первая проверка	Полученный балл	–	–	2	0	15	23					40
	Σ баллов прописью	сорок (40)						Подпись проверяющего				
								Подпись проверяющего				
Вторая проверка	№ задания	1	2	3	4	5	6	7	8	9	10	Σ
	Полученный балл	–	–	2	0	18	23					43
	Σ баллов прописью	сорок три						Подпись проверяющего				
								Подпись проверяющего				

№5

В указанной задаче нет описания алгоритма. Комментарии не являются описанием.

```
mod = 10**9+7  # Тут задаем модель
n, r, d = ...  считываем входные значения n, r, d ...
new_r = r-1  # количество красных клеток в новом дереве
cur_green = n-r  #
cur_red = r      # } количество красных / зеленых клеток после первого дня
for i in range(d-1): # проделаем по всем дням кроме 1
    cur_green += (n-r)*cur_red  #
    cur_red += new_r*cur_red    # } увеличиваем популяцию каждый день
print(cur_red % mod, cur_green % mod)  # выводим ответ по модулю
```

Асимптотика: O(d)

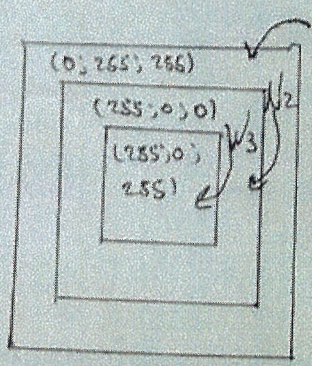
Ответ не приведен. Можно повысить до 185

(150)

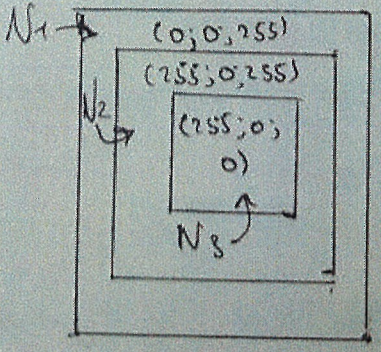
согласно критериям систем

185

№4



(0)



Алгоритм:

1) сделаем значение зеленого минимальным для всей колонки
2) для квадрата N2 сделаем значение синего максимальным
3) для квадрата N3 сделаем значение красного синего минимальным

(0)

Олимпиада школьников «Гранит науки» Шифр работы 438-180
БЛАНК ОЛИМПИАДНОЙ РАБОТЫ (не заполнять)

Вариант № 8

№6

n, m = ... считываем n, m...

matrix = ... считываем поле...

Асимптотика: ~~$O(\log_2 n \cdot n^2 m)$~~ $O(\log n \cdot n^2 m)$

```
def is_place(x, y, size) -> bool :  # функция для проверки можно ли поставить size в (x, y)
    if (x+size-1 > n-1) or (y+size-1 > m-1):  # проверка выхода за границы
        return False
    return (matrix[x][y] == matrix[x+1][y] == matrix[x][y+1] == matrix[x+1][y+1] == ".")
visited = []
def dfs(x, y, size) -> bool:  # функция для нахождения возможности поставить size из (x, y) до правого края поля
    global visited
    if x<0 or y<0 or x>n-1 or y>m-1 or not is_place(x, y, size) or visited[x][y]:  # проверка возможности нахождения там
        return False
    visited[x][y] = True  # помечаем точку как посещенную
    if (m-1-y == size):  # проверяем что достигли цели
        return True
    else:
        for sh in [(1,0), (-1,0), (0,1), (0,-1)]:  # продолжаем обход
            res = dfs(x+sh[0], y+sh[1], size)
            if res:
                return res
    return False  # если из текущей точки пути нам возвращаем False
def check(x, size) -> bool:  # функция для проверки можно ли поставить size длинной от (x,0) до правого края
    global visited
    visited = [[False for _ in range(m)] for _ in range(n)]  # пересоздание visited
    return dfs(x, 0, size)  # проверка
l, r = 1, n
while r != l:  # бинарный поиск по ответу
    m = (r+l)//2
    for i in range(n):  # поиск правильной стартовой позиции
        if check(i, m):
            break
    else:
        r = m-1; continue
    l = m
print(l)
```

(235) (235)

Олимпиада школьников «Гранит науки» Шифр работы У38-080
БЛАНК ОЛИМПИАДНОЙ РАБОТЫ (не заполнять)

Вариант № 8

№ 3

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

f - результирующий столбец

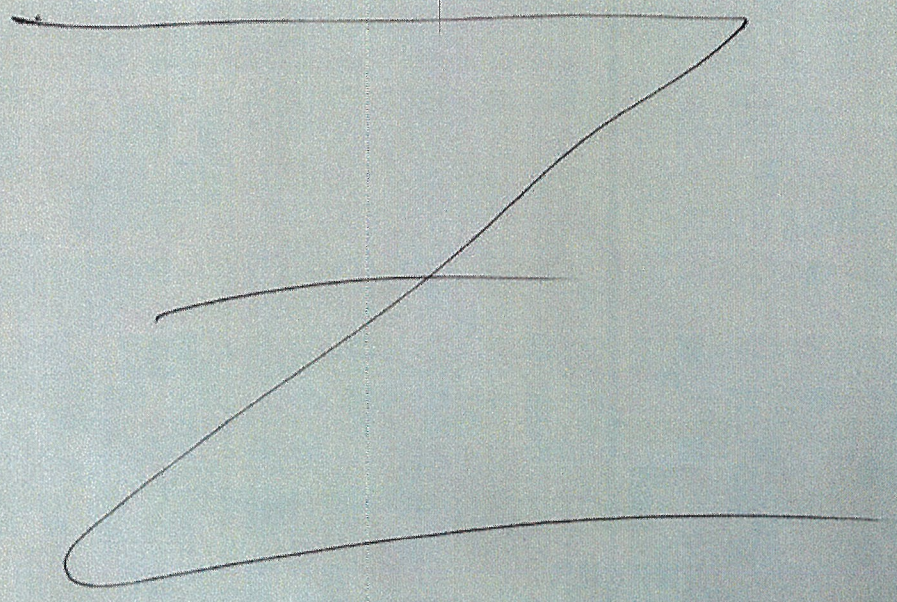
← значение истина

← значение истина

← значение истина.

05

20



```python
mod=int(1e9)+7 #задаем модуль
n, r, d = map(int, input().split()) #считываем входные значения N, R и D
new_r=r-1 #количество красных клеток в новом дереве
cur_green=n-r #количество зеленых клеток после первого дня
cur_red=r #количество красных клеток после первого дня
for i in range(d-1): #проверяем по всем дням кроме первого
    cur_green+=(n-r)*cur_red #увеличиваем популяцию каждый день
    cur_red+=new_r*cur_red #увеличиваем популяцию каждый день

print(cur_red%mod, cur_green%mod) #выводим ответ по модулю

```