Олимпиада школьников

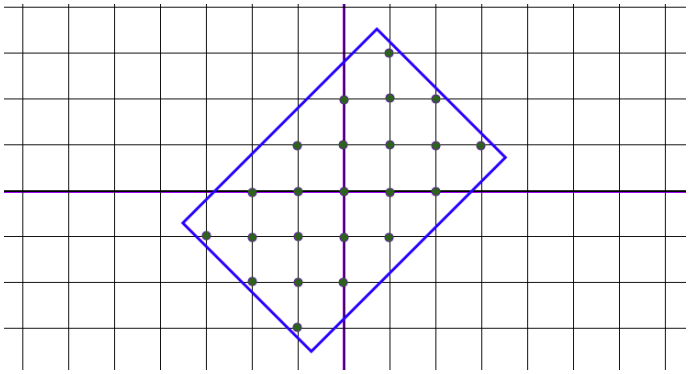
ТИИМ - Технологии. Интеллект. Информатика. Математика

Задания отборочного тура с решениями.

2021-2022 учебный год

Информатика

▷ Задача 1. Посчитай точки



Маша и Алиса учатся программировать. Алиса сообщила, что научилась рисовать прямоугольник с центром в начале координат, повернутый на любой угол. Маша сказала, что это очень интересно, но сможет ли Алиса посчитать точки с целочисленными координатами, которые попадают внутрь прямоугольника, повернутого на 45 градусов, включая те, что находятся на его границе?

Помогите Алисе решить задачу.

Входные данные

Стороны прямоугольника - целые числа $\leq 10^9$

Выходные данные:

Количество точек, содержащихся внутри прямоугольника

Ограничение времени выполнения программы: 1 секунда

Примеры тестовых данных:

Входные данные	Результат работы программы
10001 10001	100012225
111 22	2433
342 2	725
555	

Входные данные	Результат работы программы
78	43567
5 4	17
1 1	1

Решение задачи на языке Python

```
import math
def rectangle_rotation(a, b):
    a //= 2**0.5
    b //= 2**0.5
    r = (a + 1) * (b + 1) + a * b
    return r + r % 2 - 1

def main():
    a =  int(input())
    b = int(input())
    print(math.ceil(rectangle_rotation(a,b)))

main()
```

▷ Задача 2. Найди кота

Общеизвестный факт, что коты любят и умеют прятаться. В этот раз кот спрятался среди других букв. И не один, а с товарищами.

Мы знаем об этих котах следующее:

- Кот может находиться по вертикали, по горизонтали, быть написанным снизу вверх, сверху вниз, слева направо и справа налево
- Кот может и изгибаться
- Кот не может быть написан по диагонали
- Одна и та же буква не может принадлежать двум котам одновременно

Ваша задача - найти всех котов, потому что эти ребята явно что-то задумали!

Входные данные

m, n - количество строк и столбцов массива символов

массив символов

Выходные данные:

количество слов "кот"во входных данных или "нет котиков если котов не нашлось

Ограничение времени выполнения программы: 1 секунда

Примеры тестовых данных:

Входные данные	Результат работы программы
4 3 кот кот кот кот	4
8 10 ********** ********** ***к****** **кот***** ***т****** ********** **ко****** ***т******	2
3 3 bar mal ey.	нет котиков
5 20 3qsqnwIEекотVjкотLJ5 КуккотXRBNy5rkC7rrEU NAocKgкот3qqQW5rкeTu oTтgCuzRhHH2L81ooySX 8DaaaLqLnLeDтoк2тeiO	7

Решение задачи на языке Python

```
def check_word(s,i,j,word):

    siblings = [[0,1],[0,-1],[1,0],[-1,0]]

    if s[i][j]==word[0]:
        if len(word)>1:
            for sib in siblings:
                used = check_word(s,i+sib[0],j+sib[1],word[1:])
                if(used):
                    used.append([i,j])
                    return used

        else :
            used = [[i,j]]
            return used
    return 0



n, m = map(int, input().split())
s = []
word = ['к','о','т']
count = 0

for i in range(n):
    s.append(list(input()))

for i in range(n):
    for j in range(m):
        used = check_word(s,i,j,word)
        if(used):
```

```
            count = count + 1
            for coord in used:
                s[coord[0]][coord[1]] = '*'

if(count>0):
    print(count)
else:
    print('нет котиков')
```

▷ **Задача 3. Неиспорченный телефон**

Ученики одной математической гимназии любят играть в неиспорченный телефон. Но когда эта забава набрала популярность и участников стало больше 30, чемпион школы по этой игре решил написать программу, которая поможет ему в вычислениях. Игра состоит в следующем:

- Первый участник загадывает натуральное число, сообщает его второму
- Второй участник также загадывает натуральное число
- Третий игрок складывает первое и второе число
- Каждый последующий игрок знает только два предыдущих числа,

складывает их и передает следующему. Т.к. гимназия математическая, никто не допускает вычислительных или других ошибок

- Когда участники заканчиваются, последний, n-й школьник сообщает свое число первому игроку
- Первый игрок, зная только свое и последнее полученное число, должен посчитать за время меньше одной секунды, какое число загадал второй

Входные данные

$3 \leq$ количество участников ≤ 30000

первое число

последнее число

Выходные данные:

второе число

Ограничение времени выполнения программы: 1 секунда

Примеры тестовых данных:

Входные данные	Результат работы программы
4 111 333	111
11 1 199	3
4 3 15	6
10 10	

Входные данные	Результат работы программы
618	12
22 1234 8588822	22
1000 1234 210787944360045656656298097820029083916134605838245907804331643873751155650486974252802837014239727231106731181934487834612810285407190590864719747700798594734529626943071348434969244386207572475257407812487852238	22

Решение задачи на языке Python

```
#Приводим решение задачи, работающее за время $O(log N)$
#Бинарное возведение в степень
def faster_pow(m,n):

    if n==0:
        return [[1,0],[0,1]]
    else:
        if n % 2 == 0:
            t = faster_pow(m,n//2)
            return umn(t,t)
        else:
```

```
            t = faster_pow(m,n-1)
            return umn(t,m)

#умножение проще, потому что матрица симметрическая
def umn(mat1,mat2):
    a = mat1[0][0]*mat2[0][0]+mat1[1][0]*mat2[0][1]
    b = mat1[0][0]*mat2[1][0]+mat1[1][0]*mat2[1][1]
    c = mat1[1][0]*mat2[1][0]+mat1[1][1]*mat2[1][1]
    return [[a,b],[b,c]]

k  = int(input())
x = int(input())
n_k = int(input())

#матрица, позволяющая найти предыдущие числа,
#зная следующие(см алгоритм нахождения чисел Фибоначчи(Дональд Кнут.
#Искусство программирования, том 1 [ с. 112])
mat = [[-1,1],[1,0]]
mat = faster_pow(mat,k-2)
a = mat[0][0]
b = mat[1][0]
c = mat[1][1]

#Вычисляем второе число с помощью найденных коэффициентов
y = (b*x - b*b*n_k)//a + c*n_k

print(int(y))
main()
```

▷ Задача 4. Треугольник мира

Три племени(Десятичное, Двоичное и Восьмеричное), живущие вдали от цивилизации, решили заключить союз и установить в знак этого статую: треугольник мира.

Но возникла проблема: они настолько друг от друга отличаются, что используют разные системы счисления, в результате, никак не могут выяснить, а получится ли вообще треугольник из бревен, что они приготовили(каждое племя приготовило по одному бревну).

Снова назревает конфликт, а вождь Десятичного племени даже обещает должность верховного шамана тому, кто решит эту сложную задачу.

Входные данные

a - целое положительное число в двоичной системе счисления

b - целое положительное число в восьмеричной системе счисления

c - целое положительное число в десятичной системе счисления

Выходные данные:

“да будет мир” - если из исходных материалов можно сложить треугольник

“для мира требуется еще потрудиться” - если нельзя

Ограничение времени выполнения программы: 1 секунда

Примеры тестовых данных:

Входные данные	Результат работы программы
11011111 56237 5434	для мира требуется еще потрудиться
10 3 4	да будет мир
1 2 3	для мира требуется еще потрудиться
110111 112 25	да будет мир
10110 57 57	да будет мир
101101 75 57	да будет мир

Решение задачи на языке Python

```
a, b, c = int(input(), 2), int(input(), 8), int(input())
if a + b > c and a + c > b and b + c > a:
    print('да будет мир')
else:
    print('для мира требуется еще потрудиться')
```

▷ Задача 5. Поиск фальшивой монеты

Когда-то, в давние времена, когда весы еще не имели цифрового табло, и могли показывать только, совпадает ли масса того, что лежит на левой чаше, с тем, что лежит на правой, король Златоландии, Вильгельм Мудрый XIV, искал себе главного казначея. Специалист по подбору персонала королевства конечно же проверял всех откликнувшихся на вакансию на детекторе лжи, давал им психологические тесты, спрашивал, кем они себя видят через пять лет и консультировался с астрологом, но главным тестом при приеме на работу было решение задачи на поиск фальшивой монеты.

Задание:

Доподлинно известно, что среди N монет казны находится одна фальшивая и она весит меньше, чем остальные. За какое минимальное количество взвешиваний можно гарантированно найти фальшивку?

Входные данные:

число монет

Выходные данные:

Количество взвешиваний, необходимое для поиска фальшивки

Ограничение времени выполнения программы: 1 секунда

Примеры тестовых данных:

Входные данные	Результат работы программы
7777	9
1234	7
2	1
4	2
8	2
59049	10

Решение задачи на языке Python

```
import math
def how_many_measurements(n):

    print( math.ceil(math.log(n,3)))

n = int(input())
how_many_measurements(n)
```

▷ Задача 6. Оптимальный путь

Разведчик ушел вперед группы, чтобы узнать путь до города. В пути

он записывал все свои перемещения с помощью символов "^ "v « » которые соответственно означали север, юг, запад и восток. Так как местность для него новая, он иногда плутал, возвращался, ходил кругами. Теперь нужно упростить этот путь, чтобы по новому маршруту смогла пройти вся группа.

Входные данные:

Строка, содержащая путь разведчика

Выходные данные:

Короткий путь

Ограничение времени выполнения программы: 1 секунда

Примеры тестовых данных:

Входные данные работы программы	Результат
^	^
^v<>	
>><<>>^^^v	>>^^
<^^v<v>^v^>^<^><<><v>><^vv><^v< v^^^<>>^vv>v^v^^vv<^vv<^>v<v^>v> <<>v^><^v^<^^^v<><<>><^<vv<<^>>><^ >v>>^v>^^>>vv^>>^><>v^^>^>v^v>v<< v<^^^<<<^<>^^>><><vv^>^<v<>v^>^>^ <><vv^><^v^v^^<>v><>v>>^<<vv<>>	^^^^^^^^>>>>>>>>>
>v^>^^>v<v>v><<>>	>>>>>v
>>>>>>	>>>>>>

Решение задачи на языке Python

```
def solve_path(line):
    dx = 0
    dy = 0
```

```
first_x = 0
first_y = 0
res = ''

for s in line:
    if s=='>':
        dx = dx + 1
    if s=='<':
        dx = dx - 1
    if s=='^':
        dy = dy + 1
    if s == 'v':
        dy = dy - 1

if dx>0:
    path_x = '>'*dx
    start_x = line.find('>')
else:
    path_x = '<'*(-dx)
    start_x = line.find('<')

if dy>0:
    path_y = '^'*dy
    start_y = line.find('^')
else:
    path_y = 'v'*(-dy)
    start_y = line.find('v')

if start_x<start_y:
    print(path_x+path_y)
```

```
    else:
        print(path_y+path_x)


line = input()

solve_path(line)
```