МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ИНФОРМАТИКА. ОТБОРОЧНЫЙ ТУР 11 КЛАСС

Часть 1

Подготовительные курсы 1 вариант

На подготовительных курсах к ЕГЭ по информатике работает пять преподавателей. На новый учебный год набралось 10 групп учеников. Каждый преподаватель будет вести занятия в двух группах. Сколько существует способов распределить группы по преподавателям, если важно, какой именно преподаватель в какой именно группе ведёт занятия?

Ответ:

113400

Решение:

$C_{10}^2 \cdot C_8^2 \cdot C_6^2 \cdot C_4^2 \cdot C_2^2 = 113400$

Подготовительные курсы 2 вариант

На подготовительных курсах к ЕГЭ по информатике работает четыре преподавателя. На новый учебный год набралось 8 групп учеников. Каждый преподаватель будет вести занятия в двух группах. Сколько существует способов распределить группы по преподавателям, если важно, какой именно преподаватель в какой именно группе ведёт занятия?

Ответ:

2520

Решение:

$C_8^2 \cdot C_6^2 \cdot C_4^2 \cdot C_2^2 = 2520$

Шахматы 1 вариант

При игре в шахматы часто используется тактический прием - захват линии, с помощью сдваивания ладей. Ладья - фигура, которая ходит на любое количество клеток по горизонтали или вертикали. Вам дана шахматная доска (квадрат 8 на 8 клеток, всего 64 клетки) и неограниченное число ладей одного цвета. Будем считать линию захваченной (как горизонтальную, так и вертикальную), если на ней находится не менее двух ладей. Найдите минимальное число ладей, которые нужно расставить на пустой шахматной доске, чтобы каждая линия (как горизонтальная, так и вертикальная) была захвачена. В ответ запишите количество ладей.

Ответ:

16

Решение:

Идеальная расстановка ладей – по две рядом, с сдвигом на единицу вправо на каждой строке, т.е. расстановка такого вида.

XX……..
...XX……
…..XX……
…….XX…….
……XX……..
……...XX……
……..XX…….
…….XX…..

Это не единственно-возможная расстановка. Однако из условия задачи следует, что число ладей не может быть меньше 16 (2 ладьи на каждую вертикаль и горизонталь, то есть ладей не может быть 15 и менее).

Шахматы 2 вариант

При игре в шахматы часто используется тактический прием - захват линии, с помощью сдваивания ладей. Ладья - фигура, которая ходит на любое количество клеток по горизонтали или вертикали. Пусть шахматная доска имеет размер 9 на 9 клеток (всего 81 клетка) и неограниченное число ладей одного цвета. Будем считать линию захваченной (как горизонтальную, так и вертикальную), если на ней находится не менее двух ладей. Найдите минимальное число ладей, которые нужно расставить на пустой шахматной доске (размера 9 на 9), чтобы каждая линия (как горизонтальная, так и вертикальная) была захвачена. В ответ запишите количество ладей.

Ответ:

18

Решение:

Идеальная расстановка ладей – по две рядом, с сдвигом на единицу вправо на каждой строке, т.е. расстановка такого вида.

XX……..
...XX……
…..XX……
…….XX…….
……XX……..
……...XX……
……..XX…….
…….XX…..

Это не единственно-возможная расстановка. Однако из условия задачи следует, что число ладей не может быть меньше 18 (2 ладьи на каждую вертикаль и горизонталь, то есть ладей не может быть 17 и менее).

Делители 1 вариант

Найдите наименьшее натуральное число имеющее ровно 1000 делителей. В ответ запишите это число.

Ответ:

810810000

Решение:

Используя основную теорему арифметики, минимальное каноническое разложение числа с 1000 делителей выглядит так:

$2^4 * 3^4 * 5^4 * 7 * 11 * 13 = 810810000$

Такое разложение даёт наибольшее количество делителей, поскольку при увеличении степени какого-либо простого числа в разложении на единицу, количество делителей увеличивается в (n + 2) / (n + 1) раз. Например у числа $8 = 2^3$ – четыре делителя (3 + 1). А у числа 2^4 – пять. Но добавление нового простого числа в разложение увеличивает количество делителей сразу в два раза. Поэтому выгодно добавлять новые простые числа в разложение, чем увеличивать степени у уже имеющихся простых чисел.

Поскольку само число 1000, в каноническом разложении выглядит так:

$1000 = 2^3 * 5^3$

Из этого следует, что в разложении исходного числа должно быть 3 степени – единицы и 3 степени – четвёрки. Для минимизации числа, степени единицы поставим у больших простых чисел, а четвёрки у меньших.

Делители 2 вариант

Найдите наименьшее натуральное число имеющее ровно 5000 делителей. В ответ запишите это число.

Ответ:

4727833110000

Решение:

Используя основную теорему арифметики, минимальное каноническое разложение числа с 1000 делителей выглядит так:

- $2^4 * 3^4 * 5^4 * 7^4 * 11 * 13 * 17 = 4727833110000$

Такое разложение даёт наибольшее количество делителей, поскольку при увеличении степени какого-либо простого числа в разложении на единицу, количество делителей увеличивается в (n + 2) / (n + 1) раз. Например у числа $8 = 2^3$ – четыре делителя (3 + 1). А у

числа 2^4 – пять. Но добавление нового простого числа в разложение увеличивает количество делителей сразу в два раза. Поэтому выгодно добавлять новые простые числа в разложение, чем увеличивать степени у уже имеющихся простых чисел.

Поскольку само число 5000, в каноническом разложении выглядит так:

$5000 = 2^3 * 5^4$

Из этого следует, что в разложении исходного числа должно быть 3 степени – единицы и 4 степени – четвёрки. Для минимизации числа, степени единицы поставим у больших простых чисел, а четвёрки у меньших.

Часть 2

На уроке Географии

На уроке географии Кузьме часто приходится сравнивать страны по населению, территории и иным признакам. Но Кузьме больше всего нравится сравнивать страны по флагам, при этом у него есть специальная система: для каждой страны он определяет два параметра:

- R_i- рейтинг флага страны S_i. R_i равен целому числу r_i, если флаг страны содержит r_i различных цветов, нравящихся Кузьме
- C_i- разноцветность флага страны S_i. C_i равен целому числу c_i, если флаг страны содержит c_i различных цветов

Соответственно при сравнении страна $S_1 > S_2$, если $R_1 > R_2$ или $R_1 = R_2$ и $C_1 > C_2$

Попробуйте расположить страны по системе Кузьмы в порядке возрастания

Формат ввода

В первой строке содержится целое число n ($2 \le n \le 1000$) - кол-во стран

Во второй строке содержится строка s ($1 \le len(s) \le 10$), содержащая различные буквы латинского алфавита - любимые цвета Кузьмы

В последующих n строках содержатся пары строк S_i и $colors_i$ ($1 \le len(S_i), len(colors_i) \le 10$) - название страны и цвета ее флага без повторений

Формат вывода

В ответ выведите возрастающую последовательность из названий стран через пробел (если страны равны, оставьте их в исходном порядке)

Пример

Ввод	**Вывод**
5 rby Austria rw Ghana brgy Belgium byr France bwr	Austria France Luxembourg Belgium Ghana

Ввод	Вывод
Luxembourg rwb	

Примечания

Austria имеет рейтинг 1 (так как только цвет r - любимый), France и Luxembourg рейтинги 2, Belgium и Ghana рейтинги 3 (при этом Ghana > Belgium, так как кол-во цветов во флаге больше 4 > 3)

Пример решения

```
def solution(num, love, country):
    arr = []
    for x in range(num):
        s = 0
        for color in country[x][1]:
            if color in love:
                s += 1

        arr.append([country[x][0], s, len(country[x][1])])

    sorted_arr = sorted(arr, key=lambda z: (z[1], z[2]))
    result = []
    for x in range(num):
        result.append(sorted_arr[x][0])

    return result


# solving
n = int(input())
lv = input()
c = []
for x in range(n):
    c.append(input().split(" "))
r = solution(n, lv, c)
print(' '.join(r))
```

МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ИНФОРМАТИКА. ОТБОРОЧНЫЙ ТУР 11 КЛАСС

На уроке Литературы

Кузьме задали прочитать список книг разных авторов. Ознакомившись со списком авторов Кузьма удивился, ведь ранее он уже читал их произведения. Для каждого автора у Кузьмы определен показатель "интересности", влияющий на скорость чтения книги следующим образом:

- в первый день чтения книги Кузьма прочитает 10 * W_{-1} страниц (где W_{-1} - текущий интерес к чтению)
- во второй и последующие дни чтения книги Кузьма будет читать 10 * W_i страниц (W_i - интересность текущей книги)

Кузьма может читать книги в любом порядке, но если он взялся за определенную книгу - он обязательно дочитывает ее до конца. В день, когда Кузьма дочитал очередную книгу, он не берется за новую (не зависимо от того, сколько страниц он прочитал сегодня), при этом завтра, начав новую книгу, его интерес к чтению будет равен текущему.

Теперь Кузьма просит Вас посчитать: какое минимальное кол-во дней ему необходимо для прочтения заданного списка книг?

Формат ввода

В первой строке содержатся два числа: N ($1 \le N \le 7$) - кол-во книг и W_0 ($1 \le W_0 \le 10$) - стартовый интерес к чтению

В последующих N строках содержатся по два числа: K_i ($1 \le K_i \le 1000$), W_i ($1 \le W_i \le 10$) - кол-во страниц в книге и ее интересность

Формат вывода

Выведите число: минимальное кол-во дней, необходимое для прочтения всех книг

Пример

Ввод	Вывод
4 5 100 10 290 5 300 7 100 1	14

Примечания

Порядок прочтения книг в примере: 2-3-1-4

Кол-во страниц по дням Книга 2: 50+50+50+50+50+40, Книга 3: 50+70+70+70+40, Книга 1: 70+30, Книга 4: 100 - итого 6+5+2+1=14 дней

Пример решения

```
import itertools


def solution(num, w_base, books):
    book_list = list(itertools.permutations(range(num)))
```

```
    days = -1
    for bl in book_list:
        cur_days = 0
        w = w_base
        for bn in bl:
            cur_days += max(0, (books[bn][0] - w * 10)) // (books[bn][1] *
10)
            if max(0, (books[bn][0] - w * 10)) % (books[bn][1] * 10) > 0:
                cur_days += 2
            else:
                cur_days += 1
            w = books[bn][1]

        if days < 0 or days > cur_days:
            days = cur_days
            # print(bl)

    return days


# solving
n, w0 = [int(k) for k in input().split(" ")]
b = []
for x in range(n):
    b.append([int(k) for k in input().split(" ")])
r = solution(n, w0, b)
print(r)
```

МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ИНФОРМАТИКА. ОТБОРОЧНЫЙ ТУР 11 КЛАСС

На уроке Биологии

Изучая грызунов, Кузьма заинтересовался тем, что мышь при ее малых размерах способна развивать скорость до 13 км/час. Кузьма не сомневается, что на открытой местности человек легко обгонит мышь, но что будет в лабиринте?

Лабиринт представляет собой массив *M*x*N*, где в каждой ячейке хранится один из символов:

- O (open) - открытое пространство
- W (wall) - стена
- H (human), M (mouse) - стартовые позиции человека и мыши соответственно
- F (finish) - позиция финиша

Перемещение по лабиринту происходит по горизонтали или вертикали (но не по диагонали), при этом нельзя перемещаться в клетку со стеной.

- В свой ход человек перемещается на одну или две клетки
- В свой ход мышь может переместиться на одну клетку. При этом один раз за забег мышь способна пройти через стену (то есть один раз она может встать на клетку стены, после чего следующим ходом обязана сойти с нее и двигаться по обычным правилам)

Побеждает участник, которому для достижения финиша понадобилось меньшее число ходов - помогите его определить!

Формат ввода

В первой строке заданы два целых числа M и N ($3 \le M,N \le 8$) - размеры лабиринта

В дальнейших строках содержится строка s длины N - строка лабиринта

Формат вывода

При победе человека выведите: HUMAN

При победе мыши выведите: MOUSE

При ничьей выведите: DRAW

Пример 1

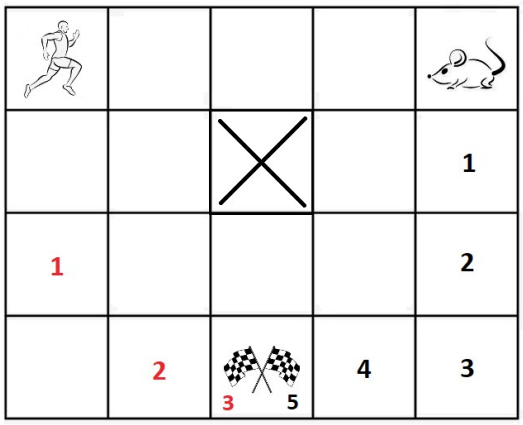
Ввод	Вывод
4 5 H O O O M O O W O O O O O O O O O F O O	HUMAN

Пример 2

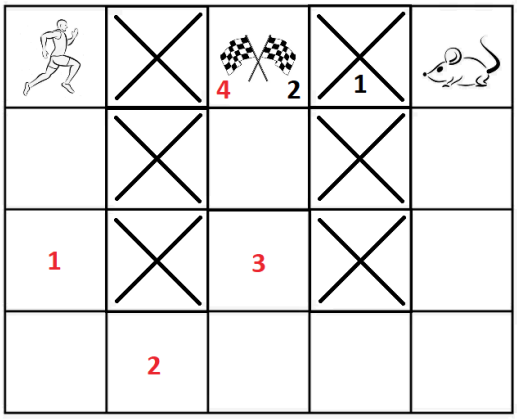
Ввод	Вывод
4 5 H W F W M	MOUSE

Ввод	Вывод
O W O W O O W O W O O O O O O	

Примечания



Первый пример (Человек за 3 хода, мышь за 5)



Второй пример (Человек за 4 хода, мышь за 2)

Пример решения

```
def solution(l_x, l_y, maze):
    maze_walled = []
    wall = []
    for x in range(l_y + 2):
        wall.append('W')
    maze_walled.append(wall)

    for x in range(l_x):
        row = ['W']
        for y in range(l_y):
            row.append(maze[x][y])
        row.append('W')
        maze_walled.append(row)
    maze_walled.append(wall)

    l_x, l_y = l_x + 2, l_y + 2

    human = [-1, -1]
```

```
mouse = [-1, -1]
finish = [-1, -1]
for x in range(l_x):
    for y in range(l_y):
        if maze_walled[x][y] == 'H':
            human = [x, y]
        if maze_walled[x][y] == 'M':
            mouse = [x, y]
        if maze_walled[x][y] == 'F':
            finish = [x, y]

maze_walled_for_mouse = []
maze_walled_reverse = []
for x in range(l_x):
    maze_walled_for_mouse.append([])
    maze_walled_reverse.append([])
    for y in range(l_y):
        maze_walled_for_mouse[x].append(maze_walled[x][y])
        maze_walled_reverse[x].append(maze_walled[x][y])

# human
road = [[0, human[0], human[1]]]
while len(road) > 0:
    cur = road.pop(0)
    maze_walled[cur[1]][cur[2]] = str(cur[0])
    if maze_walled[cur[1] + 1][cur[2]] in ['H', 'M', 'O', 'F']:
        road.append([cur[0] + 1, cur[1] + 1, cur[2]])
    if maze_walled[cur[1] - 1][cur[2]] in ['H', 'M', 'O', 'F']:
        road.append([cur[0] + 1, cur[1] - 1, cur[2]])
    if maze_walled[cur[1]][cur[2] + 1] in ['H', 'M', 'O', 'F']:
        road.append([cur[0] + 1, cur[1], cur[2] + 1])
    if maze_walled[cur[1]][cur[2] - 1] in ['H', 'M', 'O', 'F']:
        road.append([cur[0] + 1, cur[1], cur[2] - 1])

human_road = -1
if maze_walled[finish[0]][finish[1]] != 'F':
    steps = int(maze_walled[finish[0]][finish[1]])
    human_road = steps // 2 + steps % 2

# mouse
road = [[0, mouse[0], mouse[1]]]
while len(road) > 0:
    cur = road.pop(0)
    maze_walled_for_mouse[cur[1]][cur[2]] = str(cur[0])
    if maze_walled_for_mouse[cur[1] + 1][cur[2]] in ['H', 'M', 'O', 'F']:
        road.append([cur[0] + 1, cur[1] + 1, cur[2]])
    if maze_walled_for_mouse[cur[1] - 1][cur[2]] in ['H', 'M', 'O', 'F']:
        road.append([cur[0] + 1, cur[1] - 1, cur[2]])
    if maze_walled_for_mouse[cur[1]][cur[2] + 1] in ['H', 'M', 'O', 'F']:
        road.append([cur[0] + 1, cur[1], cur[2] + 1])
    if maze_walled_for_mouse[cur[1]][cur[2] - 1] in ['H', 'M', 'O', 'F']:
        road.append([cur[0] + 1, cur[1], cur[2] - 1])

mouse_road = -1
if maze_walled_for_mouse[finish[0]][finish[1]] != 'F':
    steps = int(maze_walled_for_mouse[finish[0]][finish[1]])
    mouse_road = steps

# finish -> maze
road = [[0, finish[0], finish[1]]]
while len(road) > 0:
```

```
        cur = road.pop(0)
        maze_walled_reverse[cur[1]][cur[2]] = str(cur[0])
        if maze_walled_reverse[cur[1] + 1][cur[2]] in ['H', 'M', 'O', 'F']:
            road.append([cur[0] + 1, cur[1] + 1, cur[2]])
        if maze_walled_reverse[cur[1] - 1][cur[2]] in ['H', 'M', 'O', 'F']:
            road.append([cur[0] + 1, cur[1] - 1, cur[2]])
        if maze_walled_reverse[cur[1]][cur[2] + 1] in ['H', 'M', 'O', 'F']:
            road.append([cur[0] + 1, cur[1], cur[2] + 1])
        if maze_walled_reverse[cur[1]][cur[2] - 1] in ['H', 'M', 'O', 'F']:
            road.append([cur[0] + 1, cur[1], cur[2] - 1])

    for x in range(1, l_x - 1):
        for y in range(1, l_y - 1):
            if maze_walled[x][y] == 'W':
                for dx in [[-1, 0], [1, 0], [0, -1], [0, 1]]:
                    if maze_walled_for_mouse[x + dx[0]][y + dx[1]] not in
['W', 'H', 'M', 'O', 'F']:
                        for dy in [[-1, 0], [1, 0], [0, -1], [0, 1]]:
                            if maze_walled_reverse[x + dy[0]][y + dy[1]] not
in ['W', 'H', 'M', 'O', 'F']:
                                to_wall = int(maze_walled_for_mouse[x +
dx[0]][y + dx[1]])
                                to_finish = int(maze_walled_reverse[x +
dy[0]][y + dy[1]])
                                new_road_for_mouse = to_wall + 2 + to_finish
                                if mouse_road < 0 or mouse_road >
new_road_for_mouse:
                                    mouse_road = new_road_for_mouse

    if human_road == mouse_road:
        return 'DRAW'
    elif human_road < 0:
        return 'MOUSE'
    elif mouse_road < 0:
        return 'HUMAN'
    elif human_road < mouse_road:
        return 'HUMAN'
    else:
        return 'MOUSE'


# solving
m, n = [int(k) for k in input().split(" ")]
lab = []
for k in range(m):
    lab.append(input().split(" "))
r = solution(m, n, lab)
print(r)
```