МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ИНФОРМАТИКА. ОТБОРОЧНЫЙ ТУР 10-11 КЛАСС

Часть 1 Вариант1

Задача 1. Строка

Дана строка, состоящая из 10 нулей и 10 единиц, расположенных в случайном порядке. За одно алгоритмическое действие можно поменять местами две рядом стоящих цифры. Какое наименьшее количество действий потребуется, чтобы все единицы гарантированно следовали друг за другом (не обязательно вначале или в конце строки)?

Примечание: Гарантированно - подразумевает что количество действий, указанное в ответе, хватит для любого начального распределения.

Ответ

50

Решение

Будем собирать единицы слева, а нули справа. Возьмём самую левую единицу и самый правый нолик. В худшем случае понадобится 5 ходов, чтобы сдвинуть их на края. Затем возьмем следующую пару, и вновь понадобиться 5 ходов. Таким образом получим 10 пар, по 5 ходов на каждую.

Примечание: Гарантированно - подразумевает что количество действий, указанное в ответе, хватит для любого начального распределения.

Задача 2. Соревнования

В группе по спортивному программированию учится 10 учеников на первом году и 8 учеников на втором. Учитель хочет отправить на соревнования 4 команды по 2 человека, и хочет чтобы в каждой команде был ученик как первого, так и второго года. Сколькими способами он может набрать команды?

Ответ

352800

Решение

$C_{10}^4 \cdot A_8^4 = 352800$

Задача 3. Перестановки

Рассматривается множество всех трёхразрядных чисел в восьмеричной системе счисления. Для каждого числа из множества рассматриваются все возможные перестановки разрядов. В полученных перестановках число может начинаться с нуля. После чего все полученные уникальные комбинации для каждого числа суммируются в десятичной системе счисления. Найдите количество чисел, для которых полученная сумма делится на 7.

Ответ

64

Решение

```
import itertools


def octal_to_decimal(octal_str):
    try:
        return int(octal_str, 8)
    except ValueError:
        return 0


count = 0

for i in range(int("100", 8), int("777", 8) + 1):
    octal_str = oct(i)[2:]
    digits = list(octal_str)

    unique_permutations = set(itertools.permutations(digits))

    sum_permutations_decimal = 0
    for perm in unique_permutations:
        perm_str = "".join(perm)
        decimal_val = int(perm_str, 8)
        sum_permutations_decimal += decimal_val

    if sum_permutations_decimal % 7 == 0:
        count += 1

print(count)
```

МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ИНФОРМАТИКА. ОТБОРОЧНЫЙ ТУР 10-11 КЛАСС

Часть 1 Вариант 2

Задача 1. Строка

Дана строка, состоящая из 12 нулей и 12 единиц, расположенных в случайном порядке. За одно алгоритмическое действие можно поменять местами две рядом стоящих цифры. Какое наименьшее количество действий потребуется, чтобы все единицы гарантированно следовали друг за другом (не обязательно вначале или в конце строки)?

Ответ

72

Решение

Будем собирать единицы слева, а нули справа. Возьмём самую левую единицу и самый правый нолик. В худшем случае понадобится 6 ходов, чтобы сдвинуть их на края. Затем возьмем следующую пару, и вновь понадобиться 6 ходов. Таким образом получим 12 пар, по 6 ходов на каждую.

Задача 2. Соревнования

В группе по спортивному программированию учится 8 учеников на первом году и 7 учеников на втором. Учитель хочет отправить на соревнования 4 команды по 2 человека, и хочет чтобы в каждой команде был ученик как первого, так и второго года. Сколькими способами он может набрать команды?

Ответ

58800

Решение

$C_8^4 \cdot A_7^4 = 58800$

Задача 3. Перестановки

Рассматривается множество всех трёхразрядных чисел в восьмеричной системе счисления. Для каждого числа из множества рассматриваются все возможные перестановки разрядов. В полученных перестановках число может начинаться с нуля. После чего все полученные уникальные комбинации для каждого числа суммируются в десятичной системе счисления. Найдите количество чисел, для которых полученная сумма делится на 3.

Ответ

144

Решение

```
import itertools


def octal_to_decimal(octal_str):
    try:
        return int(octal_str, 8)
    except ValueError:
        return 0


count = 0

for i in range(int("100", 8), int("777", 8) + 1):
    octal_str = oct(i)[2:]
    digits = list(octal_str)

    unique_permutations = set(itertools.permutations(digits))

    sum_permutations_decimal = 0
    for perm in unique_permutations:
        perm_str = "".join(perm)
        decimal_val = int(perm_str, 8)
        sum_permutations_decimal += decimal_val

    if sum_permutations_decimal % 3 == 0:
        count += 1

print(count)
```

Часть 2

Задача 1. Крестики-нолики

Иван любит статистику. Недавно к нему в руки попали бланки сыгранных игр в "крестики-нолики".

Бланк представляет собой игровое поле 3x3 клетки, где внутри клетки может быть крестик, нолик или пусто.

Игра "крестики-нолики" ведется по следующим правилам:

-ходят по очереди, начиная с крестиков

-в свой ход выбирается пустая клетка, в неё ставится крестик или нолик (в зависимости от того, чей ход)

-игра немедленно завершается победой крестиков, если удается получить линию (вертикальную, горизонтальную или диагональную) из трёх крестиков (аналогично для ноликов)

-игра завершается ничьёй, если победитель не определен, а пустых клеток больше нет

Иван просит Вас помочь ему посчитать статистику.

Формат входных данных

В трех строчках передаются по три символа из набора {X (крестик), O (нолик (символ заглавной буквы o латинского алфавита)), - (пусто)}

Формат выходных данных

В ответ выведите слово CROSS (если игра выиграна крестиками)

В ответ выведите слово ZERO (если игра выиграна ноликами)

В ответ выведите слово DRAW (если игра завершилась вничью)

В ответ выведите слово ERROR (если игра не доиграна или сыграна не по правилам)

Замечание

В первом примере крестики выиграли по диагонали.

Во втором примере игра закончилась вничью.

В третьем примере игра не доиграна.

Примеры

Ввод	Вывод
O-X OX- X--	CROSS
OXX XOO OXX	DRAW
OXX -OO OXX	ERROR

Пример решения

```
def solution(field):
    counts = [0, 0, 0]
    for x in range(3):
        for y in range(3):
            if field[x][y] == 'X':
                counts[0] += 1
            elif field[x][y] == 'O':
                counts[1] += 1
            else:
                counts[2] += 1
    wins, who = 0, 'no'
    for z in range(3):
        if field[z][0] == field[z][1] == field[z][2] and
field[z][0] != '-':
            wins += 1
            who = field[z][0]
        if field[0][z] == field[1][z] == field[2][z] and
field[0][z] != '-':
            wins += 1
            who = field[0][z]
    if field[0][0] == field[1][1] == field[2][2] and field[1][1]
!= '-':
        wins += 1
        who = field[1][1]
    if field[0][2] == field[1][1] == field[2][0] and field[1][1]
!= '-':
        wins += 1
```

```
        who = field[1][1]

    result = 'ERROR'
    if counts[0] - counts[1] <= 1 and (wins == 1 or wins == 0
and counts[2] == 0):
        if wins == 1:
            if who == 'X' and counts[0] - counts[1] == 1:
                result = 'CROSS'
            if who == 'O' and counts[0] - counts[1] == 0:
                result = 'ZERO'
        else:
            result = 'DRAW'

    return result

row1 = input()
row2 = input()
row3 = input()
print(solution([row1, row2, row3]))
```

Задача 2. Ожерелье

Иван и Степан на археологических раскопках нашли ожерелье с драгоценными камнями, но вот беда: цепочка ожерелья порвалась. Иван и Степан решили разделить драгоценные камни, причем решено было сделать это следующим способом:

-Драгоценный камень выбирается по очереди, причем можно выбрать только один из крайних камней (ближние к месту разрыва цепочки). Выбранный камень снимается и ход переходит.

-Жребий выбирать первым выпал Ивану.

Ценности каждого из камней известны, Иван и Степан стараются выбирать так, чтобы получить себе как можно более ценный суммарный набор. Сосчитайте, какую суммарную ценность соберет Иван.

Формат входных данных

В первой строке содержится одно число: $N\ (2 \le len(N) \le 10^3)$ – кол-во камней на ожерелье.

Во второй строке через пробел содержатся N чисел ki – ценность камней на цепочке (как если бы цепочку растянули в линию).

Формат выходных данных

В ответ выведите одно число K – сумма ценности камней, выбранных Иваном.

Замечание

В первом примере на первом ходу Иван может выбрать камни ценностью 2 (левый) или 3 (правый) - независимо от выбора, Степан выберет камень 8 (который станет крайним) и Иван заберет оставшийся. Итого Иван получит 2 + 3 = 5.

Во втором примере Иван сможет взять камни ценностью 9 + 12 = 21.

Примеры

Ввод	Вывод
3 2 8 3	5
4 7 12 1 9	21

Пример решения

```
def solution(num, numbers):

    matrix = []
    for i in range(num):
        temp = []
        for j in range(num):
            temp.append(None)
        matrix.append(temp)

    for i in range(num):
        for j in range(i, -1, -1):
            if i == j:
                matrix[i][j] = (numbers[i], 0)
            else:
                if numbers[i] + matrix[i - 1][j][1] > numbers[j]
+ matrix[i][j + 1][1]:
                    matrix[i][j] = (numbers[i] + matrix[i - 1]
[j][1], matrix[i - 1][j][0])
                else:
                    matrix[i][j] = (numbers[j] + matrix[i][j +
1][1], matrix[i][j + 1][0])
```

```
    return matrix[num - 1][0][0]


n = int(input())
row = [int(x) for x in input().split()]
print(solution(n, row))
```

Задача 3. Ладьёй ходи

Иван решает шахматную задачу:

На игровом поле размером N x N клеток расположена белая ладья, черный король и пешки двух цветов. Ваша задача – за минимальное кол-во ходов ладьей съесть короля. За один ход ладья может походить по обычным шахматным правилам:

-Походить по горизонтали или вертикали на любое кол-во клеток (при этом все клетки на пути должны быть пустыми).

-Занять место черной пешки (съесть), если до нее можно дойти по предыдущему правилу.

Необходимо посчитать минимальное кол-во ходов, за которое ладья может съесть короля (при этом ходит только ладья и нельзя есть собственные белые пешки).

Формат входных данных

В первой строке содержится одно число: N $(3 \leq N \leq 555)$ – размеры поля.

В следующих N строках содержатся по N символов, каждый из которых может быть R (белая ладья), K (черный король), W (белая пешка), B (черная пешка) и O (пустая клетка (символ заглавной буквы o латинского алфавита)).

Формат выходных данных

В ответ выведите единственное число M – минимальное кол-во ходов ладьей, чтобы съесть короля.

Если съесть короля невозможно, выведите -1.

Замечание

В первом примере достаточно двух ходов, при этом неважно походить в нижний левый угол первым ходом и потом съесть короля, или походить в правый верхний (съев пешку) и потом съесть короля - оба варианта дают 2 хода.

Во втором примере нельзя пройти через ряд белых пешек.

Примеры

Ввод	Вывод
3 ROB OOO OOK	2
3 ROB WWW BOK	-1

Пример решения

```
def solution(num, field):
    rook_pos = [-1, -1]

    new_field = ['-' * (num + 2)]
    for x in range(num):
        new_field.append('-' + field[x] + '-')
        if 'R' in field[x]:
            rook_pos = [x + 1, field[x].find('R') + 1, 0]
    new_field.append('-' * (num + 2))

    dist_field = []
    for x in range(num + 2):
        dist_field.append([])
        for y in range(num + 2):
            dist_field[x].append(-1)

    dist_field[rook_pos[0]][rook_pos[1]] = 0
    points = [rook_pos]
    king_not_found = True

    answer = -1
    while king_not_found and points:
        p = points[0]
        points = points[1:]

        for d in ([1, 0], [-1, 0], [0, 1], [0, -1]):
            k = 1
            rook_step = True
```

```
            while rook_step:
                new_p = new_field[p[0] + d[0] * k][p[1] + d[1] *
k]
                new_dp = dist_field[p[0] + d[0] * k][p[1] + d[1]
* k]
                if new_dp == -1 or p[2] + 1 <= new_dp:
                    if new_p == '-' or new_p == 'W':
                        rook_step = False
                    elif new_p == 'B':
                        if p[2] + 1 < new_dp or new_dp == -1:
                            points.append([p[0] + d[0] * k, p[1]
+ d[1] * k, p[2] + 1])
                            dist_field[p[0] + d[0] * k][p[1] +
d[1] * k] = p[2] + 1
                        rook_step = False
                    elif new_p == 'O':
                        if p[2] + 1 < new_dp or new_dp == -1:
                            points.append([p[0] + d[0] * k, p[1]
+ d[1] * k, p[2] + 1])
                            dist_field[p[0] + d[0] * k][p[1] +
d[1] * k] = p[2] + 1
                    elif new_p == 'K':
                        answer = p[2] + 1
                        '''
                        king_not_found = False
                        for x in range(num + 2):
                            print(dist_field[x])
                        '''
                        return answer
                    k += 1
                else:
                    rook_step = False

    return answer


n = int(input())
board = []
for i in range(n):
    board.append(input())
print(solution(n, board))
```