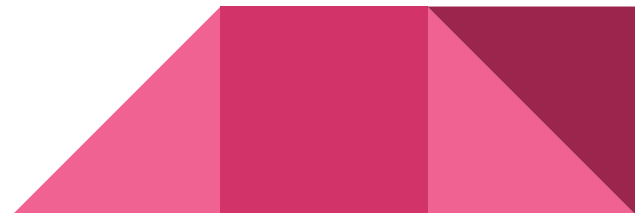


Разбор задач 2 тура 3-го этапа республиканской олимпиады по информатике 2018 года

Тур 2 Задача 1

Условие

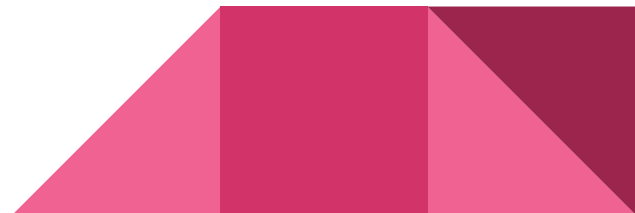
Дана квадратная матрица размера \mathbf{N} , нужно поменять местами две строки, и два столбца, чтобы числа во всех строках и столбцах шли по возрастанию.



Решение на 40 баллов

Полный перебор. Можно перебрать две строки и два столбца и проверить матрицу на соответствие условию.

Сложность $O(N^6)$.

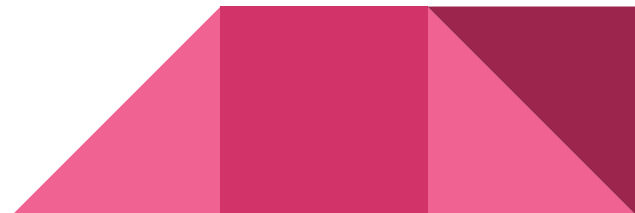


Решение на 80 баллов

Переберем две строки, которые поменяем местами. Тогда во всех столбцах должен установиться правильный порядок. Отсортируем первую строку. Если в ней найдутся два элемента, которые не стоят на своих местах, то мы обязаны поменять их местами и соответствующие им столбцы.

Проверяем матрицу на соответствие условию.

Сложность $O(N^4)$.



Решение на 100 баллов

Заметим, что если мы поменяем местами два столбца, то порядок следования элементов в столбцах не изменится, а если поменяем две строки, то порядок следования элементов в строках не изменится. Следовательно, можно решать для строк и столбцов независимо. Находим пару элементов, которые нужно обменять в первой строке и в первом столбце, получаем ответ.

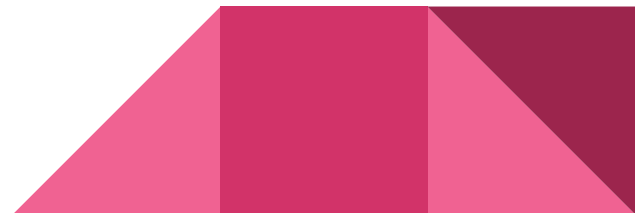
Сложность $O(N^2)$.



Тур 2 Задача 2

Условие

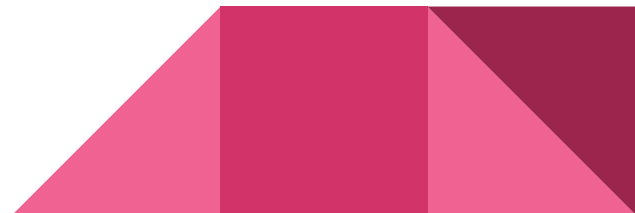
Дана матрица 2^k на 2^k в заданном в условии формате. $k \leq 30$. Требуется найти количество нулей и единиц в этой матрице.



Решение на 50 баллов

Рекурсивно строим матрицу и затем считаем количество нулей и единиц в ней.

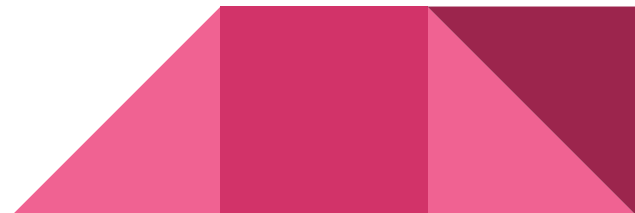
Сложность $O(2^{2k})$.



Решение на 100 баллов

Для каждой цифры (0 или 1) мы можем определить квадрат какого размера она заполняет. Достаточно посмотреть на количество незакрытых открывающихся скобок. Пусть это число **open**. Тогда, образуется квадрат со стороной $2^{(k - \text{open})}$.


Сложность **O(N)**.



Тур 2 Задача 3

Условие

Дана последовательность из N элементов со следующим свойством: для любого $1 < i \leq N$ модуль разности a_i с a_{i-1} не превосходит 1. Также дано K запросов. Каждый запрос задается двумя числами l и r . Нужно было найти максимальную возрастающую подпоследовательность среди элементов a_l, a_{l+1}, \dots, a_r .



Решение на 30 баллов

Делаем то, что просят. Максимальную возрастающую подпоследовательность можно найти с помощью динамического программирования (за квадратичное время или быстрее).

f_i - длина максимальной возрастающей подпоследовательности, которая заканчивается в i -ом элементе.

Сложность $O(K * N^2)$.

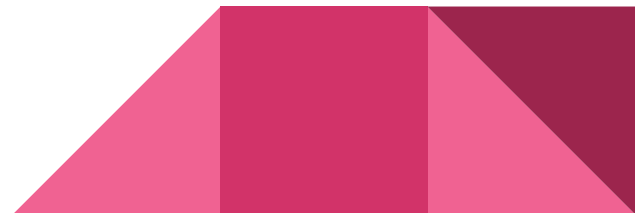


Решение на 60 баллов

На 60 баллов $a_i \leq 100$. Запишем другое динамическое программирование. $f_{i,j}$ - минимальная позиция на которую может оканчиваться возрастающая подпоследовательность длины i и последний элемент которой меньше либо равен j .

Для каждой позиции последовательности можно сохранить ближайшее справа вхождение числа x от 1 до 100.

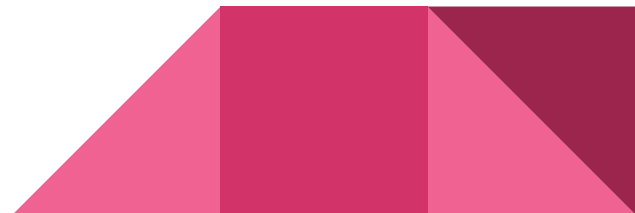
Сложность $O(\max^2\{a_1, a_2, \dots, a_n\} * K)$.



Решение на 100 баллов

Интересный факт. Если есть какие-то i, j ($i < j$) и $a_i < a_j$, то для любого q ($a_i < q < a_j$) существует такое $i < p < j$, что $a_p = q$.

Следовательно, длина максимальной возрастающей подпоследовательности, которая начинается в i и заканчивается в j равна $\max(1, a_j - a_i + 1)$. Построим дерево отрезков, где в каждой вершине будем хранить минимальное и максимальное значения на отрезке, а также максимальную возрастающую на этом отрезке.



Решение на 100 баллов

Построим дерево отрезков, где в каждой вершине будем хранить минимальное и максимальное значения на отрезке, а также максимальную возрастающую на этом отрезке.

Информацию о двух таких отрезках легко сливать (третий параметр высчитывается из значений в левом и правом отрезках и варианта объединения отрезков «максимум справа минус минимум слева»), а значит мы можем и на запрос отвечать за $O(\log N)$ (количество посещенных вершин дерева отрезков).

Сложность **$O(N + K \log N)$** .



Тур 2 Задача 4

Условие


Дана матрица **N** на **M**, а также строка длины **L**. Нужно построить путь в матрице, чтобы буква в первой клетке в пути совпадала с первой буквой строки, буква во второй клетке пути совпадала с второй буквой строки и т.д. Путь – это некоторая последовательность не обязательно соседних клеток матрицы. Длина пути – это суммарное манхэттенское расстояние между соседними клетками в пути. Требуется максимизировать длину пути.



Решения

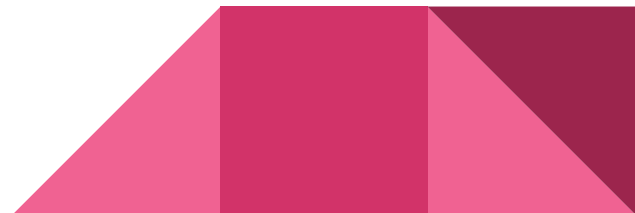
Существует много разных решений, основанных на переборе и случайных подходах. Разберём полные решения.

Напишем динамическое программирование $f_{i,j,k}$ - максимальная длина первых k прыжков, что в итоге мы окажемся в клетке i, j . $f_{i,j,k}$ можно посчитать за $O(N * M * L)$, для этого требуется находить максимум на прямоугольнике с одной из вершин в углу поля. Чтобы восстановить порядок нужно $N * M * L$ памяти, что может не поместиться в оперативную память. Опишем три подхода для решения задачи.



Первый подход

Можно доказать, что можно оставить в каждой строчке только первое и последнее вхождение каждой буквы и прыгать только между ними. Сложность решения это не изменит, но поможет значительно сократить память.



Второй подход

Пересчитывая матрицу по слоям, найдем оптимальный ответ и запомним **W**-ую позицию в обходе, а также стартовую и финишную. Тогда наш путь разделится на две части от старта к **W** и от **W** к финишу. Запустимся от этих путей рекурсивно. Будем делить путь, пока не сможем полностью сохранить путь в динамике.

Сложность $O(N * M * L * \log L)$.



*** Третий подход

Возьмем все вхождения некоторой буквы и построим по ним выпуклую оболочку. Можно доказать, что в оптимальном ответе будут только точки, которые вошли в оболочку. Воспользуемся фактом, что математическое ожидание числа точек в выпуклой оболочке, если точки случайно будут расставлены на поле N на M это $\log N + \log M$. Тогда можно реализовать то же динамическое программирование, но за $O(L * (\log N + \log M)^2)$. Также можно использовать следующее утверждение: если у всех точек целые координаты, то размер выпуклой оболочки не превосходит $O(\sqrt{N * M})$.

