,

Задача A. Жага до краси

Це задача на реалізацію.

Важливо помітити, що немає різниці в якому порядку ми будемо виконувати операції - чи спочатку міняти місцями стовпчики, чи міняти місцями рядки.

Оскільки, нам гарантується, що гарну матрицю завжди можливо отримати за допомогою максимум однієї перестановки стовпчиків й однієї перестановки рядків, можемо знайти наївним алгоритмом, першу пару стовпців, яку треба поміняти місцями (вона буде першої і єдиною) й аналогічним чином знайдемо рядки, які необхідно переставити.

Задача B. Бінарний код

Додатково кожне число візьмемо в дужки. Також запам'ятаємо для кожної відкриваючої дужки, позицію закриваючої. Тоді отримаємо дуже просту рекурсивну формулу:

$$solve(st, l, r, k) = \begin{cases} solve(st, l+1, closes_l - 1, k-1) + solve(st, closes_l + 1, closes_{closes_l+1} - 1, k-1) + ..., if l < r \\ \begin{cases} (2^{2*k}, 0), if st_l = 0 \\ (0, 2^{2*k}) \end{cases} \end{cases}$$

Де $closes_i$- позиція дужки, що закриває дужку на позиції i. Тоді рішення на задачу буде $solve(st, 1, st.size() - 2, k)$

Якщо теперішня матриця складається повністю з нулів або одиниць - тоді вертаємо $(2^{2*k}, 0)$ або $(0, 2^{2*k})$. Інакше, наша матриця розбивається на 4: ЛВ (ліва верхня), ПВ (права верхня), ПН, ЛН. Запустимо для них рекурсивно знаходження відповіді й додамо всі відповіді - це буде рішення для теперішньої матриці.

Задача C. Фотозаняття

Оскільки різниця між сусідніми елементами в масиві не більше, ніж один, тоді розмір найбільшої зростаючої підпослідовності на проміжку від l до r буде $(a_i - a_j + 1) \to max$, де $l \leqslant j \leqslant i \leqslant r$.

Підтримувати цю різницю можна як за допомогою дерева відрізків, так і за допомогою розрідженої таблиці, при чому другий варіант буде набагато ефективнішим, так як в розрідженої таблиці набагато менша константа операцій, ніж в дереві відрізків.

Принцип, за яким будемо з'єднувати відповіді дуже простий. Нехай ми знаємо для відрізка $[l, m]$ й відрізка $[m+1, r]$ наступне: мінімальний елемент на цьому відрізку(mn), максимальний елемент (mx) й довжину найбільшої зростаючої підпослідовності (ans). Тоді значення для відрізка $[l, r]$ обрахуємо наступним чином:

$mn = min(mn_l, mn_r); mx = max(mx_l, mx_r); ans = max(ans_l, ans_r, mx_r - mn_l + 1)$, де mn_l, mx_l, ans_l - мінімум, максимум й довжина НЗП в лівому відрізку, а mn_r, mx_r, ans_r - в правому.

Таким чином, загальна асимптотика такого рішення $O(q * log_2(n))$

Задача D. Клавіатура

Придумаємо наступну динаміку: для кожної літери на клавіатурі будемо зберігати найдовшу сумарну відстань набраного текста на кроці i. Тоді перерахунок динаміки буде займати на кожному кроці найгіршому випадку $O((n*m)^2)$ операцій. Всього літер в тексті - l, тоді сумарна асимптотика такого рішення буде $O(l * (n*m)^2)$.

Звичайно, запустивши таке рішення на останньому тесті, результат ми отримаємо не раніше, ніж через 2 роки. Тоді можна зробити дуже просту, але ефективну оптимізацію. Помітимо, що нам майже ніколи не вигідно переходити в літеру, яка знаходиться на центрі клавіатури, якщо є можливість перейти в якусь літеру, яка знаходиться в куті клавіатури. Тоді для кожного кута клавіатури будемо зберігати не більше, ніж 50 найближчих позицій кожної літери алфавіту. Таким чином, загальна кількість позицій кожної літери не буде більше, ніж 200. Тоді асимптотика нашого рішення буде $O(l * 200^2)$. Таке рішення дозволить отримати максимальну кількість балів на кожному тесті.