Задача A. Заявки

Щоб було n сторінок з вакансіями, треба, щоб на $n-1$ з них було по 20 вакансій, і ще хоча б одна, щоб створити нову сторінку — $20 \cdot (n-1)+1$. Щоб не створити $n+1$-у сторінку, треба, щоб було менше ніж $20 \cdot n+1$ вакансій, тобто максимальна кількість — $20 \cdot n$.

Задача B. Хрестики-нулики

Перше, що треба зробити — перевірити, що поля відрізняється не більше ніж в одній позиції. Якщо в більше ніж одній, то неможливо за 1 крок змінити два поля, і відповідь `NO`. Якщо два поля рівні, то відповідь `YES`, бо можна отримати поле B не зробивши жодного кроку.

Тепер маємо позицію (x, y) яка відрізняється у двох полях. Перевіримо, чи $A_{x,y} =$ `'.'`, якщо ні, то неможливо поставити знак в непусте поле, і відповідь `NO`. Інакше, подивимось на те, хто мав ходити наступним у полі A, хрестики чи нулики? Це можна зробити порахувавши кількість нуликів і хрестиків, якщо рівно, то хрестики, інакше нулики. Залишається лише перевірити, що поставилось саме те, що треба було.

Задача C. Дискотека

Навчимось перевіряти, чи можна вибрати набір пісень, що в нього відношення суми рейтингу до суми тривалості більше або дорівнює x.

$$\frac{\sum_{i \in X} r_i}{\sum_{i \in X} t_i} \geqslant x$$

$$\sum_{i \in X} r_i \geqslant x \cdot \sum_{i \in X} t_i$$

$$\sum_{i \in X} (r_i - t_i \cdot x) \geqslant 0$$

Таким чином, завжди оптимально брати k пісень, які мають найбільше значення різниці $r_i - t_i \cdot x$.

Зробимо бінарний пошук по відповіді, будемо проходитись по масиву, сортувати й шукати суму перших k. Якщо сума більше ніж 0, то можна отримати відповідь рівну x, інакше ні.

Загальна асимптотика — $\mathcal{O}(N \log A \log N)$.

Задача D. Послідовність

Будемо йти зліва направо по масиву і тримати значення dp_a — максимальна довжина підпослідовності, що закінчується в елементі зі значенням a. Тоді можна перераховувати $dp_{a_i} = \max(dp_{a_i}, dp_j + 1)$, так що $l \leqslant j + a_i \leqslant r$.

Що робити тепер? Перераховуймо dp за допомогою дерева відрізків. Будемо запитувати максимум на відрізку $[l - a_i; r - a_i]$, і оновлювати в точці a_i. Це можна робити за допомогою неявного дерева відрізків, або компресії координат.

Загальна асимптотика — $\mathcal{O}(N \log N)$ з компресією.

Задача E. Матчі

Перефразуємо задачу. Нам дано список ребер, треба розбити на відрізки цей список, щоб враховуючи ребра з кожного відрізку окремо виходив двудольний граф.

Давайте будемо йти й жадібно добирати ребра в граф, поки це ребро нічого не псує. Як ми можемо швидко перевірити, чи псує нам ребро двудольність? Скористаємось СНМ. Будемо в кожній вершинці підтримувати предка і парність довжини шляху до кореня. Таким чином ми "пофарбували"всі вершинки у компоненті.

Коли ми додаємо ребро, що об'єднує дві компоненти, то ми його завжди можемо додати. Інакше, треба перевірити парність відстані до кореня, якщо однакова, то нічого не робимо, інакше розпочинаємо новий відрізок.

СНМ можна модифікувати й перераховувати відстань до кореня під час пошуку кореня. Коли дві компоненти об'єднуються, ми фактично додаємо ребро не $(a; b)$, а $(par[a]; par[b])$. Тому треба сказати, що парність відстані від $par[a]$ до $par[b]$ дорівнює парності суми $dst[a] + dst[b] + 1$.

Автор усіх задач: Андрій Столітній.