Розбір задачі «Козак Вус і важлива знахідка»

Можна виписати значення важливості таблички для кожного з чотирьох її можливих положень, та вибрати із них перше з максимальним значенням.

Розбір задачі «Козак Вус і цікава задача»

Розглянемо випадок, коли ми зробимо два вертикальні розрізи кімнати (випадок, коли ми робимо два горизонтальні, аналогічний). Тоді ми отримаємо кімнати зі сторонами $n \times x$, $n \times y$ та $n \times z$, де $x, y, z > 0$ та $x + y + z = m$. Тоді загальний периметр буде рівний $p = 2 \cdot (n + x + n + y + n + z) = 6n + 2 \cdot (x + y + z) = 6n + 2m$. Отже, він не залежить від позицій де ми будемо робити розрізи. Тоді, якщо $m > 2$, існує відповідь, і нам підійде варіант зробити кімнати розмірами: $n \times 1$, $n \times 1$, $n \times (m - 2)$.

Другий випадок — це коли ми робимо один вертикальний (горизонтальний) розріз, а потім одну з кімнат ділимо на дві горизонтальним (вертикальним) розрізом. Припустимо, що ми робимо перший розріз у позиції x, тобто ділимо кімнату на дві із розмірами $n \times x$ та $n \times (m - x)$. Тоді, припустимо, що далі ми будемо ділити другу кімнату у позиції y. Загальний периметр буде рівний $p = 2 \cdot (n + x + m - x + y + m - x + n - y) = 2 \cdot (2n + 2m - x)$, тобто периметр не залежить від y. Звідси, ми можемо отримати x. Якщо $n > 1$ та $1 \leqslant x < m$, то відповідь існує, і один із можливих варіантів це кімнати із розмірами: $n \times x$, $1 \times (m - x)$, $(n - 1) \times (m - x)$.

Розбір задачі «Козак Вус і НСД»

Розглянемо рішення для першого блоку. Зробимо за мінімальну кількість кроків так, щоб числа йшли у порядку неспадання, тобто $a_i \geqslant a_{i-1}$. Це ми можемо зробити жадібно, перебираючи числа зліва направо, та збільшуючи відповідне число так, щоб воно стало більше або рівним за попереднє. Очевидно, що після цього, усі числа будуть парними, а отже НСД усіх чисел буде більший за 1.

Розглянемо рішення на передостанній блок. Припустимо, що НСД усіх чисел після усіх змін буде рівне x. Тоді ми можемо трохи змінити рішення для першого блоку. Перебираючи числа зліва направо, ми спочатку збільшимо число так, щоб воно було не менше за попереднє, а потім збільшимо його до наступного числа, що ділиться націло на x. Для фіксованого x ми можемо знайти відповідь за n операцій, отже x можна перебирати від 1 до 10^4. Таке рішення буде працювати за $O(n^2)$.

Щоб вирішити останній блок можна було помітити, що перебирати на місце x нам можна лише прості числа, приблизна кількість яких до 10^4 це 10^3. Таке рішення буде заходити на повний бал.

Розбір задачі «Козак Вус і Потоколяндія»

Для початку, трошки змінимо умову. Якщо змінити порядок подій на протилежний, тоді події можуть мати вигляд:

1. Певна дорога стає розчищеною, тобто по ній тепер можна пересуватись.
2. До якогось будинку прилітає певна кількість людей.

Тоді відповідь на запит «`x y z`» — це $d-$ (перший день коли сумарне щастя друзів x та y не менше за z), де d — це загальна кількість днів.

Розгляньмо таке рішення для окремого запиту, що є достатньо повільним, але воно нам знадобиться для повного рішення. Для фіксованого запиту «`x y z`», будемо шукати відповідь бінарним пошуком. Припустимо, що відповідь — це d. Тоді, ініціалізуємо структуру даних СНМ (Система неперетинних множин), та обробимо перші d подій. Тобто це або об'єднати дві компоненти у графі, або збільшити значення однієї компоненти. Після цього перевіримо чи достатньо цих запитів для того, щоб сумарне щастя людей x та y було не менше за z. Таке рішення для окремого запиту буде працювати за $O(d \cdot a \cdot \log d)$, де a — це час обробки кожної зміни в СНМ.

Тепер скористаємось методом паралельного бінарного пошуку. Для кожного запиту у нас є $\log(d)$ ітерацій, на кожній з яких ми перевіряємо, чи підходять нам перші w подій. Об'єднаймо пошук відповіді для кожного запиту на кожній ітерації. Тобто на певній ітерації ми для кожного запиту підтримуємо відрізок (l_i, r_i), на якому лежить відповідь до нього, і перевіряємо, у якій половині цього відрізка буде лежати відповідь на наступній ітерації. Для кожного запиту зафіксуємо $mid_i = \frac{l_i + r_i}{2}$ —

позицію яку нам треба перевірити. Тоді, будемо оброблювати усі події зліва на право, та після j-ї зміни ми перевіримо відповідь для усіх запитів у яких $mid_i = j$. Таке рішення буде працювати за $O(\log d \cdot a \cdot (d + s))$.

СНМ: http://bit.ly/2t1HxYT

Паралельний бінарний пошук: https://codeforces.com/blog/entry/45578

Розбір задачі «Козак Вус і свята»

Для початку, відсортуємо усі свята у порядку неспадання дати.

Тепер розглянемо рішення для перших трьох блоків. Будемо зберігати масив dp_i — максимальна кількість свят, які ми можемо відвідати якщо останнє свято яке ми відвідаємо буде i. Будемо перебирати наступне свято j ($j > i$). Тоді, якщо ми встигаємо дістатись від свята i до свята j, тобто День святкування свята i+ відстань між містами де проходять ці свята $\leqslant$ День святкування свята j, тоді $dp_j = max(dp_j, dp_i + 1)$. Час роботи залежить від швидкості знаходження відстані між містами, яку можна підрахувати через ДФС для кожного міста.

Перейдемо до рішення на повний бал. Побудуємо Центроїдну декомпозицію, та структуру що здатна за асимптотику $\log(n)$ знаходити відстань між двома вершинами у дереві. Тепер по черзі будемо оброблювати свята та рахувати масив dp. Припустимо, що наше свято відбувається у вершині v у день d. Будемо перебирати вершину p — будь-який предок вершини v у Центроїдній декомпозиції. Звідси, ми припускаємо, що після попереднього свята ми спочатку прийдемо до вершини p, а потім до вершини v. Давайте для кожної вершини зберігати множину подій (day, ans) — ми прийшли у цю вершину у день day, та вже відвідали ans свят. Тоді, нас цікавить подія із максимальним значенням ans, серед подій у яких $day \leqslant d - dist(v, p)$, де $dist(x, y)$ — відстань між містами x та y у початковому дереві. Так ми можемо отримати dp_i. Тепер ми повинні оновити події у вершинах. Знову будемо перебирати предка p нашої вершини у центроїдній декомпозиції, та додавати до його множини подію $(d + dist(v, p), dp_i)$.

Залишилось розібратись як оптимально підтримувати події. Помітимо, що якщо свято відбувається у вершині v, тоді ми знаємо всі перші значення подій які ми додамо, тобто їх параметр day. Тоді ми можемо у кожній вершині зберігати відсортований вектор $days$ — усі можливі значення першого параметру для цієї вершини. Після цього в кожній вершині ми можемо побудувати Дерево Фенвіка, щоб швидко оновлювати та знаходити максимум на префіксі.

Асимптотика роботи $O(m \cdot \log n \cdot \log m)$.

Центроїдна декомпозиція: https://www.geeksforgeeks.org/centroid-decomposition-of-tree/

Дерево Фенвіка: http://bit.ly/2TrxVCb