Розбір задачі «Козак Вус і екзамен»

В задачі просто необхідно правильно реалізувати визначення оцінки Вуса яке описане в умові.

Розбір задачі «Козак Вус і гармата»

Якщо за x позначити відрізок над якому знаходиться Козак, а за y висоту на якій він знаходиться, то можна імітувати всі постріли Вуса наступним чином. Спочатку $x = 1, y = 0$ (знаходиться на першому відрізку на висоті 0). Після пострілу ми кожен раз перевіряємо, чи ми впадемо на поточному відрізку, чи полетимо далі: якщо $a_{x+1} \leqslant y$ або $x = n$ ми впадемо на вірізку x, а інакше полетимо далі, причому x та y збільшаться на одиницю. Якщо ж ми падаємо, то робимо y рівним 0 і збільшуємо кількість пострілів на одиницю. Далі все повторюється допоки $x \neq n$.

Розбір задачі «Козак Вус і календар»

Помітимо, що впорядкована трійка чисел є коректною датою для конкретного формату якщо $a[M] \geqslant$ Д. Тоді x.y.z є коректною датою для всіх форматів якщо виконуються наступні умови: $a[x] \geqslant y$, $a[y] \geqslant x$, $a[x] \geqslant z$, $a[z] \geqslant x$, $a[y] \geqslant z$, $a[z] \geqslant y$ (кожна пара чисел означає номер місяця і номер доби в деякому форматі).

Рішення за $O(n^3)$: перебираємо усі трійки чисел і перевіряємо чи задовільняють вони всі умови.

Рішення за $O(n^2)$: перебираємо числа x та y, що задовільняють умови $a[x] \geqslant y$ та $a[y] \geqslant x$. Тоді на число z отримуємо такі обмеження: $a[z] \geqslant max(x, y)$ і $min(a[x], a[y]) >= z$.

Підрахуємо масив $dp[i][j]$ — кількість чисел z таких, що $z \leqslant i$ і $a[z] \geqslant j$. Масив рахується за простою формулою $dp[i][j] = dp[i-1][j] + (a[i] \geqslant j)$, де $(a[i] \geqslant j)$ дорівнює 1, якщо $a[i] \geqslant j$ і дорівнює 0 інакше.

Тоді для зафіксованих чисел x та y $dp[min(a[x], a[y])][max(x, y)]$ буде кількістю чисел z, що задовільняють усі умови.

Розбір задачі «Козак Вус і домноження»

Помітимо, що якщо $x = p_1^{l_1} \cdot p_2^{l_2} \cdot ... \cdot p_w^{l_w}$, де p_i — різні прості числа, а l_i — натуральні числа (розклад числа x на прості множники), то за 1 операцію домноження можно перетворити x на будь-яке $y = p_1^{t_1} \cdot p_2^{t_2} \cdot ... \cdot p_w^{t_w}$, де $l_i \leqslant t_i \leqslant 2 \cdot l_i$. Тоді за k домножень можна перетворити x на будь-яке $y = p_1^{q_1} \cdot p_2^{q_2} \cdot ... \cdot p_w^{q_w}$, де $l_i \leqslant q_i \leqslant 2^k \cdot l_i$.

Зрозуміємо, у якому випадку відрізок (пара описує відрізок чисел $a_l, a_{l+1}, ..., a_r$ з масиву) l, r є «гарним». По-перше, розклад на прості множники всіх чисел $a_l, a_{l+1}, ...a_r$ повинен складатися з одних і тих самих простих чисел (нехай це будуть $p_1, p_2, ..., p_w$). Тобто, відрізнятися розклади можуть лише степінню входження того чи іншого простого числа. Скажемо, що min_i — мінімальне таке число, що у розкладі деякого числа з відрізку, число p_i входить у степені min_i. Аналогічно позначимо за max_i максимальне таке число. Тоді відрізок є «гарним», якщо виконується $max_i \leqslant 2^k \cdot min_i$ $(1 \leqslant i \leqslant w)$.

Тепер зрозуміло, що якщо відрізок l, r є «гарним», і виконується $r - l \geqslant 1$, то відрізки $l + 1, r$ та $l, r - 1$ також є гарними. Тоді ми можемо можемо двома вказівниками підтримувати максимальний «гарний» відрізок. Для перевірки «гарності» відрізку будемо для кожного простого числа використовувати дерево відрізків яке буде знати мінімальне та максимальне входження простого числа на відрізку (входження у розкладі на прості множники). Так як в умові сказано, що прості дільники менші за 30, ми можемо використовувати всього 10 дерев відрізків (існує рівно 10 простих чисел менших за 30).

Розбір задачі «Козак Вус і країна»

Розв'язок: Помітимо, що країна має вигляд дерева (один із видів графів), при чому означення регіону з центром у місті v еквівалентне означенню піддерева вершини v. Тому далі все буде у термінах графа. Зокрема, a_x — число записане в вершині x, а «переселення» — обмін значеннями, записаними у вершинах x та y.

Введемо sz_v - кількість вершин в поддереві v. Далі ділення усюди цілочисельне.

По-перше, помітимо, що медіана піддерева v рівна x тоді й тільки тоді, коли в піддереві v присутнє число x, кількість чисел менших за x рівна $\frac{sz_v}{2}$, а кількість чисел більших за x рівна $\frac{sz_v - 1}{2}$.

Зафіксуємо піддерево v та розглянемо числа x, для яких воно «гарне». Введемо позначення:
Масив A - відсортовані значення всього дерева.
Масив B - відсортовані значення піддерева v.
Обидва масива 1-індексовані.
$A[0]$ поставимо рівним $-inf$, а $A[n+1]$ поставимо рівним $+inf$.

Перше обмеження на число x:
Числа $x \leqslant A[\frac{sz_v}{2}]$ не підходять, бо скільки б «переселень» ми не зробили, кількість чисел менших за x в піддереві буде менша за $\frac{sz_v}{2}$. Числа $x \geqslant A[n+1-\frac{sz_v-1}{2}]$ не підходять, бо кількість чисел більших за x в піддереві завжди буде менша за $\frac{sz_v-1}{2}$. Для усіх чисел $A[\frac{sz_v}{2}] < x < A[n+1-\frac{sz_v-1}{2}]$, присутніх в масиві A, при достатній кількості «переміщень» (наприклад $k = n$), піддерево v було б «гарним».
Отже, перше обмеження на число x: $A[\frac{sz_v}{2}] < x < A[n+1-\frac{sz_v-1}{2}]$

Друге обмеження:
Тепер за $L[x]$ позначимо мінімальну кількість «переселень», яка необхідна для «гарності» піддерева при умові, що x присутнє в дереві. Розглядати $L[x]$ будемо лише для таких x, для яких виконується перше обмеження.
Введемо $mid = \frac{sz_v}{2} + 1$. Як ми бачимо, якщо не робити «переселень», то число $B[mid]$ і буде медіаною піддерева, а отже, $L[mid] = 0$.

Доведемо, що для будь-якого $m > 0$, для всіх $B[mid-m] <= x <= B[mid+m]$ виконується $L[x] <= m$. Причому $L[B[mid-m]] = L[B[mid+m]] = m$.
Покажемо, що для будь-якого $m > 0$, для всіх $B[mid+m-1] < x <= B[mid+m]$ виконується $L[x] = m$. За таких умов, кількість чисел менших за x в піддереві на m більше ніж потрібно. Отже, ми m чисел менших за x з піддерева обміняємо з m числами більше або рівними за x не із піддерева. Причому, якщо x не знаходилося в піддереві, то його необхідно свапнути першочергово.
Покажемо, що для будь-якого $m > 0$, для всіх $B[mid-m] <= x < B[mid-m+1]$ виконується $L[x] = m$. Розглянемо окремо випадки $x = B[mid-m]$ й $x > B[mid-m]$:

- При $x = B[mid-m]$ кількість чисел менших за x на m менша ніж потрібно. Так як число x уже присутнє в піддереві, нам достатньо m свапів.
- При $B[mid-m] < x < B[mid-m+1]$ кількість чисел менших за x на $(m-1)$ менша ніж потрібно. Але через те, що x не присутнє в піддереві, нам необхідно використати одне «переселення» для того, щоб обміняти число x і будь-яке число з піддерева, яке більше за x. Отже, в цьому випадку знову необхідно m «переселень».

Твердження доведено, а тому друге обмеження на число x наступне: $B[mid-k] <= x <= B[mid+k]$, де $mid = \frac{sz_v}{2} + 1$.

Отже, конкретне піддерево v є «гарним» для усіх чисел які задовільняють обидві умови, а це усі числа з перетину відрізку $[A[\frac{sz_v}{2}]+1, A[n+1-\frac{sz_v-1}{2}]-1]$ та відрізку $[B[mid-k], B[mid+k]]$. Тоді наш алгоритм рішення: для кожного піддерева знаходимо відрізок чисел, для яких він є «гарним». Відповіддю на запит буде кількість відрізків, що перетинають x.
Тоді рішення за $O(n^2 + mlogn)$: Знаходимо масив A. Знаходимо масив sz. Перебираємо вершину v, знаходимо масив B. За формулами знаходимо необхідний відрізок. "Кількість відрізків, що перетинають x" є стандартною задачею з багатьма рішеннями. Одним із них є знаходження кількості відрізків у яких права границя менша за x, або ліва більша за x. Усі інші відрізки перетинають x.
Рішення за $O(nlogn + m)$ або $O(nlgon + mlogn)$ Помітимо, що в масиві B ми шукаємо лише два значення: $B[mid-k]$ та $B[mid+k]$. Також необхідно знати, що якщо виписати в масив C

ейлерів обхід дерева, то вершини конкретного піддерева будуть утворювати підмасив C. А отже, B для конкретного v - відсортований підмасив C. Ми звели задачу до такої: нам потрібно знайти $(mid - k)$-те та $(mid + k)$-те по величині число на відрізку в масиві C). Така задача вирішується такою структурою даних як персистентне дерево відрізків. Для кожного префіксу масива C ми зберігаємо свою версію дерева відрізків. Тоді якщо ми шукаємо відповідь на відрізку l, r, то різниця версій r та $l - 1$ і буде шуканим підмасивом.

Розбір задачі «Себек та рівняння»

Просто перевіряємо всі можливі варіанти іфами. Їх буде $3! = 6$

Розбір задачі «Себек і сусіди»

Будемо йти циклом по всім клітинкам зліва направо зверху вних і зберігати номер клітинки, яку ми зараз розглядаємо. Якщо для якихсь i та j номер клітинки співпадає з номером з запиту перевіряємо:

1. Якщо $i = j$, то клітинка знаходиться на головній діагоналі
2. Якщо $i = n - j + 1$, то клітинка знаходиться на другорядній діагоналі

Розбір задачі «Ух і сусіди»

Для початку знайдемо можливі довжини стрибків Себека. Будемо йти по стрічці і зберігати величину k — кількість символів до i включно, які співпадають з символом i. Дивимось на наступний символ, ж два варіанти:

1. Наступний символ співпадає з i-тим. Тоді просто збільшуємо k на 1 і йдемо далі.
2. Наступний символ не співпадає з i-тим. Тоді перевіряємо чи $k \leqslant 3$, запам'ятовуємо нову можливу довжину стрибка, ставимо $k = 1$ (наступний символ, що відрізняється починає новий відрізок послідовних однакових символів) і йдемо далі.

Тепер ми маємо тільки число n і знаємо можливі довжини стрибків. Тепер просто іфами перевіряємо і шукаємо найкращу відповідь. Наприклад, якщо n не ділиться на 3, а ми можемо стрибати тільки на 3 то відповідь «-1». А якщо n не ділиться на 2 а ми можемо стрибати на 1 і 2 то відповідь «$\frac{n-1}{2} + 1$»

Розбір задачі «Ілля ремонтує Пассат»

Нашому герою належить виконати n замовлень, кожне з яких має кількість розміщень t_i і дату завершення d_i. Відомо, що він обов'язково встигне зробити завдання вчасно, і він хоче лише почати роботу якомога пізніше.

У задачах такого типу головна складність полягає у визначенні найкращого порядку, в якому можна виконувати завдання. У нашому випадку, це просто та інтуїтивно зрозуміло: завдання слід опрацьовувати відповідно до їх термінів. Формально це може бути доведено так: якщо завдання не виконувалися в порядку їх закінчення, то завдання з пізнішою датою виконання було виконано безпосередньо перед іншим завданням з більш ранньою датою. Тоді ми могли б поміняти порядок виконання цих двох завдань, і ми отримали б ще одне, не гірше рішення.

Ми будемо виконувати завдання в порядку збільшення термінів завершення. Залишається зрозуміти скільки перших днів ми можему нічого не робити. Це легко зробити бінарним пошуком по відповіді, оскільки якщо ми можемо не робити нічого перші x днів, то і $x - 1$ теж підходить. Маємо рішення за $\mathcal{O}(nlogn)$, що з заданими в задачі обмеженнями набирає повний бал.

Також існує рішення за $\mathcal{O}(n)$, і не одне, але це на домашнє завдання.

Розбір задачі «Себек мстить сусідам»

Для простоти зробимо в початку $m = m + 1$

Подивимось, скільки жаб треба запустити, щоб напевно вигнати всіх сусідів з усіх квартир. Просто запустимо жаб у кожну m-ту квартиру. Бачимо, що цього вистачить, тому що всі сусіди з наступних $m - 1$ квартир після запуску переходять в m-ту наступну, куди ми і запускаємо жабу настпний раз. Таким чином, якщо $k \geqslant \lceil \frac{n}{m} \rceil$, то відповідь на задачу просто кількість всіх сусідів у всіх квартирах. Також, з цього ми можемо помітити, що вигідно між двома запусками робити відстань принаймні m.

Помітимо декілька цікавих речей. Як уже було сказано вигідно між двома запусками робити відстань принаймні m, тому що після одного сусідів в наступних і попередніх $m - 1$ квартирах

немає. Тепер якщо ми запускаємо жабу в m-ту квартиру то виганяємо всіх, що сюди переїхали і так само наступні $m-1$ переїзжають в наступну m-ту квартиру. Якщо ж ми запустили жабу в $m+1$-шу наступну квартиру або далі, то ми створюємо новий такий відрізок, що складається з запусків, які виконуються кожні m квартир. Тобто кожен такий відрізок запусків на відстані рівно m можна розглядати окремо, бо він не впоиває на інший відрізок, якщо відстань до нього принаймні $m+1$.

Зараз ми вже можемо придумати ефективне розв'язання для задачі. Напишемо динамічне програмування $dp[i][k][last]$ — ми пройшли перші i квартир, зробили k запусків жаб а також останній запуск був в квартирі з номером $i - last$. Очевидно, що нам треба зберігати значення тільки для $0 \leqslant last \leqslant m$ тому що, запуски жаб, зроблені далі ніж на m не впливають на той, що ми можемо зробити в квартирі i. Але треба звернути увагу на те, що квартири розташовані по колу, тому запуски жаб в перших m квартирах можуть вплинути на запуски в останніх m. Цю проблему можна вирішити декількома способами. Можна додати ще один параметр — в яку з перших m квартир ми робили постріл. Або завжди сріляти в першу, тільки рішити задачу окремо для перших m циклічних зсувів масиву квартир і вибрати кращий варіант. Також треба врахувати, що ми можемо взагалі не запустити жабу в перші m квартир.

В динаміці маємо такі переходи:

1. $dp[i][k][0] = max(dp[i-1][j-1][m-1] + a[i-m+1] + a[i-m+2] + \ldots + a[i-1] + a[i],\ dp[i-1][j-1][m] + a[i])$
2. $dp[i][k][j] = dp[i-1][k][j-1]$ для $1 \leqslant j < m$
3. $dp[i][k][m] = max(dp[i-1][k][m-1],\ dp[i-1][k][m])$

Також в залежності, від того як ви опрацюваєте те, що в нас квартири розташовані по колу, треба до $dp[i][k][0]$ іноді додавати якесь значення типу $a[i+1] + a[i+2] + \ldots + a[i+t]$, в залежності від того, чи були зроблені запуски жаб в перші m квартир.

В динаміці маємо $\mathcal{O}(n * k * m)$ станів і $\mathcal{O}(1)$ переходів з кожного. Отже сумарна асимптотика $\mathcal{O}(n * k * m)$.