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 $\mathcal{O}(n * k * m)$.