Задача A. Новорічні подарунки

Помітимо, що під час алгоритму нам ніколи не буде вигідно змінювати число c_i.

Тоді розглянемо такі варіанти:

1) $(c_i \leqslant y_i \leqslant x)$ - в такому випадку, ми будемо весь час віднімати 1, допоки $x \neq y$. Коли $x = y$ оберемо максимально можливе d, адже потім ми не зможемо нічого віднімати, бо x стане менше за y. Тоді d стане рівним $\lfloor \frac{y}{c_i} \rfloor$. І кінцевий x, який в нас залишиться, буде $y - \lfloor \frac{y}{c_i} \rfloor$.

2) $(y \leqslant c_i \leqslant x)$ - в такому випадку, віднімаємо 1, допоки x не стане c_i, а потім віднімемо $c_i/c_i = 1$. Таким чином, кінцевий $x = c_i - 1$

3) $(c_i \leqslant x < y)$ - x змінюватися не буде, бо він менше за y

4) $(y \leqslant x < c_i)$ - x змінюватися не буде, бо він менше за c_i $(\lfloor \frac{x}{c_i} \rfloor = 0)$

5) $(x < y \leqslant c_i)$ - аналогічно випадку 3 і 5.

Розібравши всі ці випадки й пам'ятаючи, що змінювати c_i під час виконання алгоритму ми не будемо, знайдемо такий c_i, за допомогою якого, ми, врешті-решт, отримаємо мінімальний x.

Таким чином, кінцева складність алгоритму - $O(n)$.

Задача B. Надійна мережа

Можна помітити, що будь-який шлях довжиною 4 буде або сам потрібним шляхом, або його підшлях буде відповіддю.

Всі можливі варіанти шляхів довжиною 4 :

$[a, a, a, a]$ - підшлях $[a, a]$ задовільняє відповідь;
$[a, a, a, b]$ - підшлях $[a, a]$ задовільняє відповідь;
$[a, a, b, a]$ - підшлях $[a, a]$ задовільняє відповідь;
$[a, a, b, b]$ - підшлях $[a, a]$ задовільняє відповідь;
$[a, b, a, a]$ - шлях $[a, a]$ задовільняє відповідь;
$[a, b, a, b]$ - шлях $[a, b, a, b]$ задовільняє відповідь;
$[a, b, b, a]$ - підшлях $[b, b]$ задовільняє відповідь;
$[a, b, b, b]$ - підшлях $[b, b]$ задовільняє відповідь;
$[b, a, a, a]$ - підшлях $[a, a]$ задовільняє відповідь;
$[b, a, a, b]$ - підшлях $[a, a]$ задовільняє відповідь;
$[b, a, b, a]$ - шлях $[b, a, b, a]$ задовільняє відповідь;
$[b, a, b, b]$ - підшлях $[b, b]$ задовільняє відповідь;
$[b, b, a, a]$ - підшлях $[b, b]$ задовільняє відповідь;
$[b, b, a, b]$ - підшлях $[b, b]$ задовільняє відповідь;
$[b, b, b, a]$ - підшлях $[b, b]$ задовільняє відповідь;
$[b, b, b, b]$ - підшлях $[b, b]$ задовільняє відповідь.

Таким чином, достатньо з кожної вершини перевірити чи існує шлях довжиною 4. Як ми бачимо, їх всього $2^4 = 16$, тому в найгіршому випадку така перевірка буде працювати за $O(16 * n)$.

Можливий варіант, коли з жодної з вершин немає шляху довжиною 4. Цей випадок можна опрацювати, перевіряючи чи є шлях довжиною 2, в якого дві літери однакові, під час пошуку шляху довжиною 4.

Задача C. Зіткнення галактик

Для об'єднання галактик будемо використовувати структуру даних DSU (англ. Disjoint Set Union, укр. система множин, що не перетинаються). Будемо використовувати її для визначення головної зірки в галактиці.

Для кожної головної зірки, будемо зберігати `set` всіх зірок, які знаходяться в галактиці й яскравість медіанної зірки.

Тоді для відповіді на запит типу 2 потрібно буде знайти головну зірку в галактиці й вивести med_x, де x - головна вершина.

Щоб опрацювати запит типу 1, потрібно під час об'єднання галактик ще змінювати медіану. Будемо проходитися по всім зіркам 2 галактики й додавати їх в першу. Тоді нехай x - зірка другої галактики, яку ми зараз додамо в першу. Тоді медіана зсунеться праворуч, якщо розмір першої галактики парний, а $x > m$, де m - медіана першої галактики й зсунеться ліворуч, якщо розмір

першої галактики непарний, а $x < m$. Для того, щоб зсунути медіану ліворуч або праворуч, можемо використовувати звичайний бінарний пошук на сеті (за допомогою `lower_bound` і `upper_bound`)

Тоді загальна асиптотика, яку ми отримаємо буде $O(n*q*log_2(n))$, але, якщо ми будемо додавати елементи з меншої галактики в більшу, то отримаємо асиптотику $O(q * log_2(n) * log_2(n))$

Задача D. Міста на Венері

Представимо гру у вигляді послідовності типу $l_1 > l_2, \ldots, l_k$, де k - кількість слів, які були вибрані під час гри, а l_i перша літера обраного слова. Можна помітити, що всі підпослідовності типу $(a > b > a), (a > c > a)$ ніяк не впливають на результат гри, тому ми їх можемо відразу пропустити (Гравець 1 робить перехід $(a > b)$, а Гравець 2 робить зворотній перехід $(b > a)$). Також потрібно помітити, що для переходів типу $(a > a)$ нам важлива тільки парність кількості доступних переходів, і якщо вона парна, то вона ні на що не впливає (Гравець 1 робить перехід, а Гравець 2 робить зворотній).

Врешті-решт, видаливши всі зайві переходи, можемо написати звичайну рекурсію, в якій буде в найгіршому випадку 3 розгалудження (перехід $(a > a)$, перехід $(b > b)$, перехід $(c > c)$, і після кожного з таких переходів буде ще одне розгалудження $(l_{prev} > l_{prev} > (\{a, b, c\} \cap \{l_{prev}\})_0)$ або $(l_{prev} > l_{prev} > (\{a, b, c\} \cap \{l_{prev}\})_1)$.

Наприклад, у нас є послідовність $(a > b > c > a > b >?)$. Тоді на місці знака питання не може бути a, бо утвориться підпослідовність $(a > b > a)$, яку ми вже видалили. Залишається лише розгалудження $(a > b > c)$ або $(a > b > b)$. З другого випадку з'явиться ще одне розгалудження $(a > b > b > a)$ або $(a > b > b > c)$ Розгалудження $(a > b > b > b)$ утворитися не може, бо максимальна кількість переходів $(b > b)$ дорівнює один, оскільки нас цікавить лише парність.

Таким чином загальна кількість розгалуджень буде 6, тому складність алгоритму буде $O(6 * n)$. Перебравши всі можливі стартові літери отримаємо кінцеву асимптотику $O(18 * n)$. Оскільки максимальне значення $n \approx 3460$ (це випливає з того, що слова не повторюються й загальна їх довжина 2×10^6), то це рішення буде працювати з дуже великим запасом часу.