Розбір задачі «Рядки та запити»

Спочатку відсортуємо усі рядки в лексикографічному порядку. Тоді, якщо маємо якийсь рядок w, усі рядки, префікс яких рівний відповідному рядку, будуть знаходитись поруч у відсортованій послідовності. Тобто усі рядки між першим рядком, префікс якого рівний w, та останнім таким рядком, будуть мати префікс рівний w.

Тоді, щоб знайти k-й рядок в лексикографічному порядку, що має префікс рівний w, нам достатньо знайти перший рядок із таким префіксом та взяти $k-1$-й рядок після нього. Для цього можна використати бінарний пошук.

Джерело: USACO 2014 February Contest, Bronze Problem 2. Auto-Complete

Розробник: Даніїл Смельський

Розбір задачі «Бамбук»

Легко помітити, що для початку бамбуки можна розбити на чотири групи, одна з яких буде відповідати за безкорисні бамбуки(ті що ми не будемо використовувати у наших операціях, і довжина яких не рівна жодній з a, b, c), а інші будуть відповідати одному з трьох бамбуків, які нам потрібно отримати. Після цього, для кожної групи ми рахуємо суму довжин усіх бамбуків з неї, та до відповіді додаємо модуль різниці сумарної довжини бамбуків в групі та довжини бамбуку, що нам необхідно отримати. Тобто операції першого та другого типу можна використовувати вже після усіх операцій третього типу.

Реалізувати перебір усіх можливих поділів бамбуків на групи можна за допомогою рекурсії, або перебравши усі маски довжини n за основою 4.

Джерело: AtCoder Beginner Contest 119 C - Synthetic Kadomatsu

Розробник: Даніїл Смельський

Розбір задачі «Бобер Марк»

Розглянемо для початку приклад, де виконується обмеження $q = 1$, тобто ми маємо лише один запит. Легко помітити, що якщо використовувати міста як вершини графу та канали як його ребра, то ми отримуємо орієнтований ациклічний граф. Знайдемо довжину максимального шляху від вершини, де буде проходити вечірка до усіх вершин, які з неї досяжні. Відповіддю буде максимальне значення цієї довжини серед усіх вершин, до яких ми можемо потрапити, і за виключенням тих вершин, де живуть зайняті бобри. Таке рішення буде працювати за $O(n+m)$ на кожен запит.

Для вирішення задачі без обмежень, треба розбити запити на дві групи: запити, де кількість заблокованих вершин менше за деяку константу c, та запити, де кількість цих вершин більше за c.

Кількість запитів другого типу не перевищує $\frac{m}{c}$. Отже, якщо брати $c = \sqrt{m}$, можна отримати, що кількість таких запитів не більше за c. Якщо вирішувати кожен з цих запитів за $O(n+m)$, як ми вже розглянули, то отримаємо асимптотику $O((n+m) \times \sqrt{m})$.

Залишилось вирішити запити, де кількість заблокованих вершин менше за c. Для цього, перш ніж вирішувати усі запити, можна відсортувати усі вершини графи за топологічним порядком, та починаючи від вершин, що не мають вихідних ребер, зберігати для кожної вершини список з $c+1$ максимально віддалених від неї вершин.

Для цього, спочатку додамо в кожну вершину саму себе. Потім, перебравши сусідів вершини, до яких від неї проведено ребро, можна додати максимально віддалені вершини від неї, та обрати з усіх вершин $c+1$ максимально віддалену. Оскільки обидва списки будуть вже у відсортованому порядку, їх можна об'єднати за лінію, тобто за суму довжин двох списків.

Тепер, отримавши запит, де кількість заблокованих вершин менше за c, можна взяти першу найбільш віддалену вершину в списку вершини t_j, яка не є заблокованою.

Отримаємо рішення за $O((n+m) \times \sqrt{m})$

Джерело: The 17th Japanese Olympiad in Informatics (JOI 2017/2018). Spring Training Camp/Qualifying Trial. Contest Day 3 – Bitaro's Party

Розробник: Даніїл Смельський

Розбір задачі «Максимальна сума»

Для вирішення задачі на повний бал, найголовніше, що треба помітити — якщо ми маємо три рівні квадрати, то завжди можна провести вертикальну, чи горизонтальну пряму так, щоб вона не перетинала жодний квадрат, та щоб з одного боку від прямої знаходився один квадрат, а з іншого відповідно два інші.

Припустимо, що ми будемо проводити тільки горизонтальну пряму (випадок з вертикальною прямою аналогічний, нам достатньо обернути один раз матрицю на 90 градусів та ще раз вирішити ту саму задачу). І припустимо, що зверху від нашої прямої буде знаходитись лише один квадрат (для випадків, де він буде знаходитись знизу від прямої, достатньо обернути матрицю на 180 градусів, та вирішити ту саму задачу).

Припустимо, що ми зафіксували, де буде знаходитись верхній квадрат. Нехай d — максимальний номер рядка, що належить цьому квадрату. Тоді, оскільки після нього ми проводимо горизонтальну пряму, та два інших квадрати будуть знаходитись під нею, нам треба для підматриці $[d+1, 1][n, m]$, знайти максимальну суму у двох квадратах, що не можуть перетинатись.

Треба зробити попереднє підрахування $dp[i]$ — максимальна сума у двох квадратах, що не перетинаються, та знаходяться на підматриці $[i, 1][n, m]$. Розглянемо спочатку ті випадки, де один з цих двох квадратів починається у рядку i (щоб враховувати інші випадки, достатньо брати $dp[i] = max(dp[i], dp[i+1])$). Тепер, зафіксуємо праву межу цього квадрату — число r. Звідси, квадрат буде займати клітинки підматриці $[i, r-k+1][i+k-1, r]$. Залишилось усього три варіанти, де буде знаходитись другий квадрат:

1. Його правий нижній кут буде знаходитись у підматриці $[i, k][n-k+1, r-k]$.
2. Його правий нижній кут буде знаходитись у підматриці $[i+k, k][n-k+1, m]$.
3. Його лівий нижній кут буде знаходитись у підматриці $[i, r+1][n-k+1, m-k+1]$.

Щоб знайти оптимальний квадрат, що залишився, можемо застосувати додаткову квадратну динаміку.

Таке рішення буде мати складність $O(n * m)$.

Джерело: Asia-Pacific Informatics Olympiad, 2009 - Digging for Oil

Розробник: Данііл Смельський