Домашняя работа №5

Шишацкий Михаил, 932

25.04.2020

Задача 1. Заметим, что

$$\begin{pmatrix} a_n \\ a_{n-1} \\ 1 \end{pmatrix} = \begin{pmatrix} 5 & 2 & 3 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{n-1} \\ a_{n-2} \\ 1 \end{pmatrix} = \begin{pmatrix} 5 & 2 & 3 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}^{n-1} \begin{pmatrix} a_1 \\ a_0 \\ 1 \end{pmatrix}$$

Размер матрицы константный, умножение двух матриц размера 3×3 мы умеем производить за $O(1)$. Таким образом, можно применить бинарное возведение матрицы в степень и за $O(log(n))$ получить $(n-1)$-ую степень матрицы, а затем умножить её на столбец за $O(1)$.

Задача 2. Две матрицы размера $q \times q$ мы умеем перемножать за $O(q^3)$ наивным алгоритмом (можно улучшить асимптотику, умножая алгоритмом Карацубы). Также мы умеем возводить любую матрицу в степень n за $O(q^3 log(n))$ путём бинарного возведения в степень.

Решим задачу поиска суммы рекурсивно. Пусть функция $poly\_sum(n)$ возвращает нам $A + A^2 + A^3 + \cdots + A^n$. База рекурсии: $poly\_sum(0) = 0 \in M_{q \times q}$, $poly\_sum(1) = A$. Переход:

$$poly\_sum(k) = \begin{cases} poly\_sum(\frac{k}{2}) \cdot (E + A^{\frac{k}{2}}) & \text{если} \quad k \equiv 0(mod 2) \\ poly\_sum(\frac{k}{2}) \cdot (E + A^{\frac{k}{2}}) + (A^{\frac{k}{2}})^2 \cdot A & \text{если} \quad k \equiv 1(mod 2) \end{cases}$$

Выносить множители корректно, т.к. в матричных многочленах от одной переменной умножение коммутирует. Пока что наш алгоритм работает за $O((q^3 + q^3 log(n)) log(n))$, т.к. на каждом шаге происходят арифметические операции с матрицами за $O(q^3)$, а также бинарное возведение в степень матрицы за $O(q^3 log(n))$. Сделаем так, что наша функция будет возвращать пару - $A^k, poly\_sum(k)$. Внутри функции при знании значений для $\frac{k}{2}$ значения A^k и $poly\_sum(k)$ вычисляются за $O(q^3)$, так как предполагают несколько операций сложения и умножения:

$$\{A^k, poly\_sum(k)\} = \begin{cases} \{(A^{\frac{k}{2}})^2, & poly\_sum(\frac{k}{2}) \cdot (E + A^{\frac{k}{2}})\} \quad \text{если} \quad k \equiv 0(mod 2) \\ \{A^k := (A^{\frac{k}{2}})^2 \cdot A, & poly\_sum(\frac{k}{2}) \cdot (E + A^{\frac{k}{2}}) + A^k\} \quad \text{если} \quad k \equiv 1(mod 2) \end{cases}$$

Каждый раз k уменьшается в 2 раза, поэтому глубина спуска - $log(n)$. Таким образом, мы умеем вычислять $poly\_sum(n)$ за $O(q^3 log(n))$

Задача 3. Далее будем считать, что функция $dist(u, v)$ возвращает расстояние между двумя точками (в реализации можно возвращать квадрат расстояния, чтобы не волноваться о дробных числах).

Заведём матрицу dp размера $2^n \times n$, для которой в ячейке $dp[mask][v]$ будет храниться номер вершины, в которую наиболее оптимально пойти из текущей (в какую-то вторую или на "базу", назовём "базу" '-1'-точкой). Если из вершины оптимально идти на "базу", запишем в соответствующую ячейку -1. Так как расстояния симметричны, $dp[mask][u] = v \leftrightarrow dp[mask][v] = u$ в случае, если обе вершины не подвязаны к -1. Инициализируем эту матрицу значениями -1 (все вершины подвяжем к $(0; 0)$).

Также заведём массив d размера 2^n, для которого в ячейке $d[mask]$ будет храниться минимальное, расстояние, которое надо пройти, чтобы собрать все предметы из этой маски.

Пройдёмся циклом по маскам в возрастающем порядке. В этом случае, для конкретной маски все подмножества этой маски уже рассмотрены ранее. Сначала пройдёмся по всем вершинам v в маске и запишем в $d[mask] := \sum 2dist(v, (0; 0))$. Теперь для каждой вершины v в маске будем искать наилучшую вершину u в текущей маске, к которой её можно подвязать. Если $d[mask] > dist(v, (0; 0)) + dist(v, u) + dist(u, (0; 0)) + d[mask \backslash \{v, u\}]$, обновим значение $d[mask]$. В этом переборе допускается также совпадение v и u, в таком случае $v(u)$ подвяжется к -1.

В процессе обновления значения $d[mask]$ будем запоминать, на какой паре значений был достигнут минимум. Подвяжем эти два значения друг к другу (или к -1, если это одно и то же значение) и перенесём остальные подвязки из $dp[mask \backslash \{v, u\}][*]$.

Таким образом, для каждого значения $mask$ мы умеем получать оптимум за $O(n^2)$, значит, все значения $mask$ будут обработаны за $O(2^n n^2) = O(2^n poly(n))$. Алгоритм действий для девочки будет содержаться в $dp[(1 \ll n) - 1][*]$, а наименьшая длина пути - в $d[(1 \ll n) - 1]$.