

Теоретическое решение задачи С.

В задаче требовалось найти количество путей фиксированной длины из первой вершины графа. В условии даны количество вершин в графе (n), количество ребёр (m) и длина искомых путей (k), а также m ребёр. Зададим граф матрицей смежности (a): $a[i, j]$ будет равняться числу рёбер из i -й вершины в j -ю вершину. Будем искать матрицу g , где $g[i, j]$ будет равняться количеству путей из k рёбер из i -й вершины в j -ю.

Будем решать задачу методом динамического программирования: пусть g_k – посчитанная матрица ответов для k . Требуется найти g_{k+1} – искомую матрицу ответов для $k+1$.

База динамики: $g_1 = a$ – матрица смежности.

Пересчёты динамики: $g_{k+1}[i, j] = \sum_{l=1}^n g_k[i, l] * g_1[l, j]$.

Заметим, что пересчёты динамики – это всего лишь произведение g_k и g_1 . Т.е. $g_{k+1} = g_k * g_1$. А из этого следует, что $g_k = g_1^k$.

Возводить же матрицу в степень мы будем эффективно с помощью бинарного возведения в степень:

$$\begin{aligned} \text{Имеют место формулы: } 1. \forall \text{чётного } n: a^n &= (a^{\frac{n}{2}})^2 = a^{\frac{n}{2}} * a^{\frac{n}{2}} \\ 2. \forall n: a^n &= a^{n-1} * a \end{aligned}$$

Т.е. для чётного n мы сводим степень к вдвое меньшей за одну операцию, а от нечётного n мы переходим к чётному.

Итого в полученной матрице g_k в $g_k[i, j]$ будет содержаться количество путей из i -й вершины в j -ю фиксированной длины k . Осталось посчитать количество путей фиксированной длины из первой вершины графа, а это будет сумма элементов первой строки матрицы.

Оценим время работы описанного алгоритма. Он состоит из трёх шагов:

1. Сначала мы создаём матрицу смежности. Для этого мы проходимся по всем рёбрам и увеличиваем соответствующую им ячейку матрицы смежности на 1. Всего действий – $m \Rightarrow$ этот шаг занимает $O(m)$ времени.
2. Далее мы возводим матрицу смежности в степень k :
 - I. Посмотрим, за какое время выполняется один шаг: умножение матриц (возведение матрицы смежности в квадрат).

- Дана матрица A размером $n \times n$.
- Пусть R будет новой матрицей $n \times n$.
- Для i от 1 до n :
 - Для j от 1 до n :
 - $sum = 0$
 - Для k от 1 до n :
 - $sum += A[i, k] * A[k, j]$
 - $R[i, j] = sum$
 - Возвращаем R

Три вложенных цикла, каждый из которых проходит n итераций, на каждой из которых мы делаем ровно одно действие (которое выполняется за $O(1)$). Т.е. умножение матриц выполняется за $O(n * n * n) = O(n^3)$.

- II. Посмотрим, сколько всего действий выполнится. В бинарном возведении в степень всего будет не более, чем $2 * \log k$ переходов, прежде чем мы придём к базе рекурсии. Значит, алгоритм работает за $O(2 \log k) = O(\log k)$.

Итого возведение матрицы в степень работает за $O(n^3 * \log k)$.

3. Третьим шагом мы получаем ответ, считая сумму элементов первой строки матрицы. Всего будет ещё n итераций, которые выполняются за $O(n)$.

Итого весь алгоритм работает за $O(m + n^3 * \log k + n) = O(n^3 * \log k)$.

Оценим требуемую память для данного алгоритма. Как уже говорилось, количество умножений матриц будет не более, чем $2 * \log k$, нам же нужно хранить каждую из этих матриц. Памяти на каждую из матриц требуется $O(n * n) = O(n^2)$. Таким образом, алгоритм требует $O(n^2 * \log k)$ памяти.