Теоретическое решение задачи C.

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 ребёр 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.

Для того, чтобы найти все пути длины k+1 от вершины i к вершине j, будем перебирать все промежуточные вершины p, и будем пытаться «улучшить» этот путь (от i до p) на один шаг всеми возможными способами, пытаясь прийти в j. Т.е. путь от i до p должен быть равен k, а между p и j можно пройти напрямую (то есть путём длины 1). То есть, количество путей длины k+1 от вершины i к вершине j будет равняться сумме количеств путей от i до j через все промежуточные p. Для этого на каждом p мы умножим количество путей длины k от i до p на количество рёбер между p и j. Итого:

База динамики: g_1 = a – матрица смежности.

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

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:

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

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 => 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). 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 $O(n*n*n) = O(n^3)$.

   - Дана матрица A размером n x n.
   - Пусть R будет новой матрицей n x 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

   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(2log 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 выполнятся 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.