Теоретическое решение задачи B – Крысиные бега

Алгоритм решения и доказательство его правильности

Представим условие задачи, как ориентированный граф с N вершинами и M дугами. Представим этот граф в программе в виде матрицы смежности размером NxN. Так, как по условию задачи, граф ориентирован, а также может быть несколько различных путей из вершины a в вершину b, то заполним матрицу смежности таким образом:

d[i][j] = $M_{i,j}$(число дуг из вершины i в j)

Задача сводится к нахождению числа путей фиксированной длины K на заданном графе, все искомые пути начинаются с первой вершины. Рассмотрим три случая с разными заданными K:

1. K = 0. При длине маршрута 0, ответ на задачу получается 1, так как маршрут длиной 0 только один – оставаться в первой комнате.
2. K = 1. Количество путей длины 1 из первой вершины графа хранится в составленной матрицы смежности. В первой строке матрицы хранятся количество дуг (дуга – маршрут длиною 1) выходящих из первой вершины ко всем остальным. Поэтому ответ ищется по формуле:
$$\sum_{i=1}^{N} d[0][i]$$
3. K > 1. Допустим, что у нас есть матрица ответов для некоторого k: d_k. Каждая i, j клетка данной матрицы хранит количество путей из вершины i в вершину j длиною k. Для нахождения количество путей вершины i в вершину j длиною k + 1, нам требуется искать путь через некоторую вершину p. Допустим, из вершины i есть путь в вершину p длиною k, а из этой вершины p есть путь в вершину j длиною 1, это значит, что существует путь из вершины i в вершину j, проходящий через вершину p, длина которого k + 1. Для подсчета всех путей длины k + 1 из i в j, надо рассмотреть пути через все вершины графа и просуммировать их. Тогда справедлива следующая формула:
$$d_{k+1}[i][j] = \sum_{p=1}^{N} d_k[i][p] * g[p][j]$$
Данная формула работает с графами, у которых есть петли, так как в качестве вершины p рассматриваются также вершины i и j (увеличить длину пути путем прохождения из вершины в эту же вершину). А так же алгоритм работает с графами, у которых есть множество различных дуг между двумя вершинами, так как в формуле у мы

перемножаем количество маршрутов, что дает нам возможность рассмотреть каждый маршрут (Пример: есть 10 путей из i в p длиною k, и есть 10 путей из p в j длиною 1, значит путей из i в j длиною k + 1 – 10*10=100). А значит, что данный алгоритм поиска путей фиксированной длины подходит для решения нашей задачи. Формула выше - это ничто иное, как произведение двух матриц - d_i и g:

$$d_{i+1} = d_i * g$$

Очевидно, что тогда нахождение матрицы ответов для K сводится к:

$$d_k = g^K$$

После чего, находим ответ по формуле:

$$\sum_{i=1}^{N} d_k[0][i]$$

Для оптимизации работы алгоритма применим алгоритм бинарного возведения в степень:

$$g^K = (g^{K/2})^2 = g^{K/2} * g^{K/2}$$

А для нечетного k:

$$g^k = \ g^{k-1} * g$$

Для реализации создадим вспомогательный логический массив размера $\log_2 k$. Достаточен массив такого размера, так как количество возведений матриц в квадрат равно $\log_2 k$. В него записываем значения:

1. True – нужно домножать матрицу на g
2. False = не нужно домножать матрицу

Составив “карту” умножения, заполнив логический массив, берем за основу матрицу смежности и начинаем итеративно ($\log_2 k$ раз) возводить в квадрат, и если нужно, домножать на матрицу смежности, если в логическом массиве стоит значение True в данном номере итерации. Получив на выходе матрицу ответов для K, находим сумму количества путей длины K, которые начинаются с первой комнаты по выше описанной формуле (1).

Временная сложность

Алгоритм сводится к перемножению матриц K раз. Перемножение матриц имеет сложность O(n^3), сложность алгоритма бинарного возведения в степень – O(logk). Итоговая сложность программы: O($n^3 logk$).

Затраты памяти

Всего для реализации алгоритма создается три массива размером nxn: массив для хранения матрицы смежности, массив для хранения матрицы, над которой сейчас производится действие возведения в квадрат и матрица для хранения ответа. Итоговые затраты памяти – O(n^2).