Теоретическое решение задачи С (1-ое ДЗ)

В данной задаче нам был дан лабиринт, состоящий из N комнат и M переходов между ними (1 ≤ N ≤ 50, 0 ≤ M ≤ 105). В лабиринт были запущены крысы (все крысы стартовали в 1ой комнате) и пробежали К переходов (0 ≤ K ≤ 109). Нам нужно узнать сколько существует различных маршрутов из первой комнаты длинны К.

Давайте переформулируем данную задачу: нам дан ориентированный граф, состоящий из N вершин и M ребер. Нам нужно найти количество путей длинны К, которые начинаются в 1ой вершине. Эту задачу можно решить методом динамического программирования. Давайте сначала поймем какой будет база нашего ДП. Для К = 0 существует только 1 вариант – остаться в 1ой комнате. Сколько же существует путей из 1ой вершины длинны 1? Однако это тоже самое что и «сколько у нас существует способов выйти из 1ой вершины» и равно количеству ребер из первой вершины. То есть для базы ДП нам нужно для каждой вершины знать сколько существует ребер из нее во все остальные вершины. Мы можем это легко находить если представим граф в виде матрицы смежности. Теперь нам нужно определить переход в нашем ДП. Давайте поймем, как мы можем узнать сколько существует путей длинны 2. Для этого выберем две точки А и В. Найдем количество путей длинны 2 которые начинаются в точке А и заканчиваются в точке В. Проберем все точки. Допустим сейчас мы рассматриваем произвольную точку С. Посмотрим сколько существует путей из точки А в т С и из т С в т В (т - точка). Как нам определить сколько существует путей из точки А в точку В, которые проходят через точку С? Пусть из т А в т С мы можем прийти Wa способами, а из т С в т B - Wb. Тогда количество путей из т А проходящие через т С в т В равно Wa\* Wb. Подведем итоги – зная количество способов попасть из любой выбранной точки в любую другую за К шагов мы сможем перейти к следующему состоянию ДП (узнать количество путей из любой выбранной точки в любую другую длинны К+1). Также можно доказать, что если мы проберем все промежуточные точки, то учтем все варианты путей, поскольку для того, чтобы наш путь стал на 1 длиннее, мы должны будем пройти через какую-то из вершин. Мы нашли переход нашего ДП.

Однако если мы еще раз посмотрим на условие задачи, то заметим, что К может быть очень большим (порядка 109). Если мы будем каждый раз находить количество путей длинной на один больше, то нам потребуется О(К) операций, что не пройдет по времени. Однако давайте еще раз пересмотрим что мы делаем чтобы получить количество путей длинной К+1. В ДП мы храним количество путей длинны К от всех вершин до всех остальных (в виде матрицы смежности). Затем мы выбираем две вершины и находим количество способов прийти из вершины А в вершину С за К ходов и от вершины С в вершину В за 1 ход. А что, если мы будем рассматривать переход из вершины С в вершину В также за К ходов? Мы получим количество способов пройти из вершины А в вершину В за 2\*К ходов. Можем ли мы это как-то использовать для того, чтобы ускорить наш алгоритм? Первое что приходит в голову: давайте сохраним какие-то большие состояния ДП (количество путей большой длинны К) и на каждой итерации будем находить путем «прыгать» на большое количество шагов. К примеру мы знаем ДП[10000], тогда мы сможем найти состояние ДП[K+10000].

Уже быстрее, но, возможно, мы можем сделать еще быстрее? Вспоминаем что любое число можно представить в виде степеней двойки. Давайте сохраним состояния нашего ДП для степеней двойки (сначала получаем ДП[1], потом ДП[2], ДП[4], ДП[8], ДП[16]… Для этого потребуется хранить log2 К состояний ДП. ДП[0] это матрица в которой на диагонали стоят 1, а везде кроме диагонали 0, поскольку пройдя 0 шагов мы останемся там же, где и начали). Сколько шагов нам теперь понадобится чтобы получить нужно состояние ДП? Пусть нам нужно узнать ДП[Х]. Давайте представим Х в двоичной системе. Тогда 1 будет означать что мы должны взять определенное состояние ДП (само состояние ДП определяется по позиции бита), а 0 – не должны. Покажем на примере для лучшего понимания: Х = 11710 = 11101012 = 26+25+24+22+20 = 64+32+16+4+1. Тогда для тогда чтобы получить состояние ДП[117] (ДП степеней двойки мы уже знаем как посчитать за log K) мы можем использовать ДП[0] и ДП[1], получаем ДП[1]. Далее используем ДП[4] и получаем ДП[5]. По аналогии получаем ДП[117] за 5 шагов. Сколько же шагов нам потребуется в худшем случае? Можно заметить, что количество шагов, это количество активных битов в двоичной записи Х, коих всего максимум О(log2 X). Этот алгоритм уже будет работать за О(log K), что приемлемо. Однако реализация будет довольно неудобной. Возможно есть способ как-то ее упростить?

Можно заметить, что мы имеем действия с матрицами (поскольку мы решили представить наш граф в виде матрицы смежности). Мы уже выяснили, что первоначальная матрица (назовем ее М1) смежности графа соответствует количеству путей из любой вершины в любую другую длинны 1. Немного подумав, можно понять, что переход нашего ДП соответствует умножению матрицы: M1\* M1 = M2. То есть для того, чтобы из состояния MK в состояние MK+1 нам нужно домножить на M1:

MK+1 =MK \* M1. Отталкиваясь от этого, можно понять, что

MK =M1 \* M1 \* M1 \* M1 … (К раз) = (M1)K

Если мы будем перемножать матрицу К раз то опять же не влезаем в ТЛ.

Однако вспоминаем трюк со степенями двойки. Давайте будем отталкиваться от данных утверждений:

(M1)K = (M1)K-1 \* M1

(M1)K = (M1)K/2 \* (M1)K/2  (если К % 2 == 0).

Данные утверждения являются базовыми операциями над матрицами, поэтому доказывать мы их здесь не будем.

Теперь, используя быстрое возведение в степень, мы сможем за log K шагов получить (M1)K , где M1 – наша первоначальная матрица смежности графа. Что нам это дает? Мы будем знать количество способов попасть с любой вершины в любую другую за К шагов (количество различных путей, которые начинаются в вершине А и заканчиваются в вершине В длинной К будет на пересечении А-ой строки и В-ого столбца матрицы). Изначальной задачей было найти количество путей длинны К которые начинаются в 1ой вершине. Значит нам необходимо подсчитать количество путей, которые начинаются в вершине 1 и заканчиваются в любой другой. Пройдемся по первой строке (M1)K и просуммируем все значения. Получим ответ на поставленную задачу. Конечно же, поскольку нам нужно получить ответ по модулю 1е9+7, каждое вычисление выполняем по модулю дабы избежать переполнения.

Оцениваем сложность алгоритма: у нас будет О(log K) переходов динамики, на каждом из которых нам необходимо перемножить матрицы, что занимает O(N3) операций. Также не забываем, что нам необходимо в асимптотике учесть ввод данных, ведь нам потребуется ввести 3+2\*M чисел, которые формируют условие задачи, что добавит нам асимптотически О(M). Итого имеем О(N3logK+M) асимптотику по времени.

Поскольку у нас будет O(log K) переходов динамики и на каждом переходе нам нужно будет хранить константное количество матриц (матрица занимает О(N2) памяти), то асимптотическая сложность по памяти будет О(N2 \* logK).

Рябцев Денис