Теоретичне рішення задачі B.

Алгоритм рішення та доведення його правильності.

Задача : знайти кількість маршрутів довжиною рівно K в орієнтованому графі, що виходять з вершини 0.

Алгоритм: Треба піднести матрицю A до ступеня K, де A[i][j] – кількість способів потрапити з вершини i у вершину j за один крок (матриця суміжності). І вивести суму чисел у рядку з номером 0.

Доведення: Припустимо, що ми знаємо скільки існує способів потрапити з кожної вершини до кожної рівно за X кроків, тоді припустимо нам треба вказати скількома способами ми можемо потрапити у вершину i рівно за (X + 1) крок. Зрозуміло, що це буде кількість способів потрапити у іншу вершину j (від 0 до n-1), помножена на кількість способів потрапити за один крок з цієї вершини j до вершини i, тобто:

У нас є матриця dx
Треба знайти матрицю d(x+1)

$$d_{x+1}[i][j] = \sum_{k=0}^{n-1} d_x[i][k] \cdot A[k][j]$$

А це і є формула перемноження матриць. Початкова матриця – одинична, і вона помножується на матрицю A – початкову рівно K раз (підведення у ступінь K).

Таким чином ми отримаємо матрицю d, де d[i][j] – кількість маршрутів довжиною K, що виходять з вершини номер i та закінчуються у вершині з номером j. Тому відповіддю на задачу буде сума кількостей маршрутів довжиною K до кожної з вершин, що починаються у вершині 0 – сума усіх чисел у 0 рядку матриці.

Алгоритм підведення у ступінь матриці:

За умовою K <= 10^9, тобто множити матрицю K раз буде занадто довго. Можна помітити, що для отримання результату f(x) ^ k достатньо порахувати f(x) ^ (k / 2) підвести результат до квадрату, а f(x) ^ (k / 2) можна порахувати як f(x) ^ (k / 4) та піднести до квадрату і так далі… Тож використовується бінарне піднесення до ступеню.

Часова складність

Нам треба робити бінарне піднесення у ступінь матриць (використовуючи множення матриць) .

1) Множення матриць працює (як відомо) за O(N^3), бо нам треба для кожної клітинки матриці (їх усього N^2) перебирати усі можливі вершини (їх усього N).
2) Бінарне піднесення у ступень працює за O(log n)

Тому сумарна складність – O(N^3 log N).

Витрати пам’яті

Для вирішення цієї задачі мені треба було зберігати матрицю, я зробила це за допомогою вектора векторів розміру N^2, а також була використана фіксована кількість допоміжних змінних. Тому витрати пам’яті у цьому рішенні – O(N^2).