Last update: May 2, 2025 Translated From: e-maxx.ru

Number of paths of fixed length / Shortest paths of fixed length

The following article describes solutions to these two problems built on the same idea: reduce the problem to the construction of matrix and compute the solution with the usual matrix multiplication or with a modified multiplication.

Number of paths of a fixed length

We are given a directed, unweighted graph G with n vertices and we are given an integer k. The task is the following: for each pair of vertices (i, j) we have to find the number of paths of length k between these vertices. Paths don't have to be simple, i.e. vertices and edges can be visited any number of times in a single path.

We assume that the graph is specified with an adjacency matrix, i.e. the matrix $G[][]$ of size $n \times n$, where each element $G[i][j]$ equal to 1 if the vertex i is connected with j by an edge, and 0 is they are not connected by an edge. The following algorithm works also in the case of multiple edges: if some pair of vertices (i, j) is connected with m edges, then we can record this in the adjacency matrix by setting $G[i][j] = m$. Also the algorithm works if the graph contains loops (a loop is an edge that connect a vertex with itself).

It is obvious that the constructed adjacency matrix is the answer to the problem for the case $k = 1$. It contains the number of paths of length 1 between each pair of vertices.

We will build the solution iteratively: Let's assume we know the answer for some k. Here we describe a method how we can construct the answer for $k + 1$. Denote by C_k the matrix for the case k, and by C_{k+1} the matrix we want to construct. With the following formula we can compute every entry of C_{k+1}:

$$C_{k+1}[i][j] = \sum_{p=1}^{n} C_k[i][p] \cdot G[p][j]$$

It is easy to see that the formula computes nothing other than the product of the matrices C_k and G:

$$C_{k+1} = C_k \cdot G$$

Thus the solution of the problem can be represented as follows:

$$C_k = \underbrace{G \cdot G \cdots G}_{k \text{ times}} = G^k$$

It remains to note that the matrix products can be raised to a high power efficiently using Binary exponentiation. This gives a solution with $O(n^3 \log k)$ complexity.

Shortest paths of a fixed length

We are given a directed weighted graph G with n vertices and an integer k. For each pair of vertices (i, j) we have to find the length of the shortest path between i and j that consists of exactly k edges.

We assume that the graph is specified by an adjacency matrix, i.e. via the matrix $G[][]$ of size $n \times n$ where each element $G[i][j]$ contains the length of the edges from the vertex i to the vertex j. If there is no edge between two vertices, then the corresponding element of the matrix will be assigned to infinity ∞.

It is obvious that in this form the adjacency matrix is the answer to the problem for $k = 1$. It contains the lengths of shortest paths between each pair of vertices, or ∞ if a path consisting of one edge doesn't exist.

Again we can build the solution to the problem iteratively: Let's assume we know the answer for some k. We show how we can compute the answer for $k + 1$. Let us denote L_k the matrix for k and L_{k+1} the matrix we want to build. Then the following formula computes each entry of L_{k+1}:

$$L_{k+1}[i][j] = \min_{p = 1 \ldots n} \left(L_k[i][p] + G[p][j]\right)$$

When looking closer at this formula, we can draw an analogy with the matrix multiplication: in fact the matrix L_k is multiplied by the matrix G, the only difference is that instead in the multiplication operation we take the minimum instead of the sum.

$$L_{k+1} = L_k \odot G,$$

where the operation $\odot$ is defined as follows:

$$A \odot B = C \iff C_{ij} = \min_{p = 1 \ldots n} \left(A_{ip} + B_{pj}\right)$$

Thus the solution of the task can be represented using the modified multiplication:

$$L_k = \underbrace{G \odot \ldots \odot G}_{k \text{ times}} = G^{\odot k}$$

It remains to note that we also can compute this exponentiation efficiently with Binary exponentiation, because the modified multiplication is obviously associative. So also this solution has $O(n^3 \log k)$ complexity.

Generalization of the problems for paths with length up to k

The above solutions solve the problems for a fixed k. However the solutions can be adapted for solving problems for which the paths are allowed to contain no more than k edges.

This can be done by slightly modifying the input graph.

We duplicate each vertex: for each vertex v we create one more vertex v' and add the edge (v, v') and the loop (v', v'). The number of paths between i and j with at most k edges is the same number as the number of paths between i and j' with exactly $k + 1$ edges, since there is a bijection that maps every path $[p_0 = i,\ p_1,\ \ldots,\ p_{m-1},\ p_m = j]$ of length $m \le k$ to the path $[p_0 = i,\ p_1,\ \ldots,\ p_{m-1},\ p_m = j, j', \ldots, j']$ of length $k + 1$.

The same trick can be applied to compute the shortest paths with at most k edges. We again duplicate each vertex and add the two mentioned edges with weight 0.

Contributors:
jakobkogler (91.4%) adamant-pwn (6.45%) bit-shashank (1.08%) wikku (1.08%)