added: 10 Sep 2010 16:13
Edited: 3 May 2012 1:35

Shortest paths of fixed length, the number of paths of fixed length

Contents [hide]

The following describes these two objectives, built on the same idea: to reduce the problem to the construction of the matrix to the power (with the usual multiplication operation, and modified).

Number of paths of fixed length

Let a directed unweighted graph $G$with $n$vertices, and given an integer k. Required for each pair of vertices i, and $j$find the number of paths between these vertices consisting of exactly $k$the edges. Ways at the same time we consider arbitrary, not necessarily simple (ie, vertices can be repeated any number of times).

We assume that the graph is given **adjacency matrix** , ie matrix $g[][]$size $n \times n$, where each element $g[i][j]$is equal to one if the nodes between an edge and zero if there is no edge. Described below, and the algorithm works in the case of multiple edges if between some vertices $i$and $j$is immediately $m$edges, the adjacency matrix should record this number m. Algorithm also correctly takes into account the loop in the column, if any.

Obviously, in this form, **the adjacency matrix** of the graph is the **answer to the problem when**$k = 1$ - it contains a number of paths of length 1between each pair of vertices.

Solution will be constructed **iteratively** : Let the answer for some $k$is found, we show how to build it $k + 1$. We denote $d_k$the matrix found answers for k, and through d_{k+1}- the matrix of responses you want to build. Then clear the following formula:

$$d_{k+1}[i][j] = \sum_{p=1}^{n} d_k[i][p] \cdot g[p][j].$$

It is easy to note that the above formulas - is nothing but the product of two matrices d_k, and $g$in the usual sense:

$$d_{k+1} = d_k \cdot g.$$

Thus, the **solution** of this problem can be represented as follows:

$$d_k = \underbrace{g \cdot \ldots \cdot g}_{k \text{ times}} = g^k.$$

It remains to note that the construction of the power of the matrix can be produced efficiently by the algorithm **binary exponentiation** .

Thus, the obtained solution has the asymptotic behavior $O(n^3 \log k)$and is binary exponentiation $k$of th power of the adjacency matrix of the graph.

Shortest paths of fixed length

Suppose we are given a directed weighted graph $G$with $n$vertices, and given an integer k. Required for each pair of vertices i, and $j$find the length of the shortest path between these vertices, consisting of exactly $k$the edges.

We assume that the graph is given **adjacency matrix** , ie matrix $g[][]$size $n \times n$, where each element $g[i][j]$contains the length of the edge from vertex $i$to vertex j. If between some vertices no ribs, the corresponding element of the matrix to be equal to infinity ∞.

Obviously, in this form, **the adjacency matrix** of the graph is the **answer to the problem when**$k = 1$ - it contains the length of the shortest paths between each pair of vertices, or ∞if the length of the path 1does not exist.

Solution will be constructed **iteratively** : Let the answer for some $k$is found, we show how to build it $k + 1$. We denote $d_k$the matrix found answers for k, and through d_{k+1}- the matrix of responses you want to build. Then clear the following formula:

$$d_{k+1}[i][j] = \min_{p=1 \ldots n} (d_k[i][p] + g[p][j]).$$

Look closely at this formula, it is easy to draw an analogy with the matrix multiplication: in fact, the matrix is $d_k$multiplied by a matrix g, only the multiplication operation instead of the sum of all the $p$minimum is taken over all p :

$$d_{k+1} = d_k \odot g,$$

where the operation $\odot$of multiplication of two matrices is defined as follows:

$$A \odot B = C \quad \Longleftrightarrow \quad C_{ij} = \min_{p=1 \ldots n} (A_{ip} + B_{pj}).$$

Thus, **a solution** to this problem can be represented by this multiplication as follows:

$$d_k = \underbrace{g \odot \ldots \odot g}_{k \text{ times}} = g^{\odot k}.$$

It remains to note that the construction of the power of this multiplication can be performed efficiently using the algorithm **binary exponentiation** , because the only required for a property - associativity of multiplication - obviously there.

Thus, the obtained solution has the asymptotic behavior $O(n^3 \log k)$and is binary exponentiation $k$of th power of the adjacency matrix of the graph with the modified matrix multiplication.

Generalization to the case when the required path length, no more than a specified length

The above solutions solve problems when you need to consider the way a certain, fixed length. However, these solutions can also be adapted to solve problems when you need to consider paths that contain **no more** than a specified number of edges.

You can do this by modifying the input bit count. For example, if we are only interested in the path ending in a particular vertex t, the graph can **add a loop** (t, t) of zero weight.

If, however, we are still interested in the answers to all pairs of vertices, the simple addition of loops to all vertices spoil the answer. Instead, you can **bifurcate** each vertex: for each vertex $v$to create an additional vertex v', edge hold $(v, v')$and add a loop (v', v').

Deciding on a modified graph problem of finding ways to fixed length, the answers to the original problem will be obtained as the answers between the vertices $i$and j'(ie, more vertices - the vertices-end, in which we can "whirl" many times as necessary).