K.Kvitka, **Ph.D student**
A.Panishev, Dr.Eng., Prof., research advisor
I.Orlovska, language advisor
Zhytomyr State Technological University

ALGORITHMS OF CONSTRUCTING THE SHORTEST PATHS IN GRAPHS

In graph theory, the problem of the shortest route means to find a path between two vertices (or nodes) of the graph, when the sum of the weights of the edges of which it is composed, is minimal. The finding of the shortest path between two points on a road map (in this case, the vertices are points and the edges are the way parts with weights equal to the time required to overcome this part) can be an example.

This problem is often referred to as the shortest path between a pair of nodes, but there are other versions of the shortest path problem, which differ in several parameters:

- the problem of the shortest path with one input (you must find the shortest path between the input vertex and all other vertices);
- the problem of the shortest path with one output (the shortest paths from all vertices to one output vertex should be found);
- the problem of the shortest paths for all pairs (the shortest paths between each pair of vertices in the graph should be found).

These options have much more effective algorithms than the simplified approach which launches the algorithm of searching the shortest path between all nodes.

For any two vertices there may be several paths which connect them. The path connecting the tops and having a minimum possible length is called the shortest path.

Mathematically, the problem of the shortest path is written as following:

Let $G = (V, E)$ is a directed graph (Fig.1), each edge of which is marked with a nonnegative number (the weight of an edge). Let us mark one vertex 1 and name it leak. It is necessary to find the shortest path from the source 1 to all other vertices of G.

One of the most efficient algorithms of finding the shortest path from the source vertex to all other vertexes of the weighted graph is the algorithm suggested by Dijkstra in 1959. The algorithm works only for the graphs without the edges of negative weight.

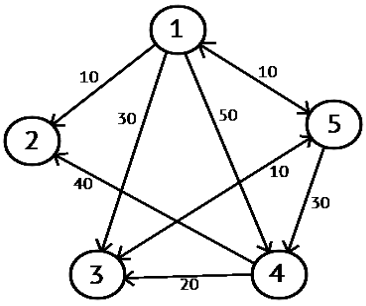

Fig. 1

The idea of the algorithm is as follows. Each vertex is assigned with a label that is the known minimal distance from this vertex to its neighbors. The algorithm works step by step - at every step, it "visits" one top and tries to reduce the marks. The work of the algorithm ends when all vertices are "visited".

This algorithm is widely used in computing and technology, for example, it is used by the routing protocols OSPF and IS-IS.

Dijkstra algorithm can be generalized in the case, where some of the arcs are negative. The idea of this generalization belongs to Ford. The algorithm is as follows. At first, the magnitude of the flow is set to 0. Then the flux is iteratively enlarged by searching the increasing path (the path from the source to the drain, along which you can send a bigger flow). The process is repeating until you can find an increasing way.

Another algorithm for finding the shortest path in the graph is the algorithm of Bellman – Ford. Using this algorithm we can find the shortest path from one vertex to all the others. Unlike Dijkstra algorithm, Bellman – Ford admits an edge with negative weight.

Ford-Bellman algorithm itself consists of several phases. At each phase all edges of the graph are visible, and the algorithm tries to make relaxation (relax, easing) along each edge (a, b) of the weight c. Relaxation along the edge is an attempt to improve the value of $d[b]$ with the help of the value $d[a] + c$. Practically, it means that we try to improve the value for the top b, using the edge (a, b) and the current value for the top a. It is stated that n-1 phase of the algorithm is enough to calculate the lengths of all shortest paths in the graph correctly (negative weight cycles are absent).

The problem of finding the shortest path between each pair of vertices can be solved by means of frequent repetition of Dijkstra algorithm. But there are more efficient algorithms than repeating Dijkstra algorithm many times. It is Floyd and

Dantzig algorithm. Both algorithms allow negative values for the arches length, but they don't allow the presence of negative contour length.

In Floyd's algorithm matrix D^0 serves as the initial data. Initially from this matrix the matrix D^1 is calculated. Then the matrix D^2 is calculated using the matrix D^1, etc. The process is repeating for as long as the matrix D^N is deducted from the matrix D^{N-t}.

Dantzig algorithm is close to Floyd's algorithm, but it differs from the previous only in the other order of the performance of the same transactions.

To solve the problem of the shortest path for each pair of vertices of a weighted directed graph $G = (V, E)$ the Floyd-Vorshella algorithm is used. Its running time is estimated proportionally to n^3. The advantage of the Floyd-Dijkstra algorithm in comparison with Vorshella's is that he either solves the problem on a graph with negative weights of arcs or finds the contours of negative weight in it.

Conclusion. The problem of the shortest path is the problem of finding a path that has the minimum length between a pair of nodes. There are many variants of the problem of the shortest path, which differ in several parameters. Each of the options has its own algorithms that are more effective in each specific case.

The most efficient algorithm for finding the shortest path from the source vertex to all other vertices of the weighted graph is Dijkstra algorithm. Ford algorithm is used when some of the arcs are negative.

To find the shortest path from one vertex to all other we use Bellman – Ford algorithm.

If you need to find the shortest path between each pair of nodes, it is advisable to apply Floyd and Dantzig algorithms.