**Ministry of Education and Research of the Republic of Moldova**

**Technical University of Moldova**

**Faculty of Computers, Informatics and Microelectronics**

**REPORT**

Laboratory work no. 5

*Empirical analysis of algorithms:*

*Dijkstra and Floyd*

Did:

St. gr. FAF-211 Andrei Ceban

Checked:

asist. univ. Cristofor Fistic

Chişinău - 2023

**Laboratory work no. 5**

**Objective:**

1. Implement the algorithms listed above in a programming language
2. Establish the properties of the input data against which the analysis is performed
3. Choose metrics for comparing algorithms
4. Perform empirical analysis of the proposed algorithms
5. Make a graphical presentation of the data obtained
6. Make a conclusion on the work done.

**INTRODUCTION**

Dijkstra's and Floyd's algorithms are two of the most fundamental and widely used algorithms in computer science, particularly in the field of graph theory. Both algorithms are used to solve the single-source shortest path problem, which is the problem of finding the shortest path between a single source vertex and all other vertices in a weighted graph.

Edsger Dijkstra, a Dutch computer scientist, developed his algorithm in 1956 while working on a project to build a compiler for the Electrologica X1 computer. His algorithm is an example of a greedy algorithm that works by iteratively building up the shortest path from the source vertex to all other vertices in the graph.

The algorithm maintains a set of visited vertices and a set of unvisited vertices, and at each iteration, it selects the unvisited vertex with the shortest path from the source and adds it to the visited set. Then it updates the distances of the adjacent vertices to the newly visited vertex, if a shorter path is found. The algorithm terminates when all vertices have been visited, and the final distances are the shortest path from the source vertex to all other vertices in the graph.

Robert Floyd, an American computer scientist, developed his algorithm in 1962. Unlike Dijkstra's algorithm, Floyd's algorithm is an example of a dynamic programming approach. It works by building a table of all pairs shortest paths, starting with paths of length 1, and gradually building up longer paths until all pairs of vertices have been considered.

The algorithm updates the table by considering all possible intermediate vertices between two vertices and choosing the shortest path. The final table contains the shortest path between all pairs of vertices in the graph.

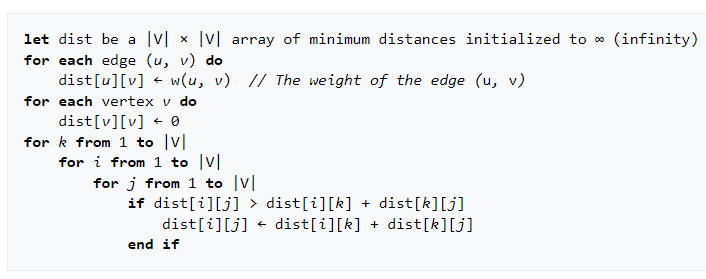
**IMPLEMENTATION**

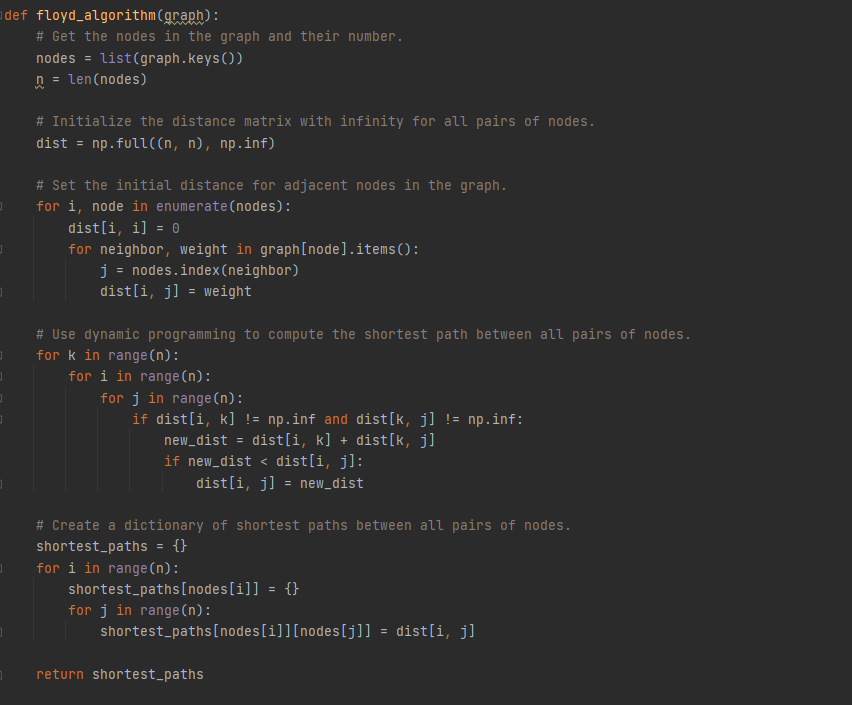
**Floyd Warshall Algorithm**

**Algorithm explanation :**

1. Initialize the shortest paths between any 2 vertices with Infinity.
2. Find all pair shortest paths that use 0 intermediate vertices, then find the shortest paths that use 1 intermediate vertex and so on.. until using all N vertices as intermediate nodes.
3. Minimize the shortest paths between any 2 pairs in the previous operation.
4. For any 2 vertices (i,j) , one should actually minimize the distances between this pair using the first K nodes, so the shortest path will be: min(dist[i][k]+dist[k][j],dist[i][j]). Dist[i][k] represents the shortest path that only uses the first K vertices, dist[k][j] represents the shortest path between the pair k,j. As the shortest path will be a concatenation of the shortest path from i to k, then from k to j.

PSEUDOCODE



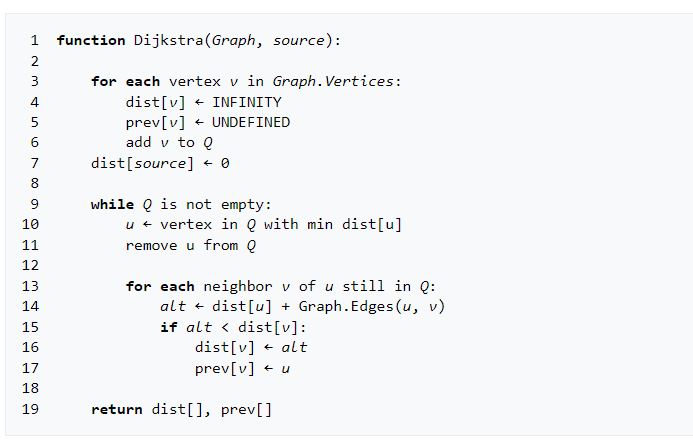


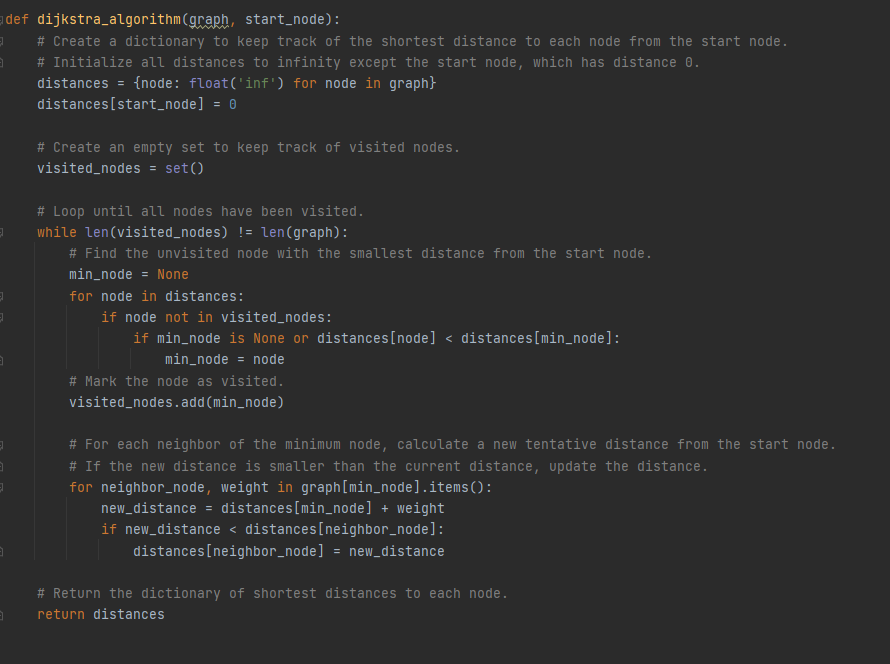
**Dijkstra's algorithm**

**Algorithm explanation :**

1. The very first step is to mark all nodes as unvisited,
2. Mark the picked starting node with a current distance of 0 and the rest nodes with infinity,
3. Now, fix the starting node as the current node,
4. For the current node, analyse all of its unvisited neighbours and measure their distances by adding the current distance of the current node to the weight of the edge that connects the neighbour node and current node,
5. Compare the recently measured distance with the current distance assigned to the neighbouring node and make it as the new current distance of the neighbouring node,
6. After that, consider all of the unvisited neighbours of the current node, mark the current node as visited,
7. If the destination node has been marked visited then stop, an algorithm has ended, and
8. Else, choose the unvisited node that is marked with the least distance, fix it as the new current node, and repeat the process again from step 4.

PSEUDOCODE





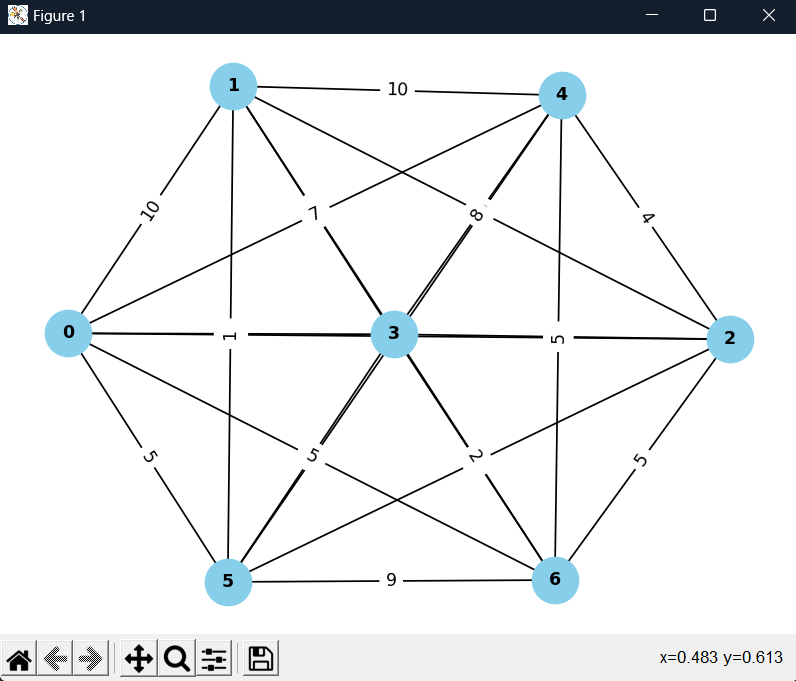
**RESULTS**

In Dijkstra’s algorithm time complexity is quadratic but in Floyd-Warshall algorithm it is cubic. In addition, the result of Dijkstra’s is just a subset of Floyd-Warshall algorithm. Dijkstra’s algorithm returns the shortest path between for a given vertex and all others but Floyd-Warshall algorithm returns the shortest path between all vertices.

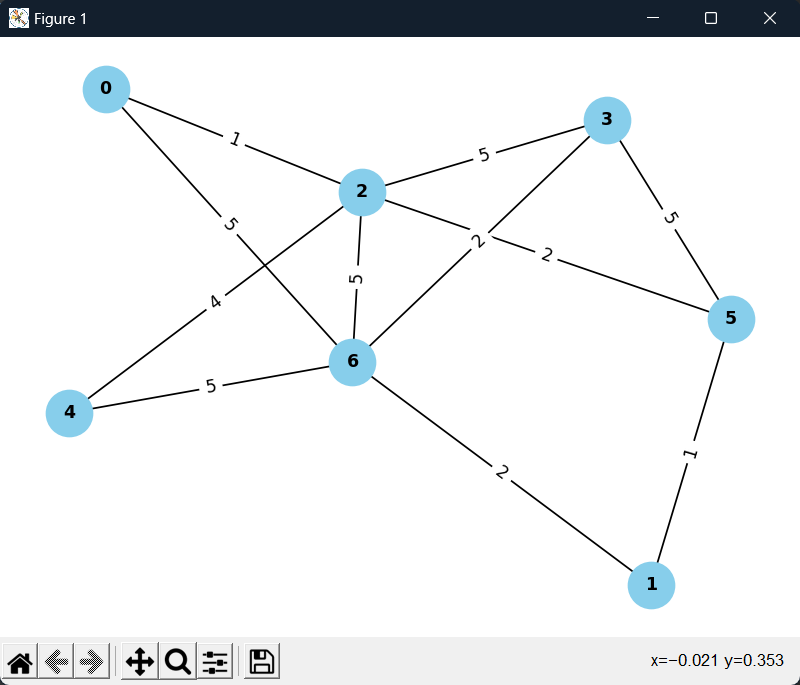
Dijkstra’s algorithm time complexity is for a given vertex, but if we try to find the shortest path for all vertex with Dijkstra’s algorithm then it will be which is equal time complexity of Floyd-Warshall algorithm .

If we try to find all shortest path between all vertex with Dijkstra’s algorithm, we will have the same efficiency and result as using Floyd’s algorithm. In conclusion in Floyd-Warshall algorithm there can be negative weighted but in Dijkstra’s algorithm there can not be.

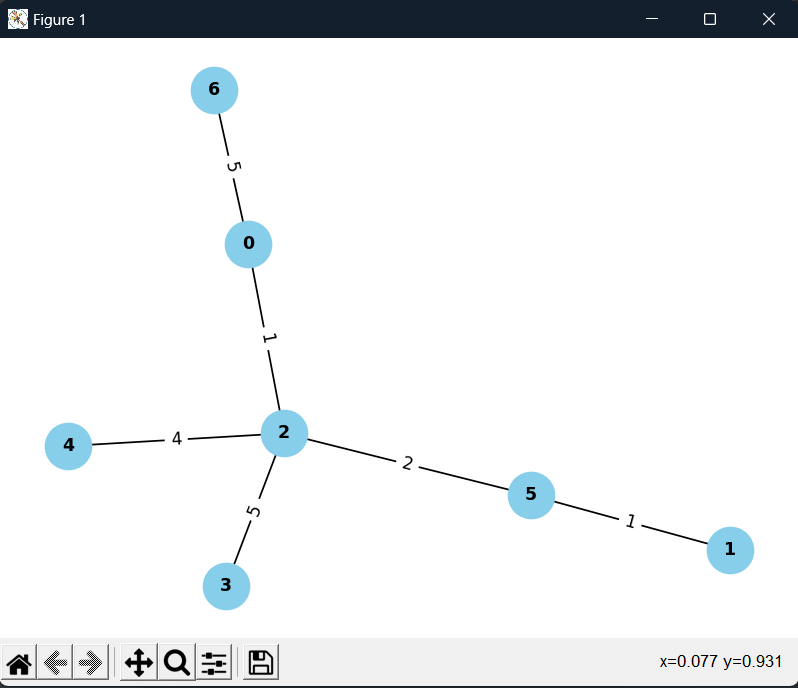
**Random Complete graph**



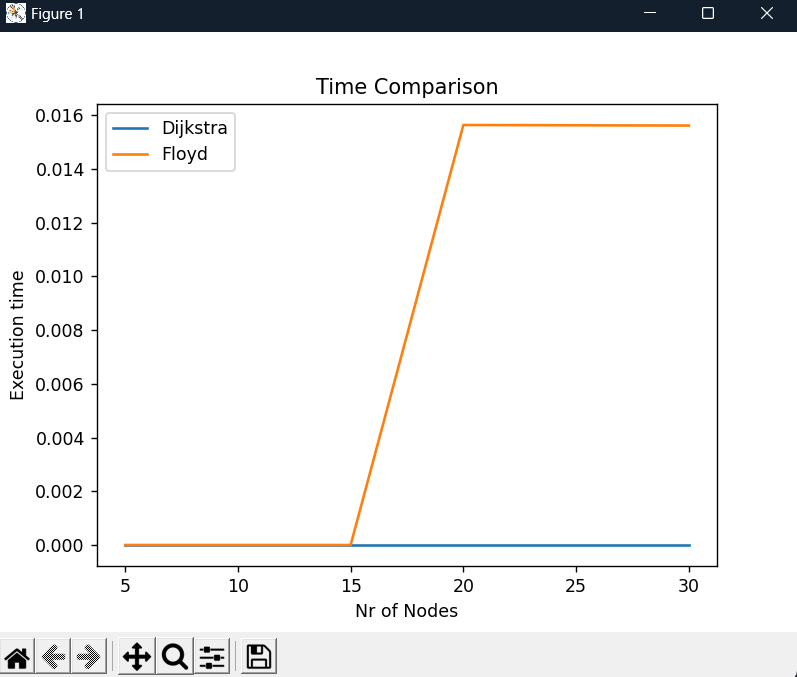
**Floyd's Algorithm Graph Tree**



**Dijkstra's Algorithm Graph Tree**



**Time Execution**



**CONCLUSION**

Both Floyd’s and Dijkstra’s algorithm may be used for finding the shortest path between vertices. The biggest difference is that Floyd’s algorithm finds the shortest path between all vertices and Dijkstra’s algorithm finds the shortest path between a single vertex and all other vertices. The space overhead for Dijkstra’s algorithm is considerably more than that for Floyd’s algorithm. In addition, Floyd’s algorithm is much easier to implement.

**Dijkstra’s solves the Single Source Shortest Path (SSSP) problem**. That is, we wish to find the shortest path from a single source node to a given destination node. A pertinent application of this algorithm is in the [link-state routing protocol](https://en.wikipedia.org/wiki/Link-state_routing_protocol), where each node uses it to create an internal picture of the network.

**Floyd-Warshall solves the All-Pairs Shortest Paths (APSP) problem**. In particular, we find the shortest paths between all pairs of nodes in the graph, which is computationally more expensive. This computational expense manifests in both the space required to store graph data and the time required to process it.

Dijkstra's algorithm is more suited for finding the shortest path from a single source to all other vertices in sparse graphs, while the Floyd-Warshall algorithm is more suited for finding the shortest path between all pairs of vertices in dense graphs. Both algorithms are widely used in various applications such as network routing, map navigation, and resource allocation.

Git Repo : https://github.com/andeiceban0352/Labs-Anul2/tree/main/Lab%20APA/Lab4