MINISTRY OF EDUCATION, CULTURE AND RESEARCH OF REPUBLIC OF MOLDOVA
TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS

Algorithm Analysis

Laboratory work 5 :Empirical analysis of algorithms: Dijkstra Algorithm, Floyd-Warshall Algorithm

Elaborated:
st.gr. FAF-211 Gazea Sandu
Verified:
asist.univ. Fiștic Cristofor

Chișinău, 2023

Content

Introduction

Dijkstra's algorithm and Floyd-Warshall algorithm are two popular algorithms used to solve shortest path problems in graphs.

Dijkstra's algorithm is a single-source shortest path algorithm, which means it finds the shortest path from a single source vertex to all other vertices in a weighted graph. The algorithm maintains a set of visited vertices and a set of unvisited vertices. It starts at the source vertex and iteratively selects the unvisited vertex with the smallest tentative distance, updating the distance of its neighboring vertices. Dijkstra's algorithm is particularly useful for finding the shortest path in a graph with positive edge weights, and it has a time complexity of O(—E—+—V—log—V—), where —E— is the number of edges and —V— is the number of vertices in the graph.

Floyd-Warshall algorithm, on the other hand, is an all-pairs shortest path algorithm, which means it finds the shortest path between all pairs of vertices in a weighted graph. The algorithm maintains a distance matrix, which stores the shortest path distances between all pairs of vertices in the graph. It iteratively updates the distance matrix by considering all intermediate vertices between any two pairs of vertices. Floyd-Warshall algorithm is particularly useful for finding the shortest path in a graph with both positive and negative edge weights, and it has a time complexity of O(—V—$^3), where|V|isthenumberofverticesinthegraph.$

Both Dijkstra's and Floyd-Warshall algorithms are widely used in various fields such as network routing, transportation planning, and computer graphics. Choosing the appropriate algorithm depends on the characteristics of the graph and the specific problem being solved.

Dijkstra's algorithm and Floyd-Warshall algorithm are fundamental algorithms for solving the shortest path problem in a graph. They are widely used in many real-world applications such as network routing, GPS navigation, airline scheduling, and more.

Dijkstra's algorithm is particularly suitable for finding the shortest path in a graph with positive edge weights. The algorithm is widely used in network routing protocols, where the weights of the edges represent the costs of the network links. Dijkstra's algorithm has a time complexity of O(—E—+—V—log—V—), where —E— is the number of edges and —V— is the number of vertices in the graph. The time complexity can be further improved to O(—E—+—V—) using a priority queue data structure.

In summary, Dijkstra's algorithm and Floyd-Warshall algorithm are powerful tools for solving the shortest path problem in graphs. They have their own strengths and weaknesses, and choosing the appropriate algorithm depends on the characteristics of the graph and the specific problem being solved.

Objectives

1. Implement Dijkstra's algorithm for finding the shortest path from a single source node to all other nodes in a graph with non-negative edge weights.
2. Implement the Floyd-Warshall algorithm for finding the shortest paths between every pair of nodes in a weighted, directed graph.
3. Generate random graphs to be used as input for both Dijkstra's and Floyd-Warshall algorithms, ensuring various sizes and edge weights.
4. Compare the execution time of Dijkstra's and Floyd-Warshall algorithms as the number of nodes in the input graph increases.
5. Visualize the input graph and the resulting shortest path trees produced by both Dijkstra's and Floyd-Warshall algorithms using the NetworkX and Matplotlib libraries.

Dijkstra

Dijkstra's Algorithm is a graph traversal algorithm that is commonly used to determine the shortest path between a starting node and all other nodes in a graph, assuming that the edge weights are non-negative. Initially, the starting node is assigned a distance of 0 and all other nodes are given a distance of infinity. The algorithm then proceeds by selecting the unvisited node with the smallest known distance at each step and updating the distances of its neighboring nodes. This is accomplished by comparing the sum of the current node's distance and the weight of the edge leading to the neighboring node to the current distance assigned to the neighbor. If the sum is less than the current distance, the algorithm updates the neighbor's distance with the smaller value. This process is repeated until all nodes have been visited, resulting in the shortest path from the starting node to each node in the graph being determined. Code:

```
 function Dijkstra(Graph, source):


for each vertex v in Graph.Vertices:
     dist[v] ← INFINITY
    prev[v] ← UNDEFINED
   add v to Q
dist[source] ← 0

while Q is not empty:
    u ← vertex in Q with min dist[u]
    remove u from Q

    for each neighbor v of u still in Q:
        alt ← dist[u] + Graph.Edges(u, v)
        if alt < dist[v]:
            dist[v] ← alt
            prev[v] ← u

return dist[], prev[]
```

Floyd-Warshall

The Floyd-Warshall Algorithm is a method to find the shortest paths between every pair of nodes in a directed, weighted graph. This algorithm works by systematically considering all combinations of nodes and updating a matrix of distances according to the shortest known path between each pair of nodes. Initially, a distance matrix is created, with the diagonal elements set to 0 and the off-diagonal elements set to the corresponding edge weights, or infinity if no direct edge exists between the node pair. The algorithm then repeatedly examines whether a shorter path between a pair of nodes can be found by going through an intermediate node. If a shorter path is discovered, the distance matrix is updated with the new, shorter distance value. This process continues until all possible intermediate nodes have been considered. At the end of the algorithm, the distance matrix will contain the shortest path distances between every pair of nodes in the graph.

Code:

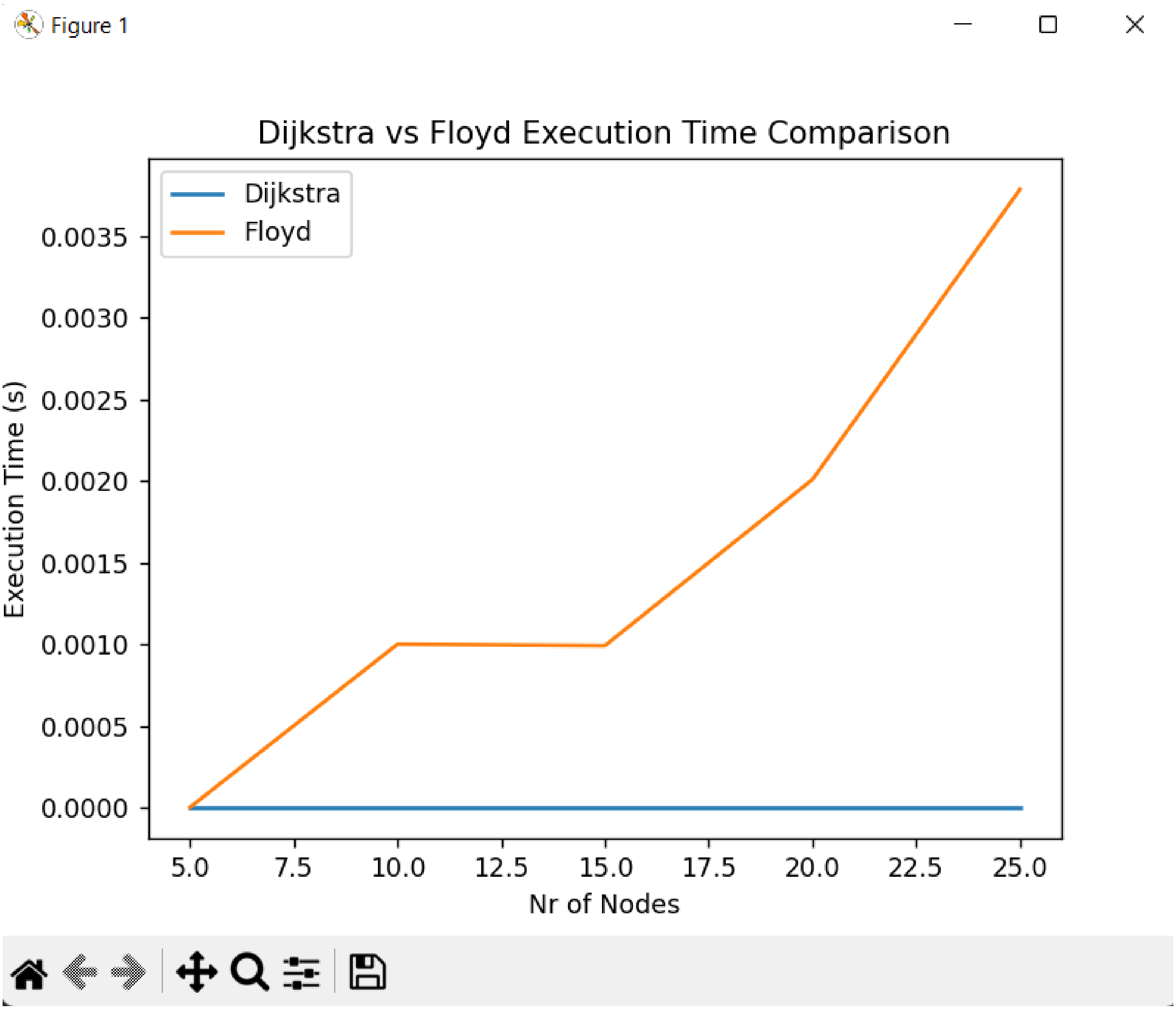
```
let dist be a |V| × |V| array of minimum distances initialized to infinity
for each vertex v
   dist[v][v] ← 0
for each edge (u,v)
   dist[u][v] ← w(u,v)  // the weight of the edge (u,v)
for k from 1 to |V|
   for i from 1 to |V|
      for j from 1 to |V|
         if dist[i][j] > dist[i][k] + dist[k][j]
            dist[i][j] ← dist[i][k] + dist[k][j]
```

Implementation

```
def dijkstra(graph, start):
    dist = {node: float('inf') for node in graph}
    dist[start] = 0
    visited = set()

    while len(visited) != len(graph):
        min_node = None
        for node in dist:
            if node not in visited:
                if min_node is None or dist[node] < dist[min_node]:
                    min_node = node
        visited.add(min_node)

        for neighbor, weight in graph[min_node].items():
            new_dist = dist[min_node] + weight
            if new_dist < dist[neighbor]:
                dist[neighbor] = new_dist
    return dist

def floyd(graph):
    nodes = list(graph.keys())
    n = len(nodes)
    dist = np.full((n, n), np.inf)

    for i, node in enumerate(nodes):
        dist[i, i] = 0
        for neighbor, weight in graph[node].items():
            j = nodes.index(neighbor)
            dist[i, j] = weight
```

```
for k in range(n):
    for i in range(n):
        for j in range(n):
            if dist[i, k] + dist[k, j] < dist[i, j]:
                dist[i, j] = dist[i, k] + dist[k, j]

return {nodes[i]: {nodes[j]: dist[i, j] for j in range(n)} for i in range(n)}
```

Screenshot:



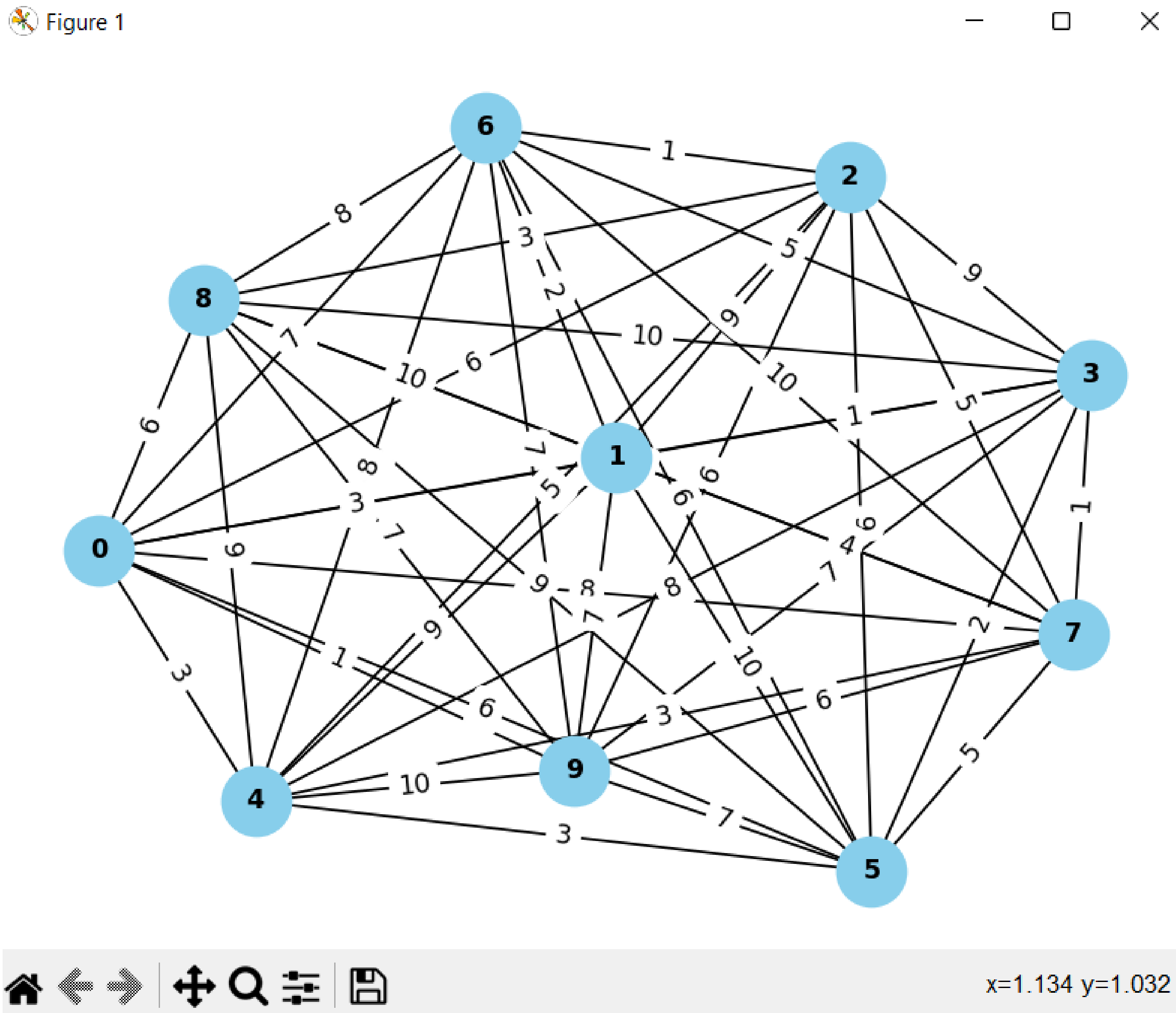


Figure 1
x=1.134 y=1.032

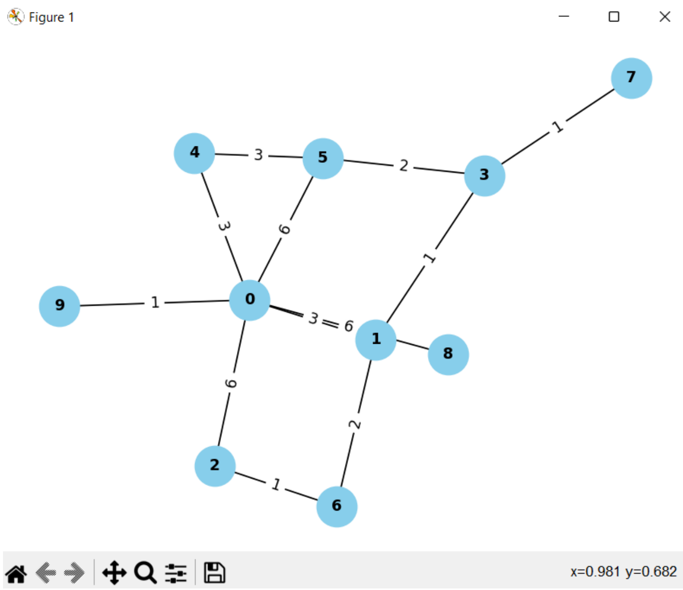


Figure 1
x=0.981 y=0.682

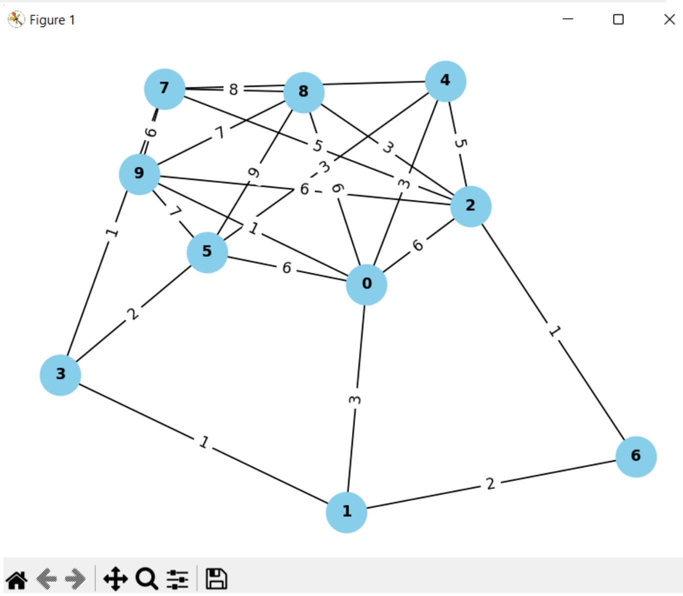


Figure 1

Conclusion

To sum up, Dijkstra's algorithm and Floyd-Warshall algorithm are two well-known and commonly used algorithms in graph theory.

Dijkstra's algorithm is a method for finding the shortest path from a single source vertex to all other vertices in a weighted graph. It works by selecting the vertex with the smallest distance and updating the distances of its neighboring vertices. It is effective in sparse graphs with non-negative edge weights but not suitable for graphs with negative edge weights or dense graphs due to its time complexity.

In contrast, the Floyd-Warshall algorithm is a dynamic programming algorithm that computes the shortest paths between all pairs of vertices in a weighted graph. It considers all possible intermediate vertices and updates the shortest path distances accordingly. It is more efficient than using Dijkstra's algorithm for every vertex pair in dense graphs with non-negative edge weights and can handle negative edge weights.

Both algorithms have strengths and weaknesses, and the choice of algorithm depends on the specific problem and graph characteristics. It is essential to note that neither algorithm is designed to handle negative cycles, and the Bellman-Ford algorithm is more appropriate for graphs with negative edge weights.

In conclusion, Dijkstra's algorithm and Floyd-Warshall algorithm are powerful tools for solving shortest path problems and have significantly contributed to the advancement of graph theory and its applications. They will continue to play an essential role in the future of graph theory and related fields.

$https://github.com/GSandu1/LabsAA.git$