Activity 3.2

José Carlos Martínez Núñez, Tania Sayuri Guizado Hernández

October 14, 2022

Contents

Problem Description 3
Algorithm Implementation 4
Dijkstra's Algorithm 4
Floyd's Algorithm 5
Test Cases 6
Hardware 7

Problem Description

- Write a C++ program that implements Dijkstra's and Floyd's algorithms for finding the shortest distance between pairs of nodes in a directed graph.

The program must read a number N followed by $N \times N$ non-negative integer values representing an adjacency matrix of a directed graph. The first number represents the number of nodes, the following values in the matrix, the value at position `i`, `j` represent the weight of the arc from node `i` to node `j`. If there is no arc between node `i` and node `j`, the value in the matrix must be -1.

The output of the program is, first with Dijkstra's algorithm the distance from node `i` to node `j` for all nodes, and then, the result matrix of Floyd's algorithm.

Sample input:

```
4
0 2 -1 3
-1 0 1 5
2 3 0 -1
3 -1 4 0
```

Sample output:

```
Dijkstra :
node 1 to node 2 : 2
node 1 to node 3 : 3
node 1 to node 4 : 3
node 2 to node 1 : 3
...

node 4 to node 2 : 5
node 4 to node 3 : 4

Floyd :
0 2 3 3
3 0 1 5
2 3 0 5
3 5 4 0
```

Algorithm Implementation

Dijkstra's Algorithm

```
void dijkstra(vector<vector<int>> graph, int src)
{
    vector<int> dist(graph.size(), INT16_MAX);
    vector<bool> visited(graph.size(), false);
    dist[src] = 0;
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
    pq.push({0, src});
    while (!pq.empty())
    {
        int u = pq.top().second;
        pq.pop();
        if (visited[u])
            continue;
        visited[u] = true;
        for (int v = 0; v < graph.size(); v++)
        {
            if (graph[u][v] && dist[u] + graph[u][v] < dist[v])
            {
                dist[v] = dist[u] + graph[u][v];
                pq.push({dist[v], v});
            }
        }
    }

    for (int i = 0; i < graph.size(); i++)
        if (src + 1 != i + 1)
            cout << "node " << src + 1 << " to node " << i + 1 << " : " << ((dist[i] == INT16_MAX)
    ? "INF" : to_string(dist[i])) << endl;
}
```

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger Dijkstra in 1956 and published three years later. This implementation uses a priority queue to store the nodes to

be visited, and a vector to store the distances from the source node to the other nodes. It has a time complexity of $O(V^2)$.

Floyd's Algorithm

```
void floydWarshall(vector<vector<int>> graph)
{
    vector<vector<int>> dist(graph.size(), vector<int>(graph.size()));

    for (int i = 0; i < graph.size(); i++)
        for (int j = 0; j < graph.size(); j++)
            dist[i][j] = graph[i][j];

    for (int k = 0; k < graph.size(); k++)
    {
        for (int i = 0; i < graph.size(); i++)
        {
            for (int j = 0; j < graph.size(); j++)
            {
                if (dist[i][k] + dist[k][j] < dist[i][j] && dist[i][k] != INT16_MAX && dist[k][j]
                    != INT16_MAX)
                    dist[i][j] = dist[i][k] + dist[k][j];
            }
        }
    }

    for (int i = 0; i < graph.size(); i++)
    {
        for (int j = 0; j < graph.size(); j++)
        {
            cout << ((dist[i][j] == INT16_MAX) ? "INF" : to_string(dist[i][j])) << " ";
        }
        cout << endl;
    }
}
```

Floyd's algorithm is an algorithm for finding the shortest paths between all pairs of nodes in a graph. It

was conceived by computer scientist Robert Floyd in 1962. This implementation uses a matrix to store the distances from the source node to the other nodes. It has a time complexity of $O(V^3)$.

Test Cases

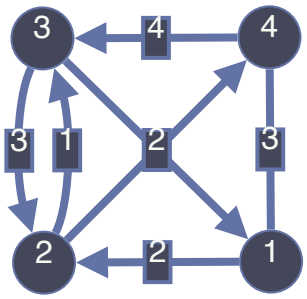

Figure 1: `example_test.txt` The Test Case given in the problem description.

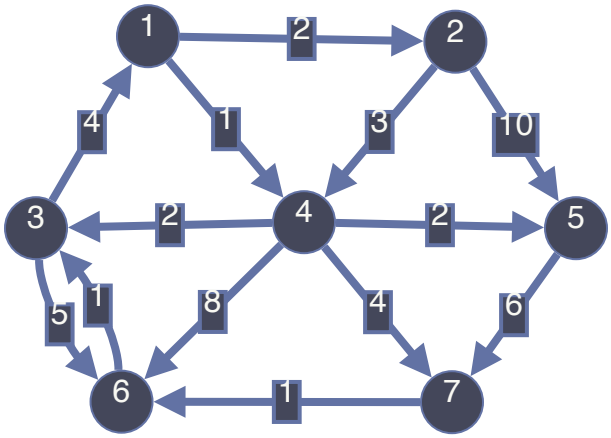

Figure 2: `all_paths_test.txt` 7 Nodes which all of them can be reached.

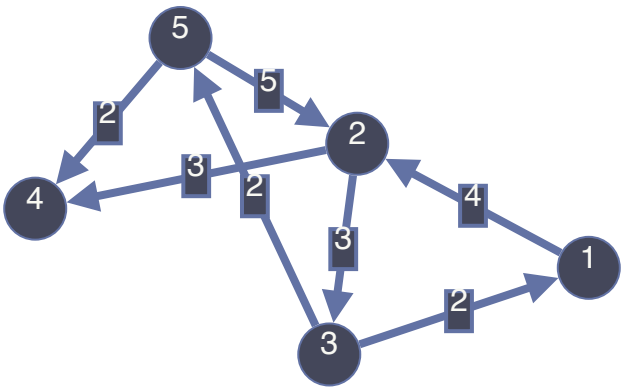

Figure 3: `dead_end_test.txt` Node 4 is a Dead End.

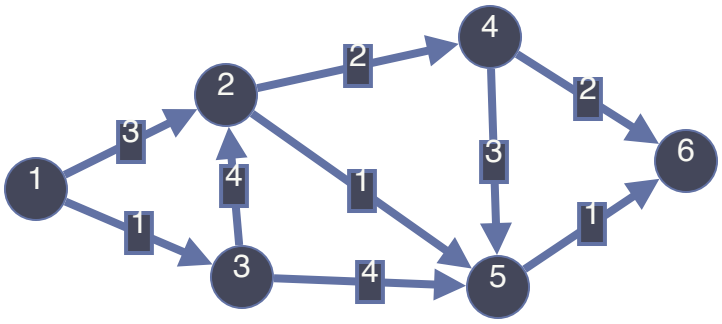

Figure 4: `missing_paths_test.txt` In this Test Case there are many missing paths between nodes.

The results of all the test cases are in the `results` folder.

Hardware

For this activity we used a MacBook Pro (13-inch, 2018, Four Thunderbolt 3 Ports) with the following specs:

- **Processor:** 2.3 GHz Quad-Core Intel Core i5
- **Memory:** 8 GB 2133 MHz LPDDR3
- **Graphics:** Intel Iris Plus Graphics 655 1536