JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

Department of CSE & IT

Bachelor of Technology, III Semester

2020-2021

Data Structure Mini-Project

Flight Management

TEAM MEMBERS

Prachi Chauhan - 19103309

Kartik Bhatia - 19103294

Saurav Sharma - 19103277

Batch: B9

PROBLEM STATEMENT

To Identify and implement the concepts of the shortest path problem (by using Dijkstra's shortest path algorithm).

INTRODUCTION

This project aims at making a software which avails the user to find the shortest path or route from a starting destination to final destination. We are implementing it with a real-life example of air lines. The program would inquire the user to input the details of current location airport and the final destination airport It will display all the routes between the two points along with the shortest route available.

There exist different types of algorithms that solves shortest path problem. Like

- Floyd - warshall algorithm
- Bellman-Ford Algorithm
- Dijkstra's algorithm

  This program focuses on the usage of Dijkstra's algorithm .

DATA STRUCTURES USED:

1.Graph

A graph is a mathematical abstract object, which contains sets of vertices and edges. Edges connect pairs of vertices. Along the edges of a graph it is possible to walk by moving from one vertex to other vertices. Depending on whether or not one can walk along the edges by both sides or by only one side determines if the graph is a directed graph or an undirected graph. In addition, lengths of edges are often called weights, and the weights are normally used for calculating the shortest path from one point to another point. In the real world it is possible to apply the graph theory to different types of scenarios. For example, in order to represent a map we can use a graph, where vertices represent cities and edges represent routes that connect the cities.

2.Array

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations.

3.Map

Maps are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values.

4.Priority Queue

A priority queue is a special type of queue in which each element is associated with a priority and is served according to its priority. If elements with the same priority occur, they are served according to their order in the queue.

Generally, the value of the element itself is considered for assigning the priority.

5.STL

-Pair

This class couples together a pair of values, which may be of different types (T1 and T2). The individual values can be accessed through its public members first and second.

-Vector

Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted

Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators.

ALGORITHMS USED :

Dijkstra

For each vertex within a graph we assign a label that determines the minimal length from the starting point s to other vertices v of the graph. In a computer we can do it by declaring an array d[]. The algorithm works sequentially, and in each step it tries to decrease the value of the label of the vertices. The algorithm stops when all vertices have been visited. The label at the starting point s is equal to zero (d[s]=0); however, labels in

other vertices v are equal to infinity (d[v]=∞), which means that the length from the starting point s to other vertices is unknown. In a computer we can just use a very big number in order to represent infinity. In addition, for each vertex v we have to identify whether it has been visited or not. In order to do that, we declare an array of Boolean type called u[v], where initially, all vertices are assigned as unvisited (u[v] = false). The Dijkstra's algorithm consists of n iterations. If all vertices have been visited, then the algorithm finishes; otherwise, from the list of unvisited vertices we have to choose the vertex which has the minimum (smallest) value at its label (At the beginning, we will choose a starting point s). After that, we will consider all neighbours of this vertex (Neighbours of a vertex are those vertices that have common edges with the initial vertex). For each unvisited neighbours we will consider a new length, which is equal to the sum of the label's value at the initial vertex v (d[v]) and the length of edge l that connects them. If the resulting value is less than the value at the label, then we have to change the value in that label with the newly obtained value. d [neighbours] = min (d [neighbours] , d[v] + l) (1) After considering all of the neighbours, we will assign the initial vertex as visited (u[v] = true). After repeating this step n times, all vertices of the graph will be visited and the algorithm finishes or terminates. The vertices that are not connected with the starting point will remain by being assigned to infinity. In order to restore the shortest path from the starting point to other vertices, we need to identify array p [], where for each vertex, where v ≠ s, we will store the number of vertex p[v], which penultimate vertices in the shortest path. In other words, a complete path from s to v is equal to the following statement

P = (s , … , p [p [p [v]]] , p [p [v]] , p [v] , v)

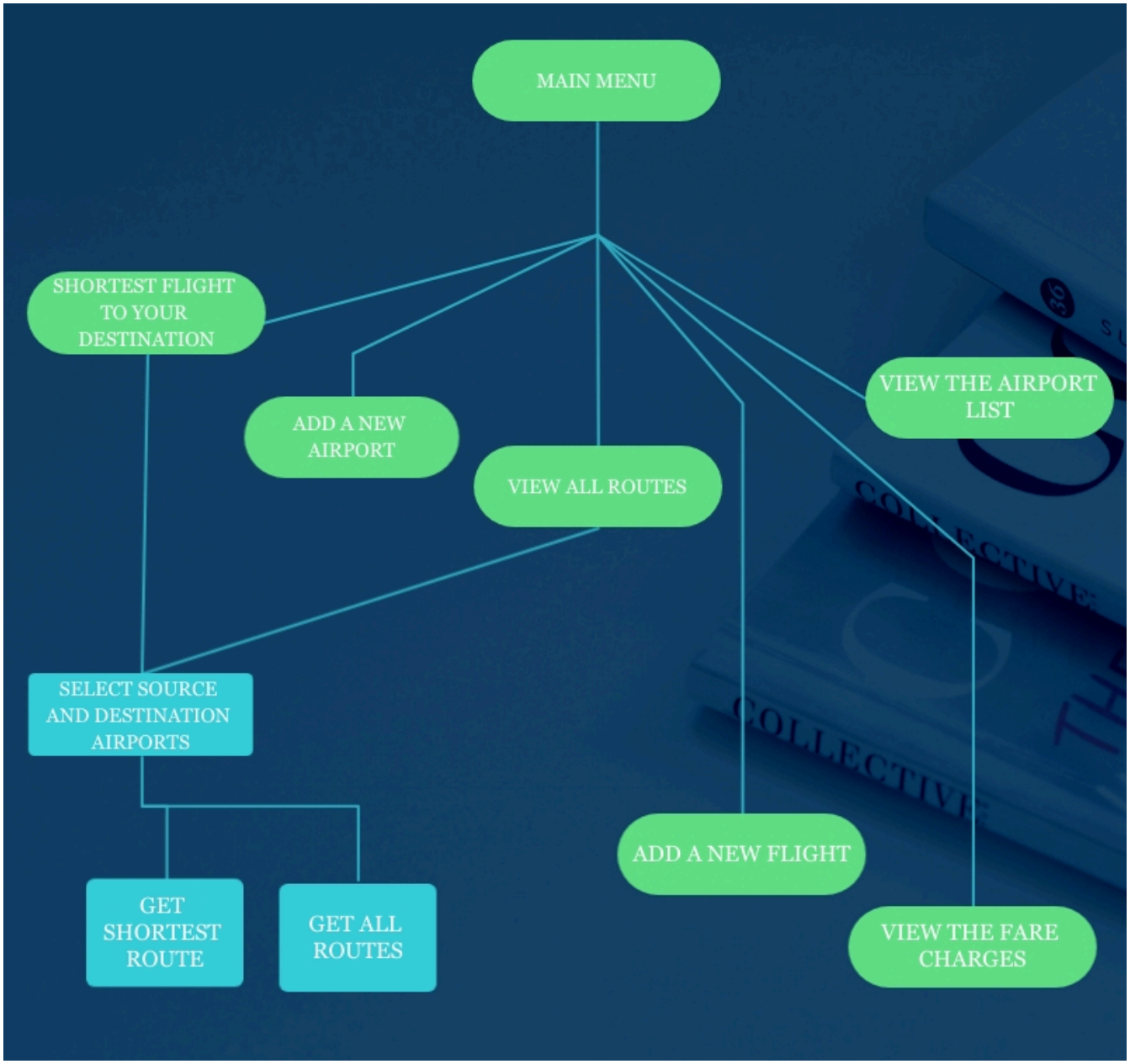
Depth First Search

The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

DETAILED DESIGN :

The following flowchart illustrates the basic design of the project



IMPLEMENTATION DETAILS AND RESULTS :

Main menu will appear in the first step which contains seven options. To proceed ahead you can choose any of the steps.

1. To Get shortest Flight Distance to your destination

- If you proceed with choosing option 1 the program would allow you to select Source airport and destination airport and will display the shortest flight distance between inputted airports.

```
                WELCOME TO TuTORIAL AIRLINES

    1. To Get shortest Flight Distance to your destination
    2. To Add a new Airport
    3. To View all Routes
    4  To Add a new Flight
    5. To View the Airport List
    6. To View Fare Charges
    7. To Exit
```

```
List of Source Airports
1 New Delhi
2 Mumbai
3 Lucknow
4 Chennai
Select Source Airport(1-4) :
1
List of Destination Airports
1 New Delhi
2 Mumbai
3 Lucknow
4 Chennai
Select Destination Airport(1-4) :
2

Shortest Flight Distance between New Delhi and Mumbai is 250

Shortest Route is: New Delhi->Lucknow->Mumbai->
```

2. To Add a new Airport

-If you proceed with choosing option 2.

The program would allow you to add a new airport to the list. As shown below.

```
 Enter Name of the New Airport

Noida



Do you want to go to the main page or not?(Y/N)
```

3. To View all Routes

- If you proceed with choosing option 3 .

Program would allow you to select Source airport and destination airport.And will display all the all the routes between inputted airports.

```
List of Source Airports
1 New Delhi
2 Mumbai
3 Lucknow
4 Chennai
5 Noida
Select Source Airport(1-5) :
1
List of Destination Airports
1 New Delhi
2 Mumbai
3 Lucknow
4 Chennai
5 Noida
Select Destination Airport(1-5) :
5
No Path Exist between New Delhi and Noida
```

```
List of Source Airports
1 New Delhi
2 Mumbai
3 Lucknow
4 Chennai
5 Noida
Select Source Airport(1-5) :
1
List of Destination Airports
1 New Delhi
2 Mumbai
3 Lucknow
4 Chennai
5 Noida
Select Destination Airport(1-5) :
2
New Delhi->Mumbai->
New Delhi->Lucknow->Mumbai->
```

4. To Add a new Flight

-If you proceed with choosing option 4.

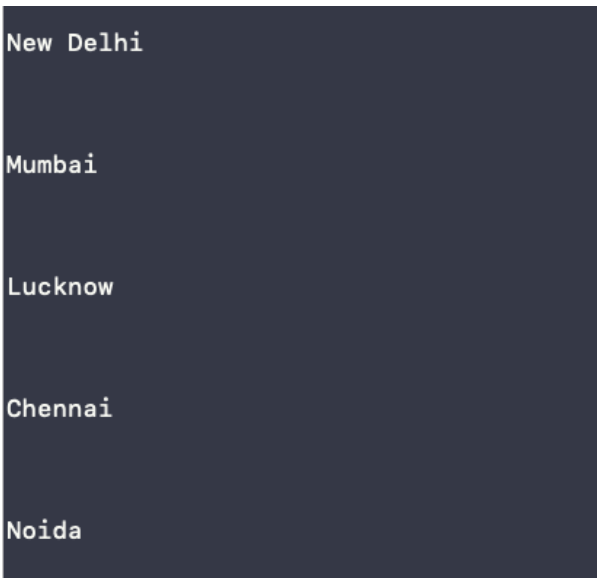
It would allow the user to add new flight. Program would allow you to select Source airport and Destination airport. Also the distance between two.

```
List of Source Airports
1 New Delhi
2 Mumbai
3 Lucknow
4 Chennai
5 Noida
Select Source Airport(1-5) :
1
List of Destination Airports
1 New Delhi
2 Mumbai
3 Lucknow
4 Chennai
5 Noida
Select Destination Airport(1-5) :
5


Enter Distance

34


Flight added successfully
```

5. To View the Airport List

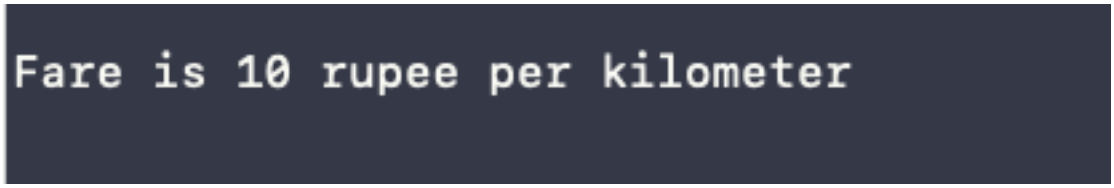
-If you proceed with choosing option 5.

Code would display a list containing names of all the airports.



6. To View Fare Charges

- If you proceed with choosing option 6. The code would display the fare per kilometres. By which user can calculate the total fare.



7. To Exit

-To exit you can simply press 7.

CONCLUSION :

We created a user friendly DOS-environment program that allows calculating shortest path between airports using a specific Dijkstra's algorithm. The program calculates the shortest routes with and without flight changes, and displays it. This is useful commercially, as in a net of a large amount of airports, it becomes tedious to search for an efficient flight route, and using this program, one can easily compare flight routes and calculate cost per kilometre, making the air routes as easily navigable as the metro routes.

FUTURE ENHANCEMENT:

We can upgrade this project by displaying more information like (flights Departure time). Can make project more user friendly by adding road routes headed towards the initial Airport.