School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC204J

Course Name: Design and Analysis of Algorithm

Title of Experiment	To implement Flight Trip Planner Using DIJKSTRA'S Algorithm
Name of the candidate	ANUJ, RHYTHM PAHWA, KARTIK AGAL
Team Members	ANUJ , RHYTHM PAHWA, KARTIK AGAL
Register Number	RA2011003010910, RA2011003010936, RA2011003010948
Date of Experiment	16-06-2022

SRMI INSTITUTE OF SCIENCE AND TECHNOLOGY

S.R.M. NAGAR, KATTANKULATHUR -603 203

BONAFIDECERTIFICATE

Register No._____

RA2011003010910, RA2011003010936, RA2011003010948

Certified to be bonafide record of the work done by _____ ANUJ, RHYTHM PAHWA, KARTIK AGAL *of C.Tech* _____, B.tech degree course in the practical 18CSC204J- Design and Analysis of Algorithms in SRM Institute of Science and Technology, Kattankulathur during the academic year 2021-22.

Date: **Lab Incharge:**

Submitted for university examination held in_____SRM Institute of Science and Technology, Kattankulathur.

Table of Contents:

1. **Chapter 1**

   Introduction

   1.1 Problem Statement

2. **Chapter 2**

   2.1 FLIGHT TRIP PLANNER

   2.2 The Dijkstra's Algorithm

   2.3 Constraints

   2.4 Pseudocode

3. **Chapter 3**

   3.1 Time Complexity Analysis

   3.2 Testing

   Conclusion

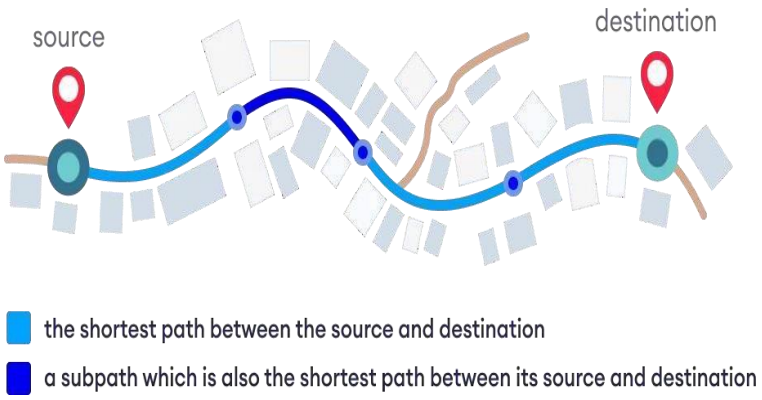
Aim:

FLIGHT TRIP PLANNER USING DIJKSTRA'S ALGORITHM

Team Members:

S No	Register No	Name	Role
1	RA2011003010948	Kartik	Rep
2	RA2011003010936	Rhythm Pahwa	Member
3	RA2011003010910	Anuj	Member

TOPIC :FLIGHT TRIP PLANNER USING Dijkstra's Algorithm



TEAM MEMBERS :

1.ANUJ(RA2011003010910)

2. RHYTHM PAHWA(RA2011003010936)

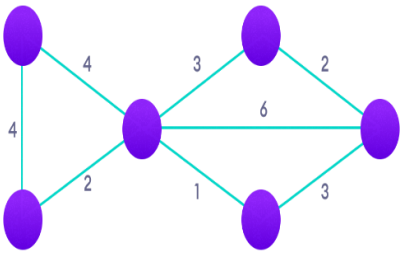
3. KARTIK AGAL(RA2011003010948)

PROBLEM STATEMENT

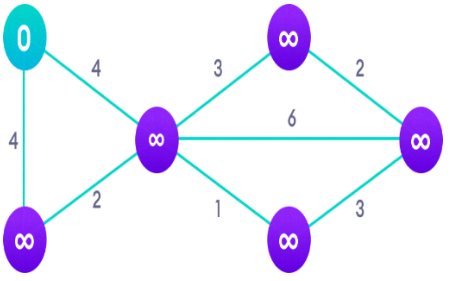
Different airline companies have different flight rates , there are problem faced by customer of how to choose the cheapest flight between two cities. The problem oforganizing inter-city flights is one of the most important challenges facing airplanes and how to transport passengers and commercial goods between large cities in less time and at a lower cost.
The task is to provide shortest route between two place.

Goal:

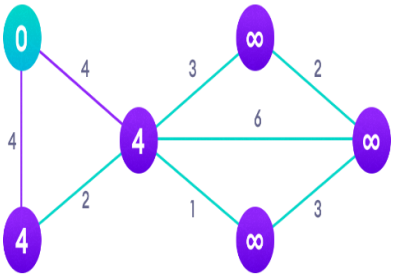
The goal of this project is to learn about how tomodel a real life problem using graphs, and to
implement Dijkstra's algorithm.



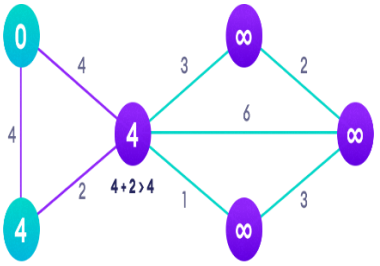
Step: 1



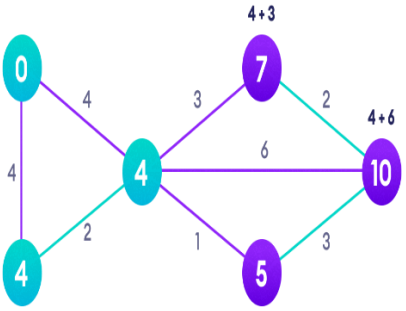
Step: 2



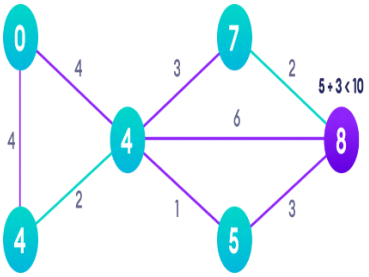
Step: 3



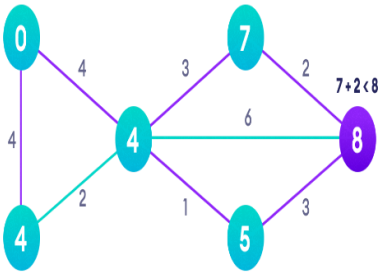
Step: 4



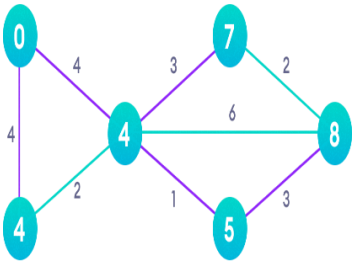
Step: 5



Step: 6



Step: 7



Step: 8

DESCRIPTION

We are given information about a set of flights between different cities. You want to go from a city A to a city B and would like to find out the earliest time by which you can reach B. The cities have names whichcan be strings, for example, “Delhi”, “Mumbai”,etc.

Each flight is described by a flight number, which is again a string (e.g., “AI 102”), source city, destination city, departure time, arrival time. Assume that no flight crosses midnight, i.e., departure time is after midnight and arrival time is before midnight. A query will specifya source city, a destination city and a departure time.

For example, it can specify source city as “Delhi”, destination city as “Mumbai” and departure time as 10:00 (assume that all times are specified in 24 Hour format, i.e. 00:00 to 23:59). There will be no two cities with the same name. Your goal is to find a sequence of flights starting from the source city (“Delhi” in this example), with the first flight starting at or after 10:00 and reaching Mumbai as early as possible.

Note that the trip could involve multiple flights. Also assume thatyou always want to reach the destination the same day(i.e., before midnight), if it is not possible to do so, should print an error message.

Constraints

There is one main complication. You must make sure that the arrival time of a flight in this trip is before the departure time of the next flight in the trip, plus a layover time. Ideal layover time is 2 hours or more. But, if cannot find an itinerary with 2+ hours layover then you may relax the problem to have a layover time 1.5 hours or more. If you still cannot find an itinerary with

1.5 hours layover, then you may relax the problem further with 1 hour or more layover. No further relaxations are permitted.

Note that for our problem (not for real world) "two 1.5+ hour but less than 2 hour layovers" and a combination of "one 1.5+ hour layover and one 2 hour layover" will be considered equivalent with respect to the layover constraint. Of these two itineraries should output the one that reaches the destination the soonest.

Dijkstra's algorithm works on the principle of relaxation where an approximation of the accurate distance is steadily displaced by more suitable values until the shortest distance is achieved.

Also, the estimated distance to every node is always an overvalue of the true distance and is generally substituted by the least of its previous value with the distance of a recently determined_path.

METHODOLOGY

The Dijkstra algorithm, also termed the shortest-route algorithm, is a model that is categorized within the search algorithms. Its purpose is to discover the shortest-route, from the beginning node (origin node) to any node on the tracks, and is applied to both directional and undirected graphs.
However, all edges must have non-negative values. The problem of organizing inter-city flights is one of the most important challenges facing airplanes and how to transport passengers and commercial goods between large cities in less time and at a lower cost. In this paper, the authors implement the Dijkstra algorithm to solve this complex problem and also to update it to see the shortest-route from the origin node (city) to the destination node (other cities) in less time and cost for flights using simulation environment.
Such as, when graph nodes describe cities and edge route costs represent driving distances between cities that are linked with the direct road. The experimental results show the ability of the simulation to locate the most cost-effective route in the shortest possible time (seconds), as the test achieved 95% to find the suitable route for flights in the shortest possible time and whatever the number of cities on the tracks application.

Example 1:

Suppose there are four cities: A, B, C, and D.

There is a flight named J1 from A to B with departure time = 1:00 and arrival time = 6:00

There is a flight named J2 from A to C with departure time = 1:00 and arrival time = 2:00

There is a flight named J3 from B to D with departure time = 7:00 and arrival time = 12:00

There is a flight named J4 from B to D with departure time = 8:00 and arrival time = 13:00

There is a flight named J5 from C to B with departure time = 4:00 and arrival time = 5:00

Suppose we want the least time consuming trip from A to D with departure time 1:00

There are four possible routes: 1. J1 to J3 2. J1 to J4 3. J2 to J5 to J3 4. J2 to J5 to J4

Of these routes, the first route is invalid since the layover time between two flights is only 1 hour.

Among the other routes, the least time consuming route is J2 to J5 to J3.

Now, Suppose we want the least time consuming trip from C to D with departure time 4:30

The possible routes are: “J5 to J4” and “J5 to J3”. However, the departure time of flight J5 is earlier at 4:00. Therefore, both the routes is invalid.

ALGORITHM

1) Create a set *sptSet* (shortest path tree set) thatkeeps track of vertices included in the shortest- path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this setis empty.

2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.

3) While *spt Set* doesn't include all vertices

a) Pick a vertex u which is not there in *sptSet* and has a minimum distance value.

b) Include u to *spt Set*.

Update distance value of all adjacent vertices of u. To update the distance values,

iterate through all adjacent vertices. For every adjacent vertex v, if the sum of distance value of u(from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

CODE

```
#include<iostream>
#include<climits> using
namespace std;

int miniDist(int distance[], bool Tset[]) // finding minimum
distance
{
    int minimum=INT_MAX,ind;

    for(int k=0;k<6;k++)
    {
        if(Tset[k]==false && distance[k]<=minimum)
        {
            minimum=distance[k];
            ind=k;
        }
    }
    return ind;
}

void DijkstraAlgo(int graph[6][6],int src) // adjacency matrix
{
    int distance[6]; // // array to calculate the minimum distance
for each node

    bool Tset[6];// boolean array to mark visited and unvisited for
each node


    for(int k = 0; k<6; k++)
    {
        distance[k] = INT_MAX;
        Tset[k] = false;
    }
```

```
distance[src] = 0;           // Source vertex distance is set 0

    for(int k = 0; k<6; k++)
    {
        int m=miniDist(distance,Tset);
        Tset[m]=true;
        for(int k = 0; k<6; k++)
        {
            // updating the distance of neighbouring vertex
            if(!Tset[k] && graph[m][k] && distance[m]!=INT_MAX
 && distance[m]+graph[m][k]<distance[k])
                distance[k]=distance[m]+graph[m][k];
        }
    }
    cout<<"Vertex\t\tDistance from source vertex"<<endl;
    for(int k = 0; k<6; k++)
    {
        char str=65+k;
        cout<<str<<"\t\t\t"<<distance[k]<<endl;
    }
}

int main()
{
    int graph[6][6]={
        {0, 1, 2, 0, 0, 0},
        {1, 0, 0, 5, 1, 0},
        {2, 0, 0, 2, 3, 0},
        {0, 5, 2, 0, 2, 2},
        {0, 1, 3, 2, 0, 1},
        {0, 0, 0, 2, 1, 0}};
    DijkstraAlgo(graph,0);
    return 0;
}
```

OUTPUT

Vertex	Distance	from source
A	0	
B	10	
C	20	
D	23	
E	20	
F	21	

Time complexity analysis:

Time complexity of Dijkstra's algorithm is O(V^2) where V is the number of vertices in the graph.

Conclusion

Hence, implement the Dijkstra algorithm to solve this complex problem and also to update it to see the shortest-route from the origin node (city) to the destination node (other cities) in less time andcost for flights using simulation environment.

The experimental results show the ability of the simulation to locate the most cost-effective route in the shortest possible time (seconds), as the test achieved 95% to find the suitable route for flights in the shortest possible time and whatever the number of cities on the tracks application.

REFRENCES:

1.https://www.crio.do/blog/mini-projects-for-computer-science-engineers/

2.https://web.stanford.edu/class/cs166/handouts/090%20Suggested%20Project%20Topics.pdf

3.https://github.com/topics/daa