FLIGHT TRIP PLANNER

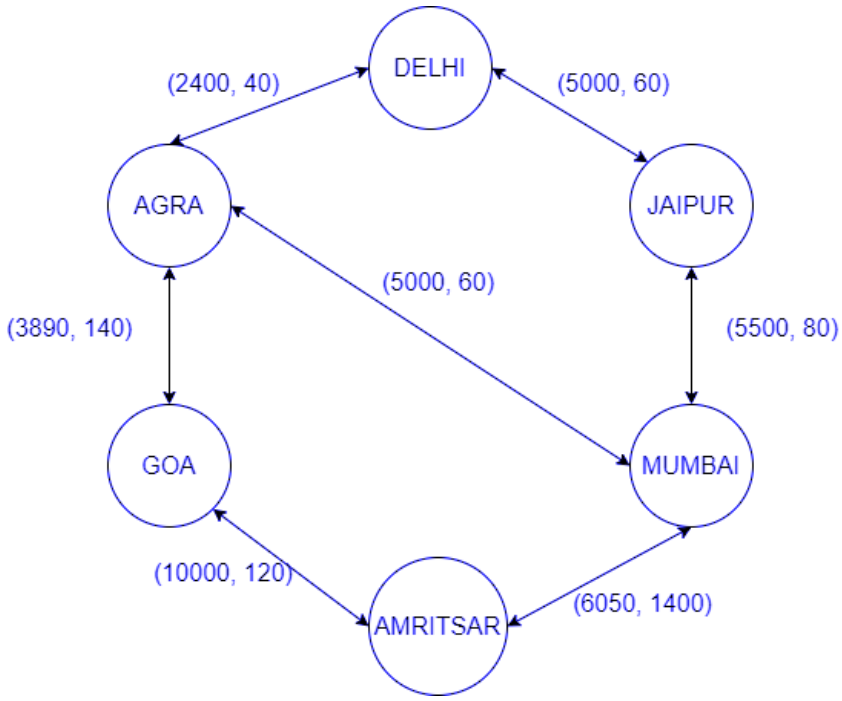
PROJECT REPORT

1. INTRODUCTION

The aim of this project is to optimise flight trip booking experience.In order to do so a particular flight is chosen for the user that is either time saving or cheapest given the departure and arrival location by the user.The entire project it made in C++ using various data structures present in standard template library ,which will be mentioned below, the major backbone of the project is Dijkstra algorithm.The project is extremely close to a real life project as the ideas and implementation is made in a similar fashion .

2. IMPLEMENTATION

The details of flights are saved as nodes in a weighted graph where the weight is a pair of cost and time taken while the nodes represent different cities. User is then asked to enter his/her departure location as well as arrival location along with the optimizable condition i.e. whether he wishes to receive information about the cheapest route or the route that takes minimal time. Dijkstra's Algorithm is applied to achieve the aforesaid. The structure of the graph is shown below.



3. TECHNOLOGY USED

Language: C++

C++ is a general-purpose programming language created by Bjarne Stroustrup as an extension of the C programming language. The language has expanded significantly over time, and modern C++ now has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation. It was designed with a bias toward system programming and embedded, resource-constrained software and large systems, with performance, efficiency, and flexibility of use as its design highlights.

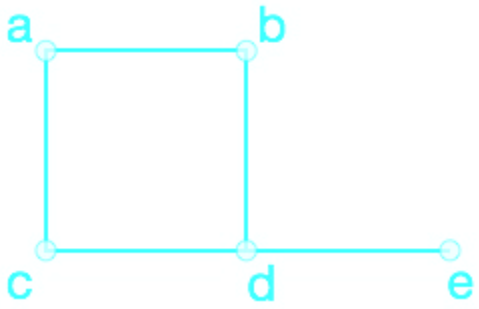
Data Structures:

1. **Graphs**

A graph is a pair of sets (V, E), where V is the set of vertices and E is the set of edges, connecting the pairs of vertices. In our project we have implemented graph using adjacency

list using data structures like map, list and pair present in standard template library. In maps we used the key as the name of a node and the list to store the name of all the nodes connected to it as values, in that list there are two sections one is name of the node the second part contains the cost and time.

A small pictorial representation of the graph.



2. **Stack**

Stacks are a type of container adaptors with LIFO(Last In First Out) type of working, where a new element is added at one end and (top) an element is removed from that end only. In Our project stack is implemented using stack of STL. It is used for a very rudimentary purpose to print the name of all the cities he should take the flight from to get the cheapest or most time saving flight.

Algorithm : Dijkstra's Algorithm

This is a greedy type of algorithm. The foundation of this algo lies in Kruskal's Minimum Spanning Tree Algorithm.To understand the Djikstra's algorithm we must understand minimum spanning tree , in layman language we can say, a minimum spanning tree (MST) or minimum weight spanning tree for a weighted, connected and undirected graph is a spanning tree with weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree. Once we understand this concept then the concept of Djikstra'sAlgorithm is extremely easy to understand. Like Kruskal's

MST, we generate a SPT (shortest path tree) with a given source as root. We maintain two sets, one set contains vertices included in the shortest path tree, the other set includes vertices not yet included in the shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.

Below are the detailed steps used in Dijkstra's algorithm to find the shortest path from a single source vertex to all other vertices in the given graph.

Algorithm

1) Create a set sptSet (shortest path tree set) that keeps track of vertices included in the shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is 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 sptSet doesn't include all vertices

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

b) Include u to sptSet.

c) 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.

4. DATASET DESCRIPTION

The dataset for the project includes rows of flight information from departure city to arrival city including the overall time taken and money it will cost for one passenger. Project's main agenda is to find the optimum route according to passenger's demand. They have the option to book a flight that could save them their time or money accordingly. The project is extremely close to a real life project as flight routes with a few stops might actually benefit the passenger in saving his recources.

SAMPLE DATASET IS AS SHOWN:

DEPARTURE CITY	ARRIVAL CITY	COST (₹)	TIME (mins)
DELHI	MUMBAI	2286	135
DELHI	AHMEDABAD	1950	80
DELHI	HYDERABAD	2088	125
GOA	MUMBAI	1953	75
GOA	DELHI	3076	155
KOCHI	GOA	3328	285
KOLKATA	CHENNAI	2693	145
INDORE	KOCHI	3376	210
CHENNAI	KOCHI	2618	80