S20190010034_A5

C Bhavesh Kumar

November 2020

1 CheapestTripPlanner

To Solve this problem, Dijkstra algorithm has been used.
In Dijsktra algorithm extra information has been augmented to be able to solve this problem.
For this problem i have used an array to store the time of arrival on city.
Also, conditions based on this new information have been added to check if there exists a flight in the needed time interval.
Graph data structure used is adjacency list.
The graph structure used for this problem is as follows

```
struct Graph{
  int vertices;
  struct node** adj;
};
```

Also each node in the graph is of type

```
struct node{
  int city;
    int cost;
    char flight[200];
    int arrival,departure;
  struct node* next;
};
```

The implementation od dijsktra algorithm for this problem is listed below

```
void cheapesttripplanner(struct Graph *graph,int source,int
    destination,int departure,int arrival){
    int n=0;
    struct heapnode a[graph->vertices];
    int pos[graph->vertices];  //Used to locate the indices of
    vertices in heap.
    int visited[graph->vertices];//maintained to keep track of
    visited vertices.
    int dist[graph->vertices];//Contains distances or path length
    from source to the vertices,
    int time[graph->vertices];//TIme array to keep track of the
    arriving time in the cities.
    for(int i=0;i<graph->vertices;i++){
        a[n].v=i;
```

```
        pos[a[n].v]=n;
        a[n].dist=INT_MAX;
        n++;
        min_heapify(a,n,n-1,pos);
        visited[i]=0;
        dist[i]=INT_MAX;
        time[i]=2400;
    }
    a[source].dist=0;
    dist[source]=0;
    time[source]=departure;
    min_heapify(a,n,source,pos);
    while(n>0){
        struct heapnode temp=extract_min(a,&n,pos);
        visited[temp.v]=1;
        int u=temp.v;
        pos[u]=graph->vertices;
        struct node *t=graph->adj[temp.v];
        while(t!=NULL){
            int v=t->city;
            int cost=t->cost;
            if(visited[v]!=1 && dist[u]!=INT_MAX && cost+dist[u]<
    dist[v] && time[u]<=t->departure && t->arrival<=arrival){
                dist[v]=dist[u]+cost;
                time[v]=t->arrival+30;
                if(time[v]%100>=60){
                    time[v]+=40;
                }
                time[v]=time[v]%2400;
                int position=pos[v];
                decreasekey(a,n,position,dist[v],pos);
            }
            t=t->next;
        }
    }
}
```

Min heap has been used to implement the dijsktra algorithm to decrease the time complexity and make it efficient.

2 Time Complexity

Time complexity of the above algorithm is $O(ElogV)$, where E is number of edges and V is number of vertices.

Since min heap is used to implement we get the time complexity of $O(ElogV)$

Each operation which operates on min heap takes time $O(logn)$