CSE 551 Programming Assignment

Manish Aakaram – 1217852896

Input:

The code takes a .csv file containing the flight details in the order origin, destination, departure time, arrival time, capacity. Flight data are taken from expedia.com between the given 10 airports on the date of 6th January 2020. This data is parsed into flights.csv file.

Algorithm:

In the algorithm, approach is to divide the time into 24 1-hour intervals and round every flight departure and arrival time to the nearest hour, i.e., the flight departing at time 10:15 is rounded to time 10:00 and the flight departing at 21:40 is rounded to time 22:00. Then, all the flights with depart time and arrival time are each taken as nodes in a graph map. The connecting flights between these nodes in the graph become edges with its capacity as weight.

For example, in the csv file, each row represents a flight.

SEA	DEN	12:35	16:21	180

For this input, the nodes are named as SEA13 and DEN16 and there is a directed edge between these nodes with weight 180.

We create an adjacency matrix from the above information with the size NxN where N is the different permutations of nodes/cities with 24 1-hr intervals. Now, we use the Ford-Fulkerson algorithm to calculate the individual maximum flow of all possible combinations of origins and destinations and add them to get total maximum capacity.

Breadth first search is used to find the augmented path from source to sink.

Pseudo Code:

Breadth First Search: (source: Wikipedia)

Algorithm 2 Breadth-first search

```
function Bfs(graph[V,V], source, sink, parent[])
    visited[V]                ▷ Create a visited array and mark all vertices as not visited
    for all v in V do
        v ← false
    end for
            ▷ Create a queue, enqueue source vertex and mark source vertex as visited
    queue q
    q.push(source)
    visited[source] ← true
    parent[source] ← −1
    while q is not empty do                                        ▷ Standard BFS loop
        u ← q.front
        q.pop
        for v is 0 to size of list do
            if visited[v] is false and graph[u][v] > 0 then
                q.push(v)
                parent[v] ← u
                visited[v] ← true
            end if
        end for
    end while
            ▷ If the sink is reached starting from the source, return true and else false
    if visited[sink] is true then
     return true
    else
     return false
    end if
end function
```

Ford Fulkerson Algorithm:

Algorithm 3 The Ford-Fulkerson algorithm

```
function MaxFlowFordFulkerson(N)
    Input: Flow network N = (G, c, s, t)
    Output: A maximum flow f for N
    for all Edges eϵN do
        f(e) ← 0
    end for
    stop ← false
    while stop is false do
        Traverse G starting at s to find an augmenting pathe for f
        if Augmenting path π exists then
            Compute the residual capacity △_f(π) of π
            △ ← +∞
            for all Edges eϵπ do
                if △_f(e) < △ then
                    △ ← △_f(e)
                end if
            end for
            Push △ = △_f(π) units of the flow along path π
            for all Edges eϵπ do
                if e is a forward edge then
                    f(e) ← f(e) + △
                else
                    f(e) ← f(e) − △
                                                        ▷ e is a backward edge
                end if
            end for
        else
            Stop ← true                                 ▷ f is a maximum flow
        end if
    end while
end function
```

Time Complexity:

Time complexity is (Edges)*(Maximum Flow).

Nodes created are 220.

Output:

Maximum Flow for the given dataset is 5855