Assignment-5

Problem Statement:

Given a list of flights with the information about various parameters we have to find the maximum number of passengers that can fly from source - Los Angles (LAX) to destination – New York (JFK) between 6:00 AM in the present day to 5:59 AM in the following day.

The possible layover airports are San Francisco (SFO), Phoenix (PHX), Seattle (SEA), Denver (DEN), Atlanta (ATL), Chicago (ORD), Boston (BOS) and Washington DC (IAD).

Solution:

To solve this problem, we can use Fold Fulkerson Algorithm for finding the maximum flow between source and destination. We however have to consider the time constraints posed on the flight arrivals and departures.

We need to consider only the flights that start after 6:00 AM and reach the destination before 6:00 AM of the following day. Also in the scenario that the passengers are hopping from flight to flight, the flight from intermediate airport to the next airport in the itinerary should be only after they reach the intermediate airport.

Adjacency Matrix:

- To consider all these scenarios we have to build and update the adjacent matrix/ graph different to a regular graph.
- We also have to consider the arrival and departure times for each airport. If we consider the airports to be the nodes of the graph and the flights between two airports the edges between the graph, the weight of the edges would be the maximum capacity of passengers who can travel in that route (sum of all the capacities of the flights between the two airports).
- To accommodate all the flights departing at different times from an airport and all the arriving flights to an airport, we consider 24 nodes for each airport, one for each hour of the day when they either depart from or arrive at the airports. This is not the case however for the source and destination servers.
- The total number of nodes would therefore be 194,
  8 * 24 = 192 24 each for all the intermediate airports and 2 for source and destination.
- The weights of the edges between the same airport nodes but with different hour values is set to infinite.

Algorithm:

1. Convert the flight arrival times and departure times to 24hrs format with 6:00 AM indicating 0 index and the following day 5:00 AM as 23.
2. Initialize the adjacency matrix to 0 for all the edges that do not belong to the same airport, and to infinite if the nodes belong to the same airport but have a different hour value associated to it and the hour value is greater than the hour value of the source node of the edge.
3. For every flight in the given dataset:
    a. Identify the source and destination based on the arrival time and departure time.
    i ← source_departure time
    j ← destination_arrival time
    b. Ignore all the flights that have the destination as the source, and any flights that arrive after the departure time after the 24 hours conversion based on 6:00 AM.
    c. Update the adjacency matrix of the source node and destination node with the capacity of the flight if the flight is not a direct flight from source to destination
    AdjacencyMatrix[i][j] ← AdjacencyMatrix[i][j] + flight capacity
    d. For any direct flights add the flight capacity to the maximum capacity
    max_flow ← max_flow + flight capacity

Ford-Fulkerson Maximum Flow Algorithm:

We find the maximum flow between the source and sink nodes, by following the steps below:

- Checking every augmented path between the source and sink in the residual graph
- Finding the bottle neck value or minimum flow in the path
- Add the minimum flow of the path to the current maximum flow value
- Updating the graph by updating the residual path or the backward label path.

Algorithm:

1. Initialize max_flow ← 0
2. For every augmented path in the residual graph from source to sink:
    a. path_flow ← the minimum flow of the path or the bottle neck
    b. max_flow ← max_flow + path_flow (add the path flow to maximum flow)
    c. Update the residual graph by reducing the capacity of all the edges in the path and adding backward labelled edges.
3. Return the max_flow when there is no augmented path present in the residual graph

Time Complexity:

1. The time complexity needed for the construction of Adjacent Matrix is $O(N^2)$ where N – number of nodes

2. The time complexity needed for the Ford-Fulkerson algorithm is $O(EN^3)$, where E – is the number of edges.
3. The overall time complexity would therefore be $O(N^2 + EN^3)$ which is equivalent to $O(EN^3)$.

Result:

The result of running the above mentioned algorithm on the flights data given as part of the flights.txt file is

8601

Steps to run the code:

1. Have python setup in the system.
2. Place the flights.txt file in the same folder as the python file – 'NAS.py'
3. Run the following python command:
   python ./NAS.py