

## **CSE 551: Foundation of Algorithms**

### **Homework – 4 (Programming Assignment)**

**Dhruv Popat**  
**ASU ID: 1217065525**

In this assignment the goal is to find the capacity of total passengers that could travel from source (LAX) to destination (JFK) on 6<sup>th</sup> Jan 2020 in simplified NAS (National Airspace System).

This capacity should follow the following constraint:

- It should be within 24 hours' time period from 00:00 to 23:59 on 6<sup>th</sup> Jan 2020
- It should only be for American Airlines (AA), Delta Airlines (DL) and United Airlines (UA)
- It should cover San Francisco (SFO), Phoenix (PHX), Seattle (SEA), Denver (DEN), Atlanta (ATL), Chicago (ORD), Boston (BOS) and Washington DC (IAD) for multi-stops. For multi-stop, the finish time of a flight at an intermediate stop must be less than the starting time of the next flight from that very same intermediate stop.

#### **Input of the program –**

The input to the program consists of List made from the CSV file (Flights\_Data.csv) with data for each flight between two nodes for AA, DL and UA.

The CSV data consists of following columns –

- DepartureAirport
- ArrivalAirport
- StartTime
- FinishTime
- Capacity
- Flight
- Airline
- FlightNumber

**Using this data two columns are made –**

flightdata['Dept\_Air\_Time'] – consisting Time dependent nodes for all Departing Airports

flightdata['Arr\_Air\_Time'] - consisting Time dependent nodes for all Arrival Airports

(E.g. For a connection from LAX to PHX from 12:30 to 14:30 following column values will be made

flightdata['Dept\_Air\_Time'] = LAX-12:30

flightdata['Arr\_Air\_Time'] = PHX-14:30

)

### **Pseudo code of Algorithm Used –**

Read data from the csv file

Make List of time dependent nodes for Departing Airport and Arrival Airport (e.g. PHX-12:00, PHX-14:30 etc.)

#### **Function: Add Edges (Graph, flightdata)**

Generate Edges between all the connections of flights with capacity

Sort all the intermediate airports according to their time

#### **Function: Sort\_nodes\_time (airport, data)**

Get List of all intermediate airport nodes which are sorted with respect to time

#### **Function: Add\_start\_end\_node (data, node)**

**If node == 'LAX'**

Make a starting LAX node with edges connecting to all LAX time dependent nodes (e.g. LAX -> LAX-12:00, LAX -> LAX-13:10)

**end**

**If node == 'JFK'**

Make an ending node JFK where edges go from JFK time dependent nodes to JFK

(e.g. JFK-16:00 -> JFK, JFK-17:00 -> JFK)

**end**

#### **Function: Add\_intermediate\_edges (airports, airport\_sorted\_nodes)**

Make edges between all the nodes with their updated capacity for same intermediate airport which can be taken keeping start and finish time constraint in mind

#### **Compute the Max-Flow between LAX (source) to JFK (sink)**

Start with initial flow as 0

While there is an augmenting path from LAX to JFK

Add this path-flow to Max-Flow

Return Max-Flow

## Time Complexity –

The time complexity of the algorithm will be the time complexity for ford Fulkerson algorithm plus the cost of finding edges for the nodes for each unique airport and adding the edges between nodes that can be taken after considering time constraint. We have X number of unique airports and  $V_x$  nodes for each unique airport.

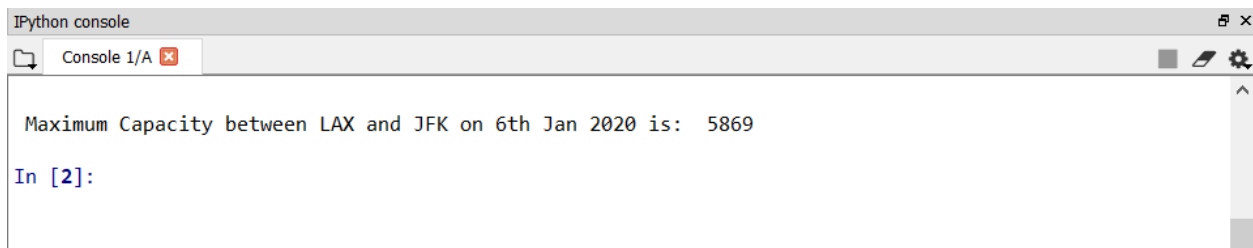
The total complexity for adding edges between the possible paths for each airport will be –  
**Big O ( $X (V_x)^2$ )**

The time complexity of ford Fulkerson is –  
**Big O (Maxflow \* E)**

Thus, the total time complexity will be-  
**Big O ( $(X (V_x)^2) + (Maxflow * E)$ )**

## Output –

The output consists of total number of passengers that would be able to travel from LAX to JFK on Jan 6<sup>th</sup>, 2020.



```
IPython console
Console 1/A
Maximum Capacity between LAX and JFK on 6th Jan 2020 is: 5869
In [2]:
```