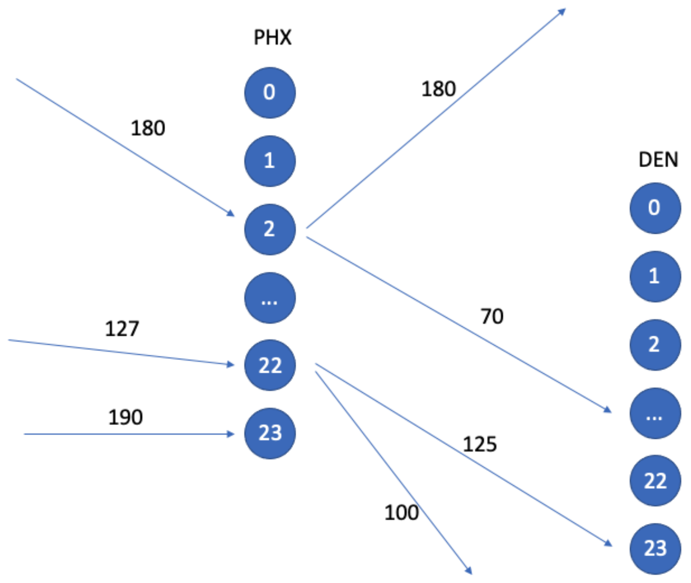
CSE 551 Programming Assignment

Intro

In this assignment, we are given a list of flights occurring at different times between 10 airports and are tasked with finding the maximum capacity of passengers that can be transported from LAX to JFK within a 24-hour time period. This problem combines constraints and complexity from both the Max Flow problem and the Weighted Interval Scheduling problem.

Algorithm Methodology

The algorithm that I used for this project was a combination of greedy, dynamic programming, and brute-force. Each of the above strategies works well for a specific part of solving the problem. I represented the map of airports as a graph in which each city is further represented by a subgraph of 24 nodes, 1 node for each hour (0-23). Each of the nodes, when connecting to another node of a different city has a different weight for each edge representing a distinct flight's capacity. However, within a city, a node representing a certain hour is connected to all other nodes in that city representing any later hour. This represents the idea that a passenger can arrive at city C at time *t* and leave city C at time *t* + x. Edges leaving city C will connect to other cities, for example city D, and connect to the nodes of the corresponding arrival time of that flight. This forces the max flow algorithm to only take multi-hop flights such that any subsequent flight arrives after the previous flight leaves. This is an example of how a few cities are depicted:



Then I ran a modified version of Ford-Fulkerson from LAX to JFK which uses dynamic programming to determine the maximum capacity that can flow through a sequence of flights from LAX to JFK, based on the times of the flights. Within a sequence of cities, I used a combination of a greedy algorithm and brute-force search to determine the best scheduling of flights leaving a city and arriving at another city that yields the maximum capacity. This is necessary since there may be multiple flights leaving a city at the same time and have the same capacity, but arrive at different cities at different times. Or, the capacities may be different, with the times being the same, or the start times different and capacities and arrival times same. For any given starting city, any combination of these 4 parameters (depart time, arrival time, destination city, capacity) can be the same or different, which is why a bottom-up dynamic programming approach is needed. I started this by considering the flights arriving at JFK and working backwards from there. I determined that this would result in the correct solution due to the weighted interval scheduling problem (job scheduling with profits). A greedy approach when selecting intervals would be optimal, specifically when there were multiple flights with equal parameter values.

Given the 10 connected cities, using the path finding function, I calculated that from LAX to JFK, there are 109601 possible routes (few of them are shown):

```
All possible paths from LAX to JFK: 109601

[['LAX', 'SFO', 'SEA', 'PHX', 'DEN', 'ATL', 'ORD', 'IAD', 'BOS', 'JFK'],
 ['LAX', 'SFO', 'SEA', 'PHX', 'DEN', 'ATL', 'ORD', 'IAD', 'JFK'],
 ['LAX', 'SFO', 'SEA', 'PHX', 'DEN', 'ATL', 'ORD', 'BOS', 'IAD', 'JFK'],
 ['LAX', 'SFO', 'SEA', 'PHX', 'DEN', 'ATL', 'ORD', 'BOS', 'JFK'],
 ['LAX', 'SFO', 'SEA', 'PHX', 'DEN', 'ATL', 'ORD', 'JFK'],
 ['LAX', 'SFO', 'SEA', 'PHX', 'DEN', 'ATL', 'IAD', 'ORD', 'BOS', 'JFK'],
 ['LAX', 'SFO', 'SEA', 'PHX', 'DEN', 'ATL', 'IAD', 'ORD', 'JFK'],
 ['LAX', 'SFO', 'SEA', 'PHX', 'DEN', 'ATL', 'IAD', 'BOS', 'ORD', 'JFK'],
 ['LAX', 'SFO', 'SEA', 'PHX', 'DEN', 'ATL', 'IAD', 'BOS', 'JFK'],
 ['LAX', 'SFO', 'SEA', 'PHX', 'DEN', 'ATL', 'IAD', 'JFK'],
 ...]
```

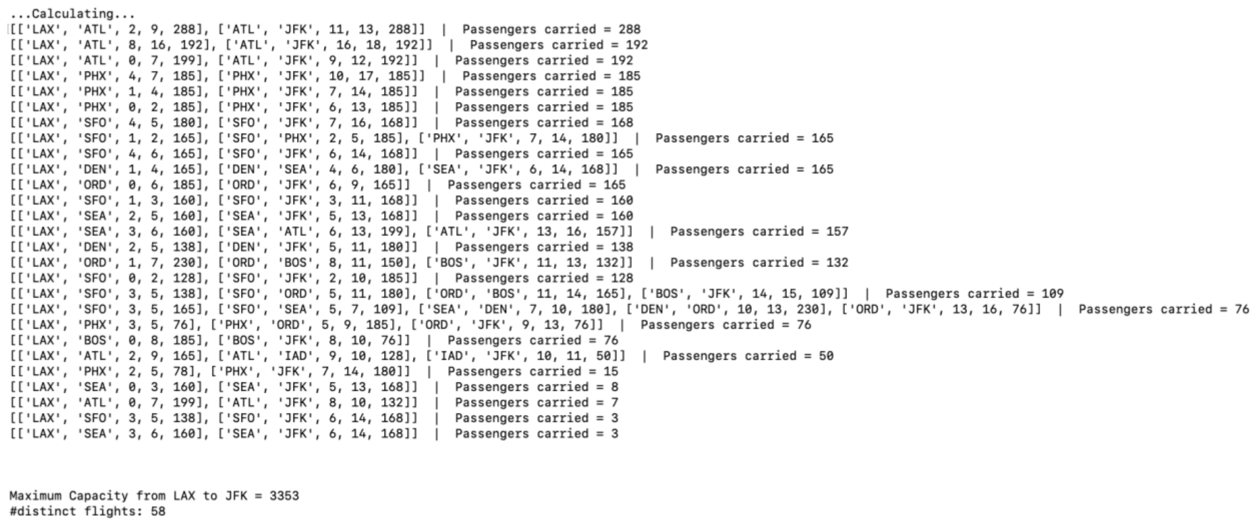
However, not all of the routes are permissible as they are constrained by the 721 available flights in flights.txt, and more importantly, the time restrictions of the flights, since any flight between 2 cities in any one of the paths listed above must depart after the previous flight arrives at that city. In order to determine the number of permissible flights that follow this constraint, I generated a tree such that a node corresponds to a flight between two cities (B -> C), the parent node represents the flight to that departure city (A -> B), and the child node represents a flight from the arrival city to another city (C -> D). By using pruning and backtracking, I was able to find all possible combinations of flights, both direct and indirect, from LAX to JFK. There are only 279 paths permissible, meaning that only for each of these 279 paths, there exists at least 1 sequence of flights between each pair of cities such that the time intervals are monotonically increasing, as required by the problem specification.

Code

In order to run the code, download “airports.py” in the same directory as “flights.txt”. Python 3.6 or above is needed. Then run the command: “python3 airports.py”. The code will take about 20 seconds to run. It will then generate the list of flight paths from LAX to JFK in addition to the number of passengers that can be carried through that flight path in order to contribute to the maximum flow. The total capacity will be printed out at the end. In the code, the first part is meant to parse the "flights.txt" file line by line and format all of the flight data in a usable and comparable way. Then a function is used to find all flights between pairs of cities for a given pair of cities. One part of the code was used to generate a graph to represent the connections between the airports, which was used for the structure of determining all of the paths. After all paths were determined, the graph structure was no longer used as all representations of flow and time intervals could be represented by the paths. Therefore, it was necessary to create several functions to determine the maximum possible capacity carried by any single path. I also coded some functions to check for compatible intervals and determine the optimal path selection. Finally, some functions were used to search for the optimal ordering and combinations of flights that would yield the maximum flow. *Some of the code for the graph generation and path enumeration was inspired by resources found online. The website and correct attributes/citation are given in the source code with the appropriate modifications.*

Results

After coding the proposed algorithm, the output yielded a total of 3353 passengers that can travel from LAX to JFK in a 24-hour time period. This involves only 27 different paths from LAX to JFK, using a total of 58 distinct flights.



```
...Calculating...
[['LAX', 'ATL', 2, 9, 288], ['ATL', 'JFK', 11, 13, 288]]  |  Passengers carried = 288
[['LAX', 'ATL', 8, 16, 192], ['ATL', 'JFK', 16, 18, 192]]  |  Passengers carried = 192
[['LAX', 'ATL', 0, 7, 199], ['ATL', 'JFK', 9, 12, 192]]  |  Passengers carried = 192
[['LAX', 'PHX', 4, 7, 185], ['PHX', 'JFK', 10, 17, 185]]  |  Passengers carried = 185
[['LAX', 'PHX', 1, 4, 185], ['PHX', 'JFK', 7, 14, 185]]  |  Passengers carried = 185
[['LAX', 'PHX', 0, 2, 185], ['PHX', 'JFK', 6, 13, 185]]  |  Passengers carried = 185
[['LAX', 'SFO', 4, 5, 180], ['SFO', 'JFK', 7, 16, 168]]  |  Passengers carried = 168
[['LAX', 'SFO', 1, 2, 165], ['SFO', 'PHX', 2, 5, 185], ['PHX', 'JFK', 7, 14, 180]]  |  Passengers carried = 165
[['LAX', 'SFO', 4, 6, 165], ['SFO', 'JFK', 6, 14, 168]]  |  Passengers carried = 165
[['LAX', 'DEN', 1, 4, 165], ['DEN', 'SEA', 4, 6, 180], ['SEA', 'JFK', 6, 14, 168]]  |  Passengers carried = 165
[['LAX', 'ORD', 0, 6, 185], ['ORD', 'JFK', 6, 9, 165]]  |  Passengers carried = 165
[['LAX', 'SFO', 1, 3, 160], ['SFO', 'JFK', 3, 11, 168]]  |  Passengers carried = 160
[['LAX', 'SEA', 2, 5, 160], ['SEA', 'JFK', 5, 13, 168]]  |  Passengers carried = 160
[['LAX', 'SEA', 3, 6, 160], ['SEA', 'ATL', 6, 13, 199], ['ATL', 'JFK', 13, 16, 157]]  |  Passengers carried = 157
[['LAX', 'DEN', 2, 5, 138], ['DEN', 'JFK', 5, 11, 180]]  |  Passengers carried = 138
[['LAX', 'ORD', 1, 7, 230], ['ORD', 'BOS', 8, 11, 150], ['BOS', 'JFK', 11, 13, 132]]  |  Passengers carried = 132
[['LAX', 'SFO', 0, 2, 128], ['SFO', 'JFK', 2, 10, 185]]  |  Passengers carried = 128
[['LAX', 'SFO', 3, 5, 138], ['SFO', 'ORD', 5, 11, 180], ['ORD', 'BOS', 11, 14, 165], ['BOS', 'JFK', 14, 15, 109]]  |  Passengers carried = 109
[['LAX', 'SFO', 3, 5, 165], ['SFO', 'SEA', 5, 7, 109], ['SEA', 'DEN', 7, 10, 180], ['DEN', 'ORD', 10, 13, 230], ['ORD', 'JFK', 13, 16, 76]]  |  Passengers carried = 76
[['LAX', 'PHX', 3, 5, 76], ['PHX', 'ORD', 5, 9, 185], ['ORD', 'JFK', 9, 13, 76]]  |  Passengers carried = 76
[['LAX', 'BOS', 0, 8, 185], ['BOS', 'JFK', 8, 10, 76]]  |  Passengers carried = 76
[['LAX', 'ATL', 2, 9, 165], ['ATL', 'IAD', 9, 10, 128], ['IAD', 'JFK', 10, 11, 50]]  |  Passengers carried = 50
[['LAX', 'PHX', 2, 5, 78], ['PHX', 'JFK', 7, 14, 180]]  |  Passengers carried = 15
[['LAX', 'SEA', 0, 3, 160], ['SEA', 'JFK', 5, 13, 168]]  |  Passengers carried = 8
[['LAX', 'ATL', 0, 7, 199], ['ATL', 'JFK', 8, 10, 132]]  |  Passengers carried = 7
[['LAX', 'SFO', 3, 5, 138], ['SFO', 'JFK', 6, 14, 168]]  |  Passengers carried = 3
[['LAX', 'SEA', 3, 6, 160], ['SEA', 'JFK', 6, 14, 168]]  |  Passengers carried = 3


Maximum Capacity from LAX to JFK = 3353
#distinct flights: 58
```