Lab 6: Graphs with "AlgoJet"

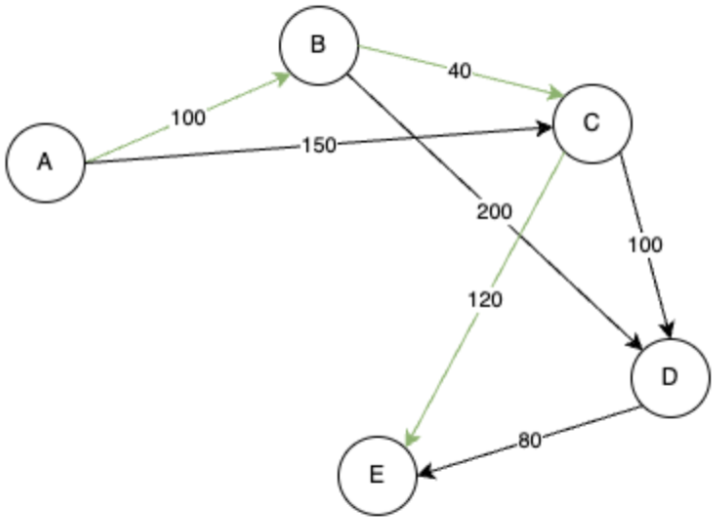
You are working for a promising new “flight hacking” startup “AlgoJet”. The startup aims to find ultra-cheap flights by combining multiple flights from various airlines and by taking portions of multi-leg flights.

A database of flights is loaded into the AlgoJet algorithm every morning with all offered flights. Flights will have: source, destination, and price.

After initializing the system, customers can enter a source and destination city and get the cheapest flight (You will only need to return the price of the flight).

Here is an example of a list of flights, the resulting flight graph, and the cheapest flight from A to E which would cost 260.

```
("A", "B", 100)
("A", "C", 150)
("B", "C", 40)
("B", "D", 200)
("C", "D", 100)
("C", "E", 120)
("D", "E", 80)
```



If no flight path reaches the destination from the source, return -1.

You must solve this problem using a handmade graph data structure and Dijkstra's algorithm.

You are given the flight class which will be used to pass the list of flights.

You will implement two methods:

- initializeFlightGraph(List of flight objects)
- getCheapestFlight(source, destination)

Deliverable Explanation

You must efficiently solve this problem in either Python or Java, using one of the attached starter code files. Several unit tests are provided to help you test your code.

Having the optimal runtime will grant you extra credit. On the other hand, you may also lose points if your code is substantially inefficient. You must also provide an explanation of the running time of your code in the provided placeholder in the starter code.

Your code will be evaluated against a set of visible and hidden test cases (the visible ones are also provided in the starter code for your convenience) to test correctness. The hidden test cases are intended to verify that you have a robust solution that considers possible edge cases and various inputs.

If you are using Java, your file's package must be set to "student".

Submission Instructions

Submit the following file on Gradescope:
AlgoJet.java to Lab6 (Java)
OR
algo_ejet.py to Lab6 (Python)

Grading Methodology

The lab is worth 100 points and is broken down into the following categories:

- Test Cases Passing (80 points)
- Big O explanation (15 points)
- Overall code performance and style (5 points)
- Having Optimal Running Time (5 points of Extra Credit)