Flight Planner

COL106 Assignment 5

November 2024

1 Background

Effective flight trip planning is essential in today's fast-paced world, especially when travelers have specific requirements. Many aim to reach their destinations quickly by minimizing connections, while others prioritize the **lowest fare** or **timely arrivals**. This assignment simulates a flight planning system that selects trips based on these diverse needs.

The task involves designing a flight planner using a set of available flights, characterized by departure and arrival cities, times, flight numbers, and fares. Given that multiple flights may exist between any two cities, the planner must efficiently choose the best itinerary based on three key objectives: minimizing the number of flights, reducing costs, or balancing both factors.

Given that multiple flights may exist between any two cities, the planner must efficiently choose the best itinerary based on three key objectives:

- **Minimizing the number of flights**
- **Reducing costs**
- **Reaching earliest**

When planning consecutive flights, a **time gap** is required between the arrival of the first flight and the departure of the second. We will model and solve these **real-world situations** to develop an effective flight trip planner.

2 Modelling

2.1 Representation

We will have data about the flights, containing details about the the start and end city, cost, time etc. For simplicity, all these will be represented using integers as follows:

- **Cities:** There are n cities, each represented by an integer $\{1, 2, \cdots n\}$.

- **Time:** The flights will be leaving and arriving at different times. We will model time as an integer in minutes. For simplicity, we assume time running from $t = 0$ to $t = \infty$
- **Flights:** There will be m flights, each with the following attributes:
  - **Flight No.:** Unique identifier of a flight, represented by an integer in $\{1, 2, \cdots m\}$
  - **Start City:** City from which the flight starts.
  - **Departure Time:** The time at which the flight departs.
  - **End City:** City where the flight ends.
  - **Arrival Time:** Time at which the flight reaches its destination.
  - **Fare:** Cost of the flight.

2.2 Constraints

- **Total flights in a City:** There are limited resources like fuel and space in one city, therefore we are guaranteed that **atmost 100 flights arrive or leave one particular city**.
- **Connecting Flights:** A person may take multiple flights to reach their destination. However, **after deboarding flight F at city A, the next connecting flight must depart from city A atleast 20 min after the arrival time of flight F**, to ensure that the person is able to fulfill all formalities of a connecting flight.

2.3 Tasks

As discussed in the background, we have to plan routes for a particular trip based on three different possible optimizations:

Basic Goal: Given parameters start_city, end_city, t1, t2, find a route (i.e., a list of flights) such that you can reach end_city from start_city by departing after or at t1 and reaching before or at t2

Since there could be multiple routes satisfying this goal, we want to display three possible routes which are optimized for different types of customers:

BFS

- **Route 1 - Fewest Flights and Earliest:** Identify a route between two given cities A and B that uses the **minimum number of flights.** If multiple options meet this criterion, choose the route that arrives at city B the **earliest**.
- **Route 2 - Cheapest Trip:** Find a route between cities A and B with the lowest total fare, regardless of the number of flights involved.
- **Route 3 - Fewest Flights and Cheapest:** Identify a route between two given cities A and B that uses the **minimum number of flights**. If multiple options meet this criterion, choose the route that is the **cheapest**.

3 Requirements

First, refer to `flights.py` to see the `Flight` class, representing flights with the parameters as discussed in 2.1.

You need to implement the `Planner` class in `planner.py`. The class contains the following methods which will be tested:

- `__init__(self, flights)`: Create an instance of planner with the given flights (list of `Flight` objects). These flights will remain same for the route finding methods that follow.
- **Route Finding methods:** The class has three methods for finding the route based on the different objectives. Each function have the same set of arguments: `(self, start_city, end_city, t1, t2)` as described in 2.3. The methods are:
  - `least_flights_ealiest_route`: Return a route satisfying optimization goal 1.
  - `cheapest_route`: Return a route satisfying optimization goal 2.
  - `least_flights_cheapest_route`: Return a route satisfying optimization goal 3.

4 Clarifications

- Do not modify the signature of any methods given in the starter code (including `Flight` class). You are free to add new methods to existing classes or create new functions/classes.
- The route returned should be a list `route` of `Flight` objects, where we first take flight `route[0]`, then `route[1]` and so on.
- To get full marks in the assignment, the **worst case time complexity** of the methods should be:
  - `__init__`: Should not take more than $O(m)$ time.
  - `least_flights_ealiest_route`: Should run in $\underline{O(m)}$ time.
  - `cheapest_route`: Should run in $O(m \log m)$ time.
  - `least_flights_cheapest_route`: Should run in $O(m \log m)$ time.

Note that the number of cities n is $O(m)$, so do not be confused about the dependency on number of cities.

5 Submission

1. You are expected to complete this assignment in the next 2 weeks. (Deadline : 13th November 11:59 PM).
2. The starter code for this assignment is available on moodle_new.
3. The submission on Moodle_New should contain the following files:
   (a) `main.py` (This file is for debugging purposes only)
   (b) `flight.py`
   (c) `planner.py`
4. Late submission policy is available on the course website. Note that you may only use 2 buffer days.