Assignment 4 is due Sunday, May 15, 23:30.

Tourist problem Suppose that a tour guide asks you to design and develop an algorithm that computes a quickest itinerary from a given train or bus station in a departure city to every other city in a given map.[1]

The tour guide gives you the following information as input:

- a set of cities;
- for every city, at most one designated train station in the city;
- for every city, at most one designated bus station in the city;
- for every two cities, whether there is a direct train transportation between their designated train stations;
- for every two cities, whether there is a direct bus transportation between their designated bus stations;
- the average travel times between two cities by a train, if it is possible to travel between their designated train stations directly by a train;
- the average travel times between two cities by a bus, if it is possible to travel between their designated bus stations directly by a bus;
- for every city that has both a train station and a bus station, the average travel time between the designated train station and the designated bus station.

The tour guide also tells you that there is no waiting at any train/bus station between travels.

Example Consider a map with three cities: Istanbul, Bursa and Eskisehir. Suppose that the tour guide gives you the following information about these cities.

- For Istanbul, the designated train station is Bostanci station, and the designated bus station is Harem station. The average transfer time between these two stations is 30 minutes.
- For Bursa, the designated bus station is the Merkez station.
- For Eskisehir, the designated train station is YHT station, and the designated bus station is Yeni station. The average transfer time between these two stations is 10 minutes.

[1] Note that the tour guide does not ask for a quickest itinerary that visits all cities in the given map. Therefore, this problem is not Traveling Salesperson Problem.

- The average travel time between Istanbul-Harem and Bursa-Merkez is 100 minutes. The average travel time between Istanbul-Bostanci and Eskisehir-YHT is 180 minutes. The average travel time between Bursa-Merkez and Eskisehir-Yeni is 120 minutes.

Suppose also that the tour guide asks for quickest itineraries starting from Istanbul-Harem. Then, your algorithm is expected to return

- a quickest itinerary from Istanbul-Harem to Bursa (i.e., Istanbul-Harem to Bursa-Merkez, with a total travel time of 100 min), and
- a quickest itinerary from Istanbul-Harem to Eskisehir (i.e., Istanbul-Harem to Istanbul-Bostanci to Eskisehir-YHT, with a total travel time of 30+180=210 min).

Your task is

1. to design an algorithm using dynamic programming to solve the tourist problem,
2. to analyze its correctness, termination, and computational efficiency (i.e., asymptotic time/space complexities),
3. to implement your algorithm in Python, and
4. to test its functionality and evaluate its performance with a diverse set of benchmark instances.

Submit Report (PDF file) Write a report (using an editor) including the following:

(a) **Recursive formulation** of the tourist problem.

(b) **Pseudocode of your algorithm** designed using dynamic programming based on the recursive formulation.

(c) **Asymptotic time and space complexity analysis** of your algorithm.

(d) Experimental evaluations of your algorithm: **plot** the results in a graph, and **discuss** the results (e.g., are they expected or surprising? why?)

Submit Python Code, and Benchmarks (ZIP file)

(a) Implement in Python your algorithm designed using dynamic programming, to solve the tourist problem.

(b) Create a benchmark suite of at least 5 instances to test the correctness of your Python program: take into account the **functional testing** methods (e.g., white box and black box testing) while constructing the instances, and test your programs with these instances.

(c) Create a benchmark suite to **test the performance** of your Python programs: construct instances of different sizes (e.g., relative to the number of levels), evaluate the performance of your program in terms of computation times, and plot the results within a graph.

Demos Demonstrate that your Python program correctly computes solutions for your benchmark instances, and for the instances that will be provided by us. Demo day will be announced.