Assignment 3 is due Sunday, May 2, 23:30.

Tourist problem Suppose that a tour guide asks you to design and develop an algorithm that computes a quickest itinerary from Istanbul-Harem bus station to every other city in Turkey (e.g., Istanbul-Harem to Eskisehir-YHT, Istanbul-Harem to Ankara-ASTI, Istanbul-Harem to Adana-Merkez), using a train or a bus.[1]

The tour guide gives you the following information as input:

- 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 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 stations between travels.

Your task is to design an algorithm using dynamic programming to solve the tourist problem, provide guarantees about its correctness, termination, and computational efficiency (i.e., analysis of its asymptotic time/space complexities), implement your algorithm in Python, and test its functionality and evaluate its performance with a diverse set of benchmark instances.

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

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

(a) Recursive formulation of the tourist problem: Identify the subproblems, observe the optimal substructure property and the overlapping computations, and then define the problem recursively respecting a topological ordering.

(b) Pseudocode of a naive recursive algorithm based on your recursive formulation, and its complexity analysis (i.e., the asymptotic time and space complexity).

(c) Pseudocode of an algorithm designed using dynamic programming (i.e., a recursive algorithm that builds solutions to your recurrence from top down with memoization, or an iterative algorithm that builds solutions to your recurrence from bottom up), and its complexity analysis (i.e., the asymptotic time and space complexity).

(d) Experimental evaluations of these two algorithms: 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 the two algorithms above, designed to solve the tourist problem.

(b) Create a benchmark suite of at least 5 instances to test the correctness of your Python programs: 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 programs in terms of computation times, and plot the results within a graph.

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