ECE406 **Algorithm Design & Analysis** **Winter 2025**

Assignment #4

due ~~Fri Mar 7 11:59pm~~ Fri Mar 14 11:59pm

See Appendix for submission instructions

Exercise 1 (Bellman-Ford algorithm) In this question we will implement the Bellman-Ford algorithm, given a graph, represented as the adjacency matrix A, and starting index s. We use the adjacency matrix representation of the graph in this question for convenience (so that we can use `numpy`), and you do not need to convert the graph to adjacency list representation. **Download the file `A4Q1`. Steps (i) and (iv) below are already completed in the script. You will fill in parts (ii) and (iii) and submit on LEARN (plus a screenshot on Crowdmark).**

(i) (Already completed) Randomly sample a directed graph with negative edges in adjacency matrix representation.

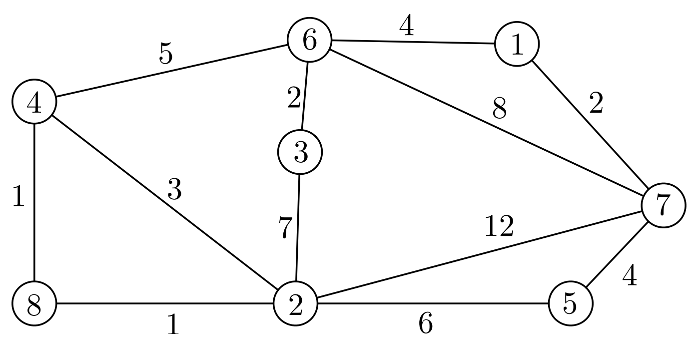
(ii) Implement the helper function `update(u, v)` that (maybe) updates the current `dist` estimates and `prev` pointer for vertex v, based on the edge (u, v).

NOTE: in the code `update` takes three arguments: a `Vertex` u, and `Vertex` v, and the adjacency matrix of the graph. Your implementation should update the properties of the `Vertex` v as needed, and does not need to return any value.

(iii) Implement `bellman_ford`, using the `update` function from part (ii). Your implementation should call `update` the appropriate number of times to find the shortest path. If a negative cycle exists in the graph, your implementation should raise a `ValueError`.

(iv) (Already completed) print out the shortest paths from s to each $v \in V$, if one exists.

Exercise 2 (Minimum spanning tree) Consider the following graph:



(i) Draw a minimum spanning tree (by hand) and give its cost.

(ii) Suppose Kruskal's algorithm is run on this graph; whenever there is a choice of edges, pick the one whose vertices sum to the lowest value. Give the order that the edges are added.

(iii) Suppose that Prim's algorithm is run on this graph; whenever there is a choice of vertices, pick the one with the lower number. Draw a table showing the cost of each vertex at each iteration of the algorithm along with list the vertices contained in H at the end of each iteration.

Exercise 3 (Spanning trees) Consider an undirected graph $G = (V, E)$ with positive lengths ℓ_e on each edge $e \in E$. Suppose you are given the minimum spanning tree T of G. Now, suppose that a new edge $e_{\text{new}} = \{u, v\}$, where $u, v \in V$ is added to the graph G to create $G' = (V, E \cup \{e_{\text{new}}\})$. Design a method to determine if T is still a minimum spanning tree of the new graph G'. If it is not, then your algorithm should return an updated MST for G'. Give the run-time of your algorithm.

Note: Your algorithm should be faster than simply recomputing the entire spanning tree by calling Prim's or Kruskal's algorithm on G'.

Exercise 4 (Greedy algorithms for scheduling) A server has n customers waiting to be served, labeled from 1 to n. The service time for customer i is t_i minutes, and is known in advance. Thus, if the customers are served in order of their labels, then customer i will wait $\sum_{j=1}^{i} t_j$ minutes until he/she has been served.

We wish to minimize the total time that customers are waiting

$$T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i).$$

Let $l(j)$ be the label of the jth customer to be served (for example, if $l(1) = 7$, then customer 7 is the first customer to be served). Then, the total time T can be written as

$$T = \sum_{i=1}^{n} \sum_{j=1}^{i} t_{l(j)}.$$

(i) Give the pseudocode of an algorithm that computes the optimal order in which to process the customers. (*Hint*: think greedy!)

(ii) Show that your algorithm is correct. (*Hint:* Suppose we swap two customers in the order, say $l(j)$ and $l(k)$. How does total time T change?)

(iii) Characterize the run-time of your algorithm.

Appendix

How do I submit this assignment? To submit the assignment, please complete the following steps.

(i) Upload your typed or handwritten work for *each* question using the ECE406 **Crowdmark** site. This includes your answers to the programming questions, where you should submit screenshots of the Python functions you have implemented.

(ii) Upload your source code for each programming question to the **A4 Dropbox submission folder** in LEARN. The starter code for each question specifies the filename convention you should follow.