Note: Your TA probably will not cover all the problems. This is totally fine. The discussion worksheets are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 Dijkstra's Algorithm Fails on Negative Edges

Draw a graph with five vertices or fewer, and indicate the source from which you would start Dijkstra's algorithm.

(a) Draw a graph with no negative cycles for which Dijkstra's algorithm produces the wrong answer.

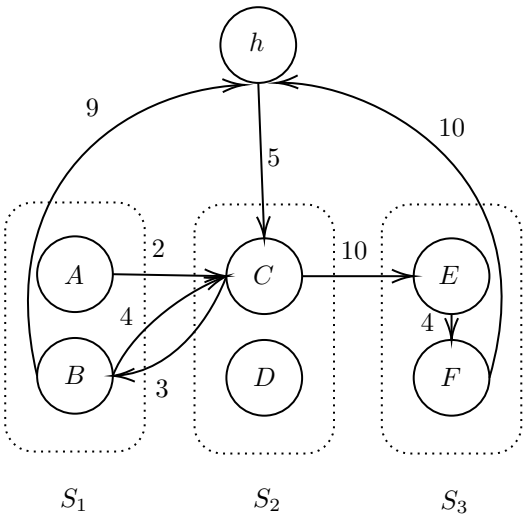
(b) Draw a graph with at least two negative weight edges for which Dijkstra's algorithm produces the correct answer.

2 Running Errands

You need to run a set of k errands in Berkeley. Berkeley is represented as a directed weighted graph G, where each vertex v is a location in Berkeley, and there is an edge (u, v) with weight w_{uv} if it takes w_{uv} minutes to go from u to v. The errands must be completed in order, and we'll assume the ith errand can be completed immediately upon visiting any vertex in the set S_i (for example, if you need to buy snacks, you could do it at any grocery store). Your home in Berkeley is the vertex h.

Given G, h, and all $(S_i)_{i=1}^k$ as input, give an efficient algorithm that computes the least amount of time (in minutes) required to complete all the errands starting at h. That is, find the shortest path in G that starts at h and passes through a vertex in S_1, then a vertex in S_2, then in S_3, etc.

For instance, in the graph below, the shortest such path is $h \to C \to B \to C \to E$ and the time needed is $5 + 3 + 4 + 10 = 22$.



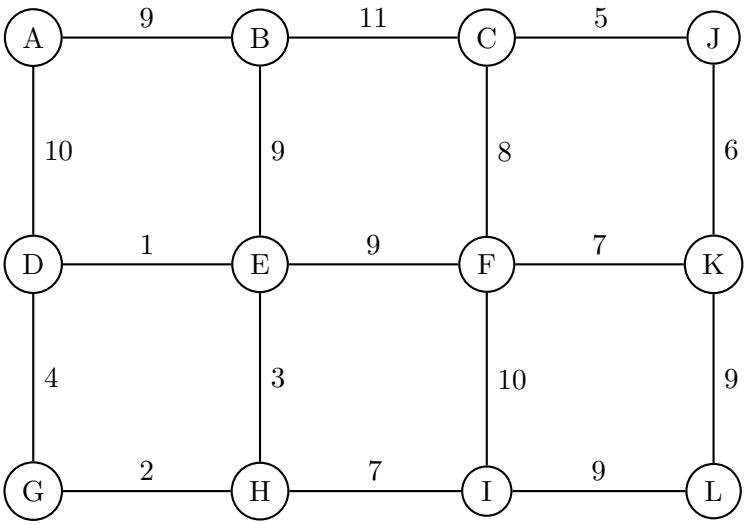
Hint: try creating copies of the graph G to help "keep track of" the errands you've completed so far.

3 Shortest Path Between Sets

Given a undirected weighted graph G with non-negative edge weights $w(\cdot)$, and let $d(s,t)$ be the shortest path length from s to t. Give an efficient algorithm that takes as input two subsets of vertices S and T and outputs $\min_{s\in S, t\in T} d(s,t)$, i.e. the shortest path from any vertex in S to any vertex in T. Your algorithm should take $O(|E|\log|V|)$ time.

Hint: create an extra dummy node so that you can find all relevant distances by running Dijkstra's from there.

4 MST Tutorial (Recommended to do after discussion)



(a) List the first **six** edges added by Prim's algorithm in the order in which they are added. Assume that Prim's algorithm starts at vertex A and breaks ties lexicographically.

(b) List the first **seven** edges added by Kruskal's algorithm in the order in which they are added. You may break ties in any way.

(c) Prim's algorithm is very similar to Dijkstra's in that a vertex is processed at each step which minimizes some cost function. These algorithms also produce similar outputs: the union of all shortest paths produced by a run of Dijkstra's algorithm forms a tree. However, the trees they produce aren't optimizing for the same thing. To see this, give an example of a graph for which different trees are produced by running Prim's algorithm and Dijkstra's algorithm. In other words, give a graph where there is a shortest path from a start vertex A using at least one edge that doesn't appear in any MST.

5 MST Practice (Recommended to do after discussion)

Let $G = (V, E)$ be an undirected, connected graph.

(a) Prove that there is a unique MST if all edge weights are distinct.

(b) True or False? If G has more than $|V| - 1$ edges, and there is a unique heaviest edge, then this edge cannot be part of a MST.

(c) True or False? If the lightest edge in G is unique, then it must be a part of every MST.