Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They 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 Dijkstra's Tiebreaking

We are given a directed graph G with positive weights on its edges. We wish to find a shortest path from s to t, and, among all shortest paths, we want the one in which the longest edge is as short as possible. How would you modify Dijkstra's algorithm to this end? Just a description of your modification is needed.

Note: if there are multiple shortest paths where the longest edge is as short as possible, outputting any of them is fine.

3 Waypoint

You are given a strongly connected directed graph $G = (V, E)$ with positive edge weights, and there is a special node $v_0 \in V$. Give an efficient algorithm that computes, for all node pairs s, t, the length of the shortest path from s to t that passes through v_0. Your algorithm should take $O(|V|^2 + |E| \log |V|)$ time.

Hint: you should only need 2 calls to Dijkstra's.

4 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.

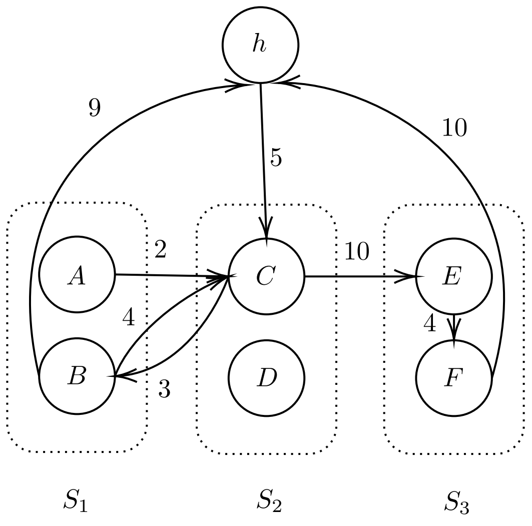
Hint: create an extra dummy node so that you can find all relevant distances by running Dijkstra's from there.

5 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.

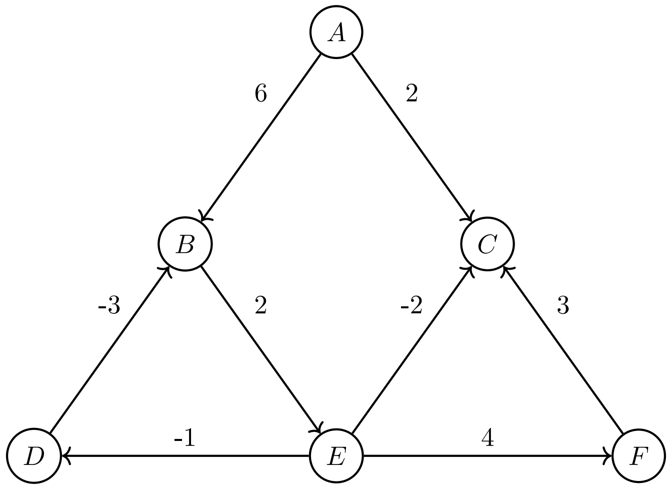
For instance, in the graph below, the shortest such path is $h \to C \to B \to C \to E$ and the time needed is $5 + 3 + 4 + 10 = 22$.



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

6 Bellman-Ford

Consider the graph below.



(a) When running the Bellman-Ford algorithm, at most how many times do we go through all the edges to determine the single source shortest paths? How can we tell if there is a negative cycle?

(b) Suppose we ran Bellman-Ford on the graph above and we process edges in the following order:

$$(A, B), (A, C), (B, E), (D, B), (E, C), (E, D), (E, F), (F, C).$$

Fill out the table below with the distances of each node from A after each iteration of Bellman-Ford. Is there a negative cycle in the graph above?

Iteration	**A**	**B**	**C**	**D**	**E**	**F**
Start	0	∞	∞	∞	∞	∞
1						
2						
3						
4						
5						
6						

7 Restaurant Orders

Andrew is the sole chef at CS 170 Diner, and today he is handling a flurry of orders from n customers. Thankfully, each customer only ordered 1 dish, and he knows that it takes c_i minutes to cook the meal for customer i ($1 \leq i \leq n$). However, Andrew is very bad at multitasking and can only cook one meal at a time. To best satisfy the hungry customers, Andrew is trying to figure out the best way to process all the orders to minimize the total wait time.

More formally, let v_i be the time at which customer i gets their food. Please help Andrew determine an efficient algorithm that finds the minimum $\sum_{i=1}^{n} v_i$ over all ways to fulfill the n orders. Please provide an efficient algorithm and runtime analysis; a proof of correctness is not required.

8 Longest Huffman Tree

Under a Huffman encoding of n symbols with frequencies $f_1, f_2, \ldots, f_n$, what is the longest a codeword could possibly be? Give an example set of frequencies that would produce this case, and argue that it is the longest possible.