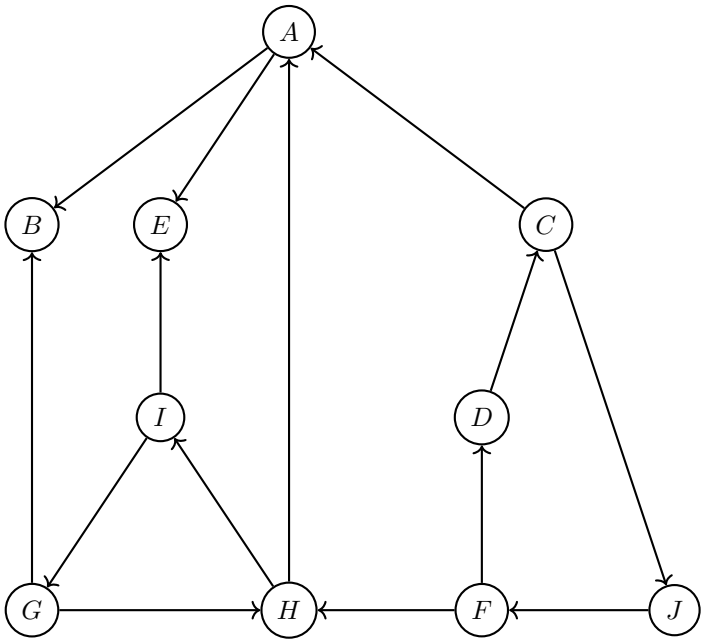
Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings,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

1 Graph Traversal



(a) Recall that given a DFS tree, we can classify edges into one of four types:

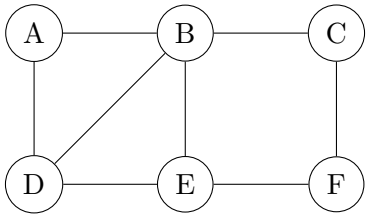
- Tree edges are edges (u, v) in the DFS tree,
- Back edges are edges (u, v) not in the DFS tree where v is the ancestor of u in the DFS tree
- Forward edges are edges (u, v) not in the DFS tree where u is the ancestor of v in the DFS tree
- Cross edges are edges (u, v) not in the DFS tree where u is not the ancestor of v, nor is v the ancestor of u.

For the directed graph above, perform DFS starting from vertex A, breaking ties alphabetically. As you go, label each node with its pre- and post-number, and mark each edge as **T**ree, **B**ack, **F**orward or **C**ross.

(b) What are the strongly connected components of the above graph?

(c) Draw the DAG of the strongly connected components of the graph.

2 Breadth-First Search



(a) Run breadth-first search on the above graph, breaking ties in alphabetical order (so the search starts from node A). At each step, state which node is processed and the resulting state of the queue. Draw the resulting BFS tree.

(b) During the execution of BFS in an unweighted undirected graph G starting from a node A, two vertices X and Y are both present in the queue at some point in time. What are the possible values of $\text{dist}[A, X] - \text{dist}[A, Y]$?

3 Dijkstra's Algorithm Fails on Negative Edges

Draw a graph with five vertices or fewer, and indicate the source where Dijkstra's algorithm will be started from.

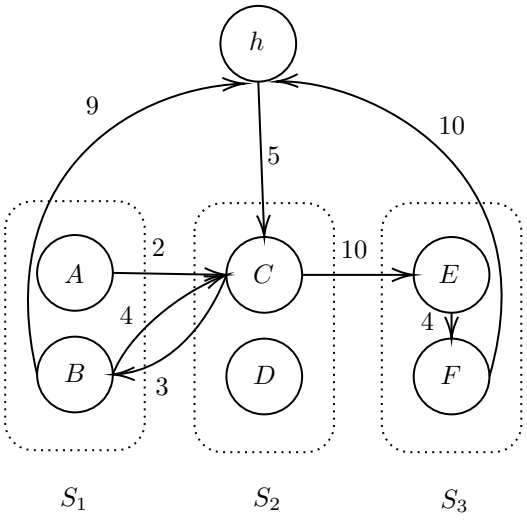
(a) Draw a graph with no negative cycles for which Dijkstra's algorithm produces the wrong shortest path length for some vertex in the graph.

(b) Draw a graph with at least two negative weight edges for which Dijkstra's algorithm produces the correct shortest path lengths for all vertices in the graph.

4 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, 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 $5 + 3 + 4 + 10 = 22$.



5 Waypoint

Throughout this question, assume that you have access to the Dijkstra's algorithm which on input a graph $G = (V, E)$ (directed or undirected) with positive edge weights l and a vertex $v \in V$ finds the shortest path length from v to all vertices in the same connected component as v. The runtime of this algorithm is $O((|V| + |E|) \log |V|)$.

(a) You are given a connected undirected 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 (the algorithm can call the Dijkstra's as a subroutine). Your algorithm should take $O(|V|^2 + |E| \log |V|)$ time (including the runtime of the Dijkstra's algorithm).

(b) Now you are given a strongly connected directed graph $G = (V, E)$ with positive edge weights, and a special node $v_0 \in V$. Again 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. What is the runtime of your new algorithm?

6 Inequalities

Given a list of n variables $x_1, \ldots, x_n$ and m inequalities of the form $x_i < x_j$ or $x_i \leq x_j$ for some $i, j \in [n]$, you would like to find values for the variables such that all inequalities are satisfied, or determine that not all inequalities can be satisfied simultaneously.

(a) Design an efficient algorithm for the case that all inequalities are inequalities are strict (i.e. of the form $x_i < x_j$).

Give a concise description of the algorithm (proof of correctness or runtime analysis are not needed).

(b) Design an efficient algorithm that solves the above problem in general (when some inequalities are of the form $x_i < x_j$, and some are of the form $x_i \leq x_j$).

Give a concise description of the algorithm (proof of correctness or runtime analysis are not needed).