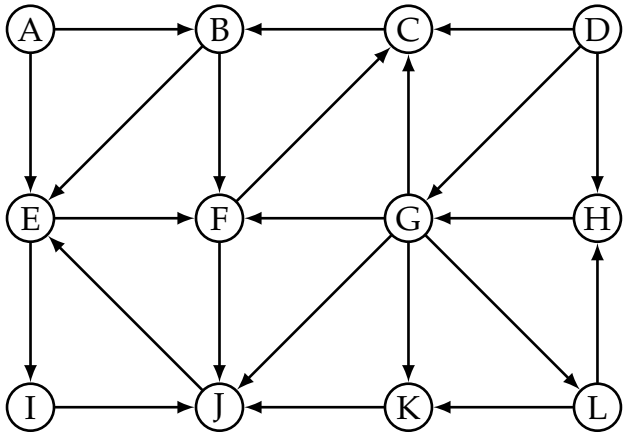
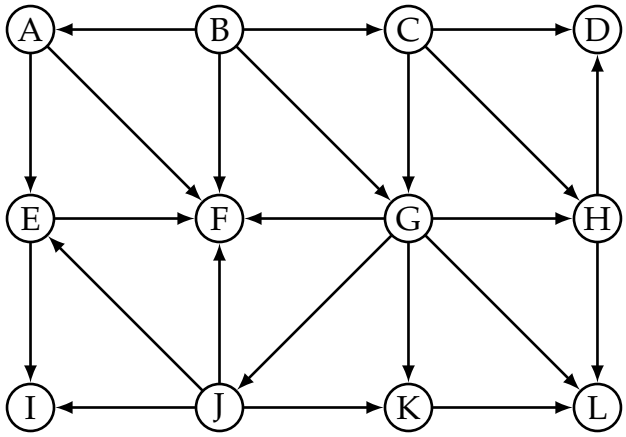
Due at 11:59PM, Tuesday, May 20th on Canvas. Submissions must be ***typed*** (except for questions labeled with a "📷" icon) and in PDF format. Show your work for full credit!

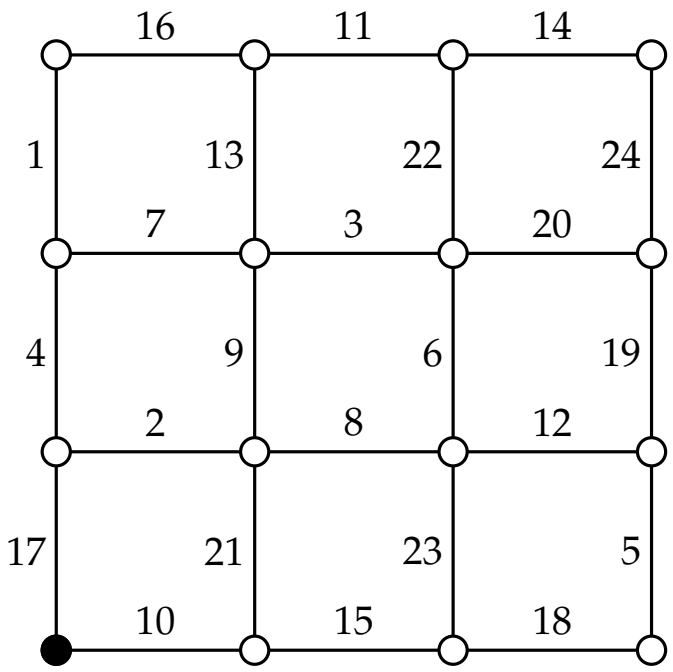
1. (📷) Calculate a depth-first search of the following two directed graphs using the "alphabetical order" convention for vertices and edges. For both graphs, draw the resulting DFS tree.





For the first graph, indicate the type (***back***, ***forward***, or ***cross***) of the remaining edges. For the second graph, which is acyclic, write down the ***pre- and post-order traversals*** of the DFS. Use that information to calculate a ***topological sort*** of the graph. (Note: because of the alphabetical ordering, there is only one acceptable DFS tree for each graph, and only one acceptable topological sort)

2. (📷) Run both ***Kruskal*** and ***Prim's algorithms*** on the following weighted graph:



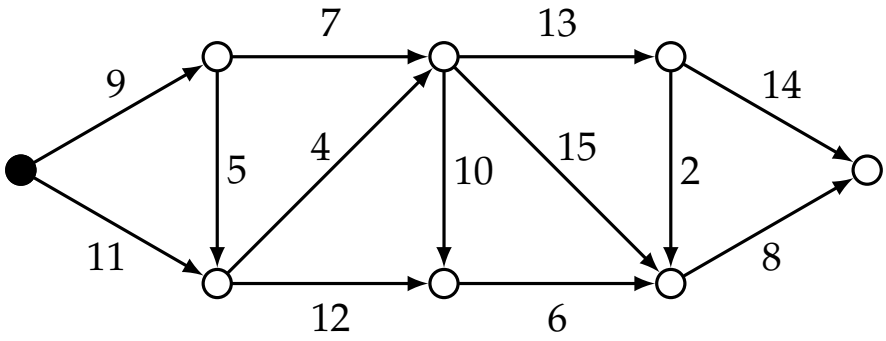
For Prim's algorithm, start at the bottom-left vertex marked in black. To show your work, ***list the weights of the edges*** in the order they are added to the MST by each algorithm. What's the ***weight*** of your minimum spanning tree?

3. Consider all weighted, undirected, connected graphs where the ***edge weights are distinct*** (i.e., there is a unique MST) and every vertex has ***degree at least 2***.

   (a) Does the minimum spanning tree of ***every*** such graph always contain the ***lightest*** edge? Why or why not?

   (b) Is there such a graph whose minimum spanning tree contains the ***heaviest*** edge? Why or why not?

4. (📷) Run Dijkstra's algorithm on the following graph, starting from the left-most vertex marked in black:



To show your work, draw the vertex labels and tree edges

(a) at the beginning of the algorithm,

(b) after processing (i.e. relaxing the outgoing edges of) four vertices, and

(c) at the end of the algorithm.