This homework covers algorithms for weighted graphs (shortest path, minimum spanning tree) and graph modeling. It is due in class Wednesday, March 26.

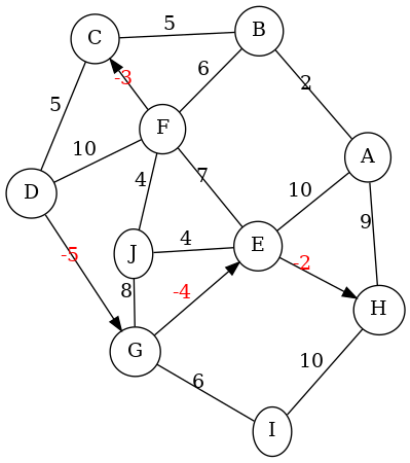
Write your solutions carefully — your work should be neat, readable, and organized.

See the Policies page on the course website for information about revise-and-resubmit, late work, and academic integrity as it applies to homework.

1. Carry out the Bellman-Ford algorithm on the graph shown below. (Negative weights are shown in red to make them easier to identify.) For clarity, when there are edges (u, v) and (v, u) with the same weight, a single undirected edge is shown.

   Go through the edges in alphabetical order. (Be sure to consider both directions — such as AB and BA — for the undirected edges.)

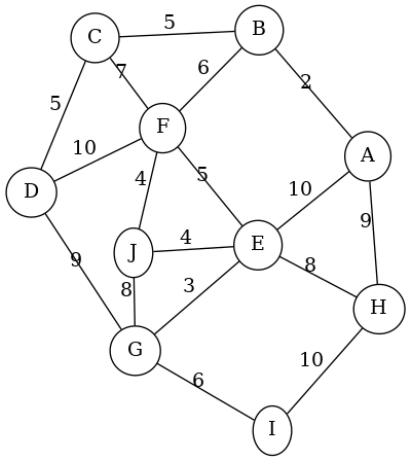
   Show your work tracing the algorithm by giving a table of the `dist` labels as they are updated in each iteration of the `repeat` loop. (Have a column of the table for each vertex and a row for each iteration.)



2. Carry out Kruskal's algorithm on the graph shown below.

   Break ties by alphabetical order — take an edge's name as its endpoint vertices in alphabetical order, then compare the edge names. For example, the two weight-5 edges incident on vertex C in the graph shown would be named BC and CD, and BC should be chosen first.

   Show your work tracing the algorithm by listing the MST edges in the order they are added to the MST.



3. Repeat #2 using Prim's algorithm instead.

For the following problems, explain how to solve the problem by modeling it as a graph problem. Address the following things in your solutions:

(a) What do the vertices of the graph represent? The edges? What is the solution to the problem in terms of the graph?

(b) Is the graph undirected or directed? Weighted or unweighted? (If weighted, what do the weights represent?) Simple or not simple? Sparse or dense? Cyclic or acyclic? Embedded or topological? Implicit or explicit? Labeled or unlabelled? Provide a brief explanation with each answer.

(c) Give an efficient algorithm for solving the problem in terms of a well-known graph algorithm (i.e. one discussed in class and/or mentioned in ADM chapter 18). Design graphs, not algorithms! Imagine that you have a library of implementations of well-known graph algorithms that you can use — design your graphs so that you can use them as-is instead of having to modify an algorithm or implement your own.

4. ADM 7-13, page 238.

5. ADM 7-15, page 238.

6. ADM 7-30, page 240.