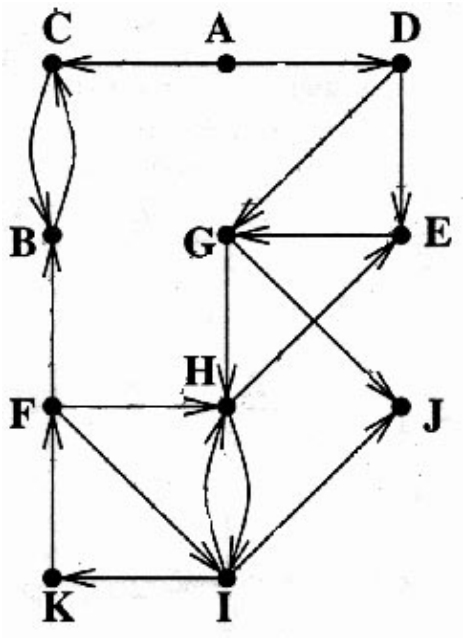
CS 170, Spring/1998
Midterm #2
Professor J. W. Demmel

Problem #1 (13 points)

What are the strongly connected components of the directed graph shown below? (Just circle them.) Note that edges that appear to cross in the figure below are not connected in any way.



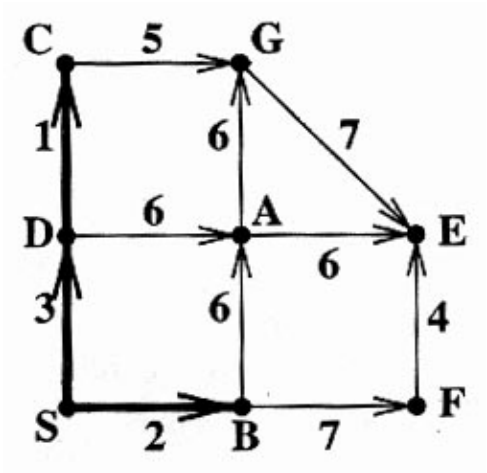
- What is the stronly connected component that will be discovered first by our algorithm (first do DFS on the reverse graph, then on the graph)? Which one will be discovered last? *As always, when DFS has a choice, it visits the alphabetically smallest vertex first.*

- Label the vertices with their postorder numbers computed by DFS, ie. the order in which they are popped from the stack.

Problem #2 (13 points)

We are running one of these three algorithms on the graph below, where the algorithm has already "processed" the bold-face edges. (Ignore the directions on the edges for Prim's and Kruskal's algorithms.)

- Prim's for the minimum spanning tree, starting from S.
- Kruskal's for the minimum spanning tree.
- Dijkstra's for shortest paths from S.



Which edge would be added next in Prim's algorithm?

Which edge would be added next in Kruskal's algorithm?

Which vertex would be marked next in Dijkstra's algorithm, i.e. deleted from the top of the heap? Which final edge would Dijkstra's algorithm choose as part of the shortest path to this vertex (i.e. which edge connects to this vertex as part of the shortest path from S)?

Problem #3 (20 points)

Give an efficient algorithm to find the *maximum spanning tree* in an undirected connected graph, i.e. a spanning tree the sum of whose edge weights is as *large* as possible. Your solution should consist of a modification of an algorithm you already know. Analyze the complexity of your algorithm, and justify its correctness (if you use an algorithm presented in class, you do not need to rederive its complexity analysis or correctness proof.)

Problem #4 (20 points)

You are given a connected undirected graph.

- show that it is possible to direct the edges so that every vetex has in-degree (the number of edges pointing to it) at least 1, if and only if the graph contains a cycle.
- Give an efficient algorithm for directing the edges in the manner just described. The algorithm should return FALSE if it is impossible to do so. Analyze the complexity of your algorithm, and justify its correctness (if you use an algorithm presented in class, you do not need to rederive its complexity analysis or correctness proof.)

Problem #5 (9 points)

True or false? No explanation required, except for partial credit. Each correct answer is worth 1 point, but 1 point will be *subtracted* for each wrong answer, so answer only if you are reasonably certain.

a. An undirected graph in which there is a unique path between every pair of vertices is a tree

b. The sum of the in-degree (the number of edges pointing to a vertex) of all vertices is equal to the sum of the out-degree (the number of edges pointing away from a vertex) of all vertices in any directed graph.

c. A graph G(V,E) with |E| = |V| - 1 is a tree.

d. The sum of the degrees of the vertices of a tree with n vertices is 2n - 2.

e. The shortest path between two vertices is unique if all edge weights are distinct

f. $\log(\log^* n) = O(\log^*(\log n))$

g. The path returned by Bellman-Ford at iteration n - 2 can be the shortest path.

h. Let G(V,E) be an undirected graph with k connected components. Then |E| >= |V| - k.