COMP 271 Design and Analysis of Algorithms
2003 Spring Semester
Question Bank 2

There is some overlap between these question banks and the tutorials. Solving these questions (and those in the tutorials) will give you good practice for the midterm. Some of these questions (or similar ones) will definitely appear on your exam! Note that you do not have to submit answers to these questions for grading. Your TAs will discuss answers to selected questions in the tutorials.

1 Basic Graph Problems

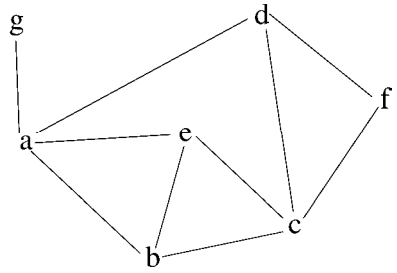
1. Given an undirected graph $G = (V, E)$, recall that the *complement*, $\overline{G}$, is a graph (V, E') such that for all $u \neq v$, $\{u, v\} \in E'$ if and only if $\{u, v\} \notin E$. Prove that either G or $\overline{G}$ is connected.

2. When a vertex and its incident edges are removed from a tree, a collection of subtrees remains. Design a linear-time algorithm that, given a tree with n vertices, finds a vertex whose removal leaves no subtree with more than $n/2$ vertices.

2 Breadth-First Search and Depth-First Search

1. The adjacency list representation of a graph G, which has 7 vertices and 10 edges, is:

$$\begin{array}{ll} a :\rightarrow d, e, b, g & b :\rightarrow e, c, a \\ c :\rightarrow f, e, b, d & d :\rightarrow c, a, f \\ e :\rightarrow a, c, b & f :\rightarrow d, c \\ g :\rightarrow a & \end{array}$$



   (a) Show the tree produced by depth-first search when it is run on the graph G, using vertex a as the source. You *must* use the adjacency list representation

given above. (Recall that the DFS tree can depend on the order of vertices in the adjacency lists; for this problem you are required to use the adjacency lists as given above.)

(b) In the DFS tree of item (a), show the edges of the graph G which are not present in the DFS tree by *dashed* lines.

2. Give an example of a directed graph G in which a vertex u of G ends up in a depth-first tree containing only u, even though u has both incoming and outgoing edges. Your example graph should have no self-loops.

3. Show that depth-first search of an undirected graph G can be used to identify the connected components of G. More precisely, show how to modify depth-first search so that each vertex v is assigned an integer label $cc[v]$ between 1 and k, where k is the number of connected components of G, such that $cc[u] = cc[v]$ if and only if u and v are in the same connected component.

4. Modify the pseudo-code for depth-first search so that it prints out every edge in the directed graph G, together with its type. Show what modifications, if any, must be made if G is undirected.

5. Prove that if G is an undirected graph, then each of its edges is either in the depth-first search tree or is a back edge.

6. Can there be any cross edges with respect to a BFS tree of an undirected graph? If yes, give an example. If not, give a proof of this fact.

7. Let $G = (V, E)$ be an undirected graph. Design an algorithm that determines whether or not G contains a cycle. Your algorithm must run in time $O(|V|)$ independent of the value of $|E|$. Give the pseudo-code for your algorithm. You should also explain how it works, prove its correctness, and derive its running time.

3 Minimum Spanning Trees

1. Let G be a connected undirected graph with weights on the edges. Assume that all the edge weights are distinct. Prove that the edge with the smallest weight must be included in any minimum spanning tree of G. *You have to prove this from first principles, i.e., you are not allowed to use the Lemmas proven in class or assume the correctness of Kuskal's or Prim's algorithm.*

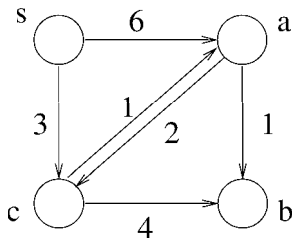
2. (a) Let $G = (V, E)$ be a connected undirected graph with weights on the edges. Assume that all the edge weights are distinct. Let C be any cycle in graph G and let e be an edge having maximum weight among the edges of this cycle C. Prove that there is a minimum spanning tree of G that does not contain the edge e.

(b) How would you use the claim in part (a) to devise an algorithm to construct the MST? Your algorithm should run in time $O(|V| \cdot |E|)$.

3. Let $G = (V, E)$ be a connected undirected graph in which all edges have weight either 1 or 2. Give an $O(|V| + |E|)$ algorithm to compute a minimum spanning tree of G. Justify the running time of your algorithm. (*Note:* You may either present a new algorithm or just show how to modify an algorithm taught in class.)

4 Shortest Paths

1. Execute Dijkstra's algorithm on the following digraph, where s is the source vertex.



   You need to indicate only the following:

   (a) the order in which the vertices are removed from the priority queue?

   (b) the final distance values $d[]$ for each vertex?

   (c) the different distance values $d[]$ assigned to vertex b, as the algorithm executes.

2. Give a simple example of a directed graph with negative-weight edges for which Dijkstra's algorithm produces incorrect answers. Why does the correctness proof of Dijkstra's algorithm not go through when negative-weight edges are allowed?

3. Suppose that we are given a cable network of n sites connected by duplex communication channels. Unfortunately, the communication channels are not perfect. Each channel may fail with certain probability (also given in the input). The probabilities of failure may differ for different channels and they are mutually independent events. One of the n sites is the central station and your problem is to compute the most reliable paths from the central station to all other sites (i.e., the paths of lowest failure probabilities from the central station to all other sites). Design an algorithm for solving this problem, justify its correctness, and analyze its time and space complexities. Let $f(u, v)$ denote the failure probability for the channel between sites u and v.