CS/Math 240: Introduction to Discrete Mathematics	7/5/2007
Homework 3	
Instructor: Jeff Kinne	TA: Mike Kowalczyk

This homework is due at the beginning of class on Thursday July 12, 2007. Mike will hold a review session at 12:45-1:45 on July 12 to discuss the solutions to these problems (that is after you have handed in the homework).

Note: All logarithms are base 2 unless otherwise specified.

Problem 1

In this problem, we look at conditions for determining whether an undirected graph is connected or not. Remember that a graph is connected if for any pair of vertices v_1 and v_2, there is some path in the graph from v_1 to v_2.

Part a

Show that if G is an undirected graph with n vertices and G is connected, then it must be the case that G has at least $n-1$ edges.
Hint: Try using induction on the number of vertices in the graph.

Part b

Is the converse of the above true? That is, if an undirected graph G with n vertices has at least $n-1$ edges, can we conclude that G is connected? If so, prove it. If not, give a counter example.

Problem 2

This problem deals with Dijkstra's algorithm for determining shortest paths in a graph.

Part a

Show that for unweighted graphs (graphs with all edge weights equal to 1), Dijkstra's algorithm is equivalent to breadth first search. Remember that Dijkstra's algorithm computes the shortest path from vertex a to all other vertices in the graph. So, show that if we assume that Dijkstra's algorithm and BFS break ties in the same order, then the order of vertices computed by Dijkstra's algorithm is the same as the order of vertices considered by breadth first search.

Part b

Show how Dijkstra's algorithm runs on the directed graph in Figure 1 when trying to find a shortest path from vertex a to vertex z. That is, for the i^{th} iteration of the algorithm, give the value of the set S containing the labels of the i closest vertices to a along with their shortest path values. So, before the 1st iteration, we have $S = (a, 0)$. If there are ties, assume that the vertex with label coming first in the alphabet is added to S first.

If you are unclear about how Dijkstra's algorithm should operate on a directed graph, please ask the TA or instructor.

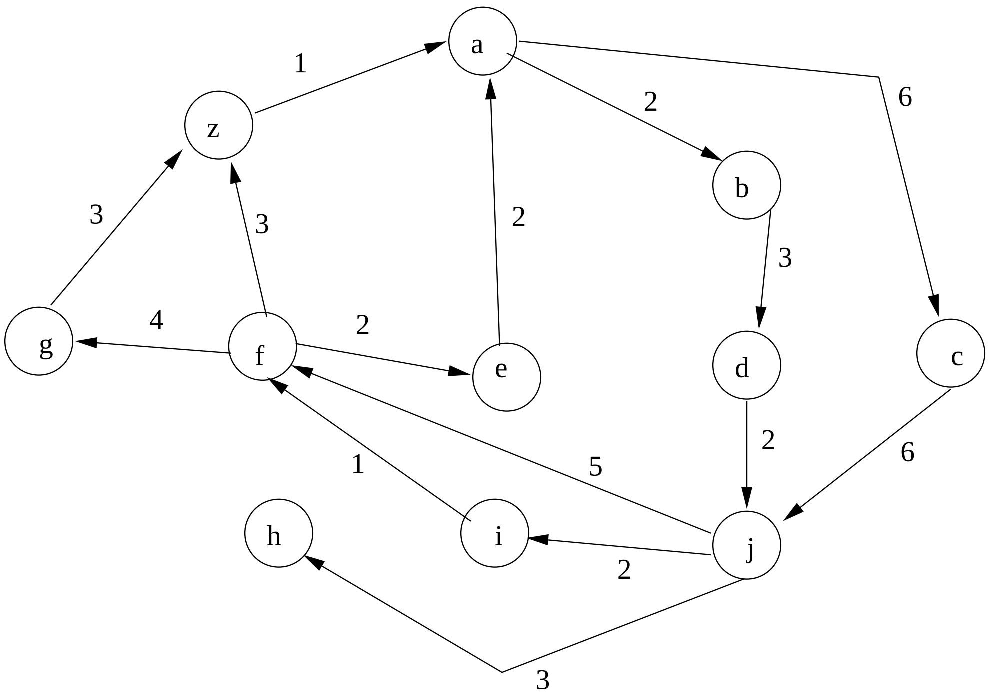

Figure 1: Graph for Problem 3, Part b.

Problem 3

This problem gives you some practice solving recurrence relations.

Part a

Suppose we have a recursive algorithm that solves a problem on inputs of size n by doing a constant amount of work and making 2 recursive calls to itself on inputs of size $n-1$. So, suppose the running time of the algorithm has the following form:

$$T(n) = 2 \cdot T(n-1) + c,$$

$$T(1) = 1,$$

where c is some constant. Use a recursion tree argument to give a big-O estimate for $T(n)$.

Notice that the recurrence relation for $T(n)$ is not suitable for using the Master Theorem.

Part b

Recall the binary search algorithm we gave in class. The running time for binary search had the form $T(n) = T(n/2) + c$ for some constant c. By solving that recurrence relation we found that $T(n) = \Theta(\log n)$.

Suppose instead that we required a logarithmic amount of work in addition to the recursive call. That is, suppose we have an algorithm where the running time $R(n)$ has the form:

$$R(n) = R(n/2) + \log n,$$

$$R(1) = 1.$$

Use a recursion tree argument to give a big-O estimate for $R(n)$. You may assume that n is a power of 2, say $n = 2^k$.

Notice that once again the recurrence relation is not in a form that is suitable for the Master Theorem.

Problem 4

In this problem, you prove some simple facts about partially ordered sets.

Part a

Let $(S, \preceq)$ be a partially ordered set, with S finite. Show that for any $x \in S$, $(S - \{x\}, \preceq)$ is a partially ordered set.

Part b

Let $(S, \preceq)$ be a partially ordered set, with S finite. Show that S has at least one maximal element.