ICS 23 / CSE 23 — Data Structures — Winter 2005 — Goodrich — Final

Name:

ID:

Percentage of lectures you believe you attended: _____ (please be truthful, as your answer will not be factored into your grade; it will only be used for comparison purposes to the attendance rosters)

1:

2:

3:

4:

5:

6:

total:

1. (50 points). Short answers.

(a) Define what it means for a binary tree to be a heap.

(b) What are the worst case running times for insertion-sort, heap-sort, and quick-sort?

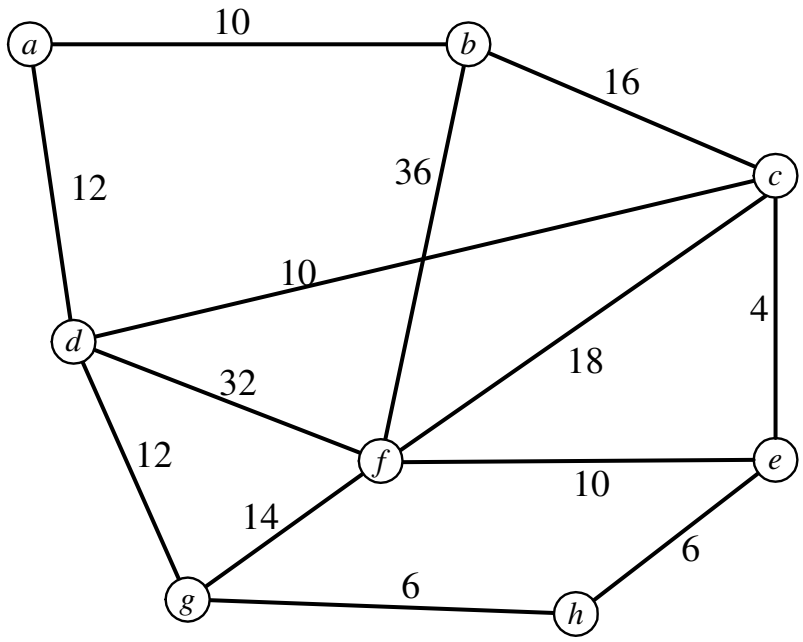
(c) Use the big-Oh notation to characterize the asymptotic amount of space that is used by a graph G with n vertices and $O(n \log n)$ edges, if we use an adjacency list to represent G.

(d) How many edges are in an undirected simple graph G that has n vertices, no cycles, and two connected components?

(e) Define what it means to say that a sorting algorithm is stable.

2. (50 points). Describe in one page or less a sorting algorithm that runs in $O(n \log n)$ time in the **worst case**.

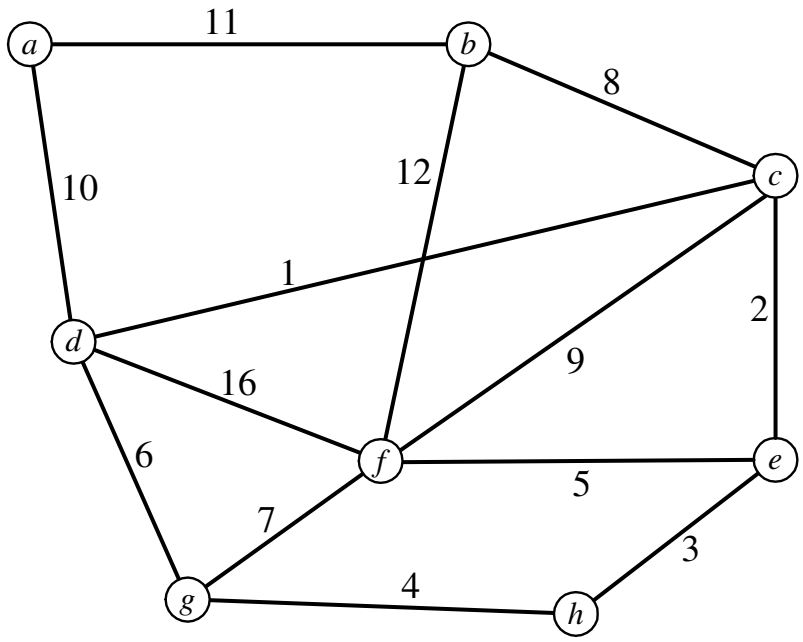
3. (50 points). Consider the following graph:



(a) Draw over and thicken all the edges that belong to the shortest path tree T rooted at a.

(b) Write down all the values the D label at f takes on during an execution of Dijkstra's shortest path algorithm, starting at a. Hint: the first value is $+\infty$.

4. (50 points). Consider the following graph:



(a) Draw over and thicken all the edges that belong to a minimum spanning tree T of this graph.

(b) Number the thickened edges in the order in which they would be added to T by Kruskal's minimum spanning tree algorithm, starting at a. That is, number the first edge added as 1, the second as 2, and so on.

5. (50 points). Suppose T is an arithmetic expression tree with n nodes, where each internal node stores an binary arithmetic operation, **op** (like $+$, $-$, $*$, and $/$), and each external node stores a number. Give a recursive algorithm for evaluating T. What is the running time of your algorithm?

6. (50 points). An undirected graph $G = (V, E)$ is a *social network* if V represents a set of people and there is an edge (v, w) in E if and only if v and w know each other. The *degree of separation* between two vertices u and w is the minimum number of edges in a path from u to v in G, taken over all such paths*. Suppose there are two vertices s and t in G who would make a perfect couple, but they don't know each other. Define a *match-maker* to be a vertex u in G that has the same degree of separation to s as it does to t. Given a social network G with n vertices and m edges, and a specific vertex u, describe an efficient algorithm for determining if u is a match-maker in G for s and t. What is the running time of your algorithm in terms of n and m? (You may use as a subroutine any algorithm discussed in class.)

*It is widely believed that there are at most six degrees of separation between any American and the actor Kevin Bacon.