ICS 161 — Algorithms — Spring 2002 — Goodrich — Final Exam

Name:

ID:

Percentage of lectures you attended: _____ (please be truthful; your answer will not be factored into your grade)

1:

2:

3:

4:

5:

6:

7:

8:

9:

10:

total:

1. (30 points). Definitions.

(a) Define "big-Oh."

(b) Define "total order."

(c) What is the "height-balance property" for AVL-trees?

2. (30 points). Short answers.

(a) If a priority queue is implemented using an unordered sequence and then used to sort n elements, what is the running time of this (selection-sort) algorithm?

(b) How much space does an adjacency matrix require in order to represent a simple graph having n vertices and m edges?

(c) How many edges are in a depth-first spanning tree of a connected undirected graph having n vertices and m edges?

3. (30 points). Consider the following sorting algorithm, which takes an array A and a range of indices $[i, j]$ and sorts all the elements in A from $A[i]$ to $A[j]$:

Procedure ArraySort(A, i, j):

```
n ← j − i + 1
if n = 2 then
  if A[i] > A[j] then
    Swap A[i] and A[j]
  end if
else if n > 2 then
  m ← ⌊n/3⌋
  ArraySort(A, i, j − m)
  ArraySort(A, i + m, j)
  ArraySort(A, i, j − m)
end if
```

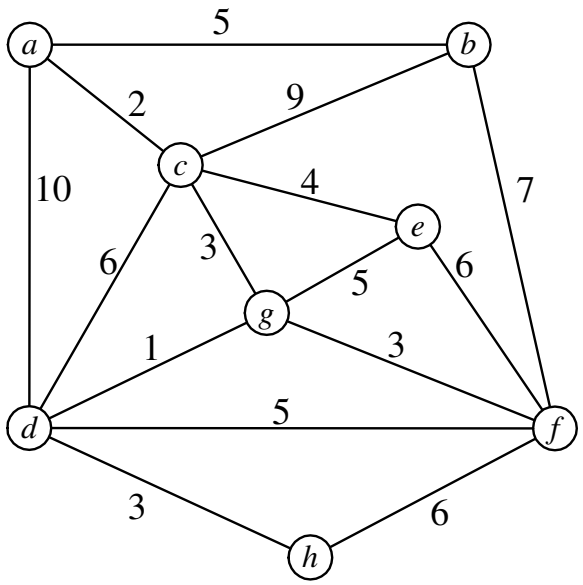
(a) Briefly argue why ArraySort is correct.

(b) Give a recurrence relation that characterizes the running time, $T(n)$, of this algorithm. You may ignore "floors" and "ceilings" so that, for example, $T(\lfloor 2n/3 \rfloor)$, $T(2n/3)$, and $T(\lceil 2n/3 \rceil)$ are all the same.

(c) Give a closed-form characterization of $T(n)$ in terms of the big-Oh notation. (Hint: you may use the master theorem.)

4. (30 points). Draw a compressed trie data structure for the following set of character strings: {`ab`, `defge`, `aears`, `bcant`, `bcers`, `defan`, `bceat`, `defls`, `dez`}. Draw the trie so that the children of any node are given in alphabetical order left-to-right.

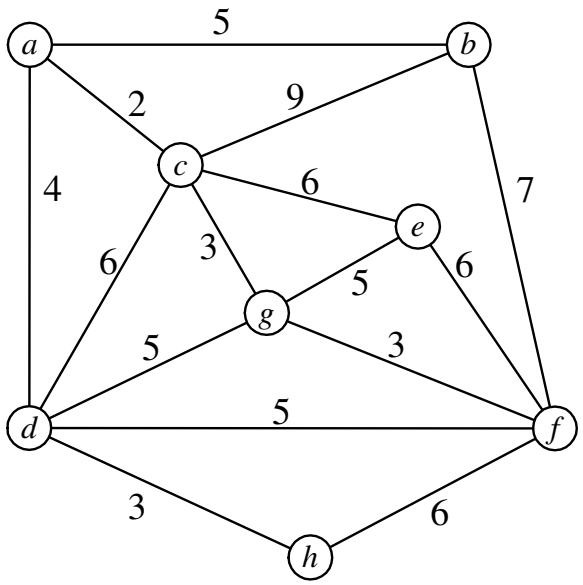
5. (30 points). Consider the following graph:



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

(b) Number the thickened edges in the order in which they would be added to T by Dijkstra's shortest path algorithm, starting at a.

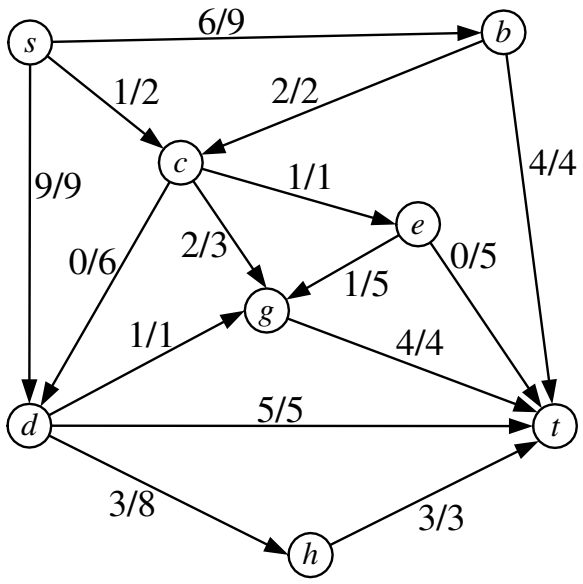
6. (30 points). Consider the following graph:



(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 the Prim-Jarnik MST algorithm, starting at a.

7. (30 points). Consider the following flow network, from s to t, where the label "a/b" on an edge indicates an edge with flow a and capacity b:



(a) Argue why this flow is a valid flow; that is, it is a flow that satisfies the constraints for a valid flow.

(b) Show that this flow is not a maximum flow by drawing over and thickening the edges of an augmenting path.

8. (30 points). **NP**-completeness.

(a) Define the complexity class, **NP**.

(b) Define the term "**NP**-complete."

(c) Define INDEPENDENT-SET as the problem that takes a graph G and an integer k and asks if G contains an independent set of vertices of size k. That is, G contains a set W of vertices of size k such that, for any u and v in W, there is no edge (u, v) in G. Show that INDEPENDENT-SET is **NP**-complete (remember that there are two parts to this). You may assume the **NP**-completeness of CLIQUE, which is the problem that takes a graph G and an integer k and asks if G contains a set X of vertices such that each pair of vertices in X is connected by an edge in G.

9. (30 points). Suppose a jeweler has a set D of n diamonds, which he wants to make into earrings. Being a perfectionist, he will make two diamonds into a pair of earrings only if the two diamonds have exactly the same weight. Unfortunately, his digital scale is broken. He has only an old merchant scale that can compare two diamonds to see which is heavier or if they are the same weight. Describe an algorithm for the jeweler to find all the equal-weight pairs of diamonds in D using an efficient number of weighings on the old merchant scale. What is the asymptotic number of weighings needed by your algorithm, in terms of n? (You may use as a subroutine any algorithm discussed in class, provided you can describe how the jeweler would "implement" it.)

10. (30 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 hate each other. Define a *diplomat* 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, describe an efficient algorithm for finding all the diplomats in G. 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.