COS 226	Algorithms and Data Structures	Fall 2005

Final

This test has 11 questions worth a total of 83 points. You have 150 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. **Write out and sign the Honor Code pledge before turning in the test.**

"I pledge my honor that I have not violated the Honor Code during this examination."

Problem	Score
1	
2	
3	
4	
5	
6	
Sub 1	

Problem	Score
7	
8	
9	
10	
11	
Sub 2	

Total	

Name:

Login ID:

1. **Analysis of algorithms. (10 points)**

A *Fibonacci heap* is a priority queue that supports the following operations in the given *amortized* running time. Here, the elements are 1 through N and the data in the keys can only be accessed via pairwise comparisons.

Operation	Description	Amortized
CREATE(N)	create an empty heap of capacity N	$O(N)$
INSERT(i, k)	insert element i, and assign it key k	$O(1)$
DECREASEKEY(i, k)	decrease the key associated with element i to k	$O(1)$
DELETEMIN()	delete and return the element with the smallest key	$O(\log N)$

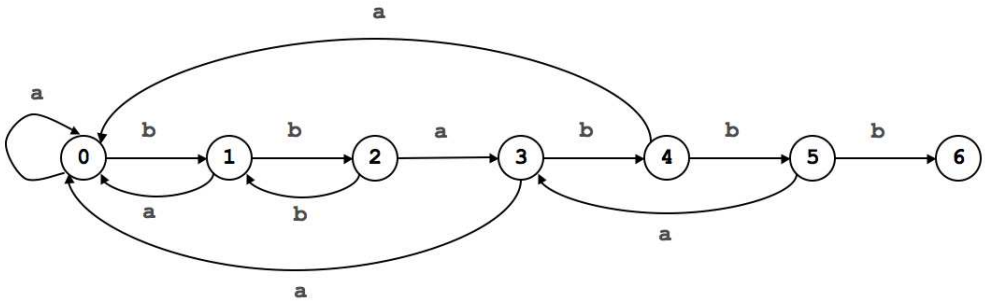
(a) Define what the amortized running times mean in this context.

(b) Suppose you implement Dijkstra's algorithm using a Fibonacci heap as the underlying priority queue. What is the resulting *worst-case* asymptotic running time of Dijkstra's algorithm as a function of the number of vertices V and the number of edges E.

(c) Explain briefly why no priority queue can implement INSERT, DECREASEKEY, and DELETEMIN in $O(1)$ time each.

2. **String searching. (6 points)**

The following DFA purports to accept precisely those strings (over the two letter alphabet) that contain `bbabbb`. Assume state 0 is the start state and state 6 is the accept state.



(a) Are there any strings containing `bbabbb` that it rejects?
If so, give the shortest such string and circle it.

(b) Are there any strings not containing `bbabbb` that it accepts?
If so, give the shortest such string and circle it.

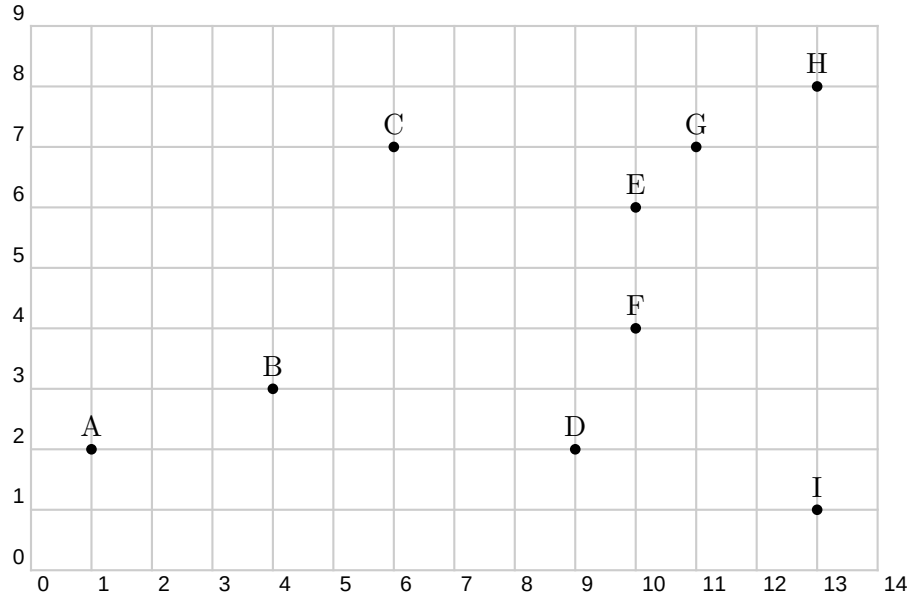
(c) Describe how to fix the DFA so that it works as intended.

3. **Pattern matching. (6 points)**

   Draw an NFA that recognizes the same language that the regular expression `a(bc)*d | e*` describes. Use the notation and construction given in lecture. Circle your final answer.

4. **Convex hull. (6 points)**

   Run the Graham scan algorithm to compute the convex hull of the 9 points below, using I as the base point, and continuing counterclockwise starting at H.



   (a) List the points in the order that they are considered for insertion into the convex hull.

   (b) Give the points that appear on the trial hull (after each of the 8 remaining points are considered) in the order that they appear.

   1.

   2.

   3.

   4.

   5.

   6.

   7.

   8.

5. **Geometry. (12 points)**

Given a set of N intervals on the x-axis of the form (a_i, b_i), design an $O(N \log N)$ sweep-line algorithm to find a value x that is contained within the maximum number of intervals. You may assume that no two endpoints have the same value.

(a) What are the events?

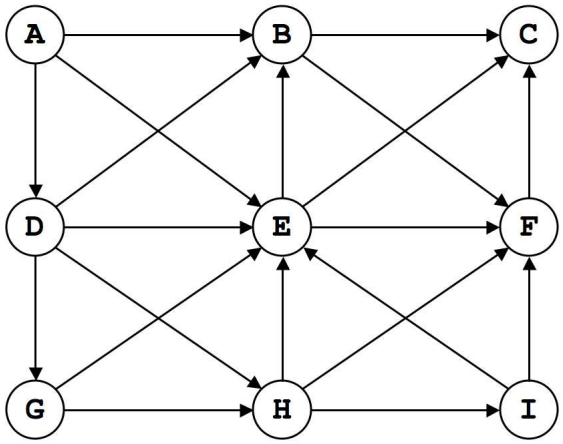
(b) How do you implement the sweep line?

(c) What data structure stores the set of intervals that intersect the sweep line?

(d) How does your sweep-line algorithm work, i.e., how do you process each event?

6. **Digraphs and DFS. (6 points)**

Consider the following DAG. Run depth first search, starting at vertex A. Assume the adjacency lists are in lexicographic order, e.g., when exploring vertex A, use A-B before A-D or A-E.



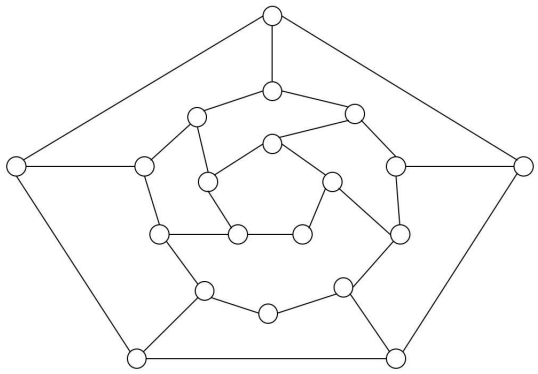
(a) List the vertices in *preorder*.

(b) List the vertices in *postorder*.

(c) List the vertices in *topological order*.

7. **Undirected graphs and BFS. (10 points)**

The *girth* of an undirected graph is the length of the shortest cycle.

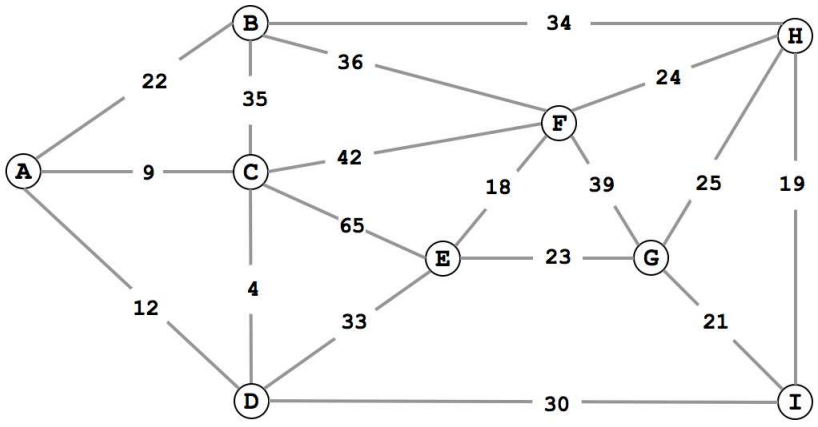


A girth 5 graph.

(a) Describe a polynomial time algorithm to find the girth of a graph.

(b) What is its asymptotic running time as a function of the number of vertices V and the number of edges E.

8. **Minimum spanning tree. (8 points)**



(a) Consider the weighted graph above. Give the list of edges in the MST in the order that *Kruskal's algorithm* inserts them. For reference, the 18 edge weights in ascending order are:

4 9 12 18 19 21 22 23 24 25 30 33 34 35 36 39 42 65

(b) Consider the weighted graph above. Give the list of edges in the MST in the order that *Prim's algorithm* inserts them. Start Prim's algorithm from vertex A.

9. **Data compression. (6 points)**

Suppose you compress the following string using LZW compression over the two letter alphabet.

```
a a b a a b a b a a a b a
```

(a) List the strings in the LZW dictionary in the order they are inserted. (Assume the dictionary is initialized to begin with the two strings `a` and `b`.)

(b) Draw the binary trie representation of the resulting LZW dictionary.

10. **Linear programming. (5 points)**

Convert the following linear program to *standard form*, i.e., a maximization problem with equality constraints and nonnegative variables. (Do not solve.)

$$
\begin{array}{lrcrcrcl}
\text{minimize} & 26A & + & 30B & + & 20C & & \\
\text{subject to:} & A & + & B & + & C & = & 100 \\
 & 2A & + & 6B & + & 3C & \geq & 145 \\
 & 7A & + & 1B & + & C & \geq & 85 \\
 & 5A & + & 1B & + & 6C & \leq & 95 \\
 & A & , & B & , & C & \geq & 0
\end{array}
$$

11. **Reductions. (8 points)**

Consider the following two decision problems.

HAM-PATH. Given a digraph, is there a path that visits each vertex exactly once?

SHORTEST-SIMPLE-PATH. Given a digraph with integer edge weights (positive or negative), two distinguished vertices s and t, and an integer L, is there a *simple* path from s to t of length at most L. (A simple path is a path that visits each vertex at most once.)

Show that HAM-PATH polynomial reduces to SHORTEST-SIMPLE-PATH. To demonstrate your reduction, draw the instance of SHORTEST-SIMPLE-PATH associated with the following instance of HAM-PATH. Be sure to include the edge weights and the value L. Assuming it is correct (and obvious how it extends to arbitrary digraphs), you need not describe it further.