COS 226	Algorithms and Data Structures	Fall 2006

Final

This test has 12 questions worth a total of 83 points. You have 180 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet (8.5-by-11, in your own handwriting). 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	
12	
Sub 2	

Total	

Name:

Login ID:

1. **Analysis of algorithms. (8 points)**

(a) Describe precisely and succinctly what the following statement means.
Do *not* use big Oh notation in your answer.

Heapsort has a worst-case running time of $O(N \log N)$.

(b) State precisely and succinctly what is *the lower bound for sorting* that we proved in class.

(c) Suppose you have an algorithm that takes about 1 hour to solve a problem of size N. You buy a new computer that is twice as fast and has twice as much memory. About how long will it take you to solve a problem of size $2N$? Assume the running time of your program is $8N^2 + 13N^{4/3} + 12N + 3$ nanoseconds. Circle the best answer.

1/2 hour   1 hour   2 hours   4 hours   16 hours

(d) Repeat the previous question, but assume the running time of your program is $2N \ln N$.

1/2 hour   1 hour   2 hours   4 hours   16 hours

2. **Algorithm analogies. (10 points)**

   Complete each of the following analogies with the best answer.

   (a) tractable : Euler path :: intractable : ____________

   (b) MSD radix sort : R-way trie :: 3-way radix quicksort : ____________

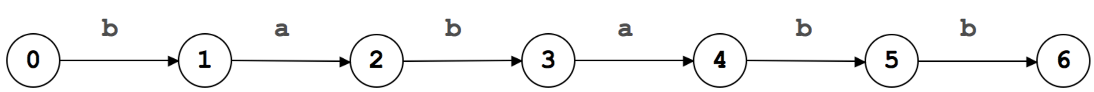
   (c) unweighted graph : BFS :: weighted graph : ____________

   (d) sorting : pairwise comparison :: convex hull : ____________

   (e) symbol table : BST :: priority queue : ____________

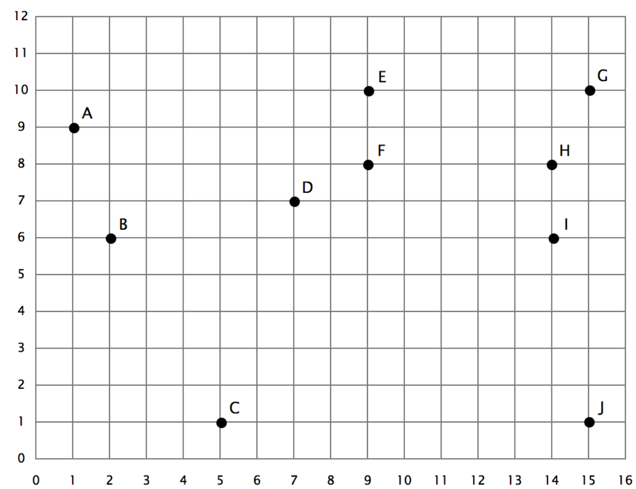
3. **String searching. (6 points)**

   Complete the following DFA to match precisely those strings (over the two letter alphabet) that contain `bababb` as a substring. State 0 is the start state and state 6 is the accept state.



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

Run the Graham scan algorithm to compute the convex hull of the 10 points below, using J as the base point, and continuing counterclockwise starting at G.



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

```
J  G  H
```

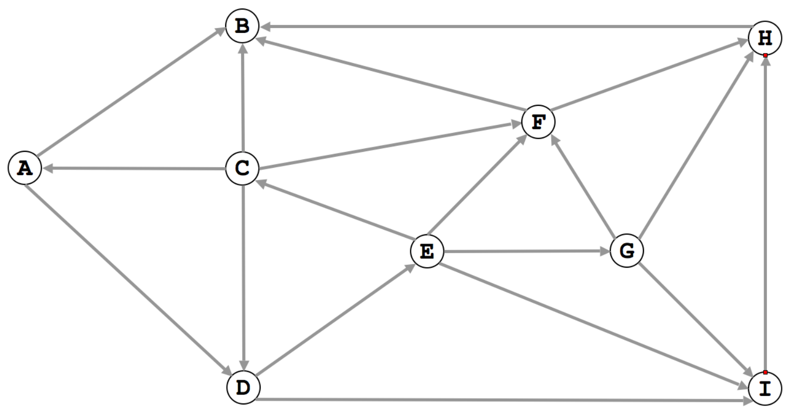
(b) Give the points that appear on the trial hull (after each of the remaining iterations).

```
1.   J -> G -> H

2.

3.

4.

5.

6.

7.

8.
```

(c) Define what it means for a set of points in the plane to be *convex*.

5. **BFS and DFS. (6 points)**

Consider the following directed graph.



(a) Run *depth-first search*, starting at vertex A. Assume the adjacency lists are in lexicographic order, e.g., when exploring vertex E, consider E-C before E-F, E-G or E-I.

i. List the vertices in preorder.

ii. List the vertices in postorder.

(b) Run *breadth-first search*, starting at vertex A. Assume the adjacency lists are in lexicographic order. List the vertices in the order in which they are enqueued.

6. **Algorithm throwdown. (10 points)**

For each of the following pairs, briefly describe one reason why you'd use one instead of the other. A familiar example is given below.

Mergesort	Quicksort
stability	in-place

(a)

Red-black tree	Ternary search trie

(b)

Dijkstra's algorithm	Bellman-Ford-Moore

(c)

Burrows-Wheeler	LZW compression

(d)

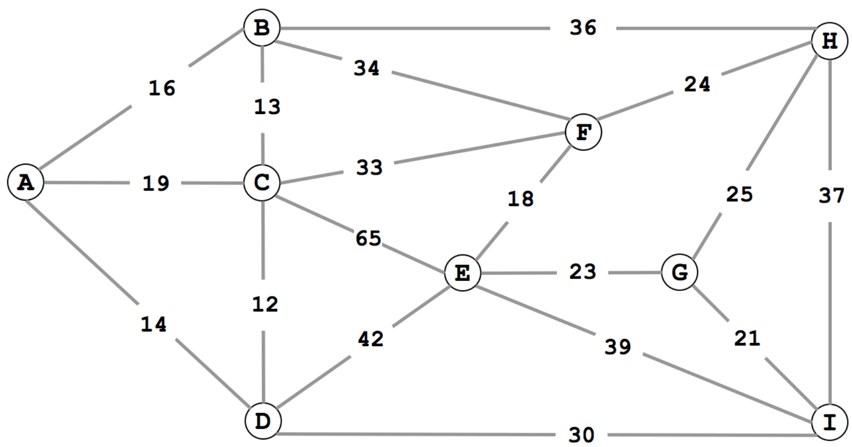
Red-black tree	Hash table

(e)

Breadth-first search	Depth-first search

7. **Minimum spanning tree. (6 points)**

Consider the following weighted graph.



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

12 13 14 16 18 19 21 23 24 25 30 33 34 36 37 39 42 65

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

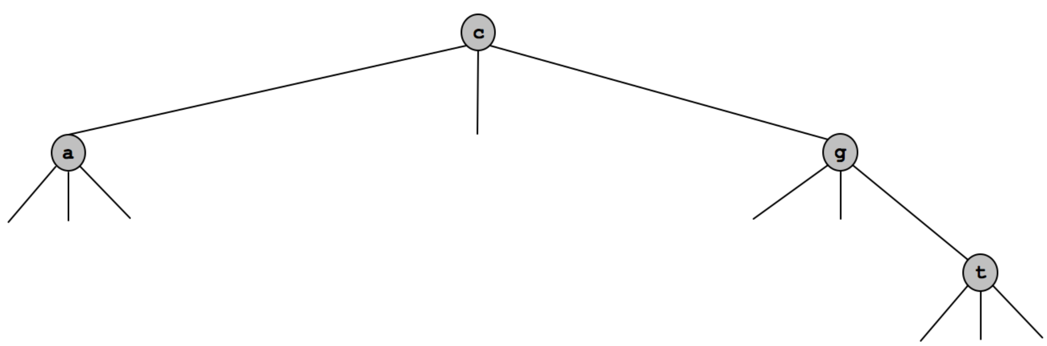
8. **Data compression and tries. (6 points)**

Suppose you apply the LZW algorithm to the following string (using the DNA alphabet).

```
a a c a a t a a c t
```

(a) List the strings in the LZW dictionary in the order they are inserted.
Assume the dictionary is initialized to begin with the four strings `a`, `c`, `g` and `t`.

(b) Complete the *ternary search trie* representation of the resulting LZW dictionary.



9. **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.)

$$
\begin{array}{lrcrcrcl}
\text{minimize} & 26A & + & 30B & + & 20C & & \\
\text{subject to:} & A & + & B & + & 2C & = & 200 \\
 & 3A & + & 6B & + & 3C & \leq & 45 \\
 & 9A & + & 2B & + & 4C & \geq & 85 \\
 & \mid 5A & + & 9B & + & 6C \mid & \leq & 95 \\
 & A & , & B & , & C & \geq & 0
\end{array}
$$

Here, $|\cdot|$ denotes the absolute value function.

10. **Reductions. (6 points)**

Consider the following two problems.

ElementDistinctness. Given N real numbers, are any two of them equal?

ClosestPair. Given N points in the plane, find a pair that is closest in Euclidean distance.

Show that ElementDistinctness linear reduces to ClosestPair. To demonstrate your reduction, give the instance of ClosestPair associated with the following instance of ElementDistinctness and describe how you could solve the element distincness problem given the solution to the corresponding closest pair problem.

```
23.0  3.14  2.72  1.41  3.14  5.32
```

11. **Sorting and hashing. (8 points)**

    Your answers will be graded on correctness, clarity, and succinctness.

    (a) Describe an algorithm for ElementDistinctness that runs in $O(N \log N)$ time in the worst-case and uses $O(1)$ extra memory. Assume the N real numbers are stored in an array.

    (b) Describe an algorithm for ElementDistinctness that runs in $O(N)$ time on average.

12. **Shortest path with landmark. (6 points)**

Given a *directed* graph G with positive edge weights and a *landmark* vertex x, your goal is to find the length of the shortest path from one vertex v to another vertex w that passes through the landmark x.

(For example, Federal Express packages are routed through x = Atlanta.)

(a) Describe a $O(E \log V)$ algorithm for the problem. Justify briefly why your proposed algorithm is correct.

(b) Now suppose that you will perform many such shortest path queries for the same landmark x, but different values of v and w. Describe how to build a data structure in $O(E \log V)$ time so that, given the data structure, you can process each query in *constant time*.