**NAME:**

**login ID:**
**precept #:**

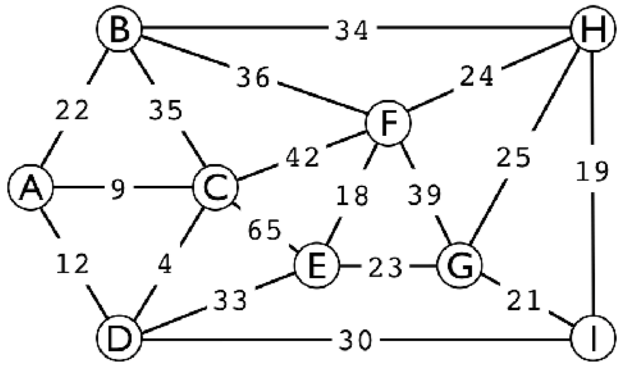
# COS 226 Final Exam, Fall 2007

This test is 15 questions, weighted as indicated. 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. *Put your name, login ID, and precept number on this page* (*now*), and write out and sign the Honor Code pledge before turning in the test. You have *three hours* to complete the test.
*"I pledge my honor that I have not violated the Honor Code during this examination."*

| | | | |
|---|---|---|---|
| 1 | /8 | | |
| 2 | /5 | | |
| 3 | /5 | | |
| 4 | /8 | | |
| 5 | /4 | | |
| | | Subtotal | /30 |
| 6 | /6 | | |
| 7 | /6 | | |
| 8 | /7 | | |
| 9 | /7 | | |
| 10 | /4 | | |
| | | Subtotal | /30 |
| 11 | /14 | | |
| 12 | /7 | | |
| 13 | /5 | | |
| 14 | /5 | | |
| 15 | /9 | | |
| | | Subtotal | /40 |
| TOTAL | /100 | | |

January 23, 2008

1. **MST** (8 points). Consider the following undirected weighted graph.



*a*. Give the list of edges in the minimum spanning tree in the order that *Kruskal's algorithm* inserts them. For reference, the 18 edge weights are listed here:

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

*b*. Give the list of edges in the minimum spanning tree in the order that *Prim's algorithm* inserts them, assuming that it starts at vertex A.

2. **KMP** (5 points). The following table gives the KMP state transitions for the string `babbabbbab`, except for the two labeled X and Y in the table. Fill in the missing values.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| a | 0 | 2 | 0 | 2 | 5 | 0 | 2 | 8 | 0 |
| b | 1 | 1 | 3 | 4 | X | 6 | Y | 1 | 9 |

X ________

Y ________

3. **Mystery code** (5 points). Circle the choice that describes a use of the following code:

```
for (Edge e : G.adj(v))
{
   int w = e.to();
   if (dist[w] > dist[v] + e.weight())
   {
      dist[w] = dist[v] + e.weight();
      pred[w] = e;
      pq.insert(dist[w], w);
   }
}
```

A. To process a vertex for Prim's algorithm

B. To compute the MST of a weighted graph

C. To topologically sort a digraph

D. To process a vertex for Dijkstra's algorithm

E. To find a cycle in a graph

4. **Graph algorithms** (8 points). Match each of the given graph-processing algorithms with a description. Write the letter corresponding to your answer for each algorithm in the space provided. In some cases, more than one answer might be argued, but you must put only one letter per space. If you pick the best matches, you will use all the letters.

| | | |
|---|---|---|
| A. Finds strong components in linear time. | _______ | Kruskal |
| B. General graph-searching scheme. | _______ | Kosaraju |
| C. Optimal MST algorithm for dense graphs. | _______ | Bellman-Ford |
| D. Fundamental recursive method. | _______ | Prim |
| E. For single-source shortest paths in unweighted graphs. | _______ | DFS |
| F. Reduces MST computation to sorting. | _______ | BFS |
| G. For single-source shortest paths in weighted digraphs with positive edge weights. | _______ | PFS |
| H. For single-source shortest paths in weighted digraphs, when edge weights could be negative. | _______ | Dijkstra |

5. **LZW compression** (4 points). Which of the following best describes the length of the code produced by the LZW compression algorithm for a string consisting of 11*N* characters made up of *N* copies of the substring `abracadabra` ? Circle your answer.

   A. log *N*

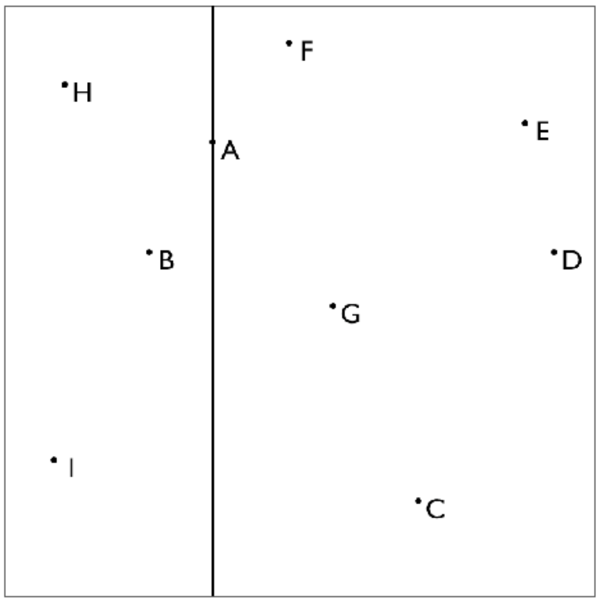
   B. (log *N*) * (log *N*)

   C. square root of *N*

   D. *N*

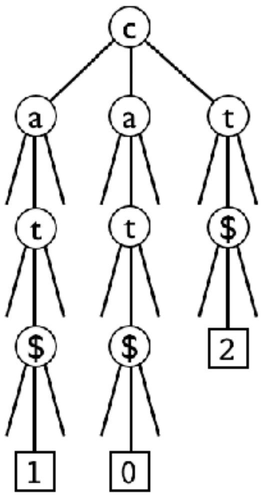
   E. *N* log *N*

6. **2D trees** (6 points). Draw the partition of the plane that results from inserting the points drawn below `A B C D E F G H I` (in that order) into a 2D tree. The first partition (a vertical line) is drawn for you.



   Draw the tree here:

7. **Suffix TST** (6 points). The suffix TST corresponding to a string is constructed by building the TST corresponding to the string suffixes, including on each an end-of string character $ that is larger than every string character. Thus, every path through the TST ends in a $, below which we put an external node corresponding to the starting point of the suffix. For example, the suffix TST corresponding to the string `cat` is drawn at left.



Draw the suffix TST corresponding to the string `yoyo` .

8. **String symbol table** (7 points). Match each of the given search data structures with one or more of the given characteristics, where N is the number of keys, L is the number of digits in a key, and M is the size of the alphabet. Write as many letters (in alphabetical order) as apply in the blank to the left of the name of the data structure.

   A. tree/trie shape is dependent on the order in which keys are inserted

   B. worst-case search time is proportional to L

   C. space usage (not including strings themselves) is not dependent on L

   D. worst-case search time is proportional to LN

   E. inorder traversal gives sorted order

| | |
|---|---|
| _________ | M-ary trie |
| _________ | BST |
| _________ | TST |
| _________ | red-black tree |

9. **RE pattern matching I** (7 points). Draw an NFA (nondeterministic finite state automata) that recognizes the same language that the regular expression `a (b | c)* e*` describes. Use the notation and construction given in lecture. Circle your final answer.

10. **RE pattern matching II** (4 points). It is easy to build a NFA (nondeterministic finite state automata) corresponding to a regular expression, and a basic theorem from automata theory says that we can convert every NFA to a DFA (deterministic finite state automata). Why do we not do so to implement RE pattern matching? Circle one of the following choices.

    A. The DFA might have an exponential number of states.

    B. There might exist a string that the DFA recognizes but the NFA does not.

    C. There might exist a string that the NFA recognizes but the DFA does not.

    D. The NFA might loop.

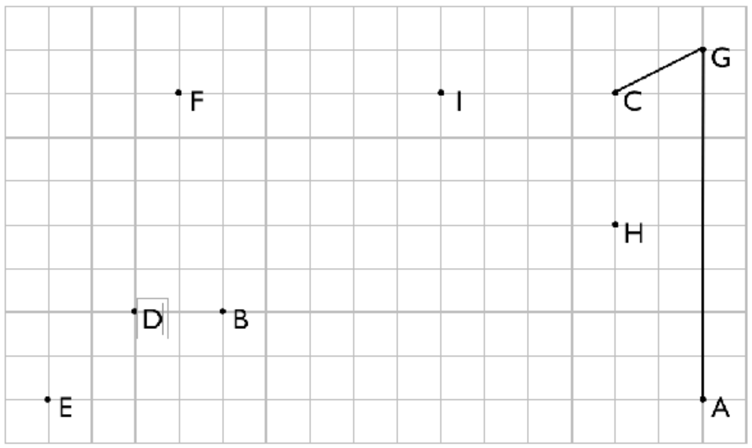
    E. The proof of the theorem is not constructive (does not tell us how to construct the DFA from the NFA).

11. **7 sorting algorithms** (14 points). The leftmost column is the original input of strings to be sorted, and the rightmost column is the sorted result. The other columns are the contents at some intermediate step during one of the 7 sorting algorithms listed below. Match up each algorithm by writing its letter under the corresponding column. Use each letter exactly once.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| fuzz | zeta | benz | zoom | faze | benz | faze | cozy | benz |
| fizz | zinc | cozy | zing | cruz | czar | cruz | fizz | cozy |
| cozy | maze | czar | zeta | cozy | cruz | cozy | fuzz | cruz |
| zinc | faze | cruz | zinc | faze | cozy | fizz | zinc | czar |
| fizz | fuze | fuzz | raze | faze | fizz | faze | fizz | faze |
| fuzz | faze | fizz | size | benz | faze | benz | fizz | faze |
| fizz | raze | fizz | fuzz | czar | fuzz | czar | fizz | faze |
| fizz | fuze | fuzz | jazz | fizz | fuze | faze | fuzz | fizz |
| maze | haze | fizz | maze | fizz | fuzz | fizz | benz | fizz |
| faze | size | fizz | fuzz | fizz | fuze | fizz | czar | fizz |
| czar | fuze | faze | fuzz | fizz | fuze | fizz | faze | fizz |
| benz | faze | fuze | lazy | fizz | faze | fuzz | maze | fizz |
| fuze | zing | fuzz | fuzz | fizz | fizz | fuze | faze | fizz |
| fuzz | zoom | faze | fuze | fuzz | fuzz | fuzz | fizz | fuze |
| faze | czar | fizz | fizz | fuze | fizz | fizz | fuze | fuze |
| fizz | cozy | fuzz | fizz | zinc | fizz | zinc | fuzz | fuze |
| jazz | lazy | fuze | fizz | jazz | fuzz | jazz | cruz | fuzz |
| zoom | fuzz | fuze | fizz | zoom | fizz | zoom | jazz | fuzz |
| cruz | fizz | faze | cruz | fuzz | fizz | fizz | zing | fuzz |
| zing | fizz | fizz | faze | zing | faze | zing | zoom | fuzz |
| fuzz | fuzz | haze | fizz | fuzz | size | fuzz | fuze | haze |
| raze | fizz | jazz | czar | raze | raze | raze | fuzz | jazz |
| fuze | fizz | lazy | fuze | fuze | zing | fuze | lazy | lazy |
| lazy | benz | maze | benz | lazy | zeta | lazy | raze | maze |
| haze | fuzz | raze | haze | haze | zoom | haze | fuze | raze |
| zeta | fizz | size | fuze | fuze | jazz | zeta | haze | size |
| size | jazz | zinc | cozy | fuzz | haze | size | size | zeta |
| fuze | cruz | zoom | fizz | size | lazy | fuze | zeta | zinc |
| faze | fuzz | zing | faze | zeta | maze | fuzz | faze | zing |
| fizz | fizz | zeta | faze | maze | zinc | maze | fizz | zoom |
| | ____ | ____ | ____ | ____ | ____ | ____ | ____ | |

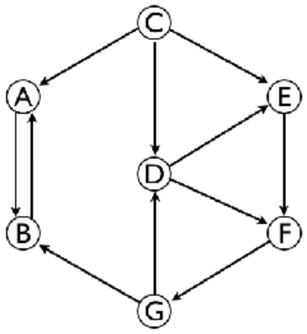
A. 3-way radix quicksort
B. LSD radix sort
C. MSD radix sort
D. Quicksort (with no random shuffle)
E. Quicksort with 3-way partitioning (with no random shuffle)
F. Bottom-up mergesort
G. Heapsort

12. **Convex hull** (7 points). Complete the trace given below for the Graham scan when used to find the convex hull of the following point set. For each point scanned, give the point followed by the current list of points on the trial hull after that point is scanned. The first three lines of the trace are filled in for you.



| *next point* | *current hull* |
|---|---|
| A | A |
| G | A G |
| C | A G C |
| ____ | ________________________________ |
| ____ | ________________________________ |
| ____ | ________________________________ |
| ____ | ________________________________ |
| ____ | ________________________________ |
| ____ | ________________________________ |

13. **Strong Components** (5 points). The following digraph has three strong components. List the vertices in each.



first component ________________________________

second component ________________________________

third component ________________________________

14. **Reduction** (5 points). You are a manager at a large and successful internet company that was built on solving problem A in cubic time. One of your mathematicians has proven that problem A linear-reduces to problem B. Furthermore, one programmer has implemented a linear-time solution to problem B and another programmer has implemented a quadratic-time solution to problem A. What should you do?

A. Fire the first programmer and promote the mathematician.

B. Fire the second programmer and promote the mathematician.

C. Promote the first programmer and fire the mathematician.

D. Promote them all.

E. Fire them all.

15. **Hard problem identification** (9 points). This question is in regard to your new job working for a software technology company. Your boss (having been told by you on the basis on your 126 knowledge that the company had better not bet its future on developing an application that finds an optimal tour connecting a set of cities) is still looking for a challenging project for you. Your boss is willing to invest in things that might be difficult, but not things that we know to be impossible or that we believe to be intractable. On the basis of your 226 knowledge, which of the following ideas can you tell your boss to forget about? Circle all that apply.

A. A pattern matcher that allows users to use a DFA to specify a richer class of patterns than with the regular expressions in `grep`

B. An algorithm that compresses any given file by one percent

C. A linear-time algorithm for finding the MST of a set of points in the plane that can only compare the distances between two points (not know their coordinate values)

D. A linear-time algorithm for finding the MST of a graph with positive edge weights

E. A linear-time algorithm for sorting an array of numbers that can only compare two numbers (not know their values)

F. A poly-time algorithm that finds the longest path connecting two vertices in a digraph with positive edge weights

G. A poly-time algorithm that finds the shortest path connecting two vertices in a digraph with positive edge weights

H. A poly-time algorithm that finds the longest path connecting two vertices in a digraph with edge weights that could be negative

I. A poly-time algorithm that finds the shortest path connecting two vertices in a digraph with edge weights that could be negative (but with no negative cycles).