COS 226 **Algorithms and Data Structures** **Fall 2008**

Midterm

This test has 7 questions worth a total of 60 points. You have 80 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	
Sub 1	

Problem	Score
5	
6	
7	
Sub 2	

Total	

Name:

Login ID:

Precept: P01 12:30 Boaz
P02 3:30 Boaz

1. **8 sorting algorithms. (8 points)**

   The column on the left is the original input of strings to be sorted; the column on the right are the string in sorted order; the other columns are the contents at some intermediate step during one of the 8 sorting algorithms listed below. Match up each algorithm by writing its number under the corresponding column. Use each number exactly once.

push	iran	axis	axis	bank	whip	axis	bush	bush	axis
poll	iraq	bank	bank	duck	vice	bank	iran	iraq	bank
iran	poll	bear	bear	axis	vote	duck	bank	iran	bear
iraq	push	bull	bill	bear	veto	evil	fdic	chad	bill
bank	axis	duck	bull	chad	town	fdic	axis	bank	bull
fdic	bank	evil	bush	fdic	pork	hall	evil	fdic	bush
axis	evil	fdic	cage	bill	iraq	iran	duck	axis	cage
evil	fdic	hall	chad	cage	iraq	iraq	chad	evil	chad
lame	duck	iran	duck	iraq	race	lame	hall	cage	duck
duck	hall	iraq	evil	iraq	poll	poll	bull	duck	evil
town	lame	iraq	fdic	bull	spin	push	bear	bill	fdic
hall	town	iraq	flag	evil	iran	town	cage	hall	flag
iraq	bear	lame	iraq	lame	iraq	iraq	bill	flag	free
iraq	bull	poll	iraq	iraq	chad	iraq	free	free	hall
bull	iraq	push	poll	bush	iraq	bull	flag	bull	iran
bear	iraq	town	iran	flag	iraq	bear	iraq	bear	iraq
veto	free	flag	veto	pork	evil	veto	iraq	iraq	iraq
race	race	free	race	jobs	lame	race	iraq	race	iraq
free	veto	jobs	free	free	free	free	iraq	iraq	iraq
whip	whip	race	whip	hall	duck	whip	iraq	whip	iraq
vice	flag	spin	vice	push	bank	vice	jobs	vice	jobs
jobs	jobs	veto	jobs	poll	jobs	jobs	spin	jobs	lame
spin	spin	vice	spin	iran	push	spin	vice	spin	poll
flag	vice	whip	hall	iraq	flag	flag	pork	iraq	pork
pork	bill	bill	pork	veto	hall	pork	vote	pork	push
vote	cage	bush	vote	race	fdic	vote	whip	vote	race
bill	pork	cage	iraq	spin	bill	bill	race	town	spin
cage	vote	chad	push	iraq	cage	cage	veto	lame	town
chad	bush	iraq	lame	vice	axis	chad	town	iraq	veto
iraq	chad	iraq	iraq	vote	bull	iraq	lame	poll	vice
bush	iraq	pork	town	town	bush	bush	poll	push	vote
iraq	iraq	vote	iraq	whip	bear	iraq	push	veto	whip
----	----	----	----	----	----	----	----	----	----
0									1

(0) Original input

(1) Sorted

(2) Selection sort

(3) Insertion sort

(4) Shellsort (*13-4-1 increments*)

(5) Mergesort (*top-down*)

(6) Mergesort (*bottom-up*)

(7) Quicksort (*standard, no shuffle*)

(8) Quicksort (*3-way, no shuffle*)

(9) Heapsort

2. **Sorting equal keys. (6 points)**

Suppose that you are sorting an array containing the following 7 equal keys (the subscript is not part of the key—its purpose is to uniquely identify each of the equal keys).

$$A_0 \;\; A_1 \;\; A_2 \;\; A_3 \;\; A_4 \;\; A_5 \;\; A_6$$

What is the result of running the standard version (from the Sedgewick textbook) of each of the following sorting algorithms?

Insertion	
Selection	
Shellsort (*13-4-1 increments*)	
Mergesort (*top-down*)	
Quicksort (*no shuffle*)	
Heapsort	

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

(a) Big-Oh notation is less precise than tilde notation at describing the growth of a function because:

I. Big-Oh notation suppresses the coefficient of the highest order term.

II. Big-Oh notation provides an upper bound on the growth of a function, but does not provide a lower bound.

III. Big-Oh notation suppresses lower order terms, so it only describes the limiting behavior of a function for large values of N.

Circle the best answer.

(a) I only.
(b) I and II only.
(c) I and III only.
(d) I, II and III.
(e) None.

(b) Suppose that your symbol table implementation supports the *insert* and *search* operations in *amortized* $4 \lg N$ compares.

I. Starting from an empty data structure, any sequence of N *insert* and *search* operations uses at most $4N \lg N$ compares. Which of the following are true? Circle the best answer.

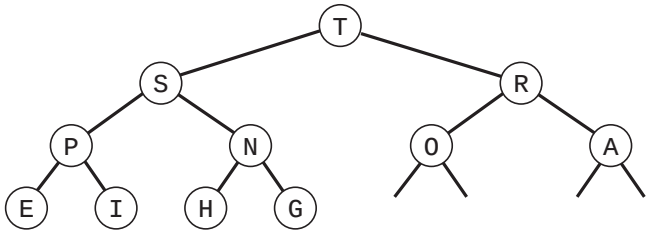
II. Any sequence of N *insert* and *search* operations uses at most $4N \lg N$ compares.

III. Starting from an empty data structure, the expected number of compares to perform N *insert* and *search* operations is $4N \lg N$, but there is a (small) probability that it will take $5N \lg N$ compares (or more).

(a) I only.
(b) I and II only.
(c) I and III only.
(d) I, II and III.
(e) None.

4. **Binary heaps. (8 points)**

Consider the following max-heap.



(a) Draw the heap that results after inserting the key `Z`.

(b) Draw the heap that results after deleting the maximum key from your answer to (a).

(c) True or false: given any binary heap with N distinct keys, the result of inserting a key larger than any of the N keys, and then deleting the maximum key yields the original binary heap.

5. **Ordered-array implementation of a set. (8 points)**

Suppose you implement the *set data type* using an ordered array. That is, you maintain the N keys in the set in ascending order in an array.

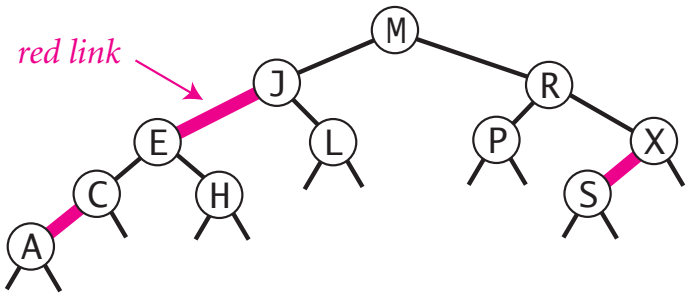
What is the order of growth of the worst-case running time of each of the operations below. Write down the best answer in the space provided using one of the following possibilities.

$1 \qquad \log N \qquad \sqrt{N} \qquad N \qquad N \log N \qquad N^2$

`add(key)`	*add the key to the set*	
`contains(key)`	*is the key in the set?*	
`ceiling(key)`	*smallest key in set* $\geq$ *given key*	
`rank(key)`	*number of keys in set* $<$ *given key*	
`select(i)`	*ith largest key in the set*	
`min()`	*minimum key in the set*	
`delete(key)`	*delete the given key from the set*	
`iterator()`	*iterate over all* N *keys in the set in order*	

6. **Red-black trees. (8 points)**

Consider the following left-leaning red-black tree.

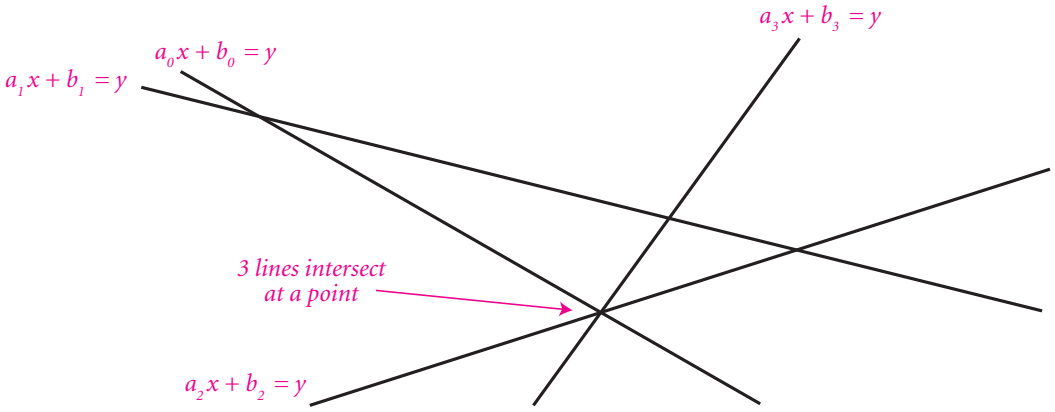


Add the key `Z`, then add the key `B`. Draw the resulting left-leaning red-black tree.

7. **Line intersection. (14 points)**

Given N lines in the plane, design an algorithm to determine if any 3 (or more) of them intersect at a single point.

For simplicity, assume that all of the lines are distinct and that there are no vertical lines.



When we ask you to design an algorithm, give a crisp and concise English description of your algorithm—don't write Java code. Your solution will be graded for correctness, efficiency, clarity, and conciseness.

(a) We specify a line by two numbers a and b such that a point (x, y) is on the line if and only if $ax + b = y$. Suppose that you are given two lines $a_0x + b_0 = y$ and $a_1x + b_1 = y$. Design a constant-time algorithm that determines if the two lines intersect and, if so, finds the point of intersection.

Assume that you can perform arithmetic operations with arbitrary accuracy in constant time. That is, don't worry about floating-point precision.

(b) Given an array of N lines, design an $O(N^2)$ algorithm to determine if any 3 (or more) of them intersect at single point. For partial credit, design an $O(N^2 \log N)$ algorithm.

Assume that you have access to a hash table in which the insert and search operations take constant time.

(c) How would you modify your algorithm in (b) if you needed to *output* all sets of 3 (or more) lines that intersect at a single point. Output each maximal set exactly once.