

COS 226	Algorithms and Data Structures	Fall 2008
Midterm Solutions		

1. **8 sorting algorithms.**

0 6 5 2 4 9 3 8 7 1

2. **Sorting equal keys.**

Insertion	$A_0 A_1 A_2 A_3 A_4 A_5 A_6$
Selection	$A_0 A_1 A_2 A_3 A_4 A_5 A_6$
Shellsort	$A_0 A_1 A_2 A_3 A_4 A_5 A_6$
Mergesort	$A_0 A_1 A_2 A_3 A_4 A_5 A_6$
Quicksort	$A_4 A_3 A_5 A_6 A_0 A_2 A_1$
Heapsort	$A_1 A_2 A_3 A_4 A_5 A_6 A_0$

as of Spr 2009 the answer is:
A5 A4 A6 A0 A1 A3 A2

sort consists of building heap (which doesn't exch any elements) and deleteMax for each element.

3. **Analysis of algorithms.**

(a) I and II only

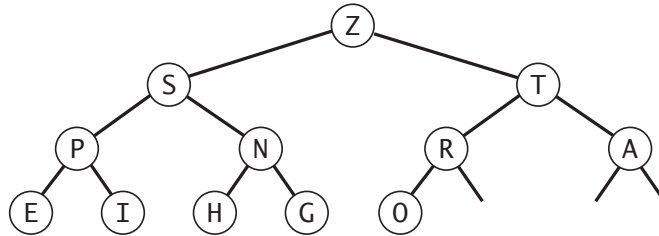
Big-Oh notation and tilde notation both suppress lower order terms.

(b) I only

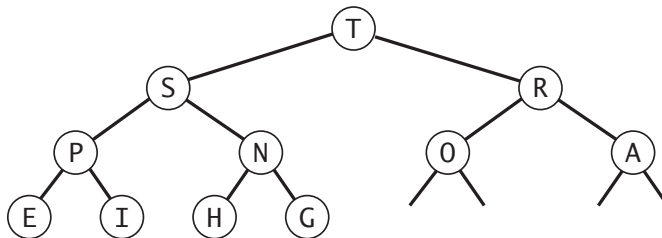
Amortized analysis provides a worst-case guarantee on any sequence of operations starting from an empty data structure.

4. Binary heaps.

(a)



(b)

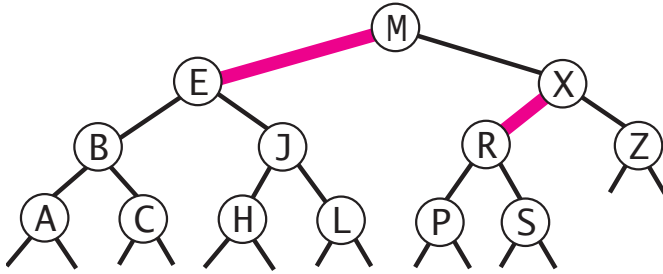


(c) True.

5. Ordered-array implementation of a set.

<code>add(key)</code>	<i>add the key to the set</i>	N
<code>contains(key)</code>	<i>is the key in the set?</i>	$\log N$
<code>ceiling(key)</code>	<i>smallest key in set \geq given key</i>	$\log N$
<code>rank(key)</code>	<i>number of keys in set $<$ given key</i>	$\log N$
<code>select(i)</code>	<i>ith largest key in the set</i>	1
<code>min()</code>	<i>minimum key in the set</i>	1
<code>delete(key)</code>	<i>delete the given key from the set</i>	N
<code>iterator()</code>	<i>iterate over all N keys in the set in order</i>	N

6. Red-black trees.



7. Line intersection.

(a) There are two cases:

- If the two lines have the same slope ($a_0 = a_1$), then return no intersection.
- Otherwise, the point (x, y) of intersection is given by:

$$x = -\frac{b_1 - b_0}{a_1 - a_0}, \quad y = a_0x + b_0$$

(b) To determine whether the i th line is involved in an intersection with 3 (or more) lines:

- Create a symbol table with key = point, value = list (say, a queue) of lines.
- For each line $j \neq i$ in order:
 - Compute the intersection point p between line i and line j .
 - If they don't intersect, continue.
 - If the key p is not already in the symbol table, add an entry to the symbol table with key = p and value = empty list.
 - Add line j to the end of the list associated with p .
- For each key in the symbol table, if it's list contains 2 (or more) lines, they correspond to 3 (or more) lines intersecting at a single point (line i , plus the lines in the list).

Implement the symbol table using a separate-chaining (or linear-probing) hash table so that each insert/search takes $O(1)$ time. Thus, the overall subroutine takes $O(N)$ time.

To determine whether any 3 (or more lines) intersect at a point, run the previous subroutine N times, once for each line i . The total running time is $O(N^2)$.

(c) Only print out a set of lines in the last step of the subroutine if the index of the first line in the list is greater than i . This guarantees we only find a set of lines once, when using the line with the smallest index as the base line.