

## Data Structures

### Prof. Loftin: Practice Test Problems for the Material after Test 2

## SOLUTIONS

1. Consider the following array of integers:

13 5 30 12 15 4 10 6 0

- (a) Trace through the above array using selection sort.

**Solution:**

13 5 30 12 15 4 10 6 0  
13 5 0 12 15 4 10 6 30  
13 5 0 12 6 4 10 15 30  
10 5 0 12 6 4 13 15 30  
10 5 0 4 6 12 13 15 30  
6 5 0 4 10 12 13 15 30  
4 5 0 6 10 12 13 15 30  
4 0 5 6 10 12 13 15 30  
0 4 5 6 10 12 13 15 30

- (b) Trace through the above array using insertion sort.

**Solution:**

13 5 30 12 15 4 10 6 0  
5 13 30 12 15 4 10 6 0  
5 13 30 12 15 4 10 6 0  
5 12 13 30 15 4 10 6 0  
5 12 13 15 30 4 10 6 0  
4 5 12 13 15 30 10 6 0  
4 5 10 12 13 15 30 6 0  
4 5 6 10 12 13 15 30 0  
0 4 5 6 10 12 13 15 30

- (c) Trace through the above array using bubble sort.

**Solution:**

Pass 1:

13 5 30 12 15 4 10 6 0  
5 13 30 12 15 4 10 6 0  
5 13 30 12 15 4 10 6 0  
5 13 12 30 15 4 10 6 0  
5 13 12 15 30 4 10 6 0  
5 13 12 15 4 30 10 6 0  
5 13 12 15 4 10 30 6 0  
5 13 12 15 4 10 6 30 0  
5 13 12 15 4 10 6 0 30

Pass 2:

5 13 12 15 4 10 6 0 30  
5 13 12 15 4 10 6 0 30  
5 12 13 15 4 10 6 0 30  
5 12 13 15 4 10 6 0 30  
5 12 13 4 15 10 6 0 30  
5 12 13 4 10 15 6 0 30  
5 12 13 4 10 6 15 0 30  
5 12 13 4 10 6 0 15 30

Pass 3:

5 12 13 4 10 6 0 15 30  
5 12 13 4 10 6 0 15 30  
5 12 13 4 10 6 0 15 30  
5 12 4 13 10 6 0 15 30  
5 12 4 10 13 6 0 15 30  
5 12 4 10 6 13 0 15 30  
5 12 4 10 6 0 13 15 30

Pass 4:

5 12 4 10 6 0 13 15 30  
5 12 4 10 6 0 13 15 30  
5 4 12 10 6 0 13 15 30  
5 4 10 12 6 0 13 15 30  
5 4 10 6 12 0 13 15 30  
5 4 10 6 0 12 13 15 30

Pass 5:

5 4 10 6 0 12 13 15 30  
4 5 10 6 0 12 13 15 30  
4 5 10 6 0 12 13 15 30

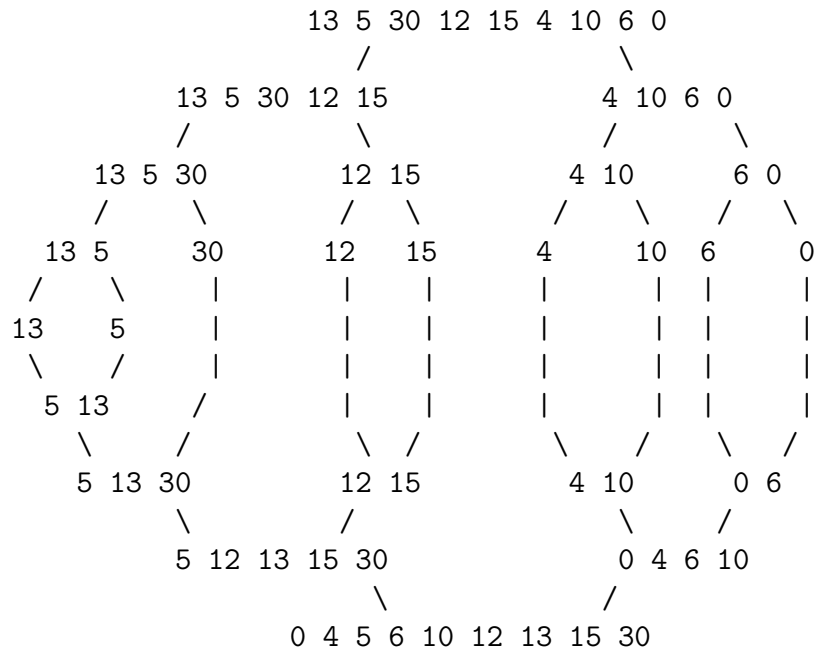
```

4 5 6 10 0 12 13 15 30
4 5 6 0 10 12 13 15 30
Pass 6:
4 5 6 0 10 12 13 15 30
4 5 6 0 10 12 13 15 30
4 5 6 0 10 12 13 15 30
4 5 0 6 10 12 13 15 30
Pass 7:
4 5 0 6 10 12 13 15 30
4 5 0 6 10 12 13 15 30
4 0 5 6 10 12 13 15 30
Pass 8:
4 0 5 6 10 12 13 15 30
0 4 5 6 10 12 13 15 30

```

(d) Trace through the above array using mergesort.

**Solution:**



(e) Trace through the above array using quicksort (with the first element as the pivot).

**Solution:**

```

13 5 30 12 15 4 10 6 0
sort via pivot 13
13 5 12 4 10 6 0 30 15
swap pivot 13
0 5 12 4 10 6 13 30 15
subarrays:
0 5 12 4 10 6          30 15
sort via pivot 0        sort via pivot 30
0 5 12 4 10 6          30 15
swap pivot 0            swap pivot 30
0 5 12 4 10 6          15 30
subarray:                (done)
5 12 4 10 6
sort via pivot 5
5 4 12 10 6
swap pivot 5
4 5 12 10 6
subarrays:
4          12 10 6
(done) sort via pivot 12
12 10 6
swap pivot 12
6 10 12
subarray:
6 10
sort via pivot 6
6 10
swap pivot 6
6 10
(done)
all together:
0 4 5 6 10 13 15 30

```

- (f) Trace through the above array using radix sort (in particular, the data should all be 2-digit numbers for radix sort to work. How should this be achieved?)

**Solution:**

Fill out the first digit with 0 if necessary:

13 05 30 12 15 04 10 06 00

First group via 1's digit:

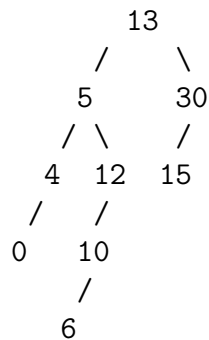
0:            2:    3:    4:    5:            6:  
 (30 10 00) (12) (13) (04) (05 15) (06)  
 30 10 00 12 13 04 05 15 06

Now group via 10's digit:

0:                    1:                    3:  
 (00 04 05 06) (10 12 13 15) (30)  
 00 04 05 06 10 12 13 15 30

- (g) Place the elements of this array, in order provided, into an initially empty binary search tree.

**Solution:**



2. True/False: Performing mergesort on an array of  $n$  elements always takes  $O(n * \log_2 n)$  steps.

**Solution:** True. The technique for mergesort always splits the array in half until the subarrays are length 1 ( $O \log_2 n$  levels), and then are merged as sorted lists (each level  $O(n)$  times). There is no shortcut or any difference for a worst case. So the mergesort always takes  $O(n * \log_2 n)$  steps.

3. If an array of  $n$  elements is already sorted in ascending order, how many steps (in terms of  $O$  notation) does bubble sort take? Answer the same question for selection sort, mergesort and quicksort.

**Solution:** The bubble sort only takes  $O(n)$  steps to sort an already sorted array. This is because the first pass verifies that each pair of

elements  $\text{ar}[i]$  and  $\text{ar}[i+1]$  are already sorted in the correct order. Then the bubble sort exits. The number of comparisons needed is  $n - 1 = O(n)$ .

For selection sort, there is no shortcut: at each pass, the largest element of the subarray from  $\text{ar}[0..i]$  is chosen and swapped with the element  $\text{ar}[i]$  (this always takes  $O(i)$  steps), while  $i$  goes from  $n - 1$  to 1. All together, this is  $O(n^2)$  steps, with no shortcuts.

For mergesort, the algorithm always takes  $O(n \log_2 n)$  steps, as there is no difference in the algorithm depending on whether the elements are already sorted.

For quicksort (with the first element chosen as pivot), an already sorted array takes  $O(n^2)$  steps. For an initial subarray of length  $i$ , when the pivot is chosen, the left subarray is empty, while the right subarray has length  $i - 1$ . This means it takes  $O(n)$  full passes to perform quicksort, since maximum length of the subarray only goes down by 1 each time from  $n, n - 1, n - 2, \dots, 1$ . Each pass takes  $O(i)$  times, and in total, there are about

$$n + (n - 1) + \dots + 2 + 1 = O(n^2)$$

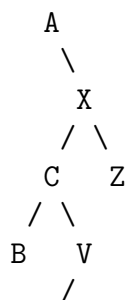
steps for quicksort in this case.

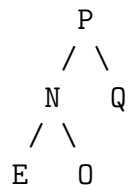
4. Consider the following array of letters:

A X Z C V P N O Q E B

- (a) In the order provided, place the elements of this array into an initially empty binary search tree.

**Solution:**





(b) For the tree in part (a), what is the result of a preorder traversal?

**Solution:** A X C B V P N E O Q Z

(c) For the tree in part (a), what is the result of a postorder traversal?

**Solution:** B E O N Q P V C Z X A

5. Write the code for a method

```
public void bubblesort (Comparable[] ar)
```

which uses bubble sort to sort the array ar.

**Solution:**

```
public void bubblesort (Comparable[] ar){
    for (int i=ar.length-2; i>=0; i--){
        boolean sorted = true;
        for (int j=0; j<=i; j++){
            if (ar[j].compareTo(ar[j+1]) > 0){
                sorted = false;
                Comparable temp = ar[j];
                ar[j] = ar[j+1];
                ar[j+1] = temp;
            } // end if
        } // end for j
        if (sorted)
            return;
    } // end for i
} // end bubblesort
```

6. Consider a class BinaryTree which has as data

```
private TreeNode<E> root;
```

From inside this class, write a method

```
public void printPostorder()
```

which prints to the screen the results of a postorder traversal of the tree. (Assume the `TreeNode` class has the usual methods `getLeft()` and `getRight()` to access the left and right child nodes.) You may also write any auxiliary methods as you see fit.

**Solution:**

```
public void printPostorder(){
    printPostorder(root);
}
private void printPostorder(TreeNode<E> tNode){
    if (tNode != null){
        printPostorder (tNode.getLeft());
        printPostorder (tNode.getRight());
        System.out.println (tNode.getItem());
    }
}
```