CSCI 200

HW #3: Quick-sort, Radix sort and search trees

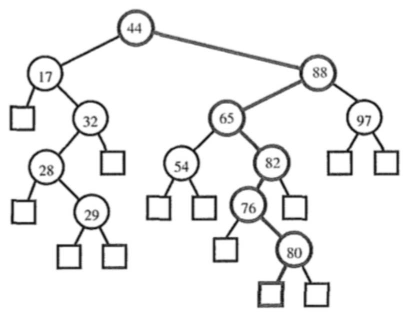
Directions: *Complete your work on a separate sheet of paper. Submit the physical copy of your work at the beginning of class. You may work in groups of up to 3 students provided that all students participate in each question. Provide a short preliminary explanation of how an algorithm works before running an algorithm or presenting a formal algorithm description, and use examples or diagrams if they are needed to make your presentation clear.*

1. (a) What is the running time of the version of quick-sort that uses the element at rank $\lfloor n/2 \rfloor$ as the pivot, provided that the input sequence is already sorted? Explain.
   (b) Does the running time of radix-sort depend on the order of keys in the input? Explain.

2. Suppose you are given the following sorted array: $A = [1, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584]$. Illustrate the execution of binary search for the number 148.

3. Insert into an initially empty binary search tree items with the following keys (in this order): 30, 40, 23, 58, 48, 26, 11, 13. Draw the single resulting tree after all insertions have been performed.

4. Remove from the binary search tree given below the following keys (in this order): 32, 65, 76, 88, 97. Draw the tree after **each** successive removal (5 trees total).



5. A different binary search tree results when we try to insert the same sequence into an empty BST in a different order. Give an example of this with at least 5 elements and show the two different binary search trees that result. Specify the order in which the items re inserted to produce the two different trees.

6. Give an algorithm that runs in $O(\log n)$ time which takes a sorted array A and two keys, x and z, which may or may not be elements of A. The algorithm should return the number of elements y in A which satisfy $x \leq y \leq z$.

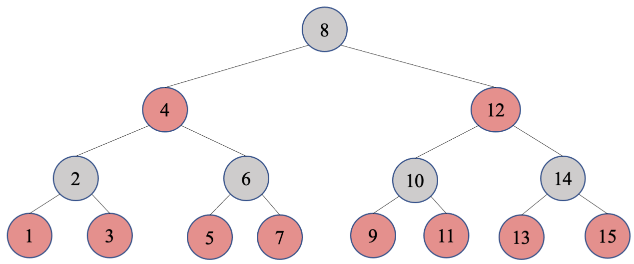
7. Suppose that we have the numbers between 1 and 1000 in a binary search tree, and we want to search for the number 363. For each of the following sequences, indicate yes/no whether it could be the sequence of nodes examined in searching for the number 363. If not, explain.
   (a) 2, 252, 401, 398, 330, 344, 397, 363
   (b) 924, 220, 911, 244, 898, 258, 362, 363
   (c) 925, 202, 911, 240, 912, 245, 363
   (d) 2, 399, 387, 219, 266, 382, 381, 278, 363
   (e) 935, 278, 347, 621, 299, 392, 358, 363

8. Let T be a binary search tree, and let x be a key. Give an efficient algorithm for finding the smallest key y in T such that $y > x$. Note that x may or may not be in T. Explain why your algorithm has the running time it does.

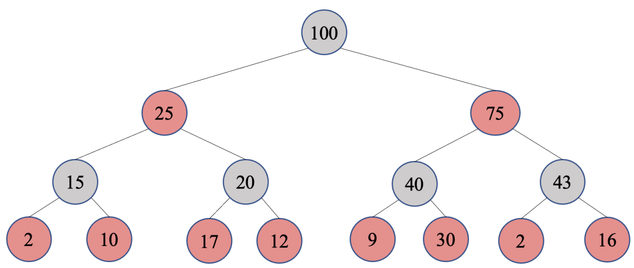
9. Design and give the **pseudocode** for an $O(\log n)$ algorithm that determines whether a red-black tree with n keys stores any keys within a certain (closed) interval. That is, the input to the algorithm is a red-black tree T and two keys, l and r, where $l \leq r$. If T has at least one key k such that $l \leq k \leq r$, then the algorithm returns true, otherwise it returns false. *Hint:* You can use the recursive or iterative TREE-SEARCH algorithm (CLRS 12.2) as a subroutine.

10. The NIL black leaf is omitted from the visualization in each of the trees shown below. For each tree, specify whether it is a red black tree (yes) or it is not a red black tree (no). If not, explain.

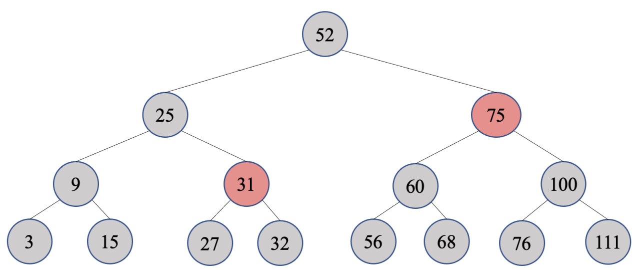
(a)



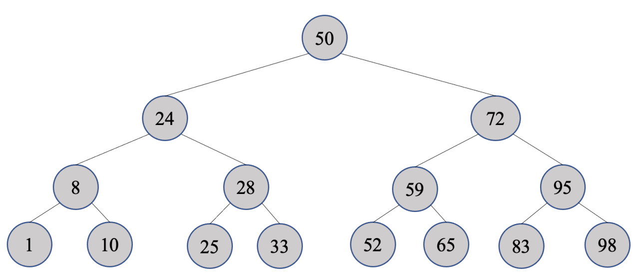
(b)



(c)



(d)



(e)