Computer Science 62

Papoutsaki/Devanny - Fall '18

Midterm II Examination

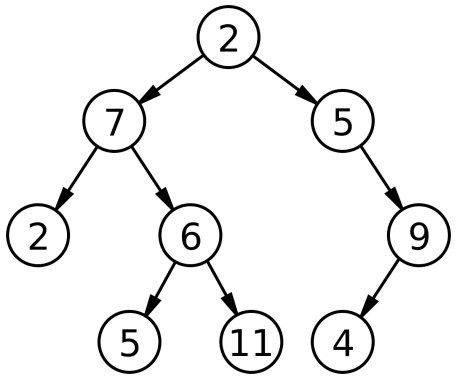
November 14, 2018

Question	Points	Score
1	11	____
2	12	____
3	10	____
4	15	____
5	20	____
6	15	____
7	8	____
8	9	____
TOTAL	100	____

Your name (Please print)

_____

1. *Binary Trees* (11 pts)
   Please write a static recursive method that adds up the values of all of the leaf nodes in a tree given as a parameter with type `BinaryTree<Integer>`. Relevant methods from the class `BinaryTree<Integer>` are: `isEmpty()`, `value()`, `left()`, and `right()`. For example, `sumLeafTree` called on the following `BinaryTree` should return 22.



SOLUTION:

```
public static int sumLeafTree(BinaryTree<Integer> root) {









}
```

2. *Heaps* (12 pts)

a. (3 pts) The following array is a representation of a min-heap:

0	1	2	3	4	5	6	7	8	9	10	11
2	3	5	10	8	7	22	11	13	20	24	16

Please draw the tree that it represents:

b. (3 pts) Suppose this heap is being used to represent a Priority Queue. Please illustrate (via a sequence of pictures) the steps that the computer must go through to add a new element 1 to the heap, and restore the remaining elements to a heap. *Your pictures should be of trees rather than arrays and the algorithm illustrated must be the one covered in class to obtain an efficient solution*.

c. (3 pts) What is the worst-case time complexity of adding an element to a Priority Queue (and restoring it to a heap) with this heap representation if there are n elements in the heap before the addition takes place (use "O" notation). Explain your answer.

d. (3 pts) Please explain why this heap implementation is better overall than using our linked list implementation of regular queues and keeping the elements in order by priority.

3. *Short answers* (10 pts, 2 points each)

a) Describe an advantage of external chaining over the open addressing scheme and vice-versa.

b) Describe a circumstance in which it might be more desirable to use heapsort rather than quicksort. Give a brief justification for your answer.

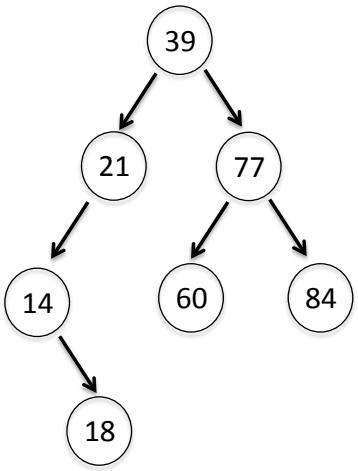
c) In an array implementation of trees, what is the upper bound on the amount of space necessary to represent a general binary tree with n nodes. Use big-"O" notation.

d) What is the complexity of a search in a binary search tree in the worst case?

e) What is the complexity of a search in a splay tree in the average case?

4. *Binary Search Trees* (15 pts)

Please write a static method `countRange` that takes a binary search tree (BST) of integers and returns the number of integers in the tree that fall in the range [`low, high`], that is, the number of integers that are greater than or equal to `low` and less than or equal to `high`. In the BST shown below, `countRange` would return 3 for `low`=14 and `high`=30 (corresponding to 21, 14, and 18).



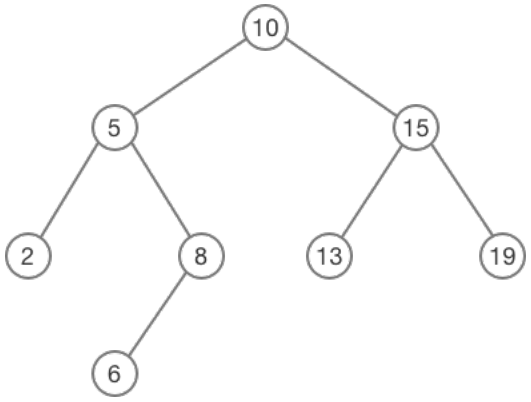
Useful methods from the class `BinaryTree<Integer>` include: `isEmpty(), value(), left(), and right()`.Your solution should take advantage of the fact that this is a binary search tree to ignore nodes that cannot contribute to the count.

```
public static int countRange(BinaryTree<Integer> tree, int
low, int high) {




}
```

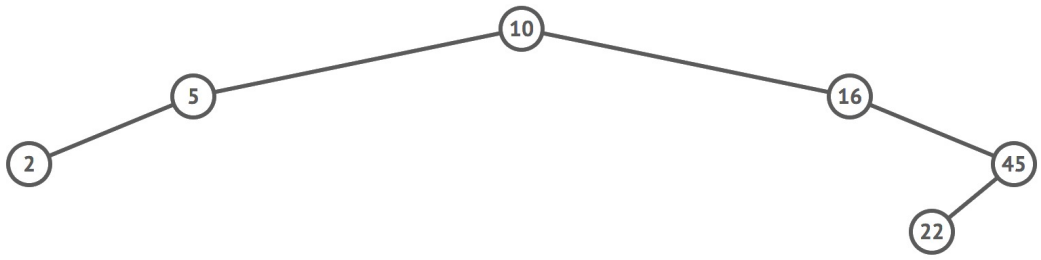
5. *Binary Search Trees and Splay Trees* (20 pts)

a) (4 pts) *Insert*, into an empty **binary search tree**, entries with keys 30, 40, 24, 58, 47, 26, 25, 11, 13 (in this order). You can draw the tree after each insertion or just the final result.

b) (4 pts) Given the following tree, *delete* the node 10 using the algorithm discussed in class and show the binary search tree after the deletion.



c) (4 pts) *Insert*, into the following **splay tree**, the node 23. Draw the tree after the insertion and name the operations you performed to adjust the tree.



d) (4 pts) *Locate*, in the splay tree given in part (c), the node 22. Draw the tree after you have located the node and name the operations you performed to adjust the tree.

e) (4 pts) *Delete*, in the splay tree given in part (c), the node 22. Draw the tree after you have deleted the node and name the operations you performed to adjust the tree.

6. *Hashing* (15 pts)

a) (10 pts) Consider inserting the keys 26, 47, 4, 41, 15, 32, 7, 25, 11, 30 into a hash table of size $m = 13$ using open addressing with the hash function $h(k) = k \bmod m$, and quadratic probing with $h(k,i) = (h(k) + i^2 + i) \bmod m$. Fill in the following hash table:

0	1	2	3	4	5	6	7	8	9	10	11	12

b) (5 pts) Now assume that instead you are using external chaining. Show what the hash table will look like.

0	1	2	3	4	5	6	7	8	9	10	11	12

7. *Various data structures* (8 pts, 2 each)

For each of the following situations choose which data structure would be most appropriate and give a short explanation why. Choose from binary tree, binary search tree, binary heap, and hash table. Each data structure is used once.

a) Alice is writing software for a special printer designed to minimize the time people spend waiting for their documents. When several people request to print documents, she wants the printer to choose to print the document with the fewest number of pages.

b) Bob is creating an AI for a two-player game where turns alternate and each player can play one of two moves when it is their turn. He needs a data structure to help the AI keep track of what moves can be played in each game position.

c) Carol is implementing a school directory for Pomona's webpage. Carol needs a data structure that will allow her to print out a list of every student sorted by name, but to also support a search feature where users can search for a specific student.

d) Dave wants to have both a free and paid version of his web application. He is looking for a data structure that will allow him to given a username efficiently look up if that user has the free or paid version of the application.

8. *Ordered Structures* (9 pts)

a. Please explain what is the consequence of having a class `C` declared to implement the functional interface `Comparable<C>`. I.e., what does that guarantee about `C`.

b. Suppose I have a list of items that I might want to order in two or more different ways (for example, sorting associations by key or by value). What kind of functional interface should I use to accomplish this? What does this guarantee about my class?