CS1102: Data Structure and Algorithms

Tutorial 8: Trees

Week of 22 March 2010

1. Binary Search Tree Operations:
   (a) Draw the binary search tree after inserting the following integers in sequence: 20, 10, 5, 7, 15, 12, 18, 30, 25, 37, and 40.
   (b) What are the corresponding output using (i) pre-order, (ii) in-order, (iii) post-order, (iv) level-order traversal?
   (c) Draw the new binary search tree after the following operations: delete 30, delete 10, delete15 (You should follow the deletion algorithm in the lecture notes). Is the new tree a complete binary tree?

2. Suppose you have a **binary tree** T containing only distinct items. The pre-order sequence of T is 8, 7, 5, 28, 4, 9, 18, 17, 16, and the in-order sequence of T is 5, 7, 4, 28, 9, 8, 18, 16, and 17.
   (a) Can you reconstruct a unique T from these two traversal sequences? If yes, reconstruct it.
   (b) Implement the algorithm if the above construction is possible.

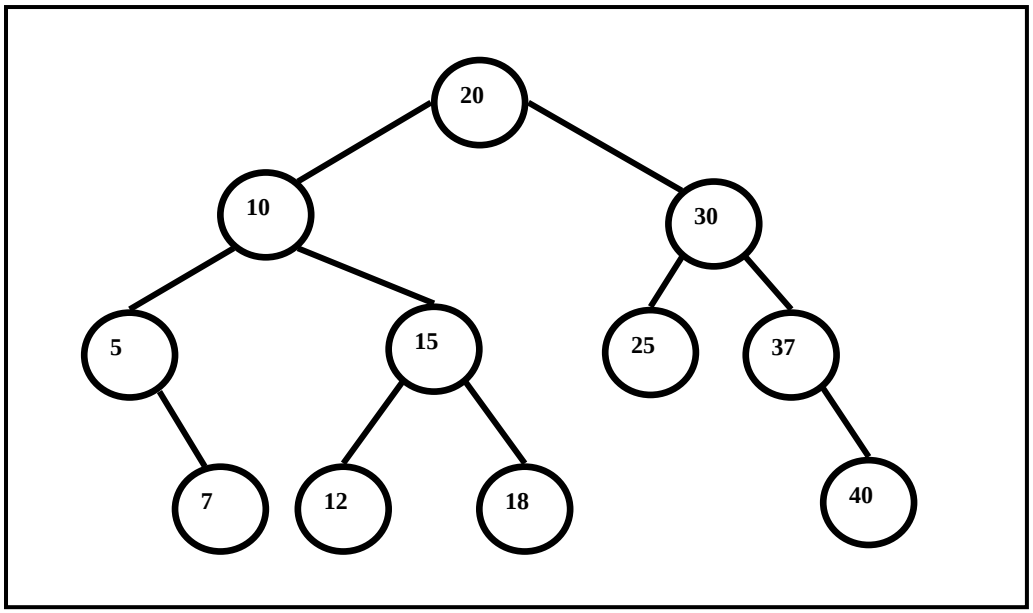
```
TreeNode constructTree(int[] inOrder, int[] preOrder, int n)
{
    //n is the number of distinct items
}
```

   (c) If you are given the pre-order and post-order sequences of a binary tree, can you construct a unique binary tree? Explain your answer.

3. Another way of representing a binary search tree is to use an array. The items in the tree are assigned to locations in the array in a level-order fashion. For example, the figure below shows an array representation of the binary search tree in question 1.
   We assume there is no repeating value in the tree and there is no node with value 0, so we use 0 to represent non-existing node.

Value	20	10	30	5	15	25	37	0	7	12	18	0	0	0	40
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

CS1102: Data Structure and Algorithms




```
public class BinaryTree {
	int[] tree;
	int MAXNUM=15;

	public BinaryTree(){
		tree= new int[MAXNUM];
	}
	public void insert(int i)
	public int search(int i)
	//return the index of integer i, -1 if it is not in the BST
	public int findMin()
	//return the minimum value in the tree
}
```

(a) Implement the recursive insertion method of binary search tree.
(b) Implement the recursive search method of binary search tree.
(c) Implement the finding minimum method of binary search tree.
(d) What are the advantages and disadvantages of array representation?

4. When deciphering ancient languages, historians often take advantage of the fact that some words are more commonly repeated than others, for example the word “I” or “We” are often repeated more than others in the English language. Hence, this idea can be used to decipher words in long texts. For example, the long text “CATTCAT” will produce the following possible words:
   Length 1: C (2 times), A (2 times), T (3 times)
   Length 2: CA (2 times), AT (2 times), TT (1 time), TC (1 time)
   Length 3: CAT (2 times), ATT (1 time), TTC (1 time), TCA (1 time)
   Length 4: …
   Length 5: …
   Length 6: CATTCA (1 time), ATTCAT (1 time)
   Length 7: CATTCAT (1 time).

   (a) Given a long string S, design an ADT to facilitate the query of the occurrence of all the possible words of length $l \leq$ length(S). (Just write down the required method headers, no need to write down their implementations) In the above example, the occurrence of “CAT” is equal to 2.

   (b) Describe the underlying data structure you would use to implement this ADT, and analyze the complexity of each method you designed in the ADT. With your data structure, is the time complexity of getting the word occurrence dependent on the number of possible words *n*? If yes, can you think of a data structure that makes the query of word occurrence independent of *n*? You may use diagrams to explain your solution. You may not use Hash tables or maps.