CS 2210 Data Structures and Algorithms

Assignment 3 (20 marks).

Due: November 8, 2024, at 11:55 pm.

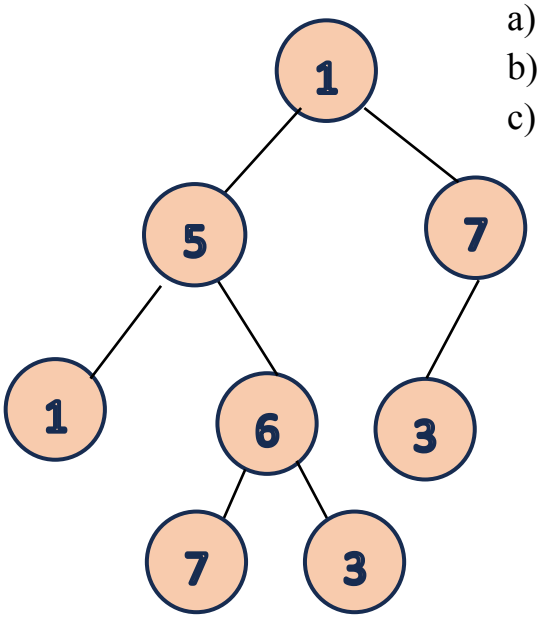
Important: No late concept assignment will be accepted

Please submit through **Gradescope** a pdf file with your solution to the assignment. You are s**trongly encouraged to type** your answers. If you decide to submit hand-written answers to the questions, please make sure that the TA will be able to read your solutions. If the TA cannot read your answers, you will not be given credit for them. Remember that concept assignments must be submitted by the due date; **no late concept assignments will be accepted** unless you have an academic accommodation.

---

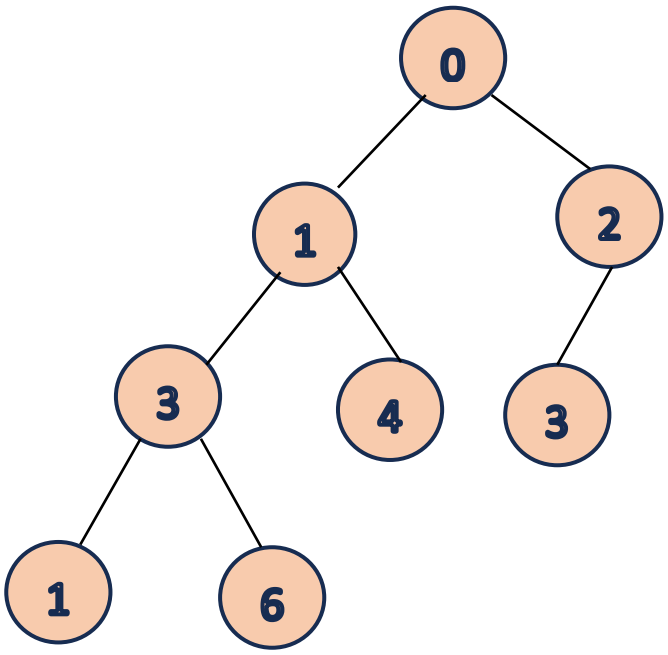
1. Consider a hash table of size M = 11 where we are going to store integer key values. The hash function is h(k) = k mod 11. Draw the table that results after inserting, in the given order, the following values: 21, 45, 56, 90, 1, 38, 67.
   i. (1 mark) Assume that collisions are handled by separate chaining.
   ii. (1 mark) Assuming collisions are handled by linear probing.
   iii. (2 marks) Assuming collisions are handled by double hashing, using a secondary hash function h′(k) = 7 − (k mod 7 + 1).

2. Given the following Binary Tree, list the order of nodes,
   a) (1 mark) for a pre-order traversal.
   b) (1 mark) for an in-order traversal
   c) (1 mark) for a post-order traversal



3. Given the pre-order traversal of any Binary Search Tree (BST),
   i. (4 marks) Write an algorithm to construct the Binary Search Tree
   ii. (1 mark) Calculate the time complexity of the algorithm. (Remember to include the number of operations per node)
   iii. (2 marks) Apply your algorithm on the pre-order traversal of a BST given below and draw the BST constructed according to your algorithm.
   List of nodes in the order Pre-order traversal – 10, 3, 1, 7, 5, 8, 11, 13, 12

4. Given any Binary Tree and any integer k,
   i. (5 marks) Write an algorithm "CountPaths" to count the number of paths from root to leaf that sum to k.
   For the below Binary tree when the value of k = 5 CountPaths will return 3, when the value of k = 2, CountPaths will return 0 and for k = 6, CountPaths will return 0.
   ii. (1 mark) Calculate the time complexity of the algorithm. (Remember to include the number of operations per node)



1a) $M=11$, $h(k) = k \bmod 11$, 21, 45, 56, 90, 1, 38, 67

0
1 → 45 → 56 → 1 → 67 ⊥
2 → 90 ⊥
3
4
5 → 38 ⊥
6
7
8
9
10 → 21 ⊥

b)

0	
1	45
2	56
3	90
4	1
5	38
6	67
7	
8	
9	
10	21

c)

0	
1	45
2	90
3	67
4	
5	38
6	1
7	56
8	
9	
10	21

2a) pre-order (root, L, R)

1 5 1 6 7 3 7 3

2b) in-order (L, root, R)

1 5 7 6 3 1 3 7

2c) Post-order (L, R, root)

1 7 3 6 5 3 7 1

3a) let A be array pre-order of BST. (root, left, right)

```
Function createTree(r, A)
  if not r or not A[0], return

  r.key = A[0]

  leftArr = [], rightArr = []

  Loop i from 1 to A:
    if A[i] > A[0], rightArr.append(A[i])
    else leftArr.append(A[i])

  CreateTree(r.left, leftArr)
  CreateTree(r.right, rightArr)
```

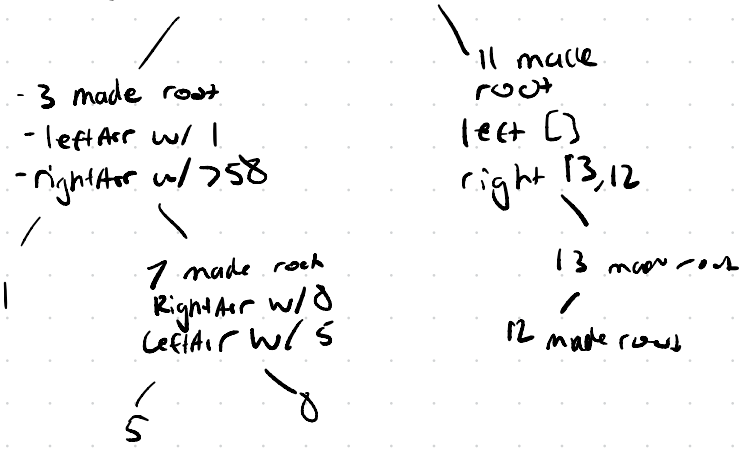
3b) Num operations per node is nC_1. All nodes are traversed, so TC is $O(C_1 n) \cdot O(n)$ where n is size of A.

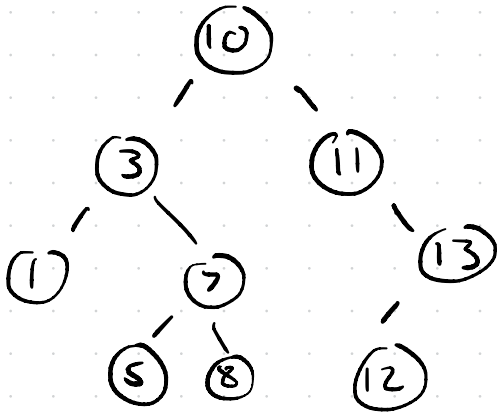
$O(C_1 n) \cdot O(n)$

$= O(n^2)$

3c) 10, 3, 1, 7, 5, 8, 11, 13, 12

- 10 made root
- leftArr with 3, 1, 7, 5, 8
- rightArr with 11, 13, 12





4a) number paths from root to leaf that sum to k

need DFS

Function count(r, k)

C_1
- if not r, return 0
- if k == 0 and r.isLeaf()
  return 1

return count(r.left, k - r.key) + count(r.right, k - r.key)

4b) All nodes are traversed

$O(n \cdot C_1) = O(n)$