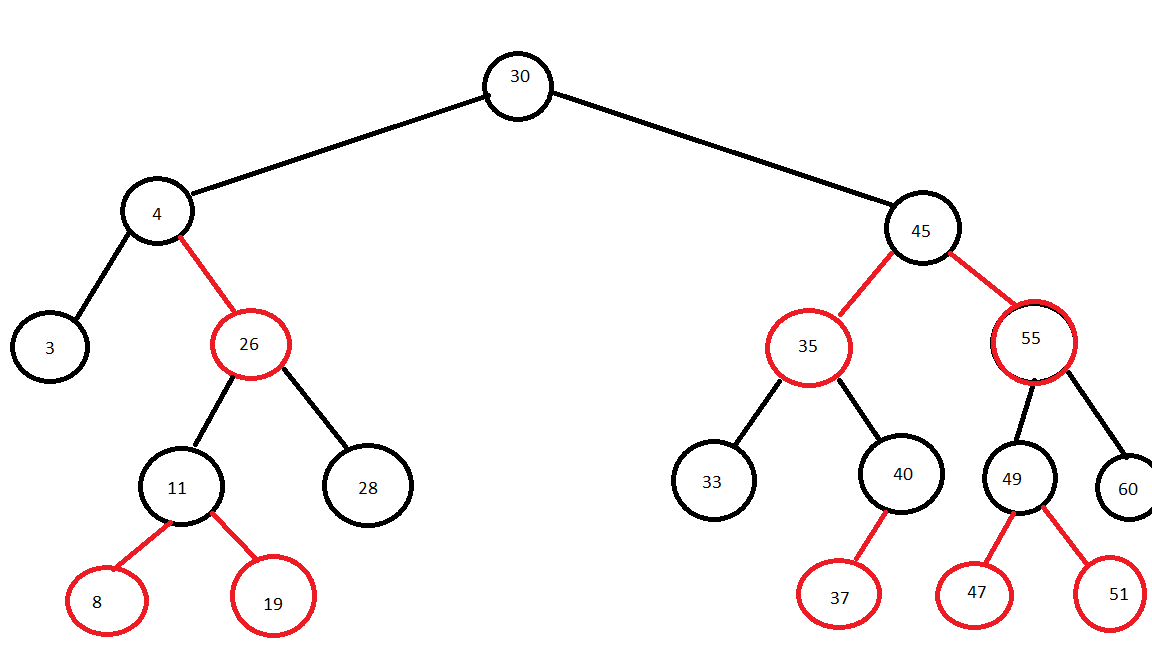
**Muhammad Arham Khan – 21701848**

**Spring 2019**

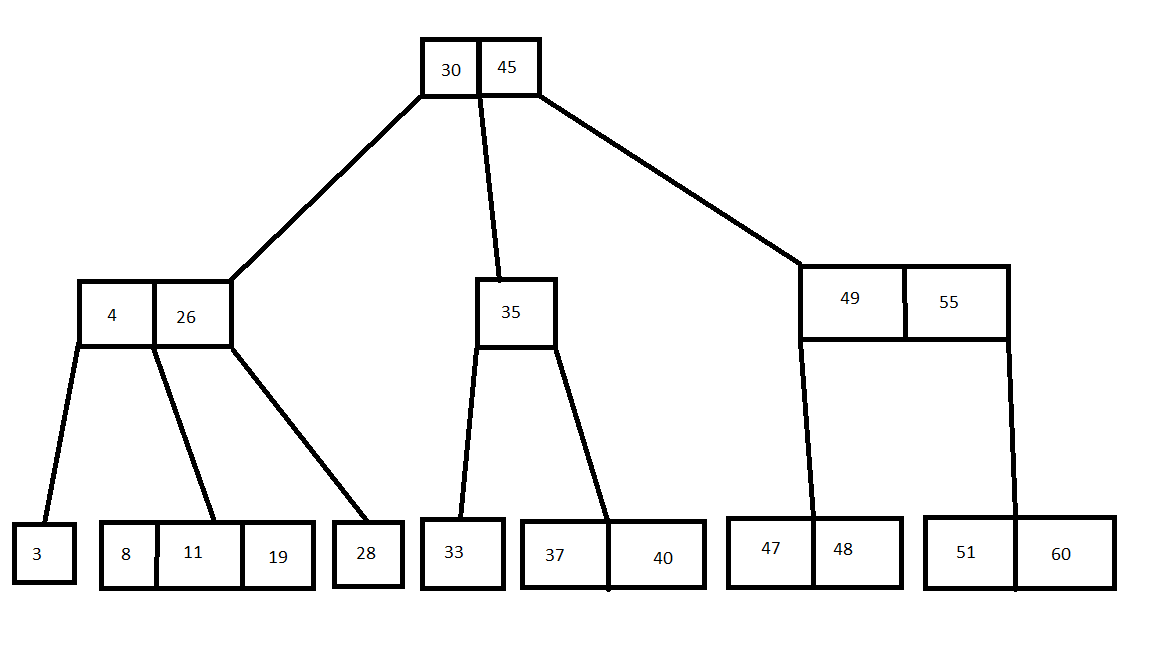
**CS 202 – Section 03**

**HOMEWORK 4**

**Question 1 (a):**

****

**Question 1 (b):**

****

**Question 2:**

|  |  |  |
| --- | --- | --- |
| **Data structure** | **insert** | **extractMin** |
| Unsorted Array | O (1) | O (n) |
| Red-Black Tree | O (log (n)) | O ( log(n) ) |
| Hashing | O (1) | O (n) |
| Min-Heap | O (log(n)) | O (log(n)) |
| Sorted Linked List | O (n) | O (1) |

**Question 3 (a):**

To find maximum number of keys in 2-3 tree, we select a tree with 3-nodes only. So for a 2-3 tree of height h, the maximum possible number of nodes is.

n = 3h – 1 / 2

So since each node is a 3-node and each node has 2 keys, So the maximum number of keys (k) that it can hold is:

k = 3h – 1

**Question 3 (b):**

Every subtree of a red-black tree is not a red-black tree. So the answer is no.

Considering that each subtree of the red-black tree is red-black tree and then consider a subtree with a red root. Since this subtree contradicts with the definition of a red-black tree, so every subtree of a red-black tree cannot be red-black.

**Question 3 (c):**

To solve this, I use a hash table. First, insert all elements from array into hash table which takes O (n) time. Now we try finding for each element in the array another element which whose sum is the target sum’s value.

To check this, I take the array element and then subtract it from the target sum value and then try finding the result in the hash table. Since searching in hash table once in:

O (1)

the total run time for this loop is:

O(n)

So the time complexity of this solution is also:

O(n)