***Name: Hassan Raza***

***ID: 21701811***

***Section: 1***

***CS202 Spring 2019***

*Q1)(a): Equivalent red-black tree of the 2-3-4 tree:*

*Q1)(b): Final resulting 2-3-4 tree after inserting 48:*

30

60

51

47

40

33

28

19

11

3

55

45

35

26

4

37

8

49

*Q2) Fill in the below table appropriately by giving expected running times in big-O notation:*

|  |  |  |
| --- | --- | --- |
| **Data structure** | **Insert** | **Extract min** |
| 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) |

*Q3)(a): What is the maximum number of keys that a 2-3 tree of height h can hold?*

We will assume that every node has three children as we have to get the maximum number of keys. We will get the maximum number of nodes for height of h which is 3h-1/2. As we have two keys at each node we will multiply our answer by 2 giving answer:

max number of keys = 3h-1

*Q3)(b): Is every subtree of a red-black tree also a red-black tree?*

No, it is not necessary for every sub-tree to be a red-black tree. It is a condition of red black tree’s root must be a black node, however this condition may not be satisfied while taking a subtree of a red-black tree.

*Q3)(c): Algorithm:*

We will put all of our elements into a hash table and then look in the table if two elements give us the required sum. Putting all elements into hash table is O(n). Then for each element we will see if it contains element for which sum is equal to required number, if it is same we exit returning the two numbers. This operation also is O(n).