Recitation 8 Spring 2023 (Student Version)

**Heaps, B-Trees, Red-Black Trees**

1. **[5 minutes]** True/False
   1. In a 2-3-4 Tree, insertion can be performed on any node so long as the other nodes are adjusted accordingly FALSE
   2. Deleting the highest priority element and reorganizing the heap will only take O(log n) time. TRUE
   3. A red-black tree may have all red nodes. FALSE
   4. The following array is a max-heap: TRUE

Int maxHeap[] = {10,5,3,4,1}; ROOT INDEX 0 🡪 LEFT: 2i + 1 , RIGHT: 2i + 2

1. **[5 minutes]** Short Answer
   1. What are the steps for Depth-First Traversal in a Binary Tree?

Left – right – root (postorder)

Recursively traverse left sub

Recursively traverse right sub

* 1. Consider an R-B tree with a black height of 5. In any given path from root to leaf, give both the minimum and maximum amount of total nodes within the path (Root = 1 Node, Leaf = 1 Node).

Min = 6 (5 + 1)

Max = (6 \* 2) = 12

* 1. **[5 minutes]** What is the max height of a Red-Black Tree with 6 nodes? Draw an example of this Red-Black Tree.



max height= 2



* 1. **[7 minutes]** Insert 2, 1, 4, 5, 9, 3, 6, 7 into an initially empty red-black tree



* 1. **[15 minutes]** Write a recursive function to check whether or not a given tree represents a max heap. You are permitted to use the following methods: getLeftChild(),

getRightChild(), getData(). Use the following function declaration and assume the first call will pass in the root for the Node argument and -1 for the parentData argument.

public boolean isMaxHeap(Node root, int parentData){

if(currentNode == null){

return true;

}

If(parentVal != null && currNode.val >= parentVal){

return false;

}

Return (isMaxHeap(leftChild, currentNode.val) && isMaxHeap(rightChild, currentNode.val));

}

7. **[10 Minutes]** Fill in the chart below with the worst-case time complexity for each of the following operations:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Search | Insert | Delete |
| B-Trees | O(log n) | O(log n) | O(log n) |
| AVL Tree | O(log n) | O(log n) | O(log n) |
| Red Black Tree | O(log n) | O(log n) | O(log n) |
|  | Peek | Push | Pop |
| Heap | O(1) | O(log n) | O(log n) |