York University
EECS 4101/5101, Winter 2023
Assignment 3

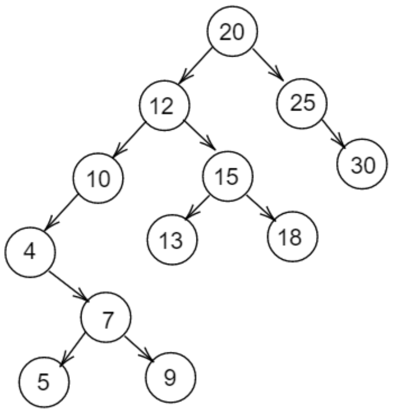
Due Date: March 2nd, at 23:59

Not being heard is no reason for silence ...
Victor Hugo

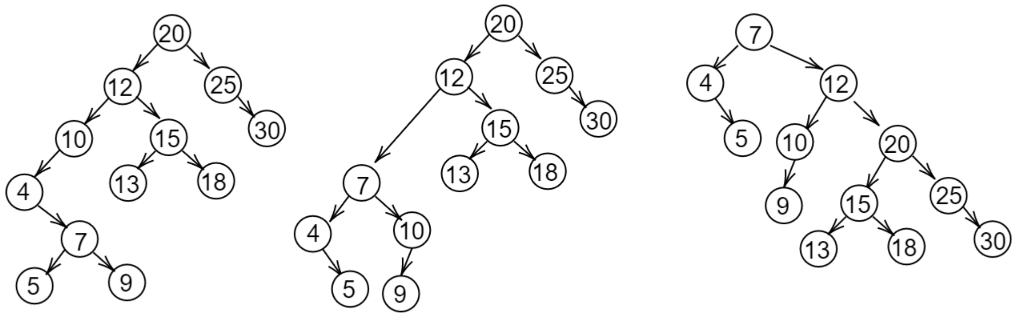
All problems are written problems; submit your solutions electronically **only via Crowdmark**. You are welcome to discuss the general idea of the problems with other students. However, you must write your answers individually and mention your peers (with whom you discussed the problems) in your solution. Please refer to the course webpage for guidelines on academic integrity.

Problem 1 Splay Trees [15 marks]

a) Apply the splay operation on the following splay tree when there is a request to node '7'. Show your steps.



Answer: See belows:



b) Prove or disprove the following statement: “the root of a splay tree always has two children”.

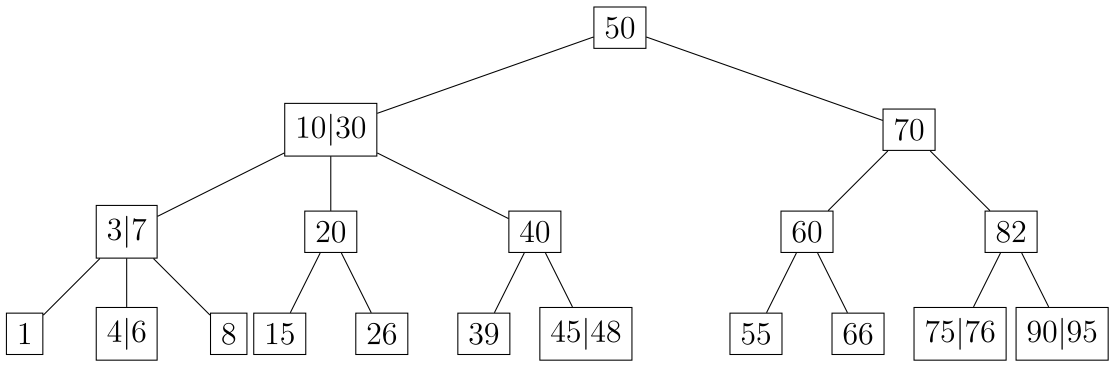
Answer: This is wrong. After access to the largest/smallest item, it becomes the root, with no node on its right/left.

c) Prove or disprove: in any splay tree formed by N nodes, there is a sequence of requests with length N that results in the tree having height $\Omega(N)$.

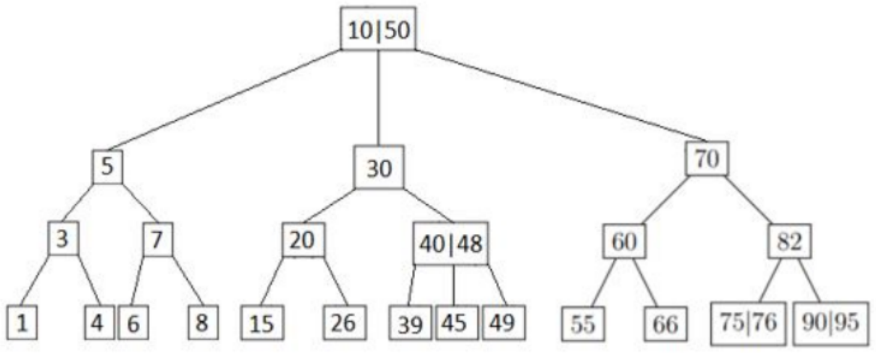
Answer: Consider a sequence of requests $a_1, a_2, \ldots, a_N$ to items in the sorted order (we have $a_1 < a_2 < \ldots < a_N$). After the first request, a_1 will be on the root, and all items will be on its right. After the second request, a_2 will be the root, and a_1 will be on its left. After the third request, a_3 will be the root, and a_2 and a_1 will form a path on its left. Similarly, after requesting a_N, all smaller vertices $a_1, a_2, \ldots, a_N$ will form a path of length N, which has height $\Omega(N)$.

Problem 2 2-3 Trees [10 marks]

This problem will concern operations on the 2-3 tree T shown in the following figure.

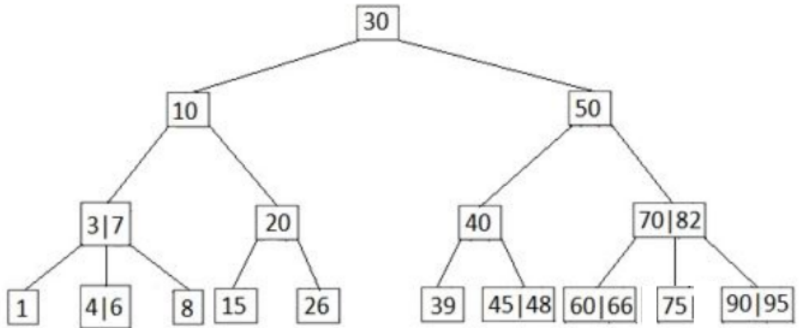


a) Draw the tree after performing operations insert(49) and insert(5). **Answer:** See the following figure.



b) Draw the *original tree* after performing operations delete(76) and then delete(55).

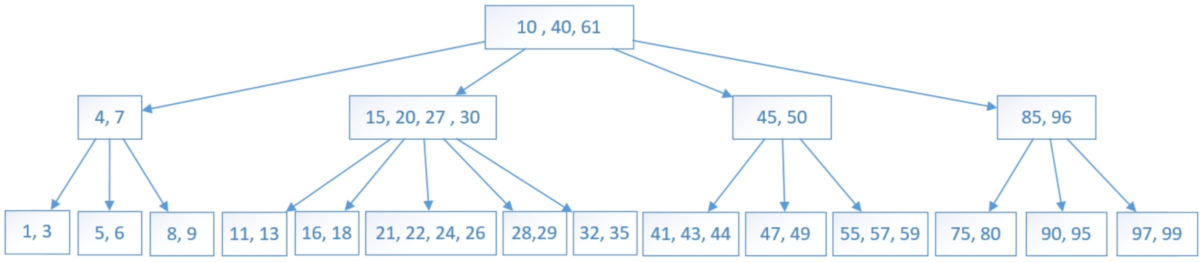
Answer: See the following figure.



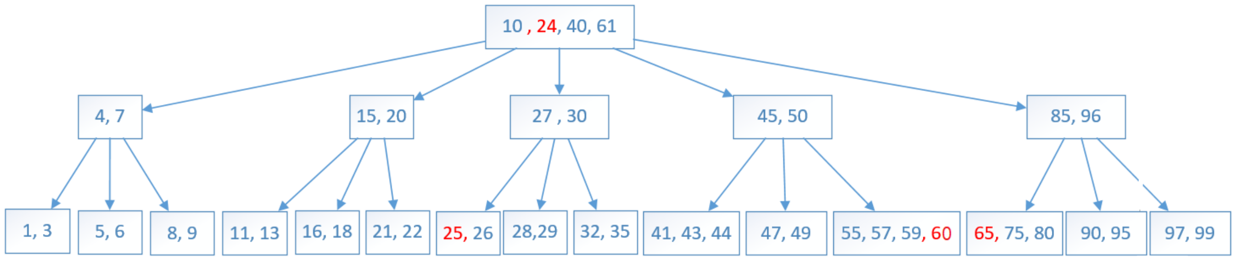
Here are some details for those of you who are still confused about delete in the 2-3 trees. For deletion, first, we swap the target item with its predecessor or successor to delete the node as a leaf. In this case, the deleted nodes are already leaves. After deleting 76, its node still has a key (75) and is not underloaded. So we can move on. Deleting 55 causes its node to have no key; looking at the direct sibling, we see it has only one key (66). So, we merge the two siblings, borrowing their parent's key. This creates a new node with keys 60 and 66. The parent who borrowed 60 is now underloaded. To fix this, we look at its direct sibling and we see it only has one key (82). So, we merge them again and borrow node 70 from the parent. The result will be a node with keys 70 and 82. Now the parent who borrowed 70 is underloaded. Looking at its sibling, we see it has two keys (10, 30). So, the parent borrows its key 50 to the empty and borrows a key 30 from the direct sibling. The pointer to 40 is updated to preserve the 2-3 tree structure.

Problem 3 B-Trees [10 marks]

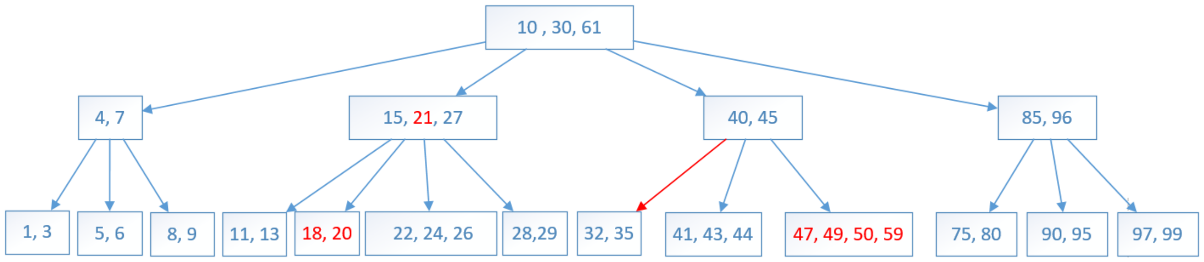
Consider the following B-tree of min-size 2. Note that each node should have at least 2 and at most 4 keys in such a tree.



(a) Insert the following keys (in the same order) into the tree: 65, 60, and 25. You need to draw one tree (the final tree after all insertions). **Answer:** Insertion of 60 and 65 does not cause an overflow. Insertion of 25 causes an overflow; as a result, its node is divided into two nodes, and the middle item (among the five in the overflowed node) is inserted to the parent (24 is recursively inserted above, which causes another overflow, and 24 is yet again moved forward to the root).

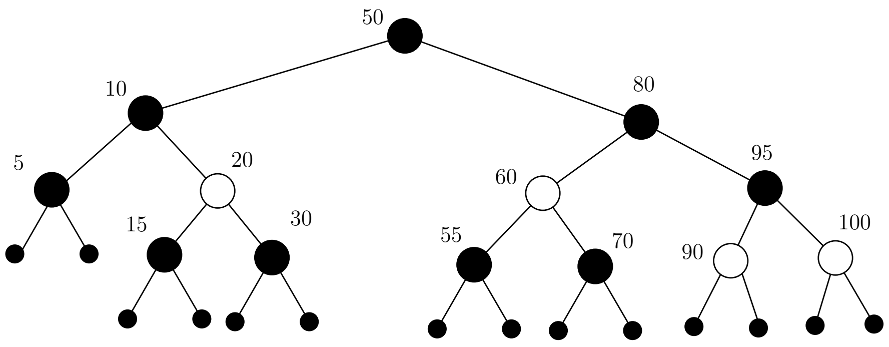


(b) From the original tree (the above tree), delete the following keys (in the same order): 55, 16, and 57.
Answer: Deletion of 55 does not cause an underflow. Deletion of 16 causes an underflow; fortunately, one of the direct siblings (the one on the right) has more than minimum keys; hence it borrows a key (21) to the parent, and the parent borrows a key (20) to resolve underflow. Deletion of 57 cause an underflow and this time, the direct sibling cannot help. We merge the underflow node with the direct sibling and borrow a key (50 from the parent). That causes another underflow in the parent which is resolved by borrowing from a direct sibling.

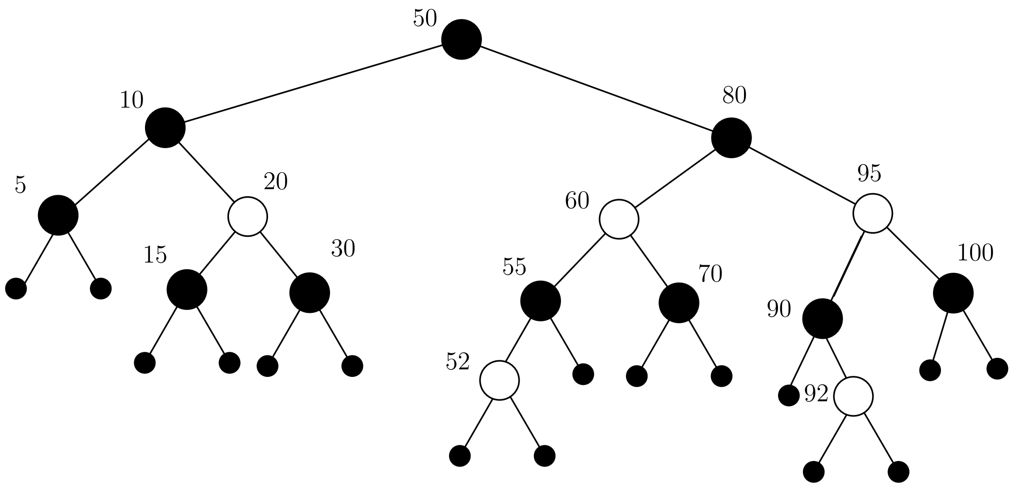


Problem 4 Red-Black trees [10 marks]

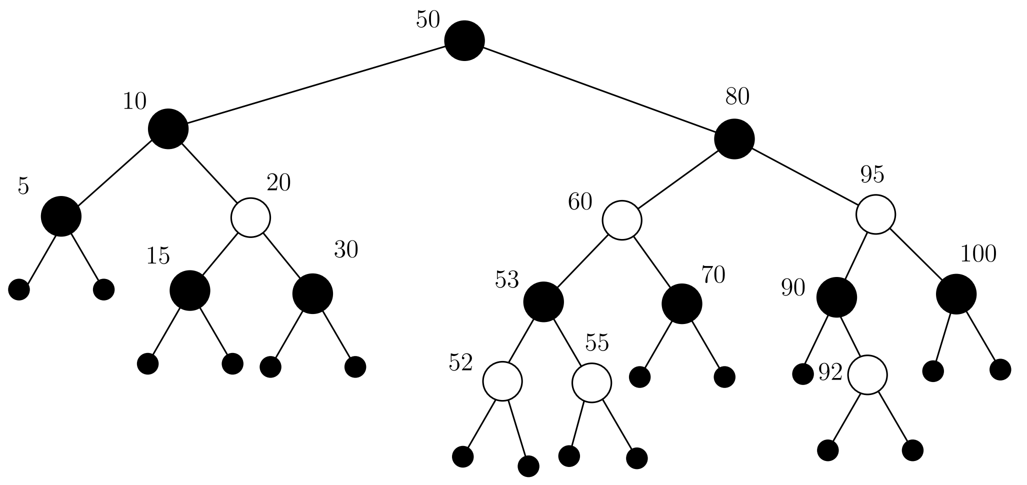
This problem will concern operations on the red-black T shown in the following figure.



(a) Draw the tree after performing operations insert(52), insert(92), and insert(53) (in the same order). It suffices to draw the final tree. **Answer:** Here is the final tree after insertions of 52 and 92:

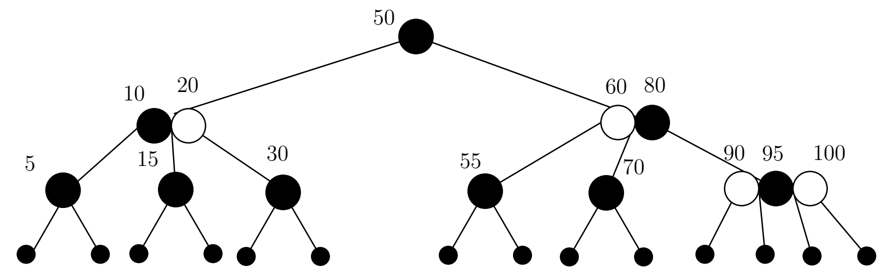


And here is the final tree after insertions of 53:

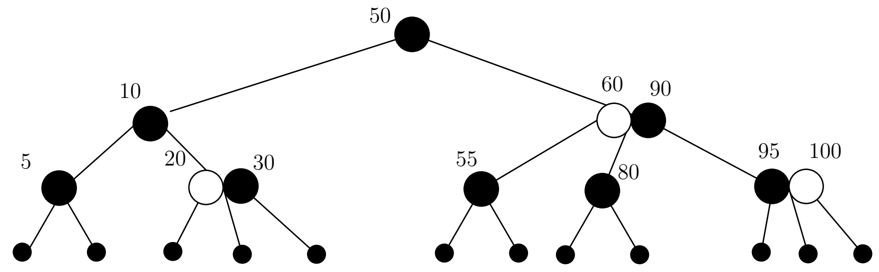


(b) Draw the original tree after performing operations delete(70), delete (15). For that, take the following steps: i) draw the B-tree associated with T ii) delete the two nodes from the B-tree. Note that after each change to the tree, you must maintain having exactly one black key at each node of the B-tree iii) draw the red-black tree associated with the resulting B-tree.

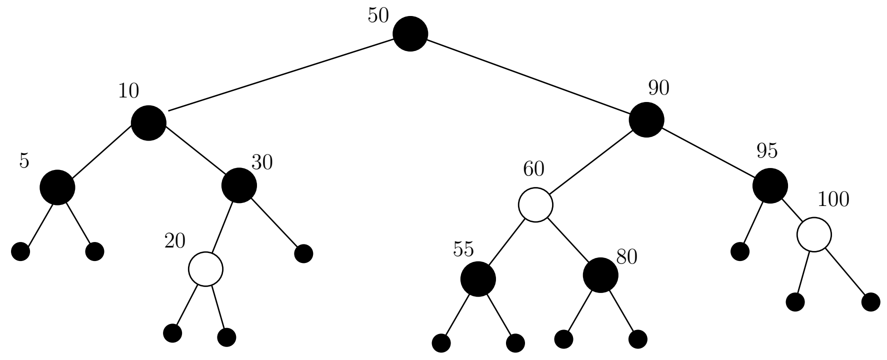
Answer: See the figures below:



initial b-tree



b-tree after deletions



final red-black tree