

Informatics II

Exercise 9

March 16, 2022

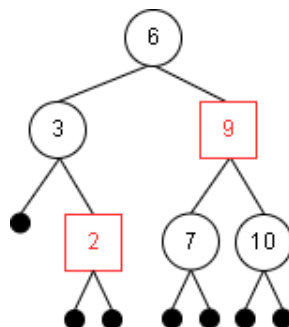
Red Black Tree

Consider the red-black tree in lecture where black nodes are denoted with a circle and red nodes are denoted with a square. We use the key to represent a node. For example, 2 represents the node with key 2. We consider four operations:

1. Creating a new node (left and right are set to NULL):
Example: create node 9: `create(9)`
2. Setting a property (color, key, left, right) of a node:
Example: set the key of node 9 to 5: `9->key = 5`. Here, 9 represents the node with key 9.
3. Rotating a node:
Example: right rotate node 5: `RightRotate(5)`
4. Deleting a node:
Example: delete node 9: `delete(9)`

Task 1

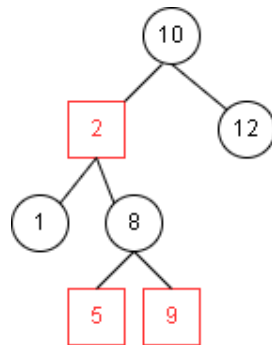
1. Consider the red-black tree below. Determine the number of internal nodes, the black height of node 3 and the black height of the tree?



2. What are the largest and smallest possible numbers of internal nodes in a red-black tree with a black-height k ? Give examples.

Task 2

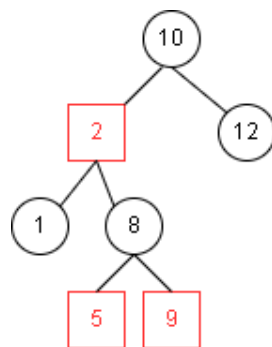
1. Consider the red-black tree shown below. State the operations that are required to insert 6 into the red-black tree.



2. Show the red-black tree that results after each of the integer keys 2, 3, 6, 7, and 1 insertion into an empty red black tree step by step.

Task 3

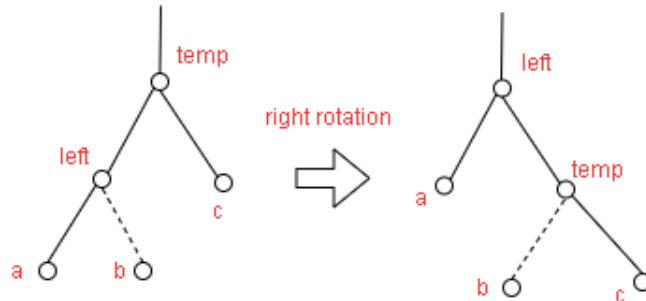
Delete 2 from the red black tree. Show state of tree after each step by drafts and explain the change.



Follow-up practises: show the states of deleting other nodes.

Task 4

1. Complete the code segment of `RightRotation(temp)` according to the diagram and instructions. Assume that the left child of node `temp` exists.



```

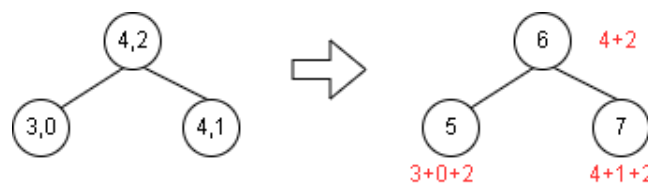
1 struct node {
2     struct node* p; // parent
3     struct node* r; // right child
4     struct node* l; // left child
5 };
6
7
8 void rightrotate(struct node* temp){
9     struct node * g = temp->p;
10    struct node* left = temp->l;
11    (a)----- // 'b' becomes left child of 'temp'
12    if(left->r) {
13        (b)----- // 'temp' becomes parent of 'b'
14    }
15    (c)----- // 'left' become the child of g
16    if (temp == g->l){
17        g->l = left;
18    }
19    else {
20        g->r = left;
21    }
22    (d)----- // 'temp' becomes right child of 'left'
23    (e)----- // 'left' becomes parent of 'temp'
24 }

```

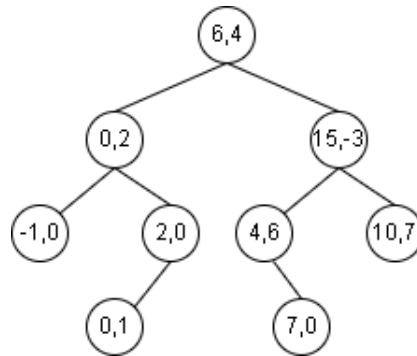
2. State what lines 16-21 have done.

Task 5

Consider an enhanced binary search tree in which each node v contains a key and an value called **addend**. The addend of node v is implicitly added to keys of all nodes in the subtree rooted at v (including v). Let $(key, addend)$ denote the contents of a node. See the following example that transforms an enhanced binary search tree to a binary search tree.



1. Show the binary search tree of the enhanced binary search tree below?



2. Consider the enhanced binary search tree below. Show the state of the enhanced binary search tree after the right rotation of node k1.

