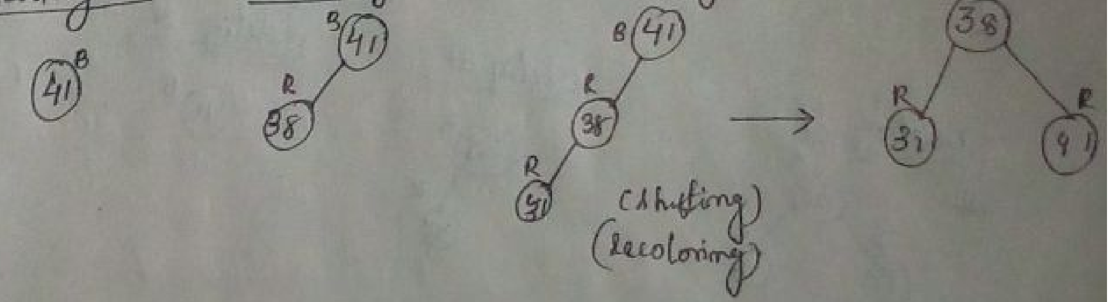
ADS

Riya Vincent
Roll no: 129

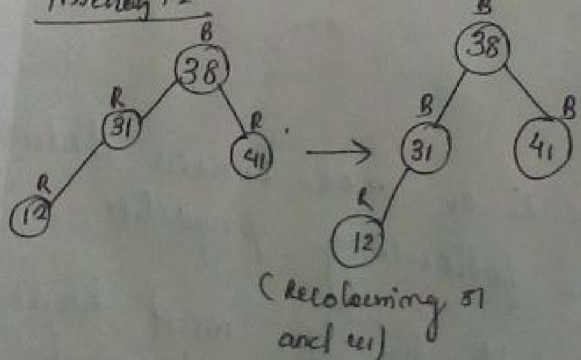
13. If we want to delete the Red black tree element in 8, 12, 19, 31, 38, 41. the Insertion process must be take place in 41, 38, 31, 19, 12, 8.
So the Insertion steps are given below.

Insertion

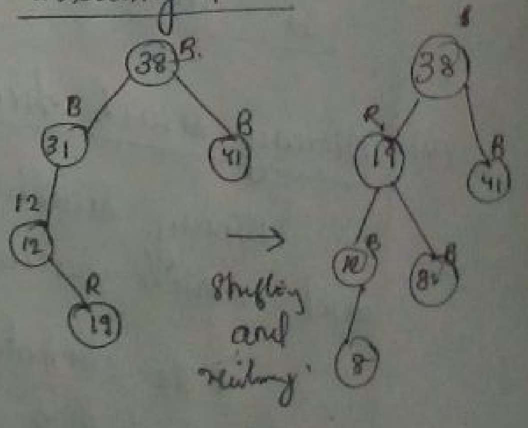
Inserting 41 | Inserting 38 | Inserting 31



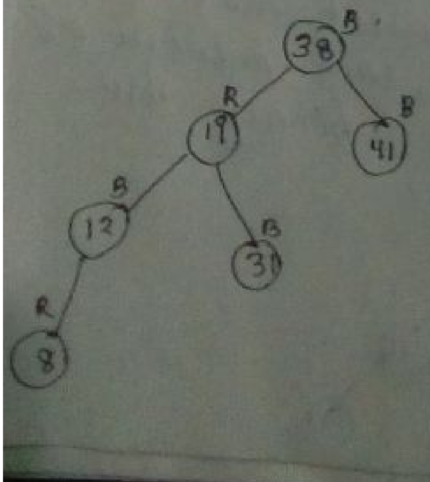
Inserting 12



Inserting 19

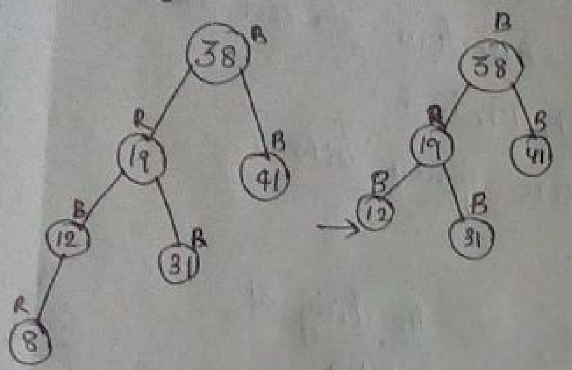


So the final tree is

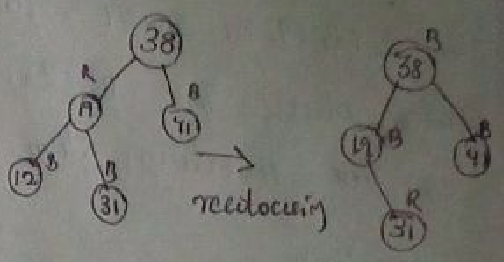


Now we are deliting the Red black tree element in 8, 12, 19, 31, 38, 41.

Deleting 8

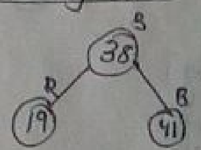


Deleting 12

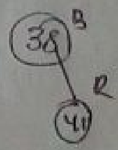


Deleting 19

all



deleting 31



deleting 38

41 B

deleting 41

Null tree.

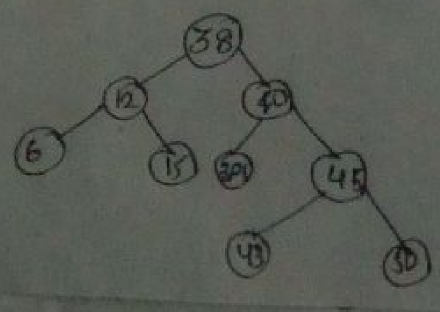
So deleted.

12) Binary search tree.

Binary search tree is a non-linear data structure and which have the following properties

* The elements in left sub tree must have the value less than that of the root node, and the right elements must have the greater value than the root node. the parent node has atleast two children and also, each sub tree of binary search tree is also a binary tree.

for example:



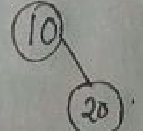
for example, if we want to insert the following elements in a binary search tree, we must follow the following steps.

elements : 10 20 15 40 5

step 1:

step2 : Inserting 20.

20 is greater than 10 so it is in the right sub tree of 10

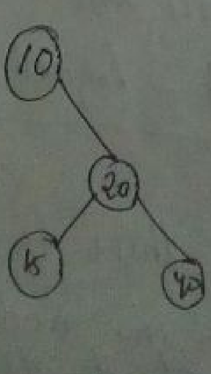


step 3: Inserting 15.

15 is also greater than 10, and it is also in right sub tree of 10, and less than 20 - ~~rig~~ left sub tree of 20

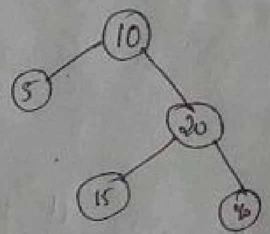
step 4 :

40 is the largest element in the list. So it in the right sub tree of 20



step 5:

to inserting 5, it is less than that of the root node. So it is the left sub tree.



the final binary search tree is given to above

Insertion steps:

step 1: start.

step 2: Insert The root element

step 3: compare the next element with root, If it is less than the root, it must be placed left side of the root, else places the element in right sub tree.

step 4: stop.

11. a) Insertion beginning and ~~end~~ mid - Singly linked list and doubly linked list

Singly linked list - beginning Insertion

step 1: start

step 2: check if there is already a node exist

step 2: if there is no node, then create a new node and set it as ptr

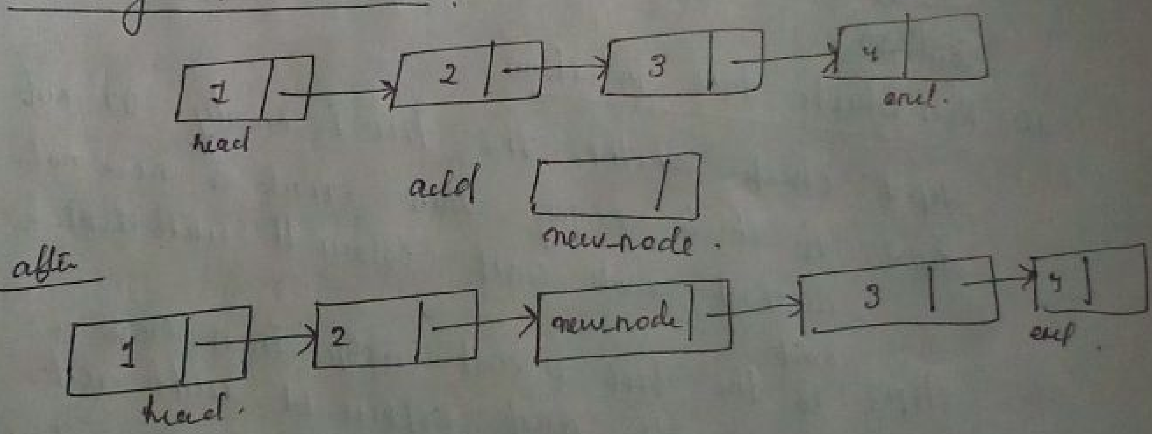
step3: set ~~the~~ ~~gvoal~~ ~~operation~~
ptr = ptr → next; (to store the address of the 1st node).

step 4: set new node,
insert the value to that node.
new node → data = value

step5: if there is no element there then sets,
new node → next = head
head = new node.

step 6: exit.

Inserting at middle



Algorithm

step1: start
step2: If check the ~~element~~ ~~is~~ list is empty. If it is empty, then create a new node and follow the above mentioned steps
step3: If is not empty, then create a new node with attributes data and link.

step4: If the list ~~empt~~ is not empty and the list has only one element, then calculate the size of the list and divide it by

2 to get mid-point of the list

step 5: define a current node and it will point to the created new node.

step 6: define another node temp which points to node of next to current.

step 7: the new node will be inserted after the current and before temp.

step 8: stop.

~~doubly linked list~~

~~insertion~~ .

10. Red black tree Insertion

step 1: check whether the tree is empty or not

step 2: If it is empty then create a new node as root node and colour it with Black. exit

step 3: If the tree is not empty then create a new node and colour it with red.
If the parent node is black then exit.

Step 4: If the parent node is ~~is~~ Red then check the colour of parent's sibling of new node.

step 5: The parent's sibling is ~~[illegible]~~ black or Null, then rotate and recolour it.

step 6: If parent's sibling is red then recolor.

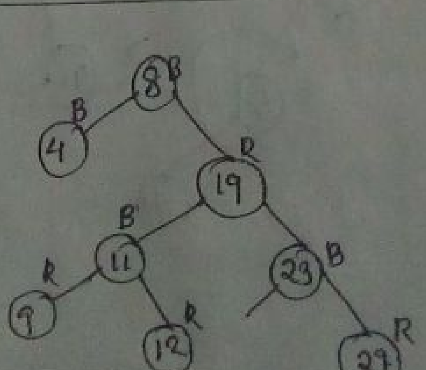
step 7: exit

7. Red black tree

Red black tree is a type of binary search tree with some additional properties that are listed below.
The root node of the red black tree is always black coloured. Every path from the root node to leaf node must have the same number of black node. No two red nodes can be adjacent, which mean that, the red node cannot be a parent of the another red node.

And a red black tree is a kind of self balancing tree, where each node has an extra bit. Each node have some colour is red or black. These colours are used to ensure that the tree remains balance during insertions and deletions.

example of red black tree.

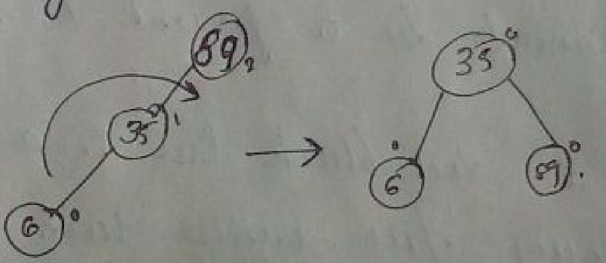


6 Balancing can be performed, using the following rotations.

1) Right rotation
2) left rotation
3) Right left rotation
4) left right rotation

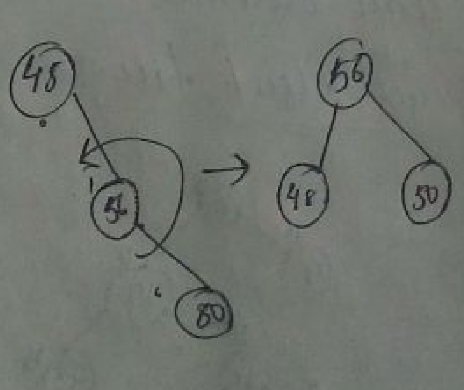
1) <u>Right rotation</u>

In the following binary search tree, there is a single rotation is required.

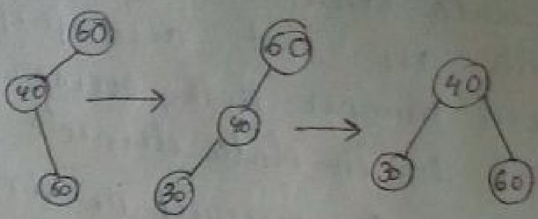


2) <u>Left rotation</u>

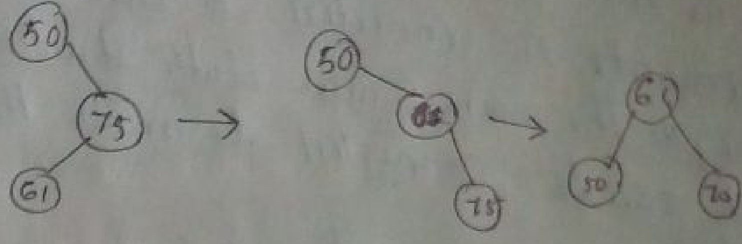
In the following binary search tree, there is single left rotation is required



3 Right-left rotation



4. Left right rotation



1) Linear data structure

A linear data structure is a structure in which the elements are stored sequentially, the elements are connected to the previous and the next ~~node~~ element. The elements are stored sequentially, so they can be traversed or accessed in a single run. The implementation process are very easy compared to the non linear ~~ds~~ data structure. The data elements are ~~ds~~ traversed one after another and can access only one element at a time

examples.
* Array
* Queue
* Stack.
* Linked list

Non linear data structure

A non-linear data structure is also another type of data structure in which the data elements are not arranged in a contiguous manner. As the arrangement is non-sequential, so the data elements cannot be traversed or accessed in single run. In the case of linear data structure, element is connected to two elements, whereas, in the non-linear data structure, an element can be connected to more than two elements.

Eg:) tree and graph.

3) Binary search tree

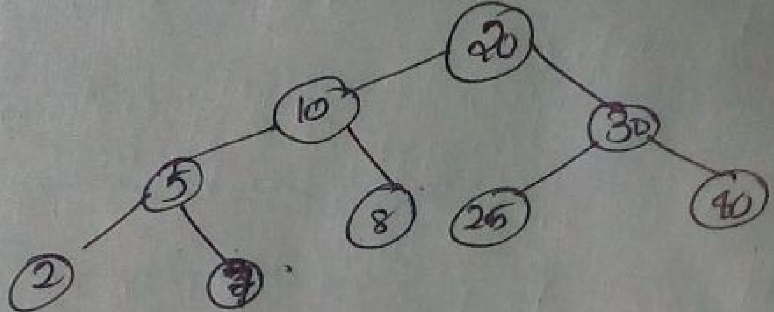
A binary search tree is a type of tree in data structure which has the following properties

→ The value of the left sub tree is less than the value of its parents node's key.

→ the value of the key of the right sub tree is greater than or equal to the node.

The binary search tree is a collection of nodes arranged in a way where they maintain BST properties.

example of binary search tree



Basic operations

→ Insertion
→ deletion
→ searching.
→ Traversal → Inorder
→ preorder
→ post order

2) A stack is a linear data structure can be implemented using one-dimensional array. Stack implemented using array stores only a fixed number of data values. The implementation of is very simple. Just define a one dimension array of a size and insert and delete the values into that array by using LIFO principle with the help of a specific variable. Initially, the top is set to -1 a value from the stack, then delete the top value and decrement the top by value 1

Algorithm

step 1 : start

step 2 : declare a 1 dimensional array with a fixed size.

step 3 : define a integer variable called top and initialize top = -1.

step 4 : And perform the stack operations
  1) PUSH
  2) POP

step 5 : stop