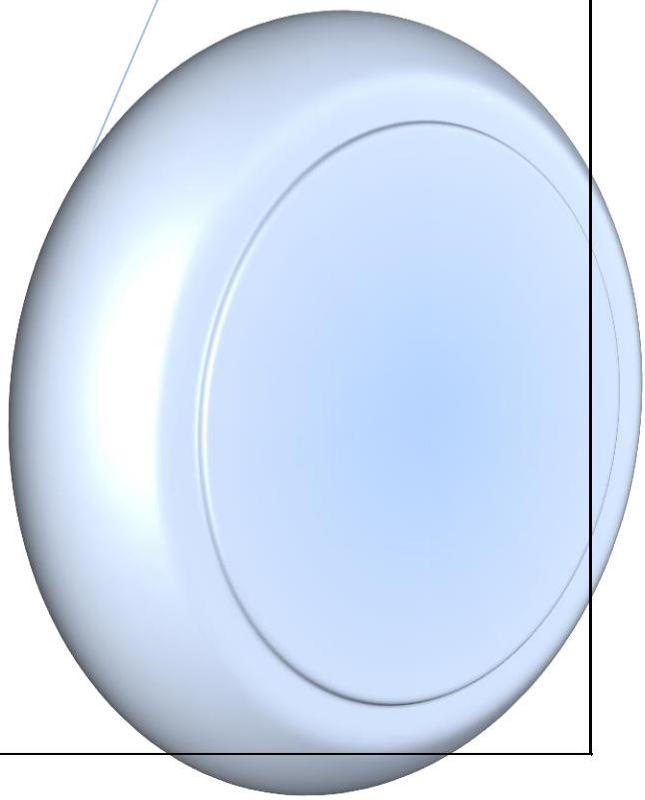
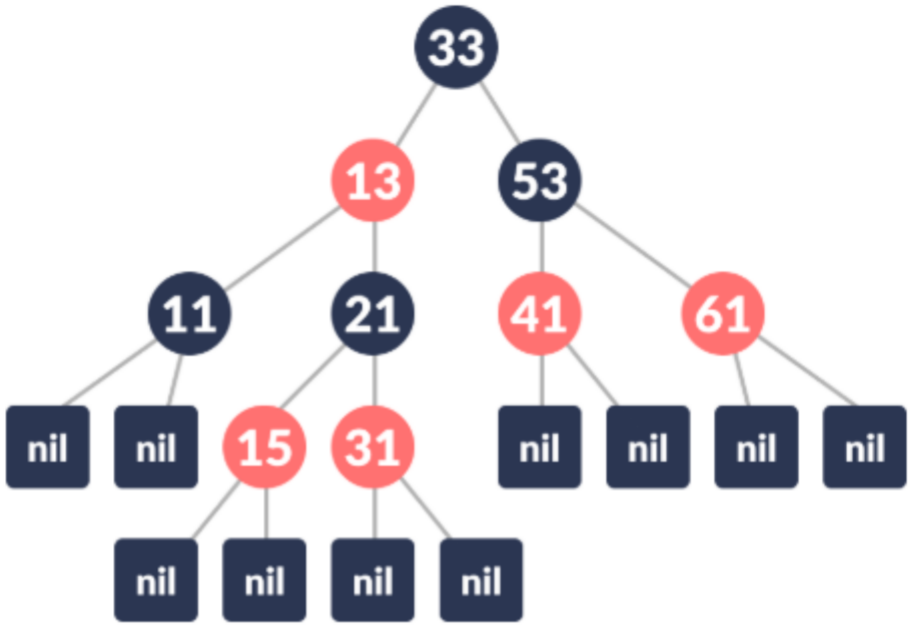


Lab 2:Red Black Tree

Name 1:Nada Fathy. ID:68.
Name 2:Omar Youssef. ID:45.

1.1 Red Black Tree.



A red black tree is a kind of self-balancing binary search tree in computer science as it shown in the figure. Each node of the binary tree has an extra bit, and that bit is often interpreted as the color (red or black) of the node. These color bits are used to ensure the tree remains approximately balanced during insertions and deletions.

Explaining each function in The Red balck tree and its time time analysis.

1-getRoot():

return the root of the given Red black tree.

2-isEmpty():

return whether the given tree is Empty or not.

3- clear:

Clear all keys in the given tree.

4- search:

return the value associated with the given key or null if no value is found.

5. contains:

return true if the tree contains the given key and false otherwise.

6-insert:

Insert the given key in the tree while maintaining the red black tree properties. If the key is already present in the tree, update its value.

7-Delete:

Delete the node associated with the given key. Return true in case of success and false otherwise.

8-Rotate Right:

It make the right rotation after insertion.

9-Rotate Left:

It make the left rotation after insertion.

10-fix tree:

It fix the properites of the whole tree after insertion.

11-Fix delete:

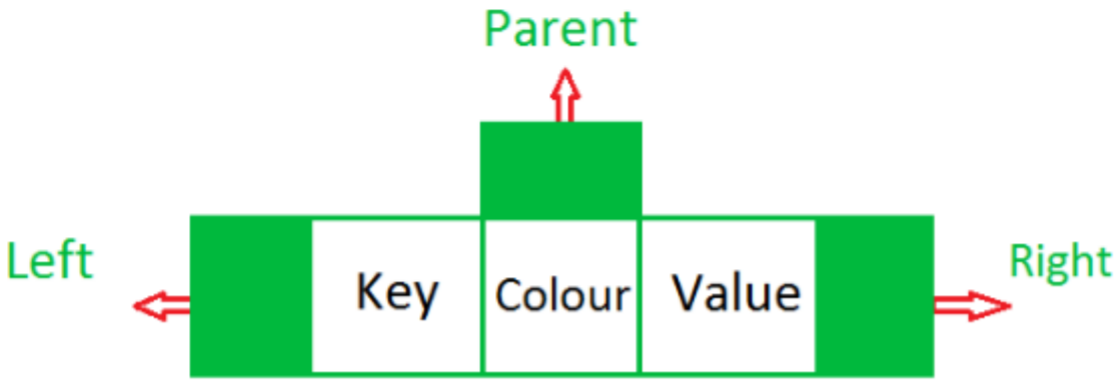
It fix the properites of tree after deletion.

12-Delet Node helper:

it delet the Node if the tree has it.

Function	*Get Root*	*Is Empty()*	*Clear()*	*Search()*	*contains*	*Insert()*	*Deletion*		
time	*O(1)*	*O(1)*	*O(1)*	*O(n)*	*O(n)*	*Log(n)*	*Log(n)*		

2-2 Tree Map.



1-ceilingEntry:

Returns a key-value mapping associated with the least key greater than or equal to the given key, or null if there is no such key.

2-ceilingKey:

Returns the least key greater than or equal to the given key, or null if there is no such key.

3-clear:

Removes all of the mappings from this map.

4-containsKey:

Returns true if this map contains a mapping for the specified key.

5-containsValue:

Returns true if this map maps one or more keys to the specified value.

6-entrySet:

Returns a Set view of the mappings contained in this map in ascending key order.

7-firstEntry:

Returns a key-value mapping associated with the least key in this map, or null if the map is empty.

8-firstKey:

Returns the first (lowest) key currently in this map, or null if the map is empty.

9-floorEntry:

Returns a key-value mapping associated with the greatest key less than or equal to the given key, or null if there is no such key.

10-floorKey:

Returns the greatest key less than or equal to the given key, or null if there is no such key.

11-get:

Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.

12-headMap:

Returns a view of the portion of this map whose keys are strictly less than to Key in ascending order.

13-headMap:

Returns a view of the portion of this map whose keys are less than (or equal to, if inclusive is true) to Key in ascending order..

14-keySet:

Returns a Set view of the keys contained in this map.

15-lastEntry:

Returns a key-value mapping associated with the greatest key in this map, or null if the map is empty.

16-lastKey:

Returns the last (highest) key currently in this map.

17-pollFirstElement:

Removes and returns a key-value mapping associated with the least key in this map, or null if the map is empty.

18-pollLastEntry:

Removes and returns a key-value mapping associated with the greatest key in this map, or null if the map is empty.

19-put:

Associates the specified value with the specified key in this map.

20-putAll:

Copies all of the mappings from the specified map to this map.

21-remove:

Removes the mapping for this key from this TreeMap if present.

22-size:

Returns the number of key-value mappings in this map.

23-values:

Returns a Collection view of the values contained in this map.

Function	*ceilingEntry*	*ceilingKey*	*clear*	*containsKey*	*containsValue*	*entrySet*	*firstEntry*	*firstKey*
Time	*Log(n)*	*Log(n)*	*Log(n)*	*Log(n)*	*Log(n)*	*Log(n)*	*Log(n)*	*Log(n)*
function	*floorEntry*	*floorKey*	*get*	*headMap*	*headMap*	*keySet*	*lastEntry*	*lastKey*
Time	*Log(n)*	*Log(n)*	*Log(n)*	*Log(n)*	*Log(n)*	*(n)*	*Log(n)*	*Log(n)*
Function	*pollFirstElement*	*pollLastEntry*	*put*	*putAll*	*remove*	*size*	*values*	
Time	*Log(n)*	*Log(n)*	*Log(n)*	*(n)Log(n)*	*Log(n)*	*O(1)*	*(n)*	