RED – BLACK Tree Insertion Algorithm

*Problem statement* : Implement the Red-black tree insertion.

*Expected result* : Insertion should be completed with the test cases.

*Achieved result* : Insertion is completed with the test cases.

**Introduction to the Red-black tree:**

Red-black tree is a self balancing binary search tree. In this every node is follows following rules.

1. Every node has a color either red or black.
2. Root of tree is always black.
3. There are no adjacent red node(means a red node can not have red parent or red child).
4. Black height must be same in every path from the root node to null node.

**Red-black tree Insertion Algorithm:**

In red-black tree insertion to overcome the unbalancing after insertion we are trying to recolor first, if recoloring does't bring back balancing property of the red-black tree, then we will do rotations.

Mainly we have two cases depending on the uncle color, if uncle color is red then we will do recoloring. If the uncle color is black we will do rotations. Color of the NULL node is always considered as black. Newly created node color should always be red.

**Fixing of the nodes:**

Let ‘A‘ be newly created node. Perform insertion operation by following steps.

1. Insert the newly created node by using standard binary search tree insertion and the newly inserted node is red.
2. If the newly inserted node is root then change the color of the node to black(then the black height of tree will increases by 1).
3. If color of newly inserted node is not black and the newly inserted node is not root do the following steps.

* If A’s uncle is red (grand parent must be black color from rule 4).
* Color A’s grand parent as red.
* Change A = A’s grand parent, repeat the steps 2 and 3 for new A.

1. If color of A’s uncle is black then there can be four cases.

* Left – Left case (P is left of G and A is left of P).
* Left – Right case (P is left of G and A is right of P).
* Right – Right case (P is right of G and A is right of P).
* Right – Left case (P is right of G and A is left of P).

*Left – Left case:*

* *Right rotate with respect to G.*
* *Swap the colors of G and P.*

*Left – Right case:*

* *Left rotate with respect to P.*
* *Apply Left – Left case as mentioned above.*

*Right – Right case:*

* *Left rotate with respect to G.*
* *Swap the colors of G and P.*

*Right – Left case:*

* *Right rotate with respect to P.*
* *Apply Right – Right case mentioned above.*

*NOTE : P is Parent.*

*G is Grand parent.*

*A is new node.*