# Red Black Tree

## Insertion

1. **Z** is Root

*Color* ***Z****, black*

1. **Z.Uncle** and **Z.Parent** are red

*Recolor* ***Z.Parent****,* ***Z.grandParent*** *&* ***Z.uncle***

1. **Z.Uncle** is black (**Z**, **Z.Parent** & **Z.GrandParent** for a Triangle)

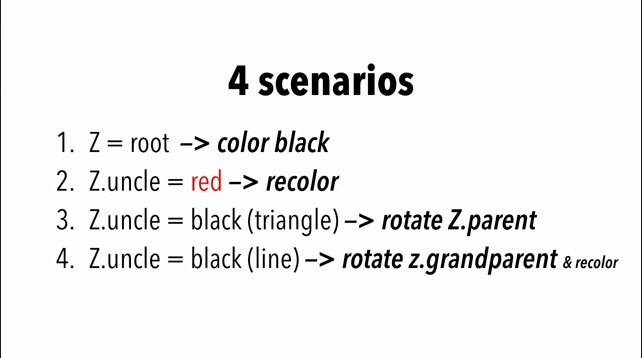
*Rotate* ***Z.Parent*** *opposite direction of* ***Z****:*

* *If* ***Z*** *is the left child, rotate right*
* *If* ***Z*** *is the right child, rotate left*

1. **Z.Uncle** is black (**Z**, **Z.Parent** & **Z.GrandParent** for a Line)

*Rotate* ***Z.GrandParent*** *the opposite direction of* ***Z*** *and recolor* ***Z.Parent*** *&* ***Z.GrandParent****:*

* *If* ***Z*** *is the left child, rotate right. Then recolor* ***Z.Parent*** *&* ***Z.GrandParent***
* *If* ***Z*** *is the right child, rotate left. Then recolor* ***Z.Parent*** *&* ***Z.GrandParent***

****

## Deletion

1. Initial Step - Choosing replacement
   * *If the node we deleted has two* ***NIL*** *children, its replacement* ***X*** *is* ***NIL***
   * *If the node we deleted has 1* ***NIL*** *child and 1* ***Non-NIL*** *child, its replacement (successor’s)* ***X*** *is the* ***Non-NIL*** *child (****X*** *is the right node of the replacement)*
   * *If the node we deleted has 2* ***Non-NIL*** *children, set the* ***X*** *to replacement’s (successor’s) right child before the replacement is spliced out. (****X*** *is the right node of the replacement)*
2. Initial Step - Coloring
   * *If the node we deleted is* ***Red*** *its replacement is* ***Red*** *or* ***NIL****, we are done*
   * *If the node we deleted is* ***Red*** *its replacement is* ***Black****, recolor the replacement and Proceed to appropriate case*
   * *If the node we deleted is* ***Black*** *its replacement is* ***Red****, recolor the replacement and Proceed to appropriate case*
   * *If the node we deleted is* ***Black*** *its replacement is* ***Black*** *Proceed to appropriate case*
3. Cases
4. Node **X** is **Red**

*Color* ***X****,* ***Black***

1. Node **X** is **Black** & Its sibling **W** is **Red**
2. *Color* ***W****,* ***Black***
3. *Color* ***X.Parent******Red***
4. *Rotate* ***X.Parent*** *in the same direction of* ***X***

* *If* ***X*** *is* ***left*** *child, rotate* ***left***
* *If* ***X*** *is* ***right*** *child, rotate* ***right***



* *If* ***X*** *is* ***left*** *child, set* ***W = X.Parent.Right***
* *If* ***X*** *is* ***Right*** *child, set* ***W = X.Parent.Left***

1. Node **X** is **Black** & Its sibling **W** is **Black** and both **W’s** children are **Black**
2. *Color* ***W****,* ***Red***
3. *Set* ***X = X.Parent***

* *If* ***New X*** *is* ***Red****, color* ***X Black***
* *If* ***New X*** *is* ***Black****, decide on other cases*

1. Node **X** is **Black** & Its sibling **W** is **Black** and
   1. If **X** is left child, **W’s** left child is **Red** & **W’s** right child is **Black**
2. *Color* ***W.Left****,* ***Black***
3. *Color* ***W****,* ***Red***
4. *Rotate* ***W*** *to the right*
5. *Set* ***W = X.Parent.Right***
6. *Proceed to case 5*
   1. If **X** is right child, **W’s** right child is **Red** & **W’s** left child is **Black**
7. *Color* ***W.Right****,* ***Black***
8. *Color* ***W****,* ***Red***
9. *Rotate* ***W*** *to the left*
10. *Set* ***W = X.Parent.Left***
11. *Proceed to case 5*
12. Node **X** is **Black** & Its sibling **W** is **Black** and
    1. If **X** is left child & **W’s** right child is **Red**
13. *Color* ***W*** *the same color as* ***X.Parent***
14. *Color* ***X.Parent****,* ***Black***
15. *Color* ***W.Right****,* ***Black***
16. *Rotate* ***X.Parent*** *to the left*
    1. If **X** is right child & **W’s** left child is **Red**
17. *Color* ***W*** *the same color as* ***X.Parent***
18. *Color* ***X.Parent****,* ***Black***
19. *Color* ***W.Left****,* ***Black***
20. *Rotate* ***X.Parent*** *to the right*