Red Black Tree

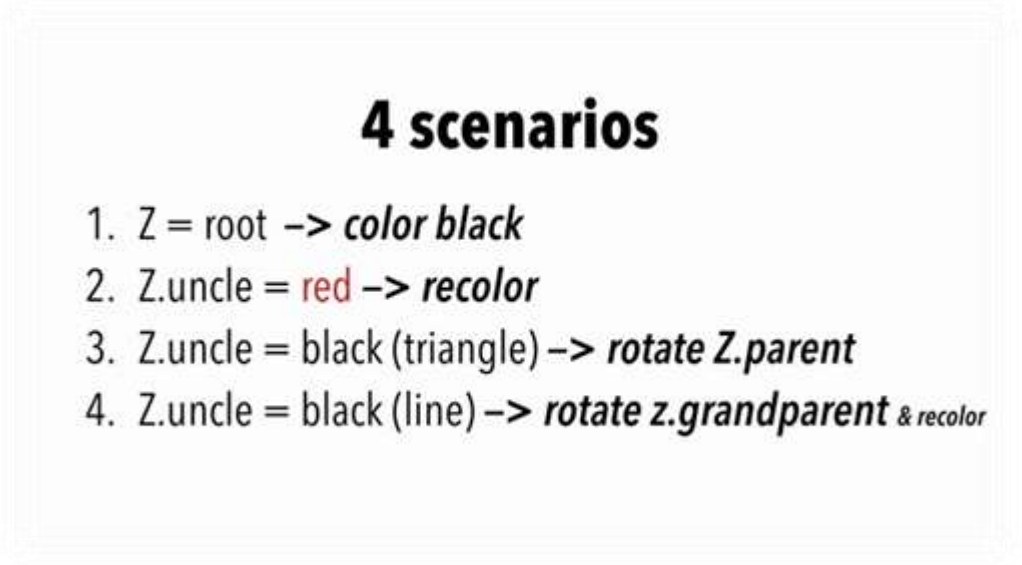
Insertion

1. **Z** is Root
   *Color **Z**, black*

2. **Z.Uncle** and **Z.Parent** are red
   *Recolor **Z.Parent**, **Z.grandParent** & **Z.uncle***

3. **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*

4. **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 is the **Non-NIL** child*
    - *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.*
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
    1) Node **X** is **Red**
       *Color **X**, **Black***

    2) Node **X** is **Black** & Its sibling **W** is **Red**
        I. *Color **W**, **Black***
        II. *Color **X.Parent Red***
        III. *Rotate **X.Parent** in the same direction of **X***
            - *If **X** is **left** child, rotate **left***
            - *If **X** is **right** child, rotate **right***
        IV.
            - *If **X** is **left** child, set **W = X.Parent.Right***
            - *If **X** is **Right** child, set **W = X.Parent.Left***

    3) Node **X** is **Black** & Its sibling **W** is **Black** and both **W's** children are **Black**
        I. *Color **W**, **Red***
        II. *Set **X = X.Parent***
            - *If **New X** is **Red**, color **X Black***
            - *If **New X** is **Black**, decide on other cases*

    4) Node **X** is **Black** & Its sibling **W** is **Black** and
        a. If **X** is left child, **W's** left child is **Red** & **W's** right child is **Black**
            I. *Color **W.Left**, **Black***
            II. *Color **W**, **Red***
            III. *Rotate **W** to the right*
            IV. *Set **W = X.Parent.Right***
            V. *Proceed to case 5*

        b. If **X** is right child, **W's** right child is **Red** & **W's** left child is **Black**
            I. *Color **W.Right**, **Black***

II. *Color **W, Red***
III. *Rotate **W** to the left*
IV. *Set **W = X.Parent.Left***
V. *Proceed to case 5*

5) Node **X** is **Black** & Its sibling **W** is **Black** and
   a. If **X** is left child & **W's** right child is **Red**
      I. *Color **W** the same color as **X.Parent***
      II. *Color **X.Parent, Black***
      III. *Color **W.Right, Black***
      IV. *Rotate **X.Parent** to the left*

   b. If **X** is right child & **W's** left child is **Red**
      I. *Color **W** the same color as **X.Parent***
      II. *Color **X.Parent, Black***
      III. *Color **W.Left, Black***
      IV. *Rotate **X.Parent** to the right*