Red Black Tree insertion

Some pseudocode:

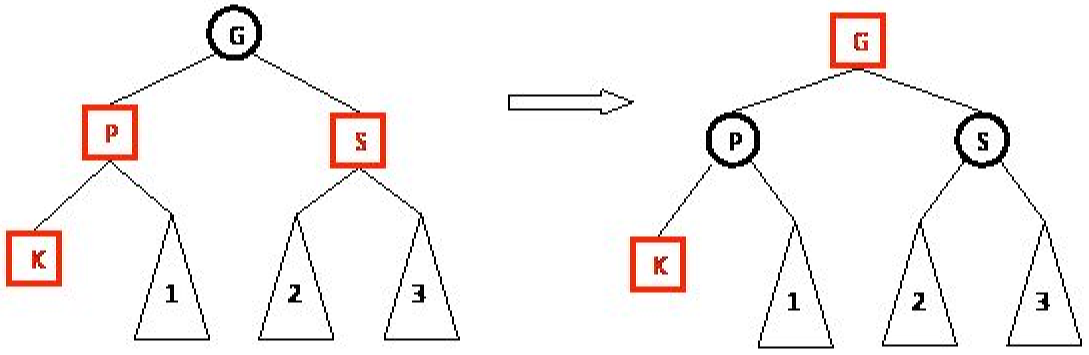
```
insert( K key)
    n = create red node( key)
    if empty tree
        root = n
        change n color to black
    else
        do BST insert of n as leaf
        if parent of n is red
            // new node and its parent are both red = must fix this!
            if uncle of n is red, then **recolor**
            else **rotate**
```

Recoloring

In the figure below… a new node (K) is added. It's parent (P) is red causing a red-red violation. If the uncle (S) is red, then recolor in two steps:

1. Make the grandparent (G) red, and
2. Color its children (P and S) black.

This resolves the red-red conflict AND maintains equal black-height.



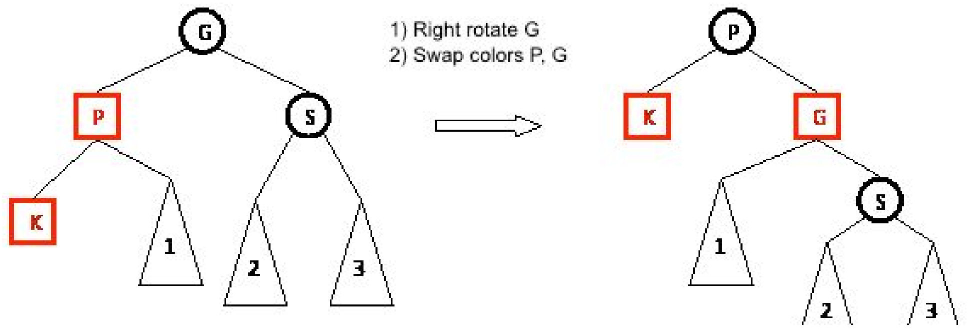
Important - making the red grandparent (G) may cause a conflict above us. Apply the same recursively to grandparent (G).

Rotation

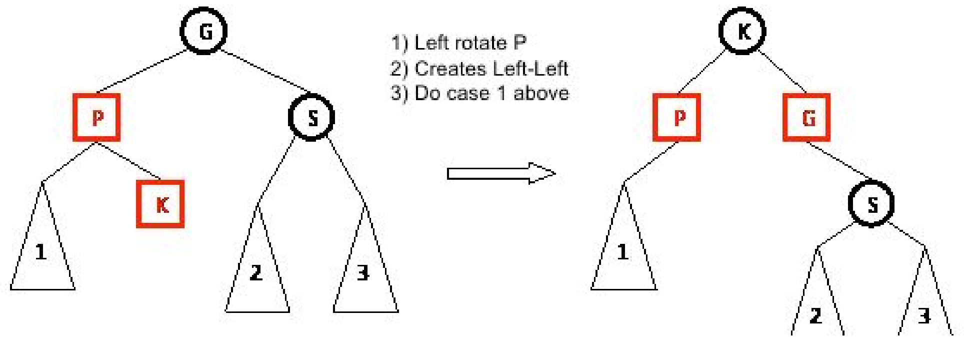
If the new node's uncle is a black node, then rotation is required.
There are 4 rotations cases (similar to AVL). They're on the next back (the back).
Source: pages.cs.wisc.edu/~paton/readings/Red-Black-Trees/

4 Rotation cases - when new node (K) has a red parent (P) and a black uncle (S).

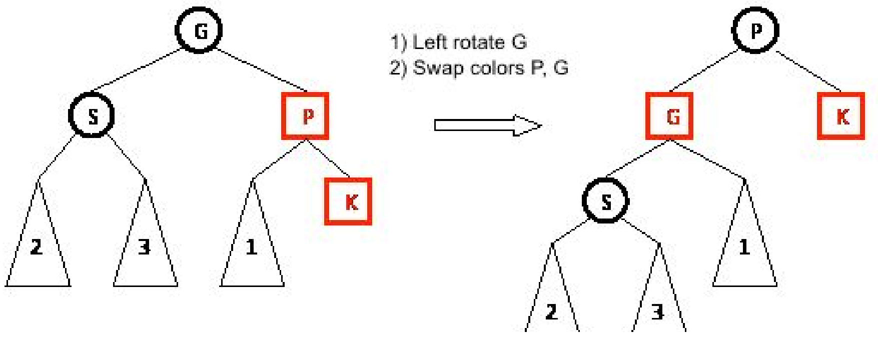
Case 1: Left-Left (P is parent, K is new (Key) node)



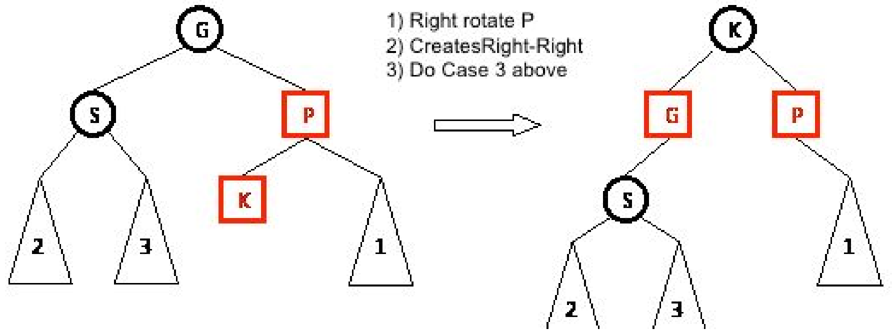
Case 2: Left-Right



Case 3: Right-Right



Case 4: Right-Left


G
S
P
K
2
3
1
1) Right rotate P
2) CreatesRight-Right
3) Do Case 3 above
K
G
P
S
1
2
3