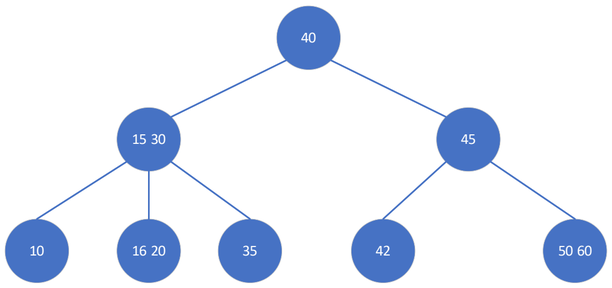


2-3树

插入40

插入30

Case1: 2节点插入数据

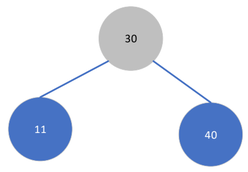
形成一个3节点

形成一个3节点

插入11

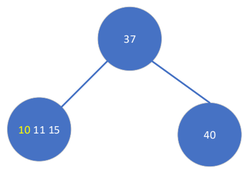
Case2: 3节点插入数据

插入11后，形成临时4节点，不满足2-3树性质，4节点分裂，30向上提

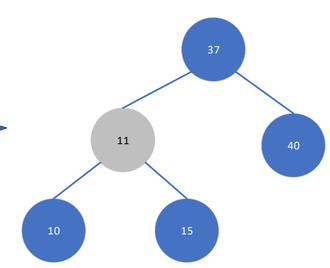


插入10

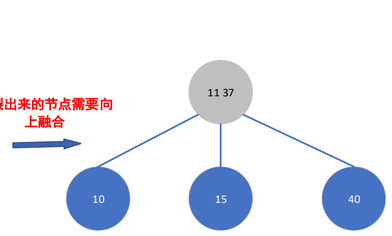
Case3: 3节点插入数据，向父节点融合，父节点是2节点



4节点分裂

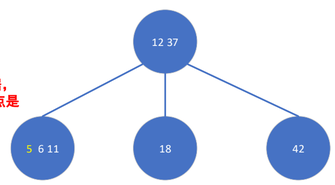


分裂出来的节点需要向上融合

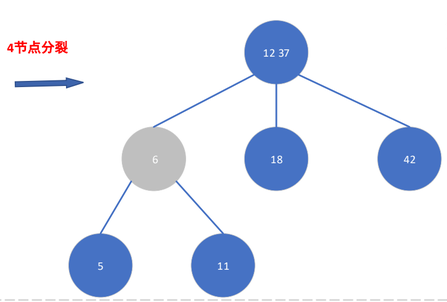


插入5

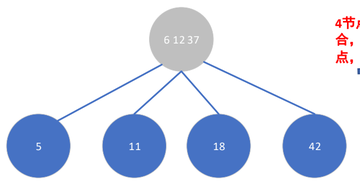
Case2: 3节点插入数据，向父节点融合，父节点是3节点



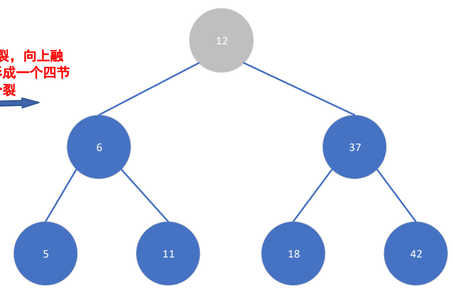
4节点分裂

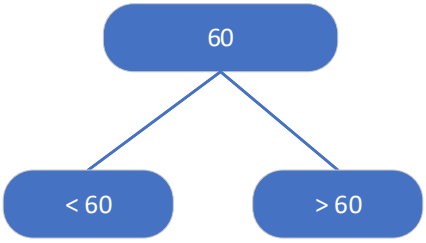


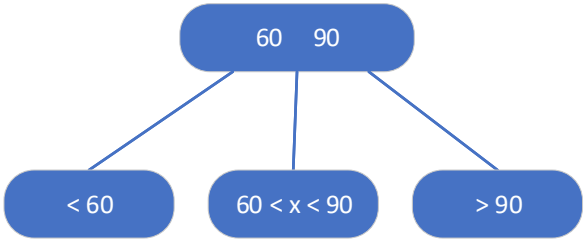
分裂后，6向上融合，又形成一个四节点

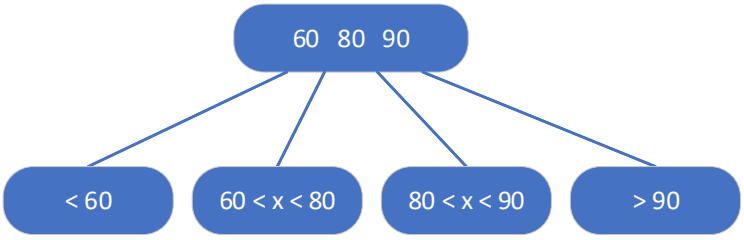


4节点分裂，向上融合，又形成一个四节点，在分裂









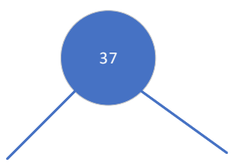
2-3-4树的节点

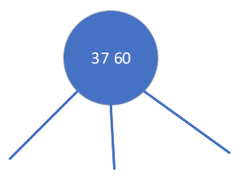
2节点

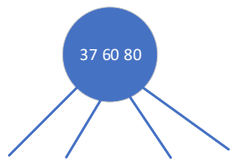
3节点

4节点

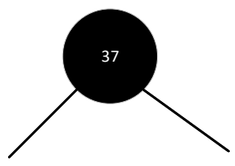
2-3-4树

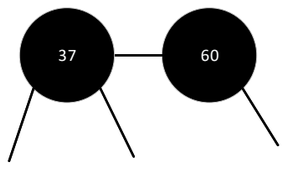


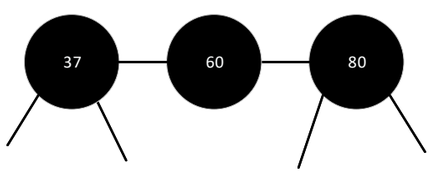


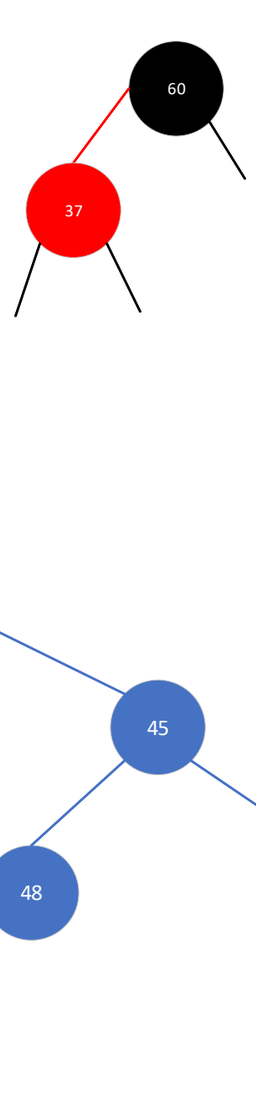


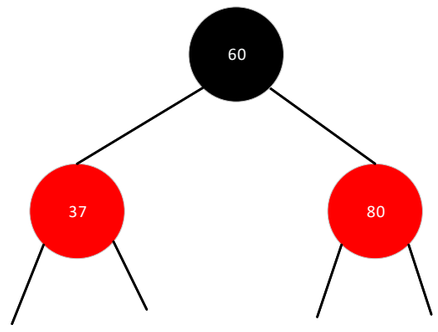
红黑树

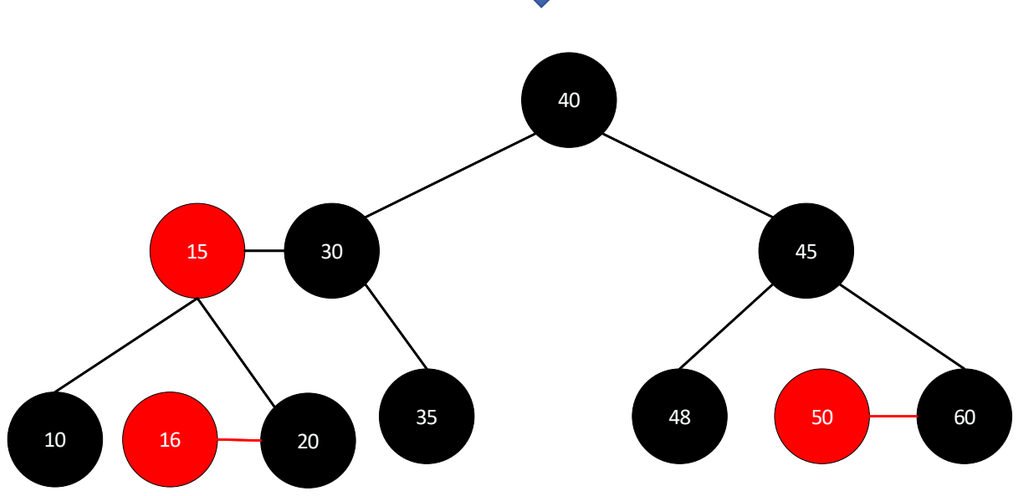


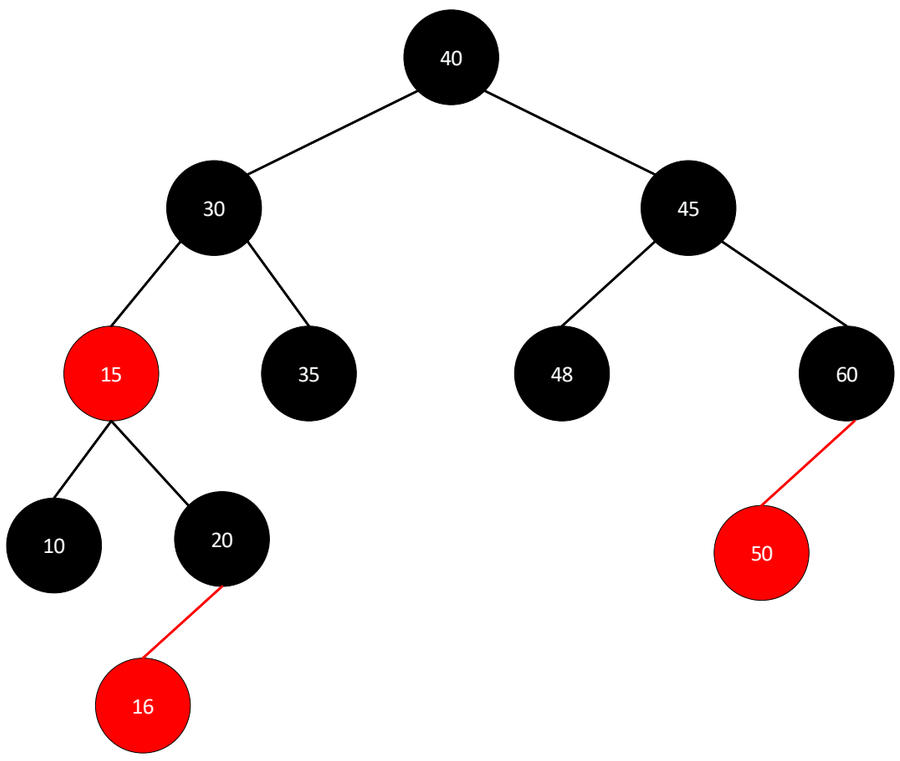


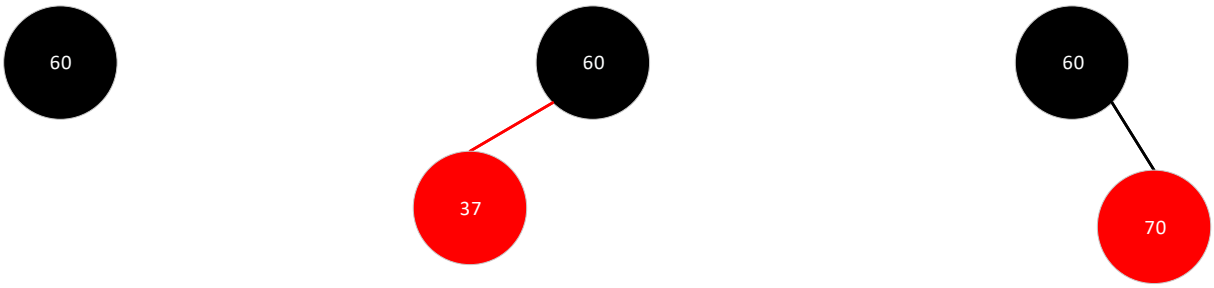


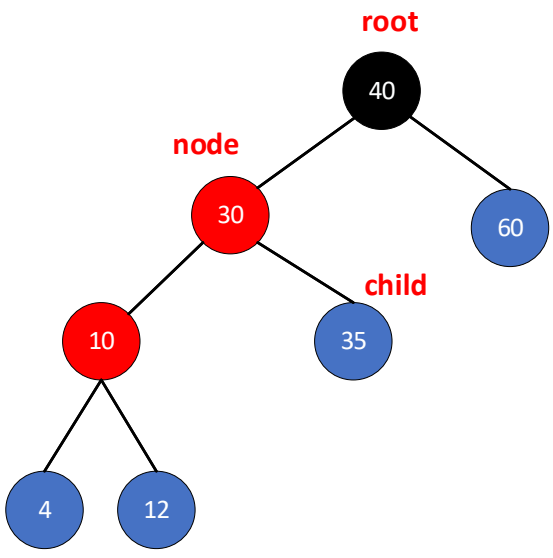


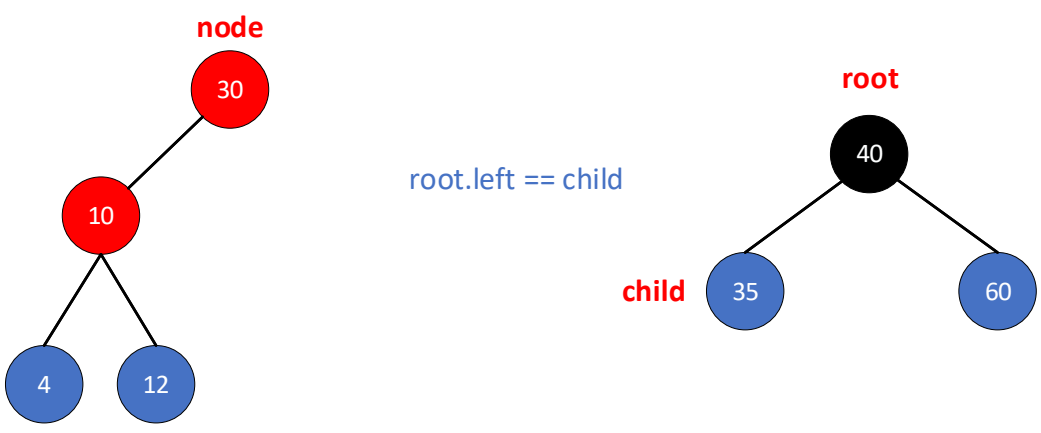


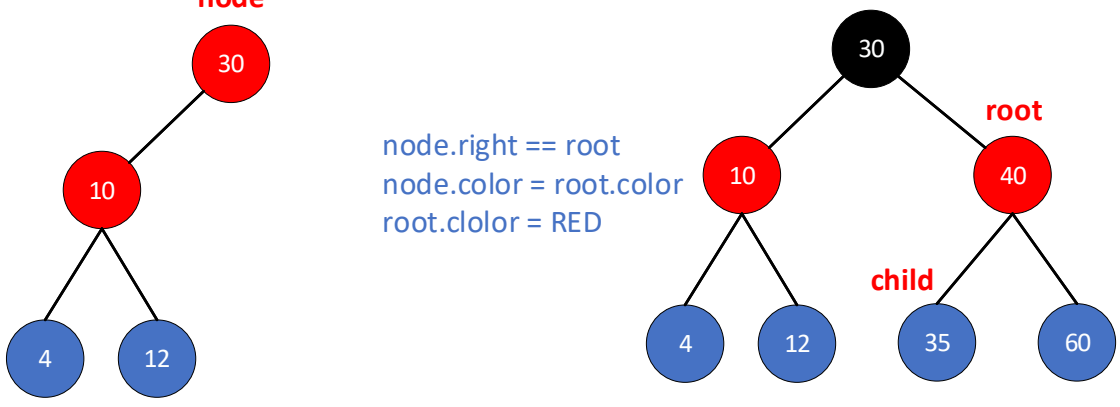


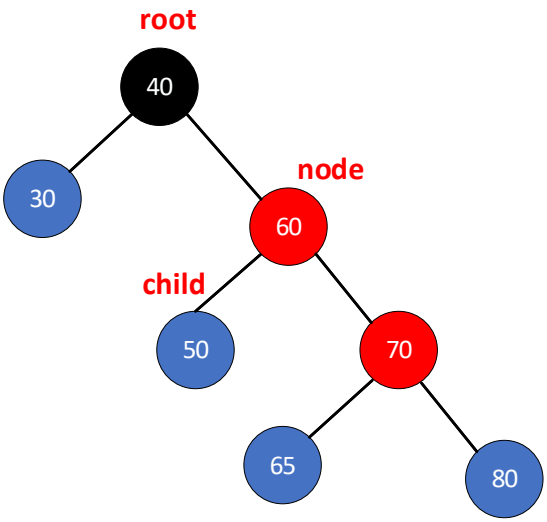


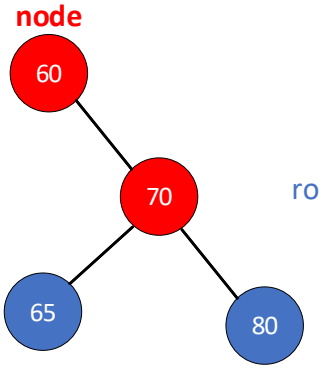

60
60
37
60
70


root
40
node
30
60
child
10
35
4
12

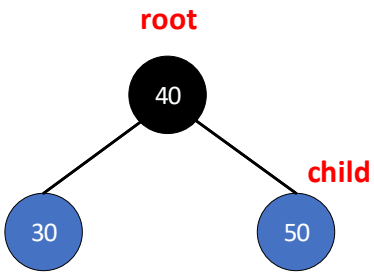

node
30
10
4
12
root.left == child
root
40
child
35
60

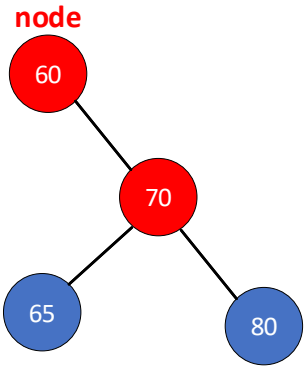

node
30
10
4
12
node.right == root
node.color = root.color
root.clolor = RED
30
root
10
40
child
4
12
35
60





root.right == child





node.left == root
node.color = root.color
root.clolor = RED