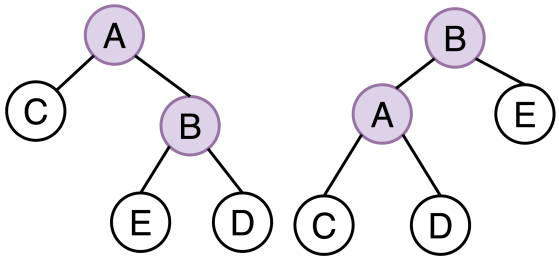
Red Black Tree

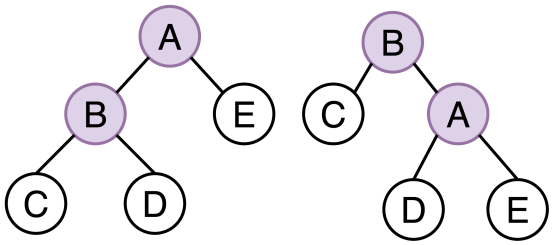
1. Root is always black
2. No two adjacent 临近的 nodes are red
3. Any path between a node and any descendant (lower) node 子孙 has the same number of black nodes

Rotate

Left



Right



Insert

Root

Change colour to black

Violated 2 & Uncle is red

- Change parent and uncle to black
- Change grandparent to red
- Make grandparent n, and repeat

Violated 2 & Uncle is black

Left Left

rotate right, swap colours of parent and grandparent

Left Right

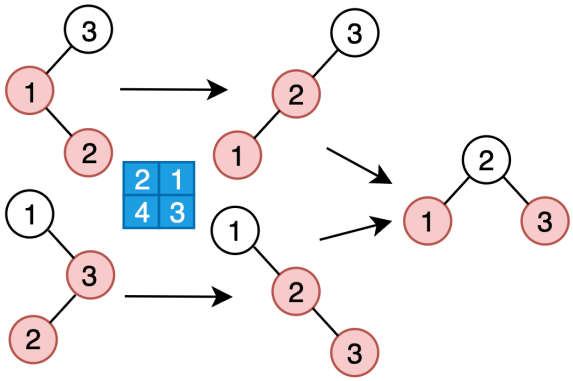
rotate left, then right, swap colours of new node and grandparent

Right Right

rotate left, swap colours of parent and grandparent

Right Left

rotate right, then left, swap colours of new node and grandparent



Delete

Simple Cases

- If a node is red with nullptr child (no children)
- If a node has 1 child and **either** *the node* OR *child* (but not both) is red

1. Delete node 2. Updated node ⟶ **black** (node replaced the deleted node)

Double Black

If both *node to be deleted* AND *child* are **black** (or the node has no children), the *updated node* becomes **double black**

- n's *sibling* is **black**
  - with at least one **red** *child*
    1. Rotate (as per insertion following path to red *child*)
    2. Recolour red *child* to **black**, *sibling* to red
  - with *all children* **black**
    1. Recolour sibling to red
    2. Push **black** up (**black** *parent* ⟶ **double black**, red *parent* ⟶ **black**)
- n's *sibling* is **red**
  1. Rotate
  2. Recolour *Sibling* to **black** *Parent* to red