- balanced BST
- balanced ST
- 红黑树
  - Insert（z）
  - fix（5）
    - case00：黑父亲
    - case0：是root
    - case1：红父亲，红叔叔
    - case2：红父亲，黑叔叔
    - case3：红父亲，黑叔叔
    - Time
  - Remove
  - Fix
    - case1：x红兄弟
    - case2：黑兄弟，两个黑侄子
    - case3:黑兄弟，左侄子红，右侄子黑
    - case4:黑兄弟，右侄子红
- 跳表SkipList
  - 随机插入

balanced BST

> 判断：树高logn

balanced ST

- AVL tree (Adelson-Velsii & Landis, 1962)
- B-tree (Bayer & McCreight, 1970)
- Red-black tree (Bayer, 1972)
- Splay tree (Sleator & Tarjan, 1985)
- Treap (Seidel & Aragon, 1996)
- Skip list (Pugh, 1989)
- and so on …

Efficient implementation of OSet

	`Search(S,k)`	`Insert(S,x)`	`Remove(S,x)`
`BinarySearchTree`	$O(h)$ worst-case	$O(h)$ worst-case	$O(h)$ worst-case
`Treap`	$O(\log n)$ in expectation	$O(\log n)$ in expectation	$O(\log n)$ in expectation
`RB-Tree`	$O(\log n)$ worst-case	$O(\log n)$ worst-case	$O(\log n)$ worst-case
`SkipList`	$O(\log n)$ in expectation	$O(\log n)$ in expectation	$O(\log n)$ in expectation

红黑树

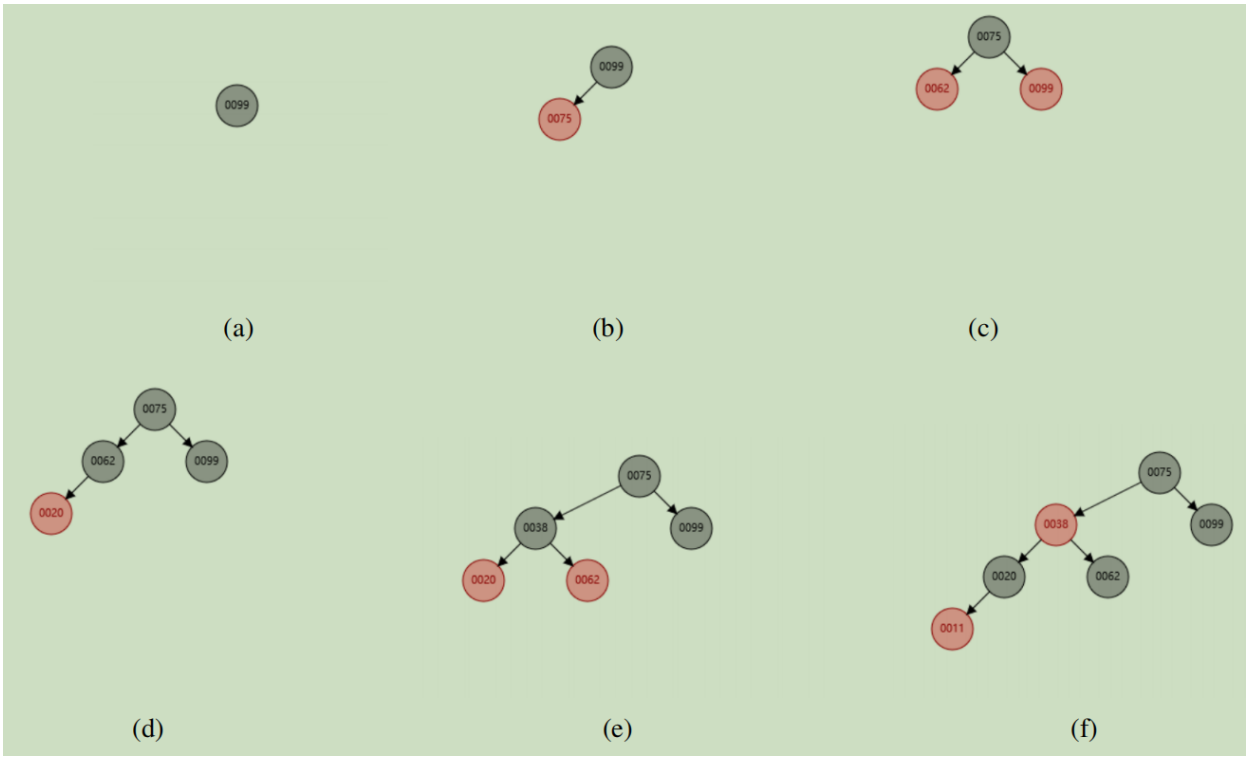
- 根叶黑
- 无父子红：内结点红，则两孩子黑
- 黑高度bh一致：所有路径上黑结点数目一致

> 可以把所有黑-红放平

Insert（z）

- **维持黑高度，当作红点插入，以BST方式**

例题ps6-4




```
c：case3：红父亲，无叔叔，左孩子：右旋
d：case1：红父亲，红叔叔：父子换色
e：case2：红父亲，无叔叔，右孩子：左旋
    case3：（把20当作新插入）：右旋
f：case1：父子换色
```

fix（5）

> 进入case0则结束
> case1可能多次执行，每次进入case1，都向根靠近一步
> case23可能改变树形状，但只会一次
> rotate只改变值的位置，颜色不跟随rotate旋转

case00：黑父亲

- 则不需修改

case0：是root

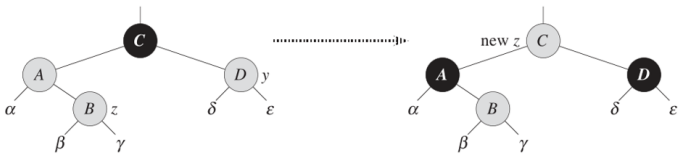
- 则染为黑

case1：红父亲，红叔叔

- 则使父亲叔叔和祖父换色

> 先维持黑高度不变

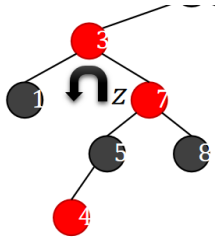
> 但祖父和曾祖父可能出现红边：把风险向上推了两层
> 进入下一种case

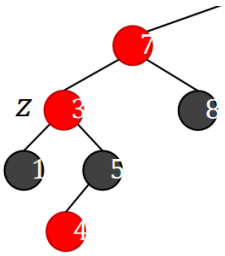


case2：红父亲，黑/无叔叔

- z是右孩子，红边靠右

> 在红父亲处left-rotate
> 再进入下一种case

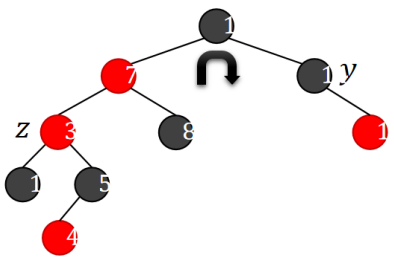


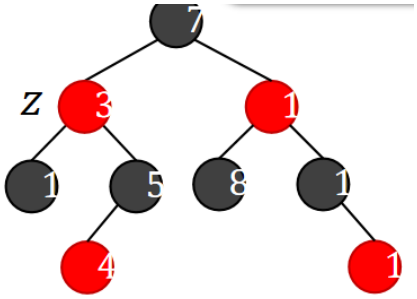


case3：红父亲，黑/无叔叔

- z是左孩子，红边靠左

> 在黑祖父处right-rotate





Time

- O（h）=O（logn）
- O（1）rotation

> 对树影响有限

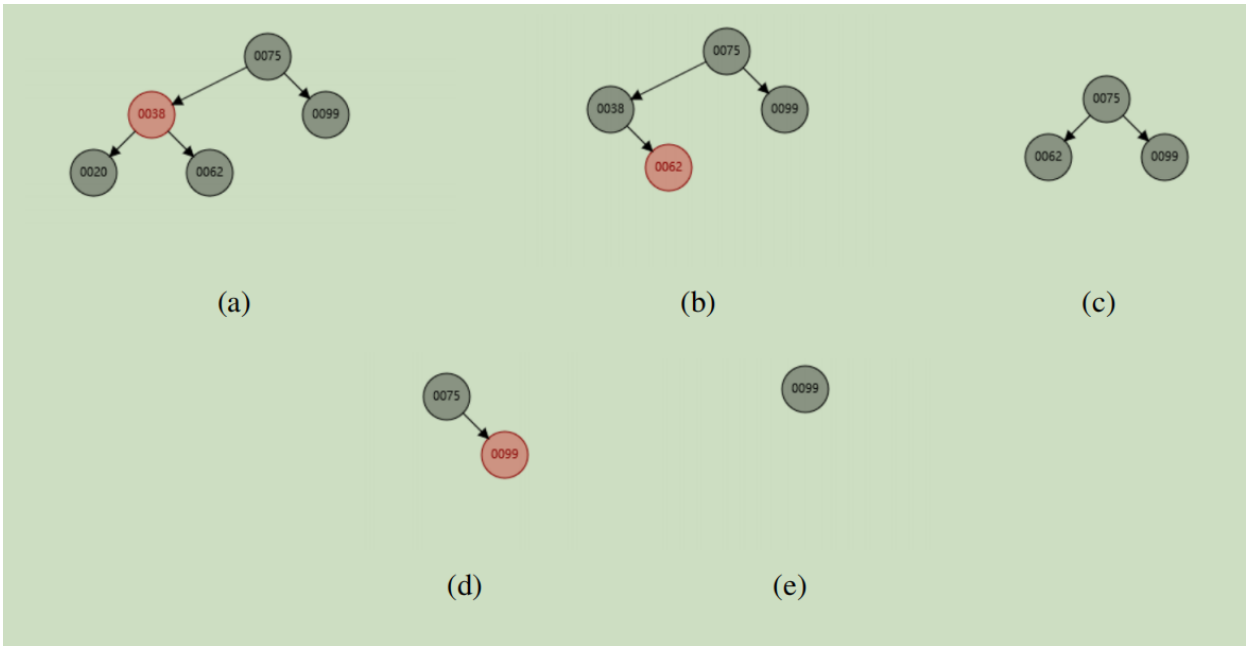
Remove

辨识删除部分

> 维持y是要删除或要移位的结点

- 若z右孩子空（external）：左子树代替z位置，y=z
- 否则（internal）：

> 记y为z右子树最小点，则y无左孩子，将y的值放在z，颜色使用z的y右子树（x）代替y



Fix

- y红：正常提上来x，无操作
- y黑x红：x变黑

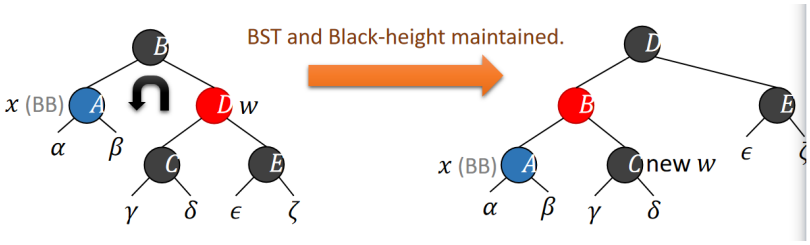
- y黑x黑（减少了一个黑高度）：case1-4

> 此时x已在原y的位置

case1：x红兄弟

- leftrotate，父兄换色

> 变为case2-4，x的新兄弟必黑



case2：黑兄弟，两个黑侄子

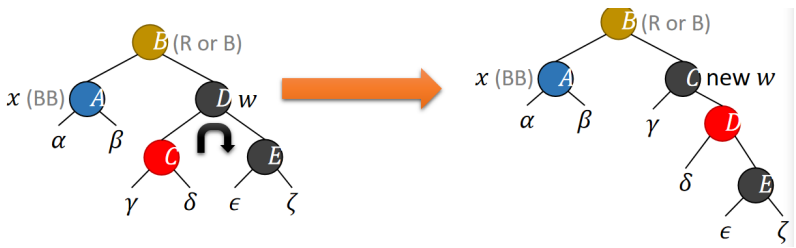
- 兄弟变红，x变成父亲

> 效果：？？？？

case3:黑兄弟，左侄子红，右侄子黑

- 对黑兄弟右rotate，右兄弟和左侄子换色

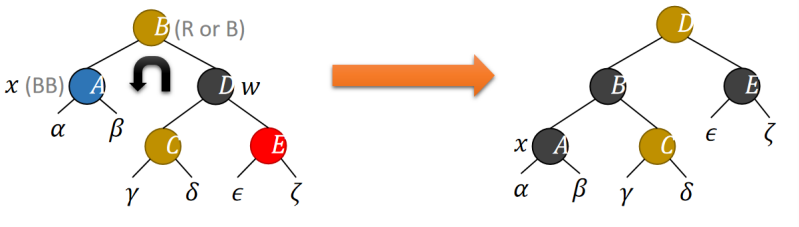
> 进入case4



case4:黑兄弟，右侄子红

- 对父亲左rotate，右侄子变黑

> 左面黑高度加1.右侧不变



跳表SkipList

> 像搜索树
>
> 实现简单，空间换时间
>
> 空间大概2*n

随机插入

> 不很均匀，但不影响查找

Insert(L,x):

```
level = 1, done = false
while (!done)
  Insert x into level k list.
  Flip a fair coin:
    With probability 1/2: done = true
    With probability 1/2: k = k+1
```