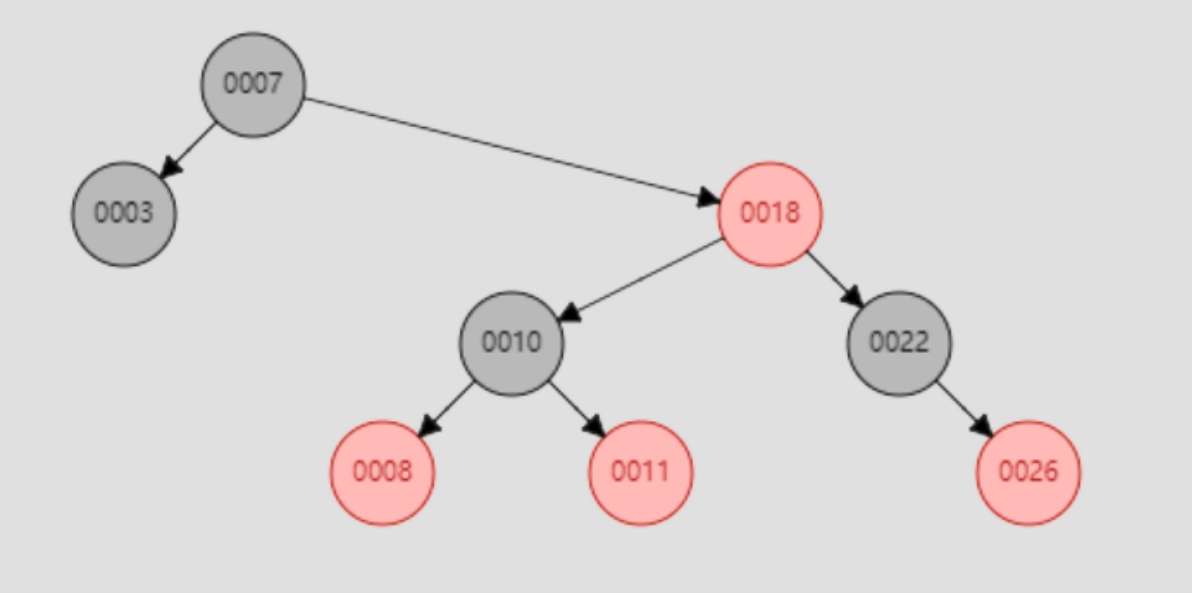
522031910747+李若彬+hw5

1.代码正确性证明

> `main.cpp`中给出了一个插入案例，插入顺序为7， 3， 18， 10， 22， 8， 11， 26。
>
> 1. 画出插入后的红黑树
> 2. 修改`inorder`函数，使该函数对树遍历时，将结点的颜色一同输出，并确保其颜色正确

红黑树:



程序运行结果:

```
PS D:\onedrive_edu\OneDrive - sjtu.edu.cn\SJTU\grade2-II\ADS\hw\hw5> .\test.exe
Inorder traversal of the constructed tree:
3 BLACK
7 BLACK
8 RED
10 BLACK
11 RED
18 RED
22 BLACK
26 RED
```

2.删除操作分析

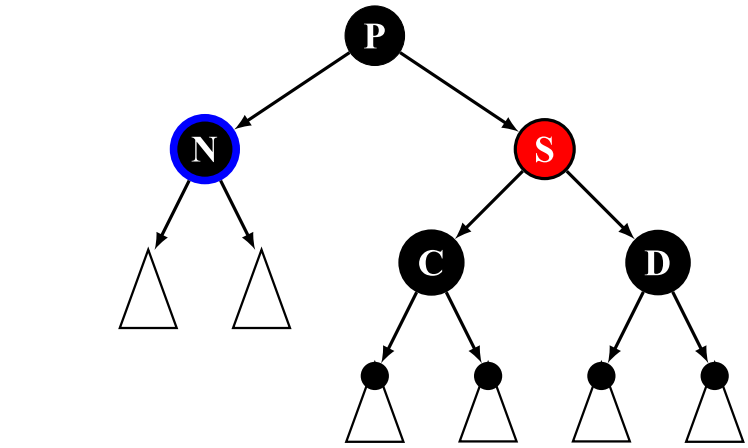
> 请按照hw4中的Part 2第一问的要求，解释删除后采取的失衡恢复操作，是如何使得红黑树依然维持其性质

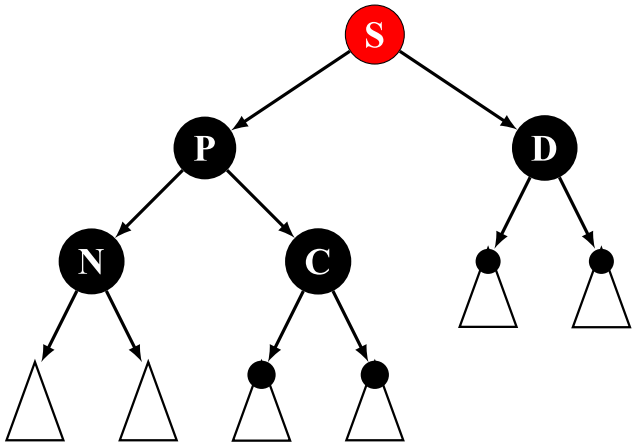
Case 1:

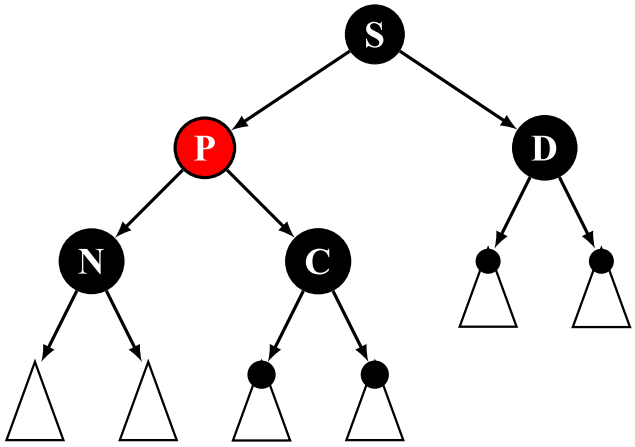
兄弟节点 (sibling node) S 为红色，则父节点 P 和侄节点 (nephew node) C 和 D 必为黑色。与这种情况下无法通过直接的旋转或染色操作使其满足所有性质，因此通过前置操作优先保证部分结构满足性质，再进行后续维护即可。

这种情况的维护需要：

1. 若待删除节点 N 为左子节点，左旋 P; 若为右子节点，右旋 P。
2. 将 S 染黑，P 染红。
3. 此时只需根据结构对以当前 P 节点为根的子树进行维护即可（无需再考虑旋转染色后的 S 和 D 节点）。

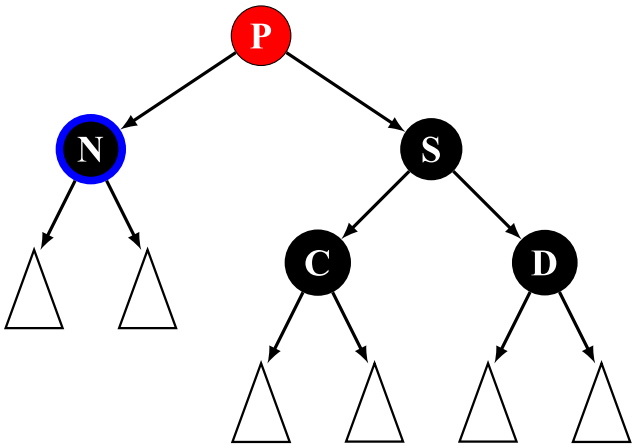


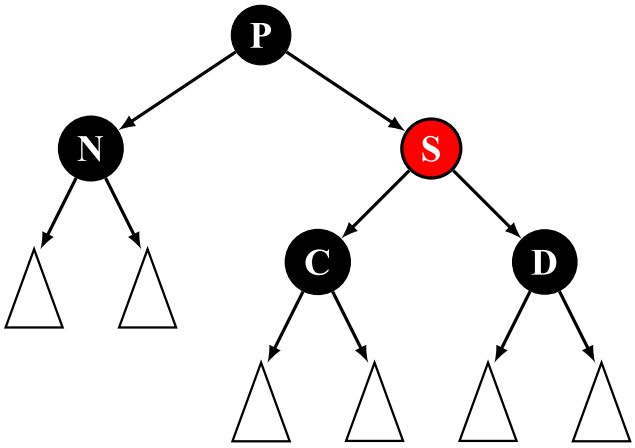




Case 2:

兄弟节点 S 和侄节点 C, D 均为黑色，父节点 P 为红色。此时只需将 S 染红，将 P 染黑。

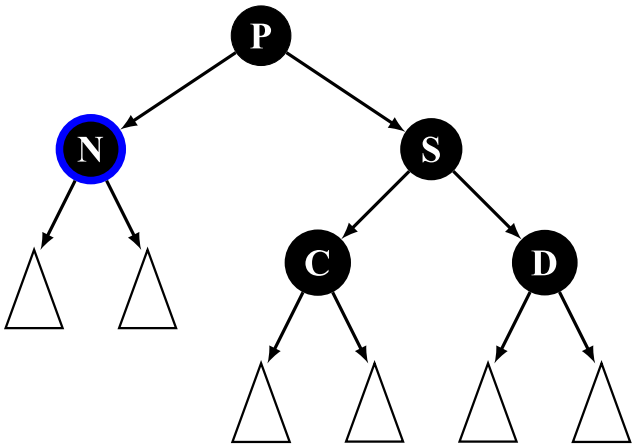


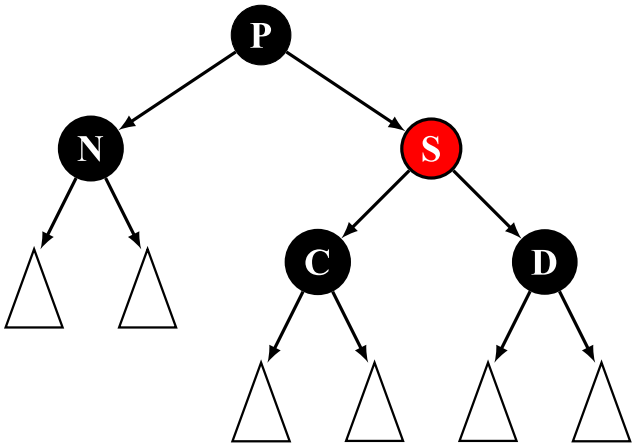


Case 3：

兄弟节点 S，父节点 P 以及侄节点 C, D 均为黑色。

此时选择将 S 染红，再递归维护 P 节点根据上部结构进行后续维护。





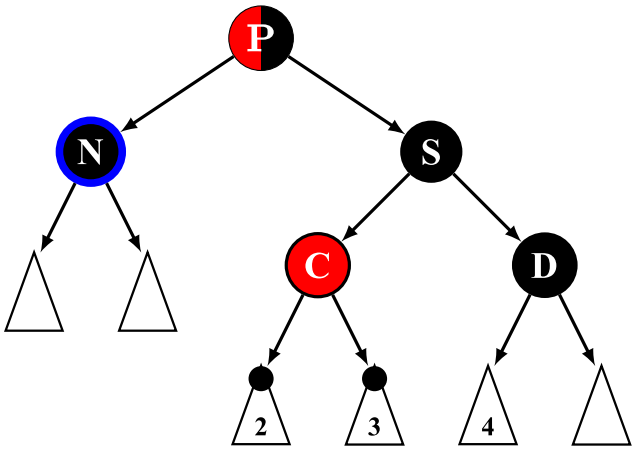
Case 4:

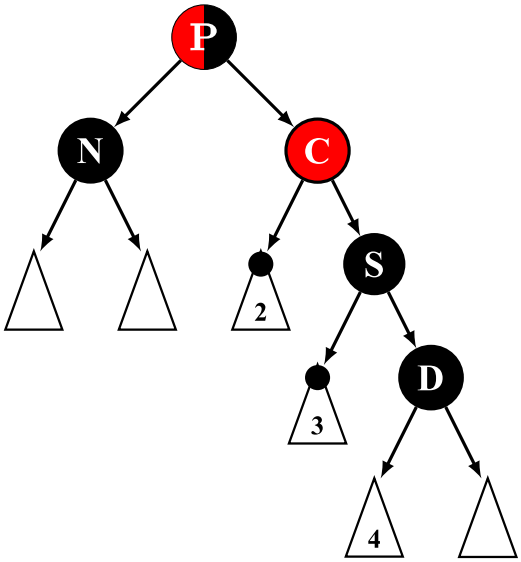
兄弟节点是黑色，且与 N 同向的侄节点 C（由于没有固定中文翻译，下文还是统一将其称作 close nephew）为红色，与 N 反向的侄节点 D（同理，下文称作 distant nephew）为黑色，父节点既可为红色又可为黑色。

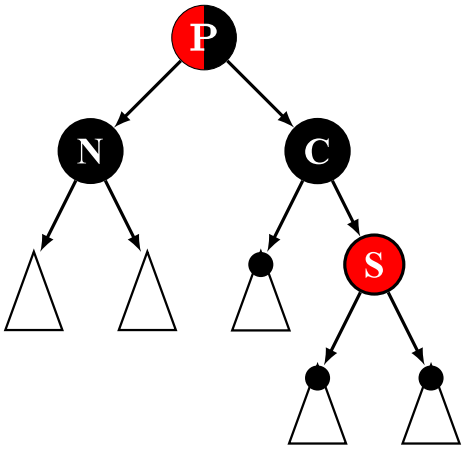
此时同样无法通过一步操作使其满足性质，因此优先选择将其转变为 Case 5 的状态利用后续 Case 5 的维护过程进行修正。

该过程分为三步：

1. 若 N 为左子节点，右旋 P，否则左旋 P。
2. 将节点 S 染红，将节点 C 染黑。
3. 此时已满足 Case 5 的条件，进入 Case 5 完成后续维护。



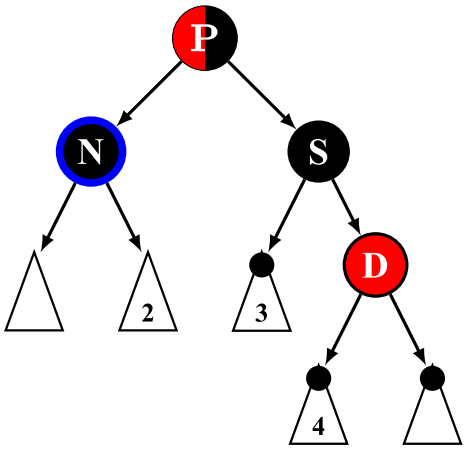


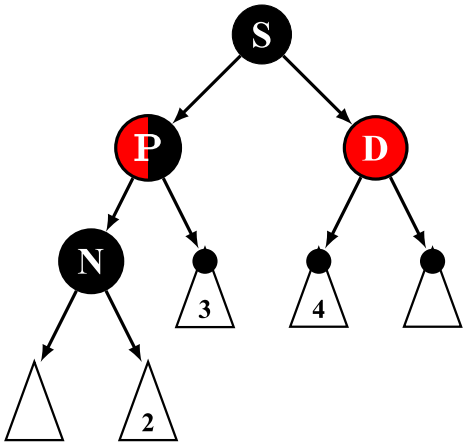


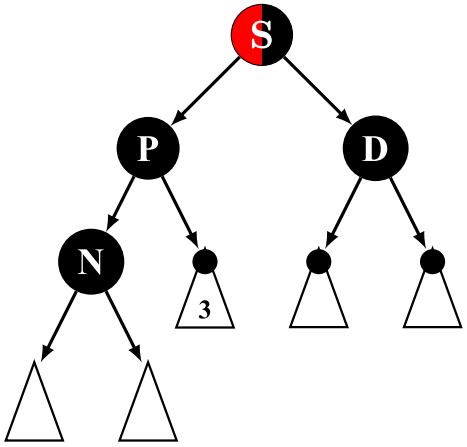
Case 5：

兄弟节点是黑色，且 close nephew 节点 C 为黑色，distant nephew 节点 D 为红色，父节点既可为红色又可为黑色。此时通过旋转操作使得黑色节点 S 变为该子树的根节点再进行染色。具体步骤如下：

1. 若 N 为左子节点，左旋 P，反之右旋 P。
2. 交换父节点 P 和兄弟节点 S 的颜色。
3. 将 distant nephew 节点 D 染黑。







3.对顺序和乱序插入的实验结果统计与分析

> hw4中曾问道，红黑树顺序插入和乱序插入会有何影响。现在，请设计实验进行探究，具体的，你应该
>
> 1. 顺序以及乱序插入10000个元素（请多次执行乱序插入）
> 2. 统计一共发生了多少次失衡恢复，失衡恢复中一共执行了多少次染色，多少次旋转
> 3. 根据所得实验结果，回答顺序插入和乱序插入对红黑树插入的影响

顺序插入：

```
PS D:\onedrive_edu\OneDrive - sjtu.edu.cn\SJTU\grade2-II\ADS\hw\hw5> .\test.exe
Sequential insertion:
fixcount: 19947
rotatecount: 9976
dyecount: 59865
```

乱序插入：

```
PS D:\onedrive_edu\OneDrive - sjtu.edu.cn\SJTU\grade2-II\ADS\hw\hw5> .\test.exe
Disorderly insertion:
fixcount: 9099
rotatecount: 5931
dyecount: 33340
```

分析

由以上实验结果可以知道：

- 乱序插入的失衡恢复、染色，旋转次数都明显少于顺序插入
- 由于顺序插入会导致树的结构趋于链表化，因此顺序插入的情况下，其调整的情况更加复杂
- 乱序插入使得数据的分布更加分散，因此需要调整的情况与顺序插入相比大大减少