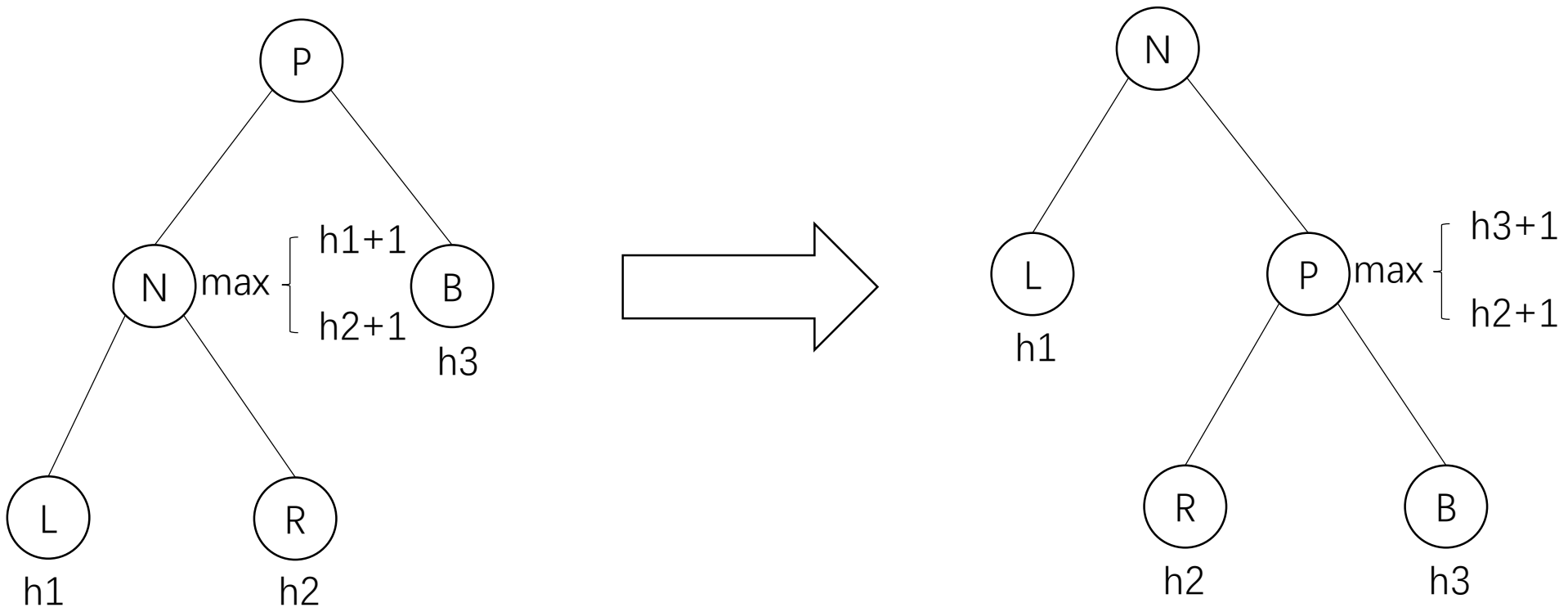
红黑树基本性质

1. 每个节点或是红色的，或是黑色的
2. 根节点是黑色的
3. 每个叶节点（null或NIL）是黑色的
4. 如果一个节点是红色的，则它的两个子节点都是黑色的
5. 对每个节点，从该节点到其所有后代叶节点的简单路径上，均包含相同数据的黑节点

- 红黑树是非严格的平衡二叉搜索树，根节点到所有叶子节点最高可能高度和最低可能高度相差不会超过2，在元素插入删除、和查询操作之间保持了平衡。
- 性质1、2、3可以说是前提设定，非常简单就可以满足。在插入删除操作中，需要重点维护的是性质4和性质5。

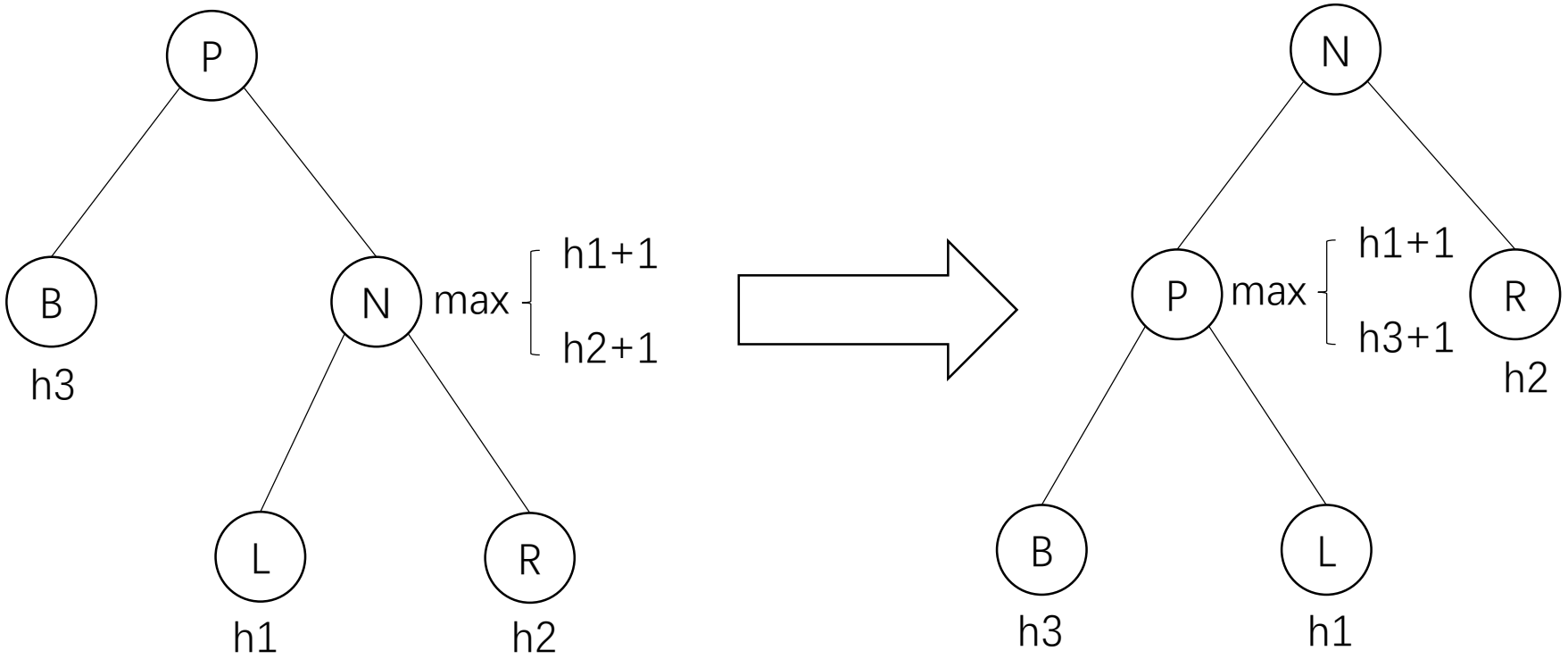
红黑树的基本操作 – 左旋

h1,h2,h3表示树高。可以看到左旋之后左子树高至少-1，右子树高至少+1。因此，对节点p的左旋操作可以调整p的左右子树的高度。



红黑树的基本操作 – 右旋

h1,h2,h3表示树高。可以看到右旋之后右子树高至少-1，左子树高至少+1。因此，对节点p的右旋操作可以调整p的左右子树的高度。

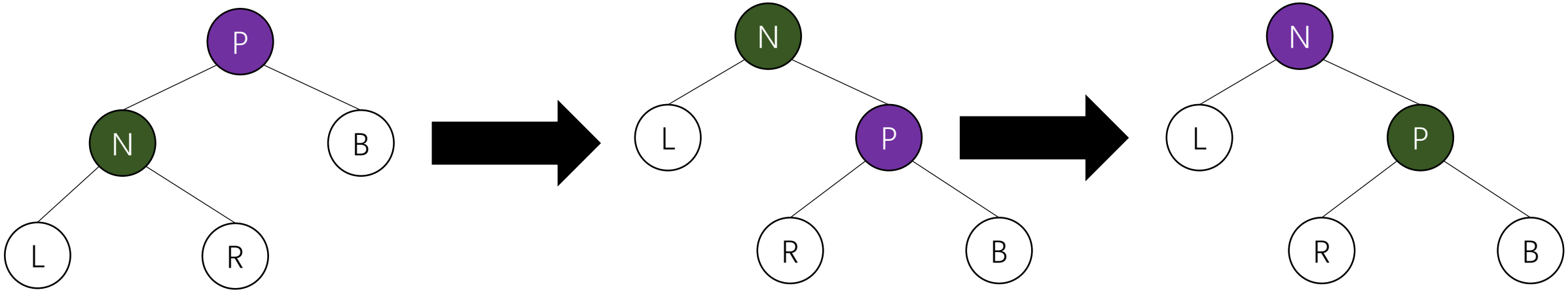


红黑树的基本操作 – 左旋+交换颜色

旋转之后交换颜色的操作在插入和删除中频繁出现。如果当前节点为父节点的左子节点，则**左旋+交换**当前节点与父节点的颜色；否则，**右旋+交换**当前节点与父节点的颜色。

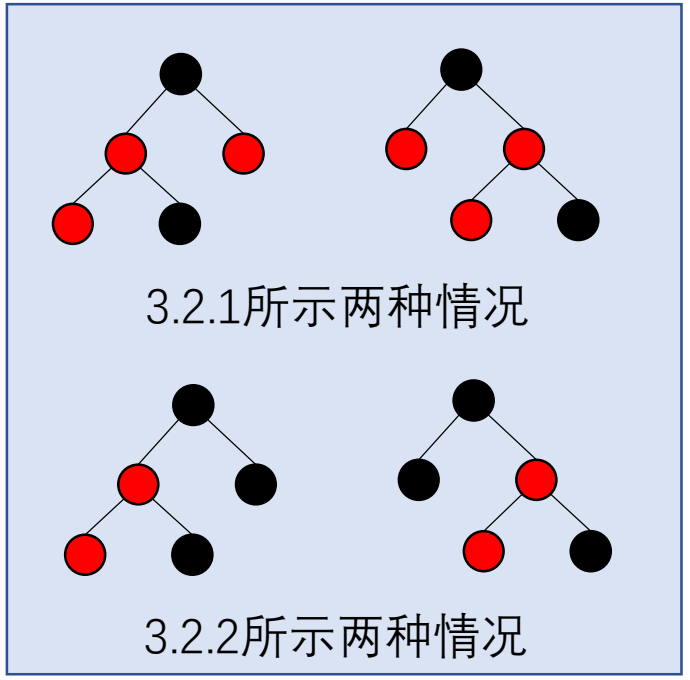
如下图所示，根据最终结果，可以发现：

1、如果绿色代表的是红黑树中的黑色，则最终左子树黑色节点-1，右子树黑色节点+1；
2、如果绿色代表的是红黑树中的红色，则最终左右子树路径的黑色节点数目不变；
3、在插入操作中，如果紫色代表的是红黑树中的黑色，则表示操作可以在此处中止；否则要继续向上追溯；
4、在删除操作中，如果紫色代表的是红黑树中的黑色，则表示双黑性质需要向上传递；否则表示操作中止。



红黑树方法 – 插入

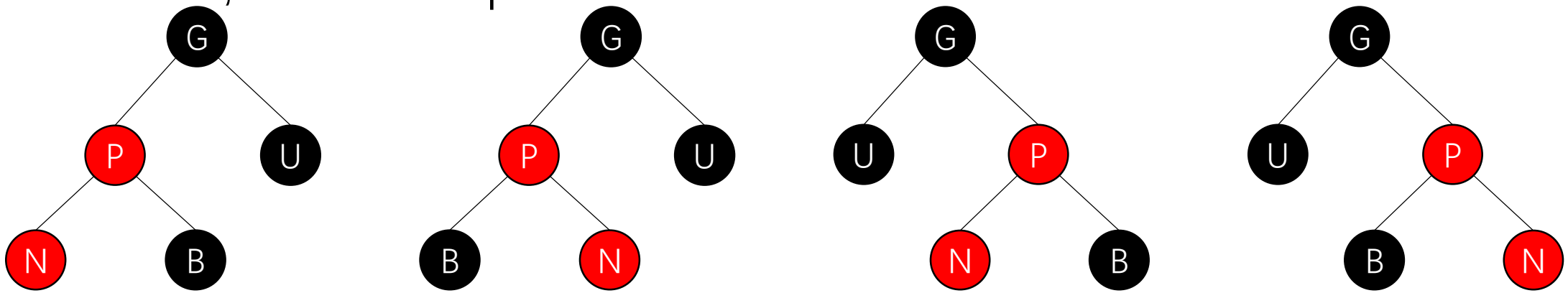
- 插入一个节点
  1. 首先寻找插入节点的位置：
     1. 如果插入节点是root节点，则直接插入为黑色节点，插入结束；
     2. 如果插入节点不是root节点，则进行下一步。
  2. 其次判断插入节点的颜色
     1. 如果插入节点颜色为黑色，则肯定违反性质5。舍弃该方案。
     2. 如果插入节点颜色设定为红色，则可能违反性质4。进行下一步。
  3. 插入节点后，判断父节点颜色
     1. 如果父节点是黑色，则没有违反性质4，插入结束；
     2. 如果父节点是红色，则违反性质4。此时必有兄弟节点是黑色，祖父节点是黑色。但是叔叔节点未知
        1. 如果叔叔节点是红色，则父节点和叔叔节点同为红色（如右上所示），可以同时变色为黑色以满足性质4；同时将祖父节点变为红色，以满足性质5。此时，因为祖父节点为红色，所以从祖父节点向上可能违反性质4，继续以祖父节点为当前节点，重复步骤3。
        2. 如果叔叔节点是黑色，则父节点和叔叔节点不同色（如右上所示）。此时如果单纯进行如上的变色必然会在修复性质4的同时使得叔叔子树违反性质5。此时只能在叔叔子树上进行黑色节点数目+1的修复，能够很快修复的可能性很小。下面一页我们专门讨论这种情况。



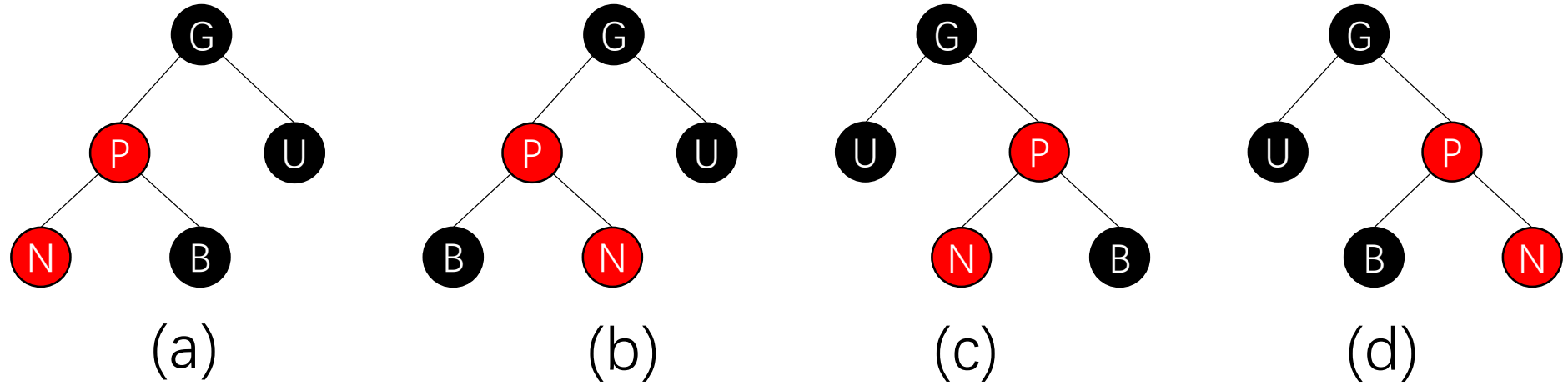
红黑树方法 – 插入

3.2.2

- 在这种情况下，可以确保的颜色为：本节点红色，兄弟节点黑色，父节点红色，叔叔节点黑色，祖父节点黑色。
- 在这种情况下，由于有三代参与，每两代之间有左右孩子两种可能，所以一共有下述四种可能。N=Node，B=Bro，P=Parent，U=uncle，G=Grandpa

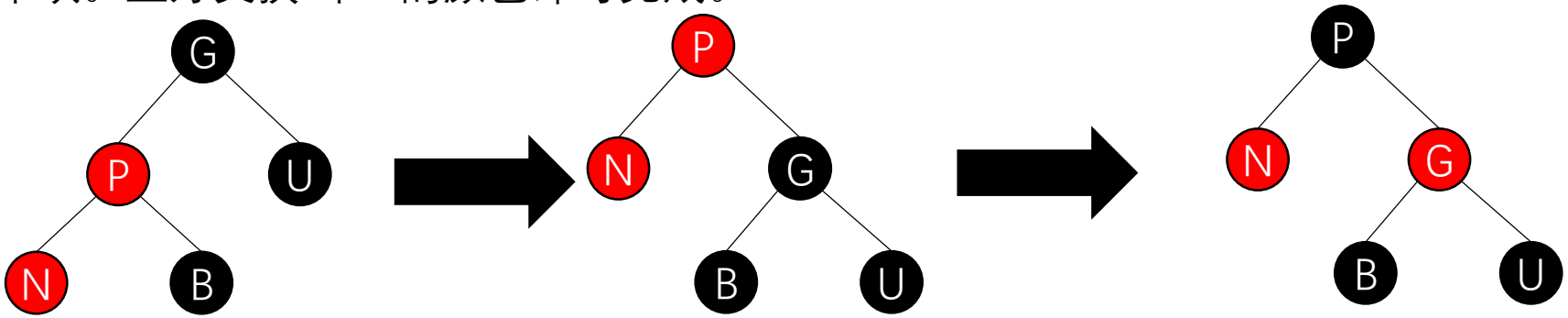


红黑树方法 – 插入

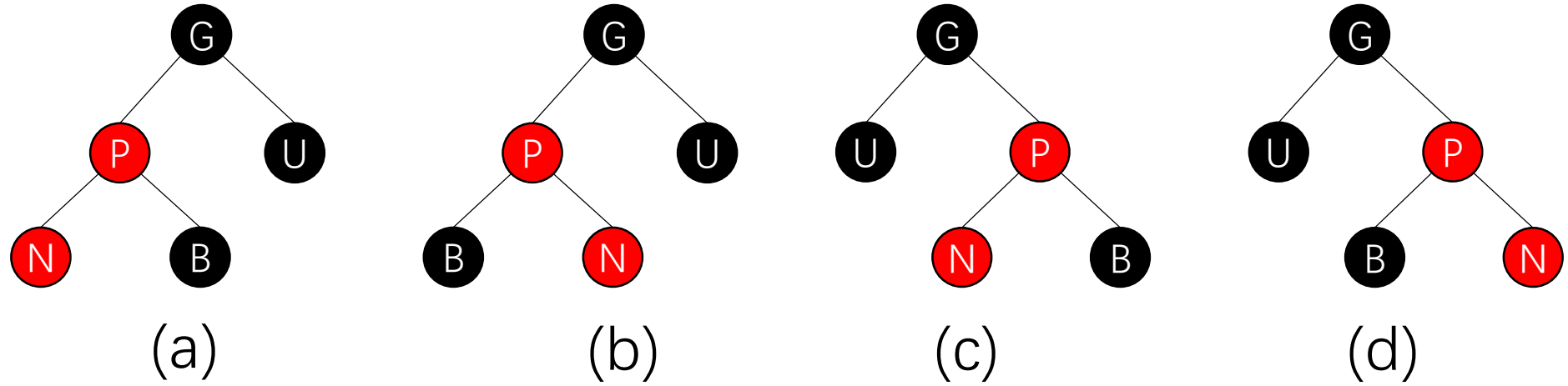


上述abcd四种情况中，a和d对称，b和c对称。因此我们直接处理a和b。

- 对于**场景a**来说，考虑旋转操作。
  - 如果旋转P，左旋和右旋分别将红色冲突的问题下移和右移，对于问题解决没有帮助。
  - 如果旋转G，左旋会对该子树结构造成更大的破坏。因此尝试**右旋**。右旋之后发现左子树缺一黑，右子树不缺。正好交换P和G的颜色即可完成。

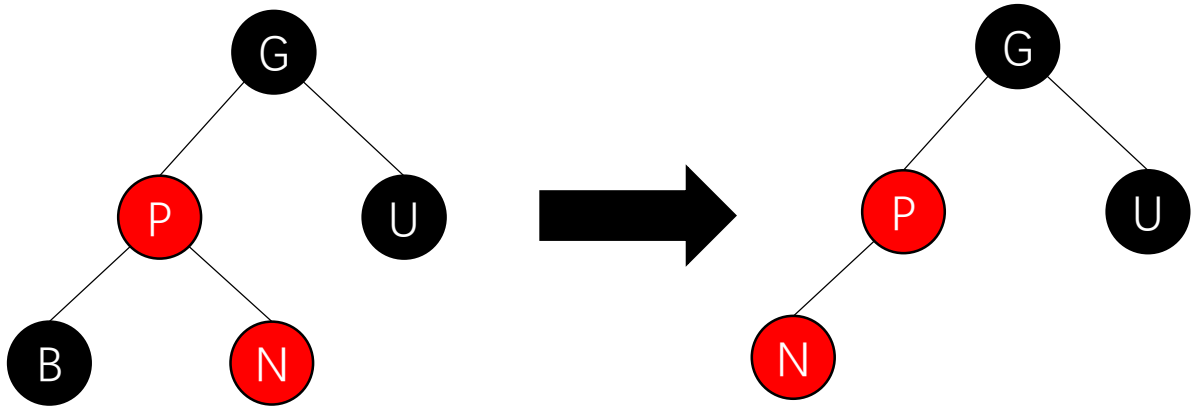


红黑树方法 – 插入



上述abcd四种情况中，a和d对称，b和c对称。因此我们直接处理a和b。

- 对于**场景b**来说，考虑旋转操作。
  - 如果旋转P，**左旋**可以变换为**场景a**（B是黑色，在左旋中不影响性质，因此无需考虑）。



红黑树方法 – 删除

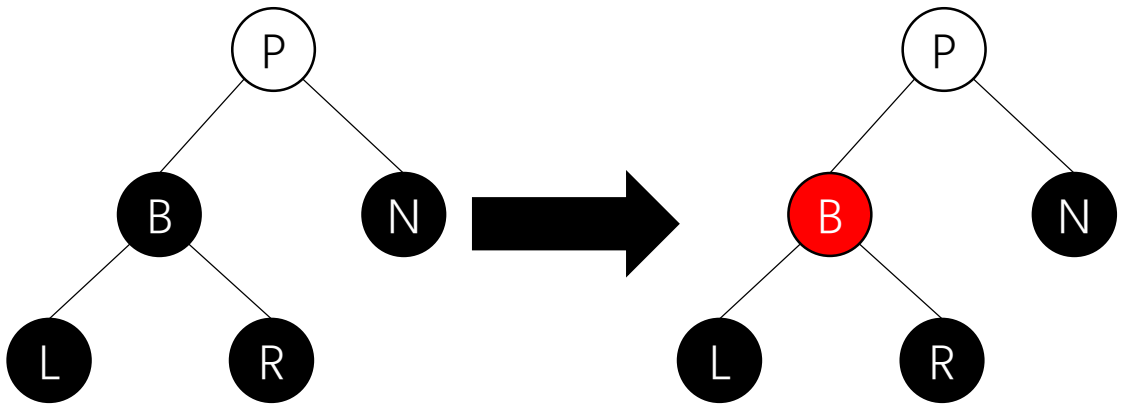
- 删除操作分为以下几步：
  1. 判断被删除节点是否存在，如果不存在，删除结束；如果存在，进行下一步
  2. 节点删除首先会影响到被删除节点后代。因此判断删除节点的后代情况：
     1. 如果删除节点有两个后代：表示节点删除会涉及到左右两个子树。寻找其中序遍历的下一个节点为替换节点，然后交换两者的值。再进行第二步的判断。
     2. 如果删除节点是root，此时root最多有一个后代，删除操作简单。
     3. 如果删除节点没有后代：表示删除节点是叶子几点，没有替换节点，或者替换节点是黑色null节点。
        1. 如果删除节点是红色，直接删除不影响性质5，删除操作结束。
        2. 如果删除节点是黑色，直接删除影响性质5，导致删除节点子树黑色节点少1，因此要在此子树上增加黑色节点数目，对被删除节点进行双黑处理。
     4. 如果删除节点只有一个后代：毫无疑问，该后代就是替换节点。
        1. 如果这两个节点不都是黑色，则直接交换键值，再根据两个节点的颜色选择是否染黑替换节点，最后删除节点。删除操作结束
        2. 如果这两个节点都是黑色，则删除操作必定会导致删除节点子树黑色节点少1，因此要对被删除节点进行双黑处理。

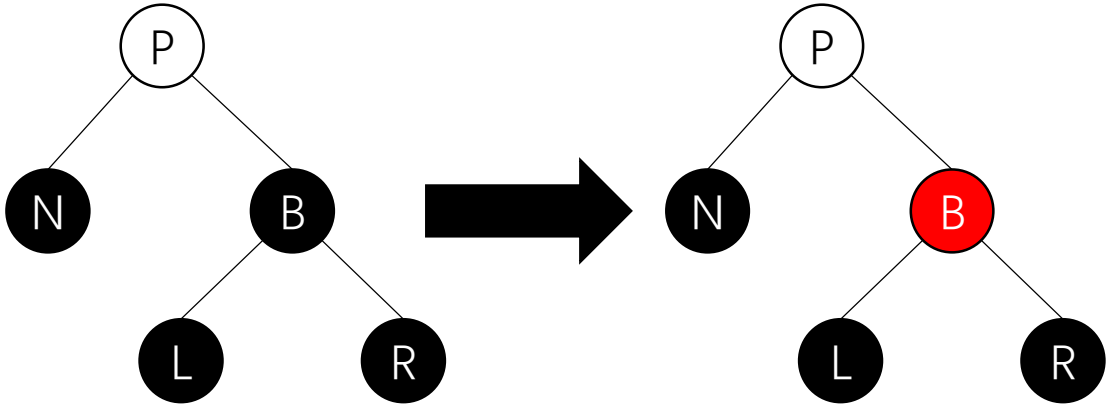
红黑树方法 – 双黑处理（全黑）

何为双黑?

双黑指的是一种节点性质。我们判断某个节点具有双黑性质，指的是经过该节点的路径上黑色节点数目比其他路径要少1。因此要想办法对该节点极其以上的节点进行调整。

1. 如果该节点是root节点，则直接染黑节点即可;
2. 如果该节点是红色节点，则直接染黑节点即可;
3. 如果该节点是黑色节点，则需根据其兄弟节点和父节点的情况，判断是否可以染色:
    1. 如果兄弟节点为黑色:
        1. 如果兄弟节点的左右子节点都为黑色。则此时可以无视父节点的情况，直接将兄弟节点染红，使得经过兄弟节点B的路径上黑色节点数目比其他路径少1。颜色变化以后，兄弟节点和当前节点情况一致，因此可以把双黑性质向上推给父亲节点，循环向上。





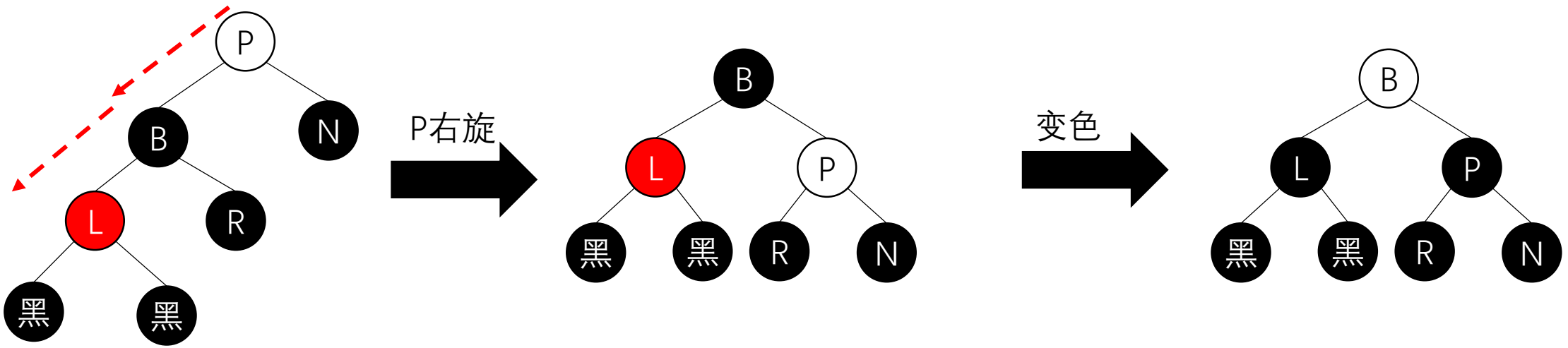
红黑树方法 – 双黑处理（一红一黑）

何为双黑?

双黑指的是一种节点性质。我们判断某个节点具有双黑性质，指的是经过该节点的路径上黑色节点数目比其他路径要少1。因此要想办法对该节点极其以上的节点进行调整。

1. 如果该节点是root节点，则直接染黑节点即可;
2. 如果该节点是红色节点，则直接染黑节点即可;
3. 如果该节点是黑色节点，则需根据其兄弟节点和父节点的情况，判断是否可以染色:
    1. 如果兄弟节点为黑色:
        1. 如果兄弟节点的左右子节点都为黑色。
        2. 兄弟节点的左右子节点只有一个为红色。如果红色节点可以和兄弟节点以及父节点形成直线型，则旋转变色，为节点N所在子树的路径增加黑色节点。

直线型1



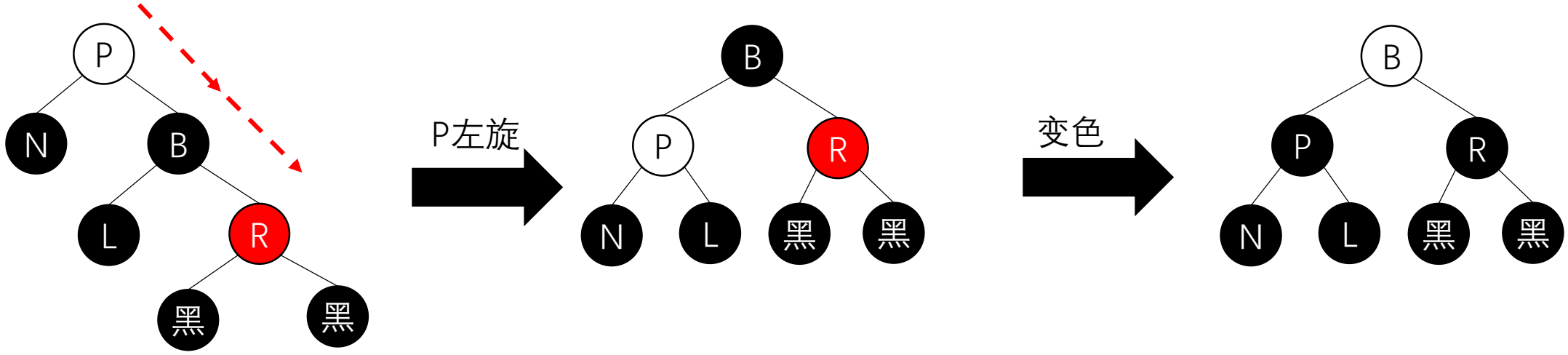
红黑树方法 – 双黑处理（一红一黑）

何为双黑?

双黑指的是一种节点性质。我们判断某个节点具有双黑性质，指的是经过该节点的路径上黑色节点数目比其他路径要少1。因此要想办法对该节点极其以上的节点进行调整。

1. 如果该节点是root节点，则直接染黑节点即可;
2. 如果该节点是红色节点，则直接染黑节点即可;
3. 如果该节点是黑色节点，则需根据其兄弟节点和父节点的情况，判断是否可以染色:
    1. 如果兄弟节点为黑色:
        1. 如果兄弟节点的左右子节点都为黑色。
        2. 兄弟节点的左右子节点只有一个为红色。如果红色节点可以和兄弟节点以及父节点形成直线型，则旋转变色，为节点N所在子树的路径增加黑色节点。

直线型2



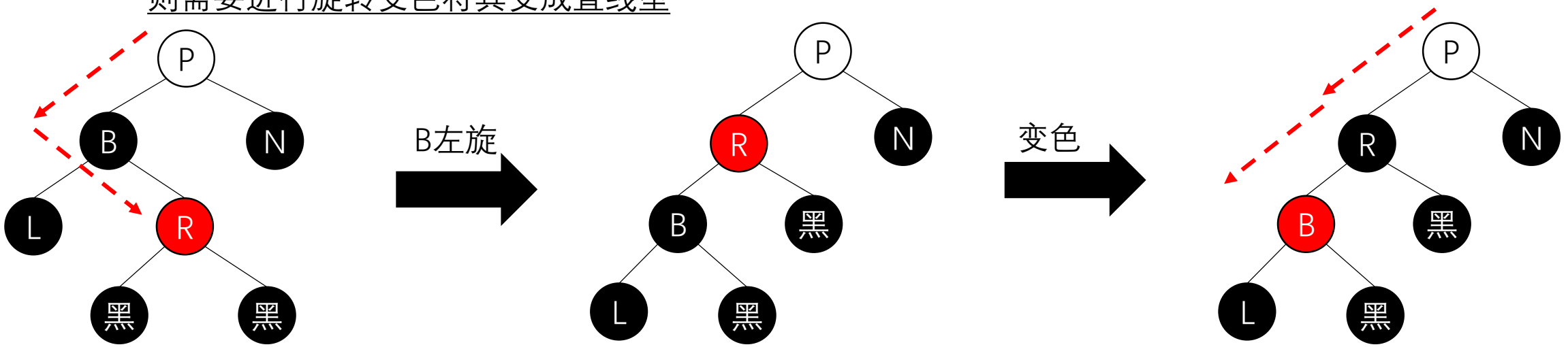
红黑树方法 – 双黑处理（一红一黑）

何为双黑?

双黑指的是一种节点性质。我们判断某个节点具有双黑性质，指的是经过该节点的路径上黑色节点数目比其他路径要少1。因此要想办法对该节点极其以上的节点进行调整。

1. 如果该节点是root节点，则直接染黑节点即可;
2. 如果该节点是红色节点，则直接染黑节点即可;
3. 如果该节点是黑色节点，则需根据其兄弟节点和父节点的情况，判断是否可以染色:
    1. 如果兄弟节点为黑色:
        1. 如果兄弟节点的左右子节点都为黑色。
        2. 兄弟节点的左右子节点只有一个为红色。如果红色节点可以和兄弟节点以及父节点形成折线型，则需要进行旋转变色将其变成直线型

折线型1



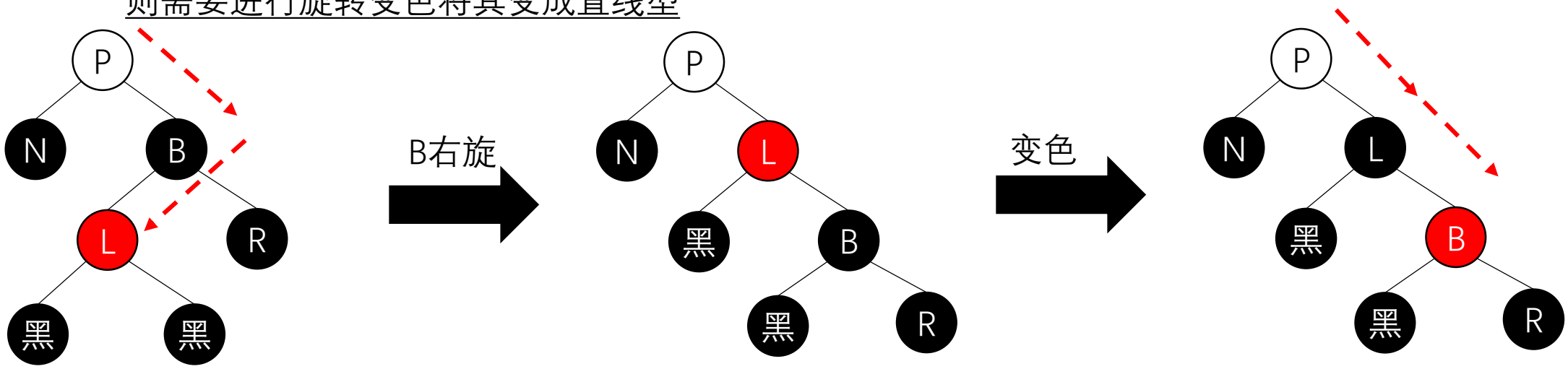
红黑树方法 – 双黑处理（一红一黑）

何为双黑?

双黑指的是一种节点性质。我们判断某个节点具有双黑性质，指的是经过该节点的路径上黑色节点数目比其他路径要少1。因此要想办法对该节点极其以上的节点进行调整。

1. 如果该节点是root节点，则直接染黑节点即可;
2. 如果该节点是红色节点，则直接染黑节点即可;
3. 如果该节点是黑色节点，则需根据其兄弟节点和父节点的情况，判断是否可以染色:
    1. 如果兄弟节点为黑色:
        1. 如果兄弟节点的左右子节点都为黑色。
        2. 兄弟节点的左右子节点只有一个为红色。如果红色节点可以和兄弟节点以及父节点形成折线型，则需要进行旋转变色将其变成直线型

折线型2



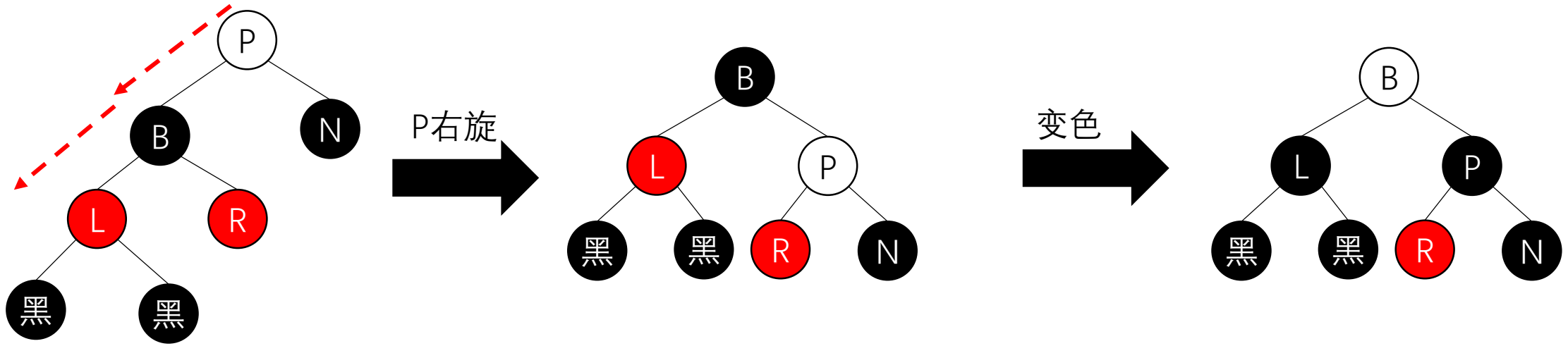
红黑树方法 – 双黑处理（双红）

何为双黑?

双黑指的是一种节点性质。我们判断某个节点具有双黑性质，指的是经过该节点的路径上黑色节点数目比其他路径要少1。因此要想办法对该节点极其以上的节点进行调整。

1. 如果该节点是root节点，则直接染黑节点即可;
2. 如果该节点是红色节点，则直接染黑节点即可;
3. 如果该节点是黑色节点，则需根据其兄弟节点和父节点的情况，判断是否可以染色:
    1. 如果兄弟节点为黑色:
        1. 如果兄弟节点的左右子节点都为黑色。
        2. 兄弟节点的左右子节点只有一个为红色。
        3. 如果兄弟节点的左右子节点都为红色。则处理方式和3.1.2中的直线型相同。

直线型1



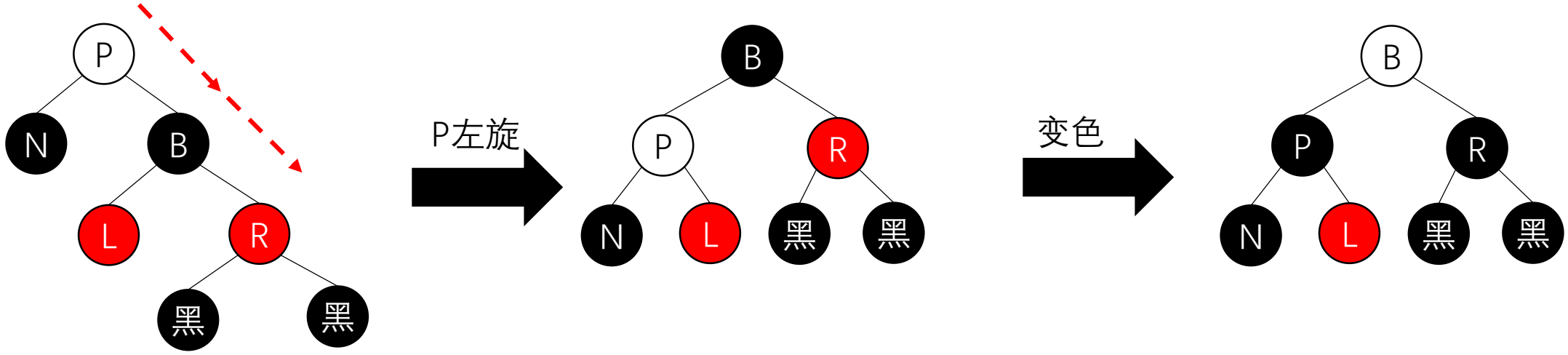
红黑树方法 – 双黑处理（双红）

何为双黑?

双黑指的是一种节点性质。我们判断某个节点具有双黑性质，指的是经过该节点的路径上黑色节点数目比其他路径要少1。因此要想办法对该节点极其以上的节点进行调整。

1. 如果该节点是root节点，则直接染黑节点即可;
2. 如果该节点是红色节点，则直接染黑节点即可;
3. 如果该节点是黑色节点，则需根据其兄弟节点和父节点的情况，判断是否可以染色:
    1. 如果兄弟节点为黑色:
        1. 如果兄弟节点的左右子节点都为黑色。
        2. 兄弟节点的左右子节点只有一个为红色。
        3. 如果兄弟节点的左右子节点都为红色。则处理方式和3.1.2中的直线型相同。

直线型2

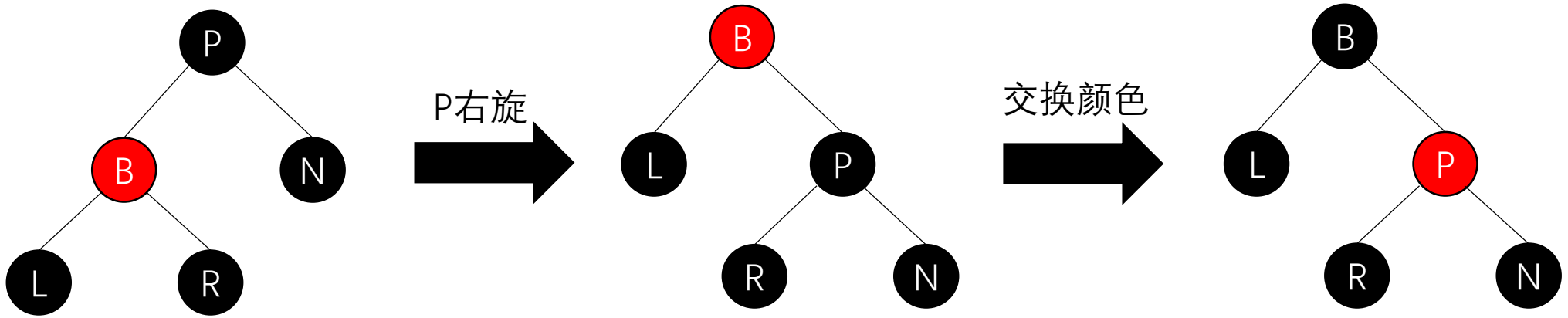


红黑树方法 – 双黑处理（兄红）

何为双黑?

双黑指的是一种节点性质。我们判断某个节点具有双黑性质，指的是经过该节点的路径上黑色节点数目比其他路径要少1。因此要想办法对该节点极其以上的节点进行调整。

1. 如果该节点是root节点，则直接染黑节点即可;
2. 如果该节点是红色节点，则直接染黑节点即可;
3. 如果该节点是黑色节点，则需根据其兄弟节点和父节点的情况，判断是否可以染色:
    1. 如果兄弟节点为黑色:
        1. 如果兄弟节点的左右子节点都为黑色。
        2. 兄弟节点的左右子节点只有一个为红色。
        3. 如果兄弟节点的左右子节点都为红色。
    2. 如果兄弟节点为红色。<u>则需要旋转+变色，使得兄弟节点变为黑色再进行处理。</u>



红黑树方法 – 双黑处理（兄红）

何为双黑?

双黑指的是一种节点性质。我们判断某个节点具有双黑性质，指的是经过该节点的路径上黑色节点数目比其他路径要少1。因此要想办法对该节点极其以上的节点进行调整。

1. 如果该节点是root节点，则直接染黑节点即可;
2. 如果该节点是红色节点，则直接染黑节点即可;
3. 如果该节点是黑色节点，则需根据其兄弟节点和父节点的情况，判断是否可以染色:
    1. 如果兄弟节点为黑色:
        1. 如果兄弟节点的左右子节点都为黑色。
        2. 兄弟节点的左右子节点只有一个为红色。
        3. 如果兄弟节点的左右子节点都为红色。
    2. 如果兄弟节点为红色。则需要旋转+变色，使得兄弟节点变为黑色再进行处理。