Deletion in Red–Black Trees

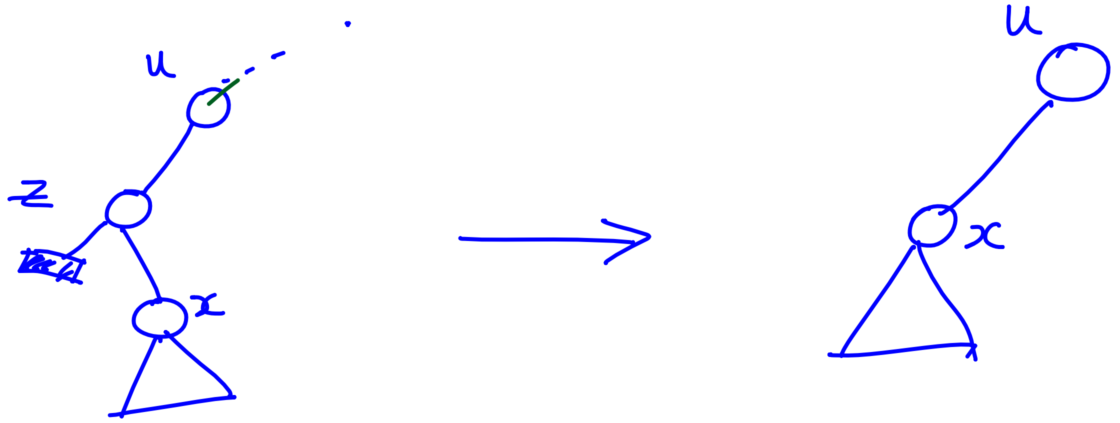
Delete(T, z) 2 phase algorithm (like insert)

I- phase : delete z using BST deletion algorithm.

II-phase : restore R-B properties.

BST deletion

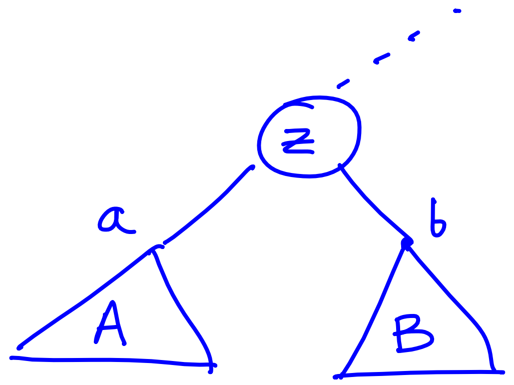
Case I



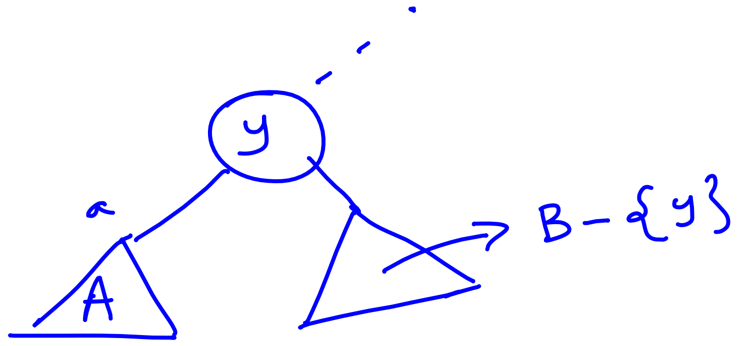
y = z

x is the node which comes in place of deleted node y.

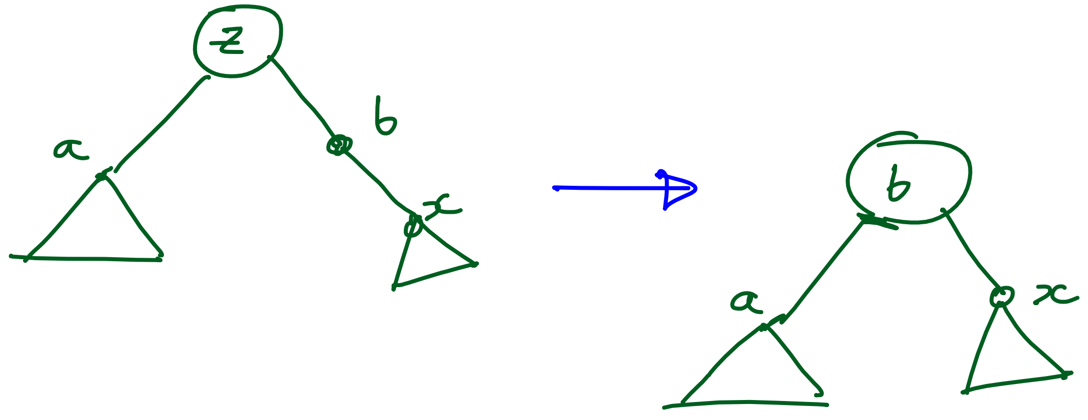
Case 3

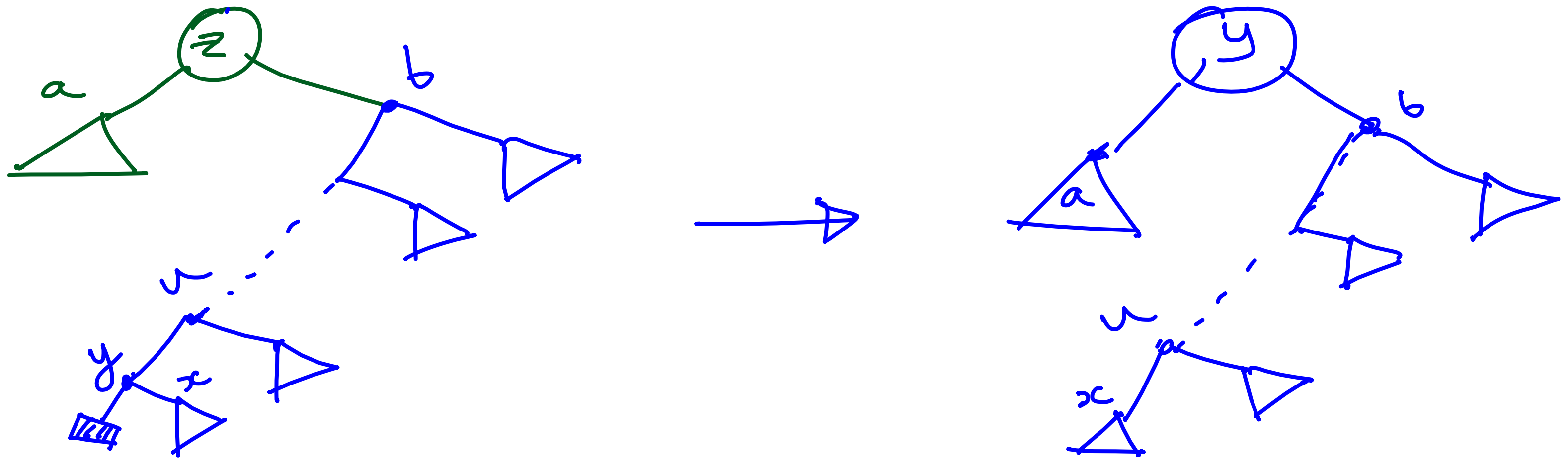


$y = \min(B)$



$y \equiv b$

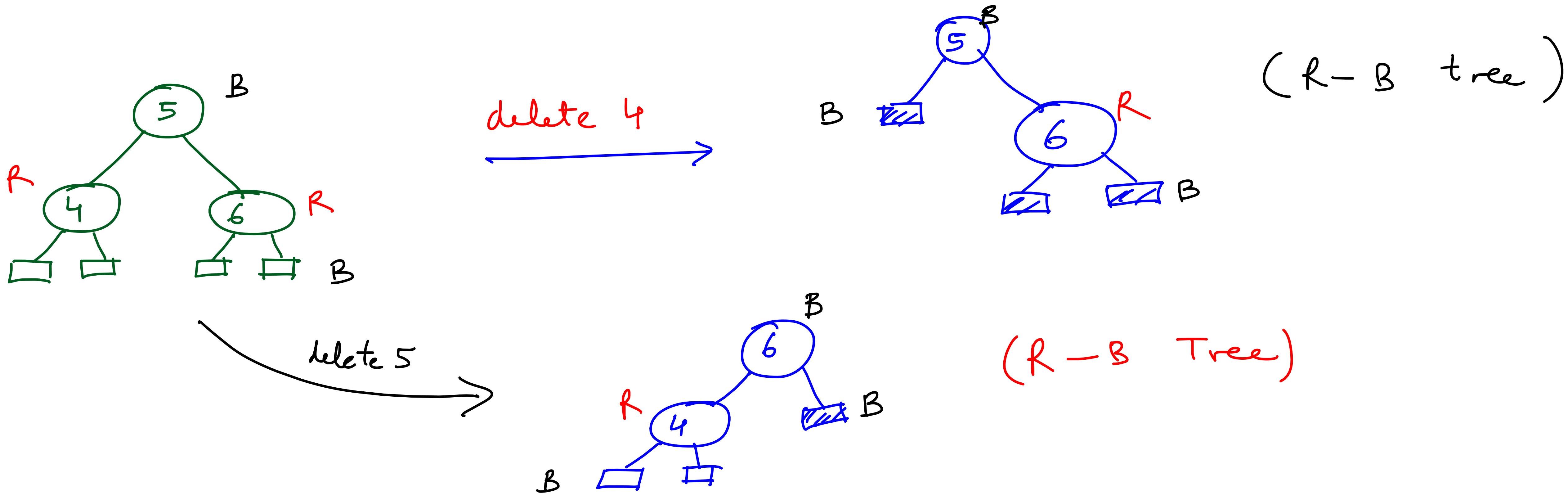




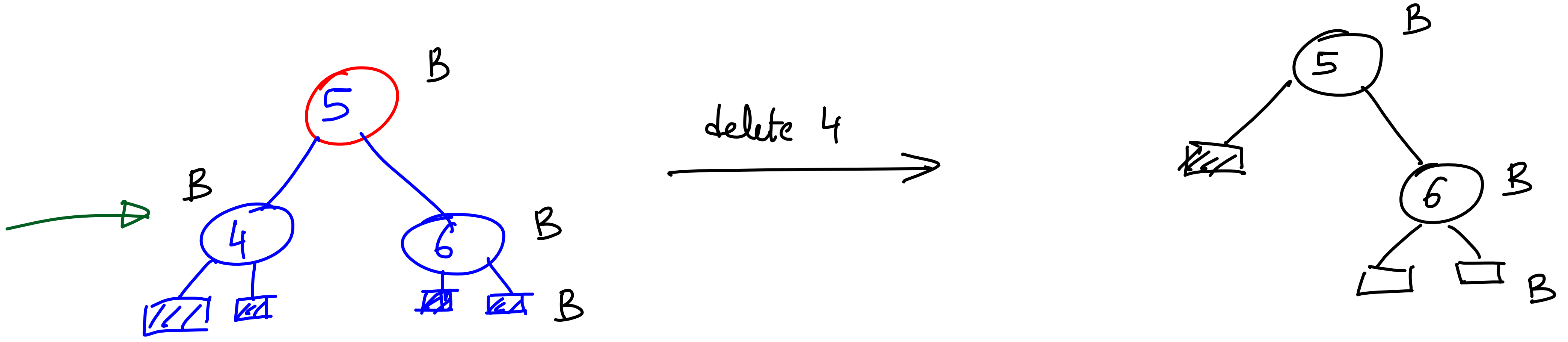
z is the node being deleted

y is the node which was at the position in T that got deleted

x is the node that comes in place of y after deletion



[In case 3, color of y is changed to col of z]



Not a R-B tree

Claim If original color of y is red then resulting tree (after applying BST delete(T, z)) satisfies R-B properties)

Proof:
(i) Every node is either Red or black.
Every node is either Red or black, as seen by considering cases 1–3.

(ii) Root remains black (color of any leaf is not changed)

(iii) Every leaf has black color

(iv) x is black ($\because$ it is a child of red node y) and no red–red adjacent pair is created

(v) As color of omitted node in any path is red, it does not affect the no of black nodes on the path. [Can be considered in more detail in each case]

$\square$

Kind of violations of R-B property when original color of y is black

(i) Every node is R or B (node x is red-black or black-black)

(ii) Root may not remain black, only if we delete root node in case 1 or 2.
x is the new root, which may be red

(iii) All the leaves are black

(iv) A red node may have a red child s.t.
x is red and parent of y is also red

(v) On all paths, passing through y, will now pass through x and will have one less black node on them.

We can fix (v), conceptually by thinking that x has an extra unit of black color.

x may have red-black or black-black

$\Rightarrow$ violation of property (1)

Pseudo-code for phase-I

```
Delete(T, z)
  y = z
  o-col-y = y.col
  if z.left == T.nil
        x = z.right
        Transplant(T, z, x)
  elif z.right == T.nil
         x= z.left
         Transplant(T, z, x)

  else  y = min(z.right)
        o-col-y = y.col
        y.col = z.col
        if (z.right).left==T.nil
            y.left = z.left
            (y.left).p = y
            Transplant(T, z, y)
        else  Transplant(T, y, y.right)

//else  Transplant(T, y, y.right)
          y.left = z.left
          (y.left).p = y
          (y.right) = z.right
          (y.right).p = y
          Transplant (T, z, y)

if o-col-y == black
      Dfixup (T, x)
```