Red-Black tree

DATA STRUCTURES & ALGORITHMS

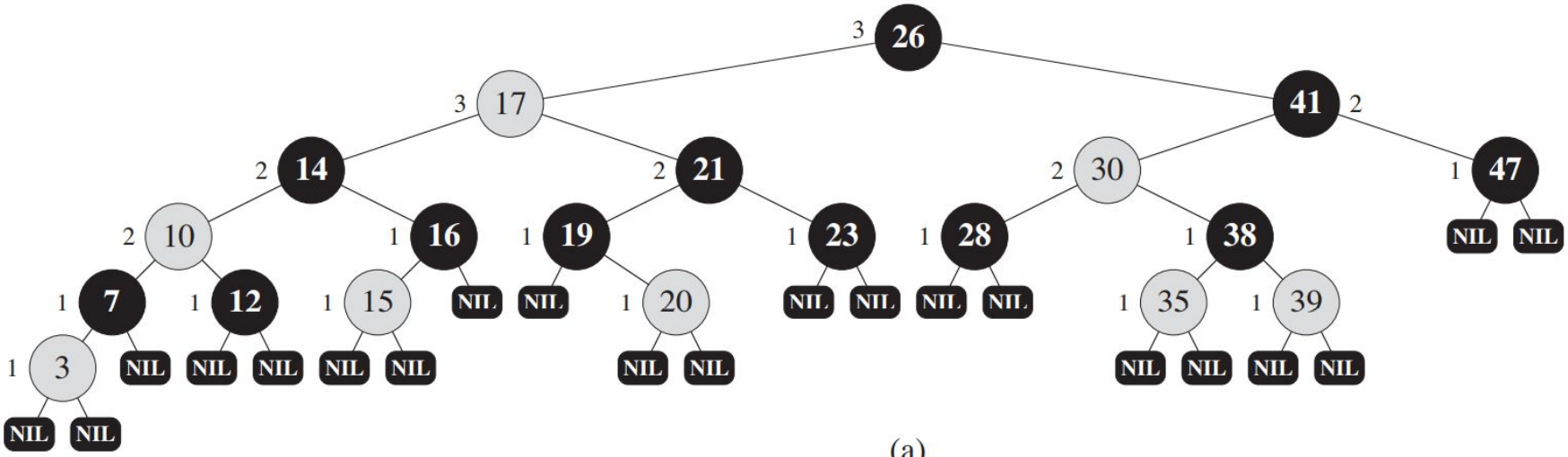
Red-Black tree properties

A red-black tree is a binary search tree with one extra bit of storage per node:

its color, which can be either **RED** or **BLACK**.

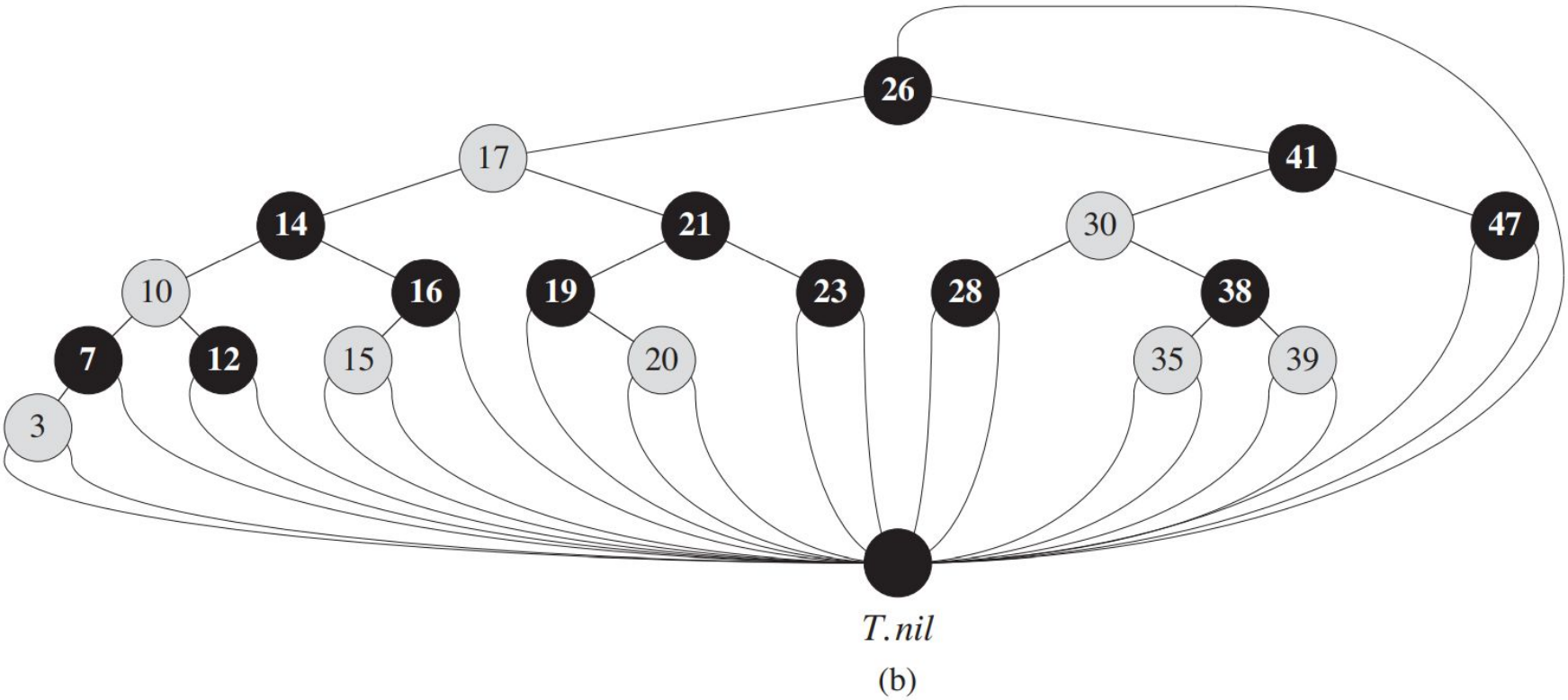
1. Every node is either red or black.
2. The root is black.
3. Every leaf (NIL) is black.
4. If a node is red, then both its children are black.
5. For each node, all simple paths from the node to descendant leaves contain the same number of black nodes.

Example


3 26
3 17
41 2
2 14
2 21
2 30
1 47
2 10
1 16
1 19
1 23
1 28
1 38
NIL NIL
1 7
1 12
1 15
NIL
NIL
1 20
NIL NIL
NIL NIL
1 35
1 39
1 3
NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL
NIL NIL

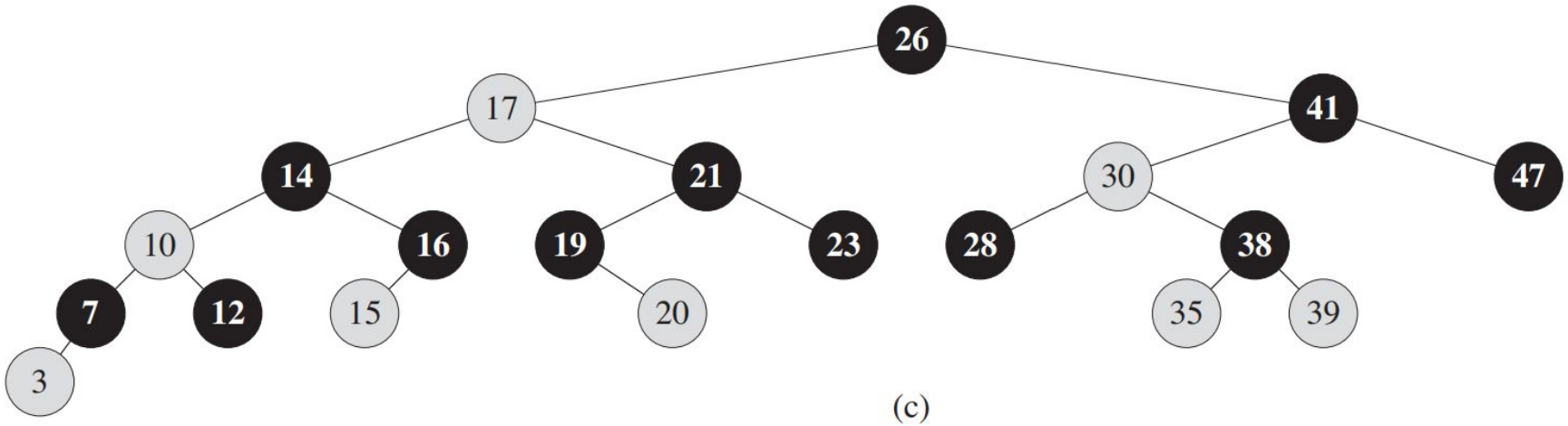
(a)

Example



(b)

Example



(c)

Black height

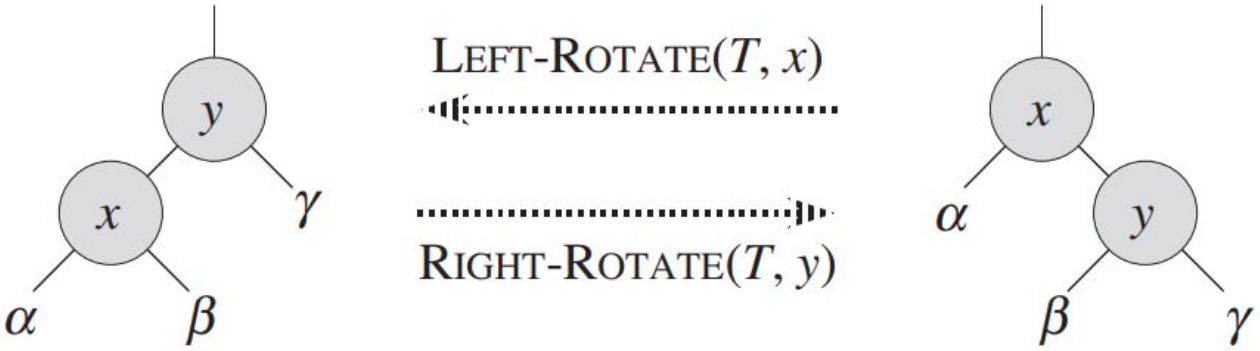
black-height of the node, denoted ***bh(x):***

the number of black nodes on any simple path from, but not including, a node ***x*** down to a leaf

Lemma 13.1

A red-black tree with n internal nodes has height at most $2\lg(n + 1)$.

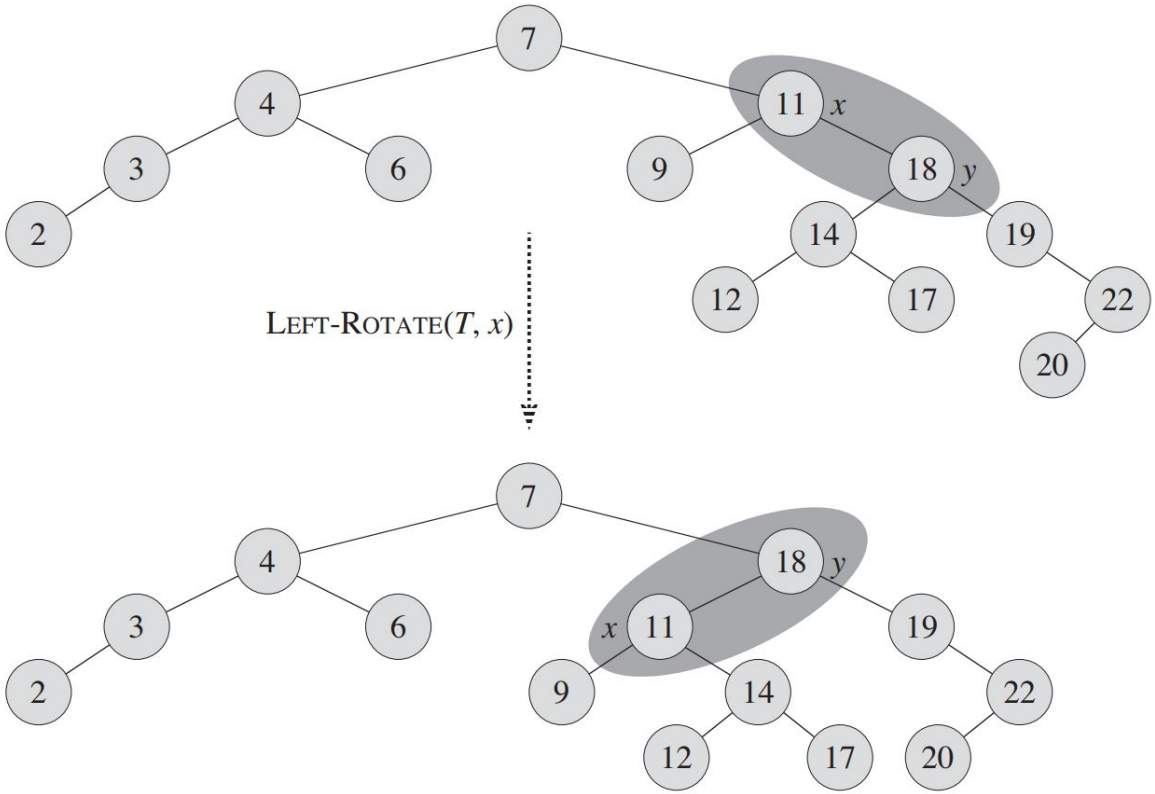
Rotation



LEFT-ROTATE(T, x)

```
1   y = x.right                 // set y
2   x.right = y.left            // turn y's left subtree into x's right subtree
3   if y.left ≠ T.nil
4       y.left.p = x
5   y.p = x.p                   // link x's parent to y
6   if x.p == T.nil
7       T.root = y
8   elseif x == x.p.left
9       x.p.left = y
10  else x.p.right = y
11  y.left = x                  // put x on y's left
12  x.p = y
```

Left rotation example

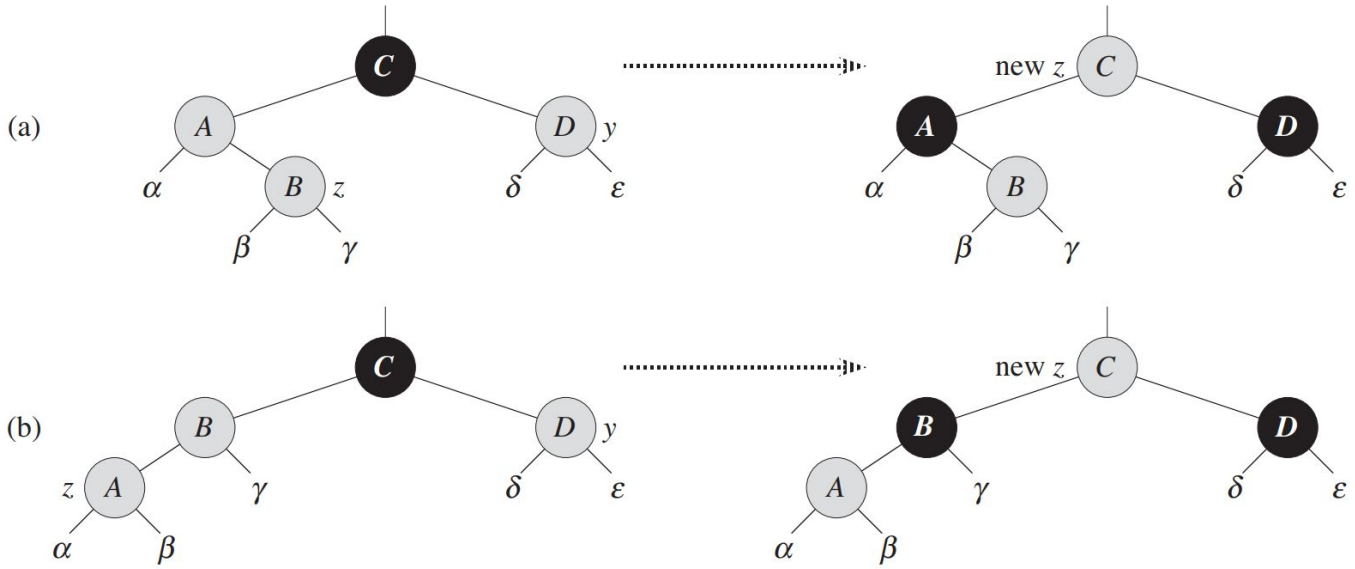


Insertion

ابتدا محل قرارگیری راس جدید را بر اساس اینکه درخت BST است پیدا کنید و یک گره با رنگ **قرمز** درج کنید.

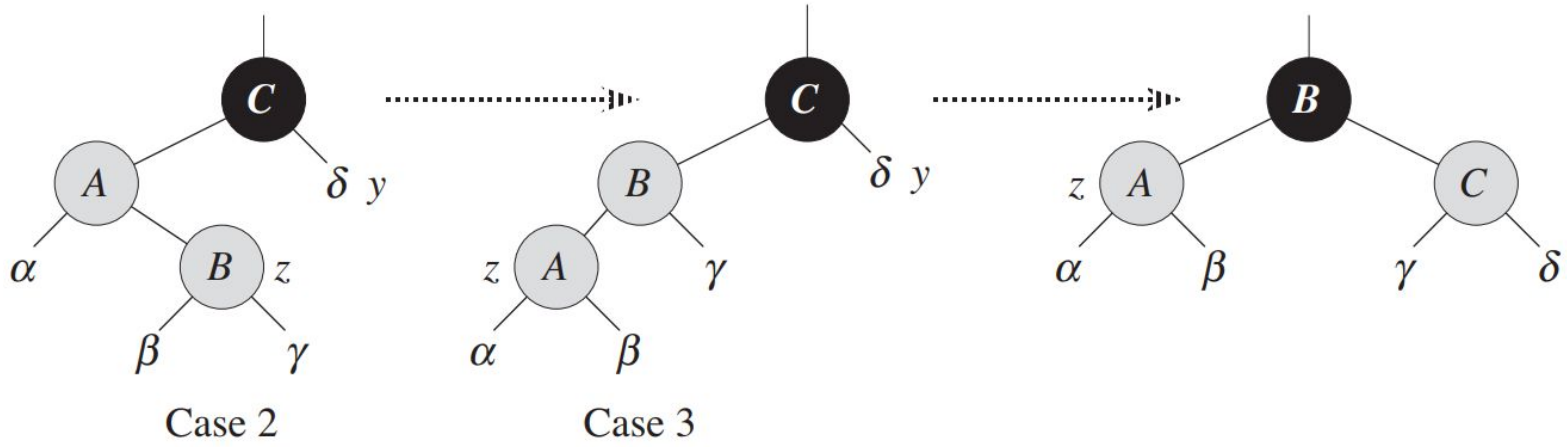
- اگر پدر گره جدید سیاه است: نیاز به انجام کاری نیست
- اگر پدر گره جدید **قرمز** است:
  - اگر عموی گره جدید قرمز است: پدر و عمو را سیاه کنید و جد را قرمز کنید و همین مراحل را برای جد تکرار کنید.
  - اگر عموی گره جدید سیاه است: ابتدا با چرخش مناسب حول پدر، گره جدید و پدر و جدش را روی یک راستا قرار دهید. رنگ پدر و جد را عوض کنید و جد را به سمت عمو بچرخانید

Insertion



Case 1: z's uncle y is red

Insertion



Case 2: z's uncle y is black and z is a right child
Case 3: z's uncle y is black and z is a left child

Deletion

فرض کنید y راسی باشد که بطور فیزیک باید حذف شود و x فرزند آن و p پدر و w برادر آن باشد، آنگاه:

- اگر y **قرمز** باشد نیاز به انجام کاری نیست.
- اگر y سیاه باشد:
  - اگر x **قرمز** باشد کافیست آن را سیاه کنید
  - اگر x سیاه باشد آنرا $double\ black$ کنید:
    - اگر w **قرمز** است: رنگ p و w را عوض کنید و پدر را به سمت x بچرخانید تا به حالت بعد تبدیل شود.
    - اگر w سیاه است:
      - اگر w بچه **قرمز** ندارد رنگ x و w را به پدر بدهید و p را به عنوان x جدید در نظر بگیرید.
      - اگر w بچه **قرمز** دارد با تعویض رنگ w و بچه قرمزش اگر با p در یک راستا نیستند و چرخش مناسب آنها را در یک راستا قرار دهید و سپس قرمز را سیاه و رنگ p و w را عوض کنید و پدر را به سمت x بچرخانید.