(1) Inorder_Tree_Walk(x) ← ریشه درخت ⟶ O(n)

```
1. if x≠null then
2.     Inorder_Treewalk(x.left)
3.     Print x.key
4.     Inorder_Tree_walk(x.right)
```

---

(2) دو اشاره‌گر چپ و راست برای x. x.left به چپ‌ترین فرزند x و x.right به برادر خواهر راست x.

درصورت عدم وجود x به پدر x اشاره می‌کند. متغیر boolean b = depth(x) mod 2 :

برای دسترسی به پدر: x.right تا جایی رو جهت b تغییر کند (0 به 1 یا 1 به 0)

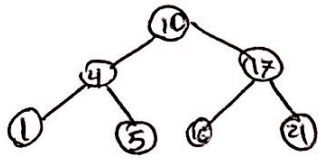
و فرزند: ابتدا x.left سپس x.right تا جایی که رو جهت تغییر کند. گره یک مانده به آخر گره فرزند و گره آخر است.

راست (چپ)

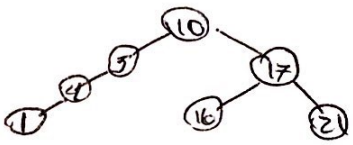
---

(3)

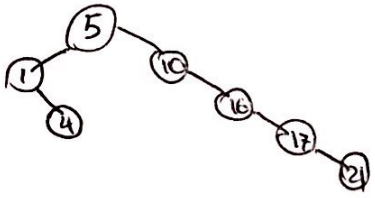
2:



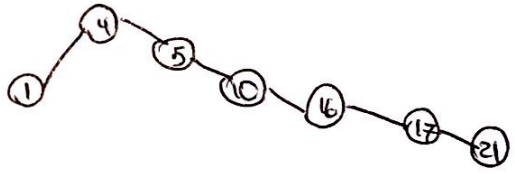
3:



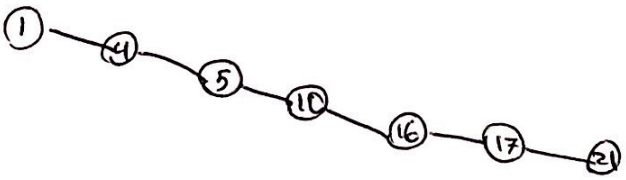
4:



5:



6:

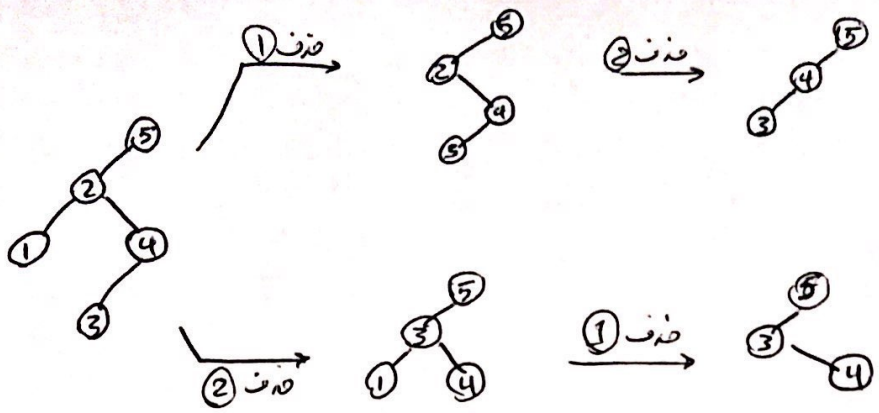


---

(4) Tree_Predecessor(x)

```
1. If x.left ≠ null then
2.     return Tree_Maximum(x.left)
3. y = x.p
4. while y≠null anel x == y.left do
5.     x = y
6.     y = y.p
7. return y
```

(5) خیر. مثال نقض:



---

(6) بیشتر تمامی فرزندان راست را به حالت زنجیر در نظر بگیرید. هرگره ای که فرزند چپ باشد را می‌توان با چرخش به زنجیره اضافه کرد؛ پس هر درخت را می‌توان با $n-1$ چرخش راست به یک زنجیر تبدیل کرد.

فرض کنید $r_1, r_2, \dots, r_m$ چرخش‌های لازم برای تبدیل درخت T_1 به T_2 و $s_1, s_2, \dots, s_k$ برای تبدیل درخت T_2 به زنجیر باشند؛ می‌توان T_1 را به T_2 با چرخش‌های $r_1, r_2, \dots, r_m, \bar{s}_k, \bar{s}_{k-1}, \dots, \bar{s}_1$ که $\bar{s}_i$ چرخش عکس s_i است تبدیل کرد.

$$\left.\begin{array}{l} m \le n-1 < n \\ k \le n-1 < n \end{array}\right\} \Rightarrow m+k < 2n \Rightarrow O(n)$$

---

(7) خط 8 از RB_Insert را حذف کرده، RB-Insert-FixUp را اضافه می‌کنیم؛ به این صورت که $z.p$ و $z.p.p$ را پیدا کرده و نگه می‌داریم (این دو گره در $\log n$ پیدا می‌شوند).

از آنجا که RB-Insert-FixUp $O(\log n)$ است، پیدا کردن $z.p$ و $z.p.p$ و ذخیره آن‌ها از نظر زمانی تغییری در RB_Insert ایجاد نمی‌کند.

```
RB_Insert (T, z)
2.  y = T.Null
2.  x = T.root
3.  while (x ≠ T.Null) do
4.      y = x
5.      if (z.key < x.key) then
6.          x = x.left
7.      else x = x.right
8.  z.p = y
9.  if y == T.Null then
10.     T.root = z
11. else if z.key < y.key then
12.     y.left = z
13.     y.right = z
14. z.left = T.Null
15. z.right = T.Null
16. z.color = RED
17. RB_Insert_FixUp (T, z)
```