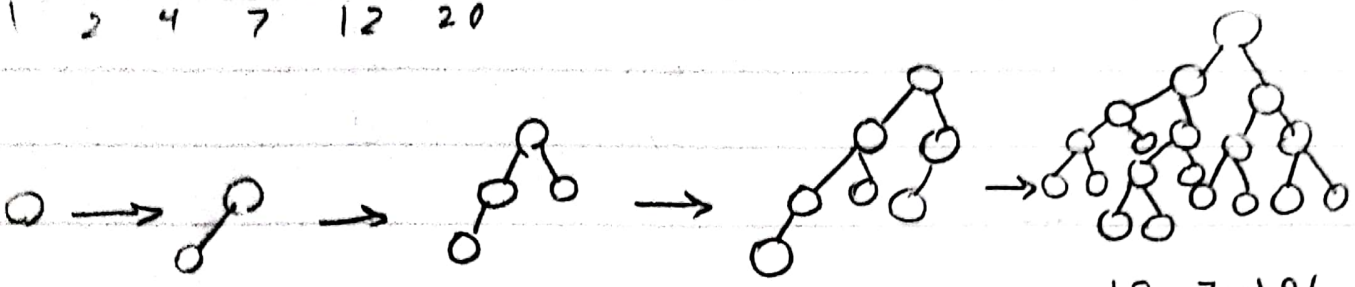
⑩ حلقه با $i=0$ تا $\frac{n}{2}-2$ $i=2$
① اگر $a[i]<a[2i+1]$ برقرار بود، false برگردان.
② اگر $a[i]<a[2i+2]$ برقرار بود، false برگردان.
③ true برگردان

⑪ ما برای تشخیص هیپ بودن حداکثر $n-1$ مقایسه نیاز داریم.

0	1	2	3	4	5
1	2	4	7	12	20

$12 < 19 < 20 \rightarrow h = 4$ (6)



$12 + 7 = 19$

(7)

5 15 10 30 45 40 20 70 100 90 50

5 15 10 30 45 40 20

30 45 40

70 100 90

5 15 10

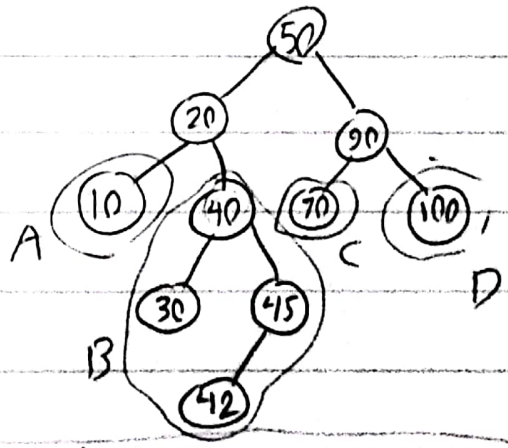
① بله ② خیر ③ در کل برای کم کردن ارتفاع درخت، باید ریشه را دوران راست یا چپ بدهید اگر عمیق‌ترین گره یعنی 42 در گره B

A ← دوران راست حول ریشه

D ← دوران چپ حول ریشه

B ← دوران نوع ③

C ← دوران نوع ④



(9) $O(k)$

* پویش میانی ترتیب با $O(n)$ انجام می‌شود، پویش را تا k امین عنصر ادامه می‌دهیم.

* فرض کنیم ارتفاع درخت $n-1$ باشد، در این صورت حداقل کار، پیمایش مسیر ریشه تا عنصر مورد نظر است.

(3)

e=18 d=15 b=12 a=10 c=8 f=7

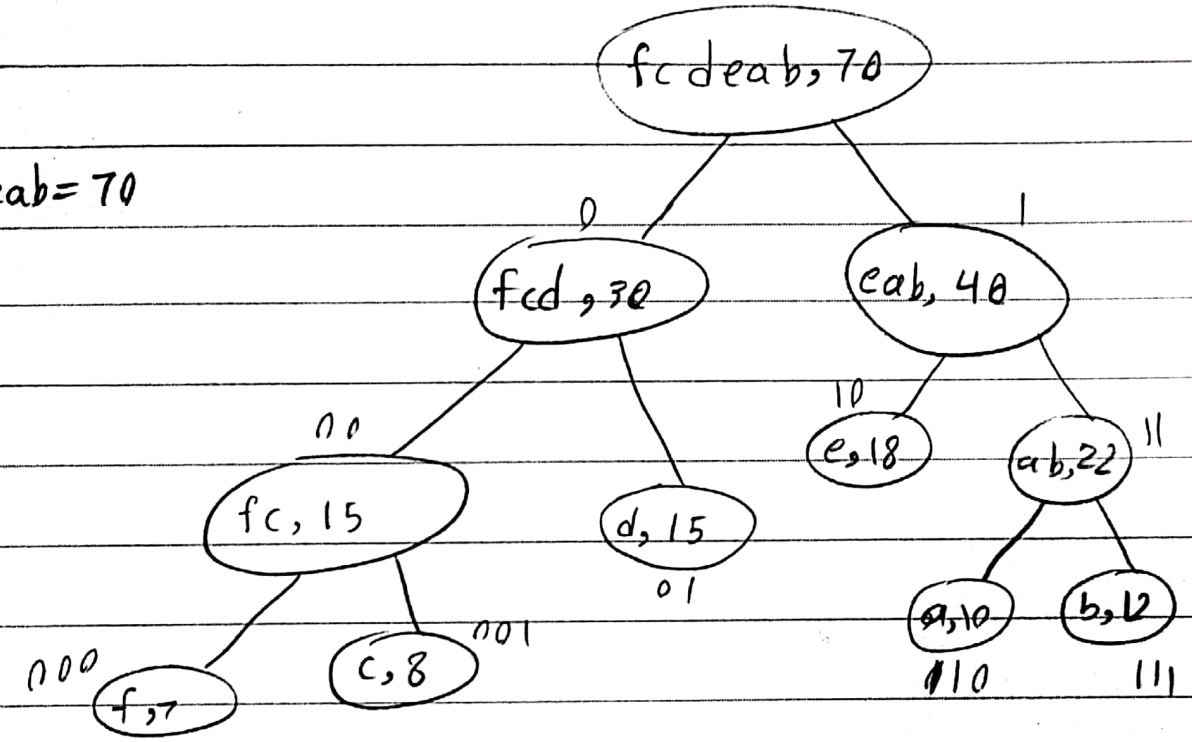
⇒ e=18 d=15 fc=15 b=12 a=10

⇒ ab=22 e=18 d=15 fc=15

⇒ fcd=30 ab=22 e=18

⇒ eab=40 fcd=30

⇒ fcdeab=70

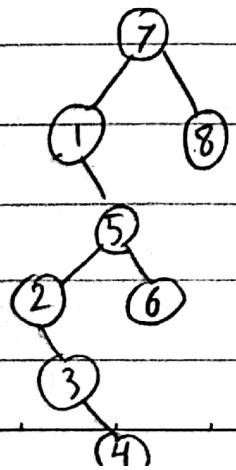


$3(7+8+10+12) + 2(15+18) = \underline{177}$

(4)

43265187 → 43265 1 → 432 6 5 → 43 2

↳ 4 3 → 4



③ * این قدر بالا برو تا زیردرخت چپ شود. اگر بالاتر null بود، برنامه را تمام کن

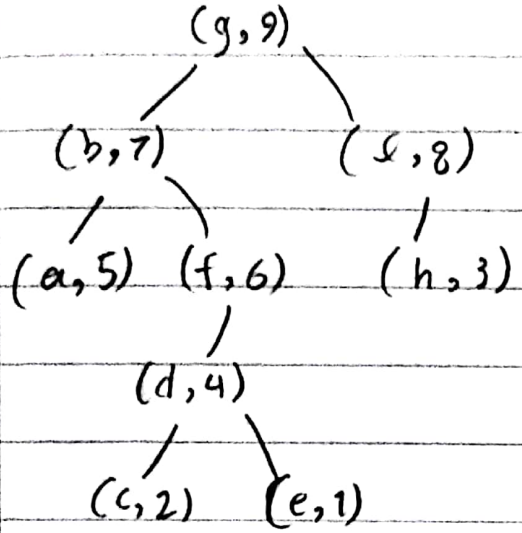
② * یکی دیگر بالا برو و برو به حلقه

```
void Q2(Node head){
    while(true){
        while(head.left != null)  head = head.left;
        print(head.value);
        if(head.right != null)  head = head.right;
        else while(true){
            if(head.parent.left == head) head = head.parent;
            else {
                while(true){
                    if(head.parent == null) return;
                    if(head.parent.left == head) break;
                    head = head.parent;
                }
                head = head.parent;
            }
            print(head.value);
            if(head.right != null){
                head = head.right;
                break;
            }
        }
    }
}
```

① الف)

آرمان حسن‌زاده

۴۰۱۲۰۲۳۰۰۸



ب) root را برابر بزرگترین عنصر prio قرار می‌دهیم

ج) (key[e], prio[e]) را برابر root قرار می‌دهیم

د) الگوریتم را به ازای key[n تا e+1], prio[n تا e+1] و root = root.right اجرا می‌کنیم در صورتی که (e+1 != n) باشد.

ه) الگوریتم را به ازای key[e-1 تا 0], prio[e-1 تا 0] و root = root.left اجرا می‌کنیم در صورتی که (e-1 != 0) باشد.

---

② ① * برو به چپ ترین زیر درخت
* چاپ کن
* برو به راست، اگر نبود برو به حلقه ②

② * اگر فرزند چپ بود، برو بالا، در غیر اینصورت برو به بخش ③
* چاپ کن
* اگر راست موجود بود، برو به راست و حلقه ①