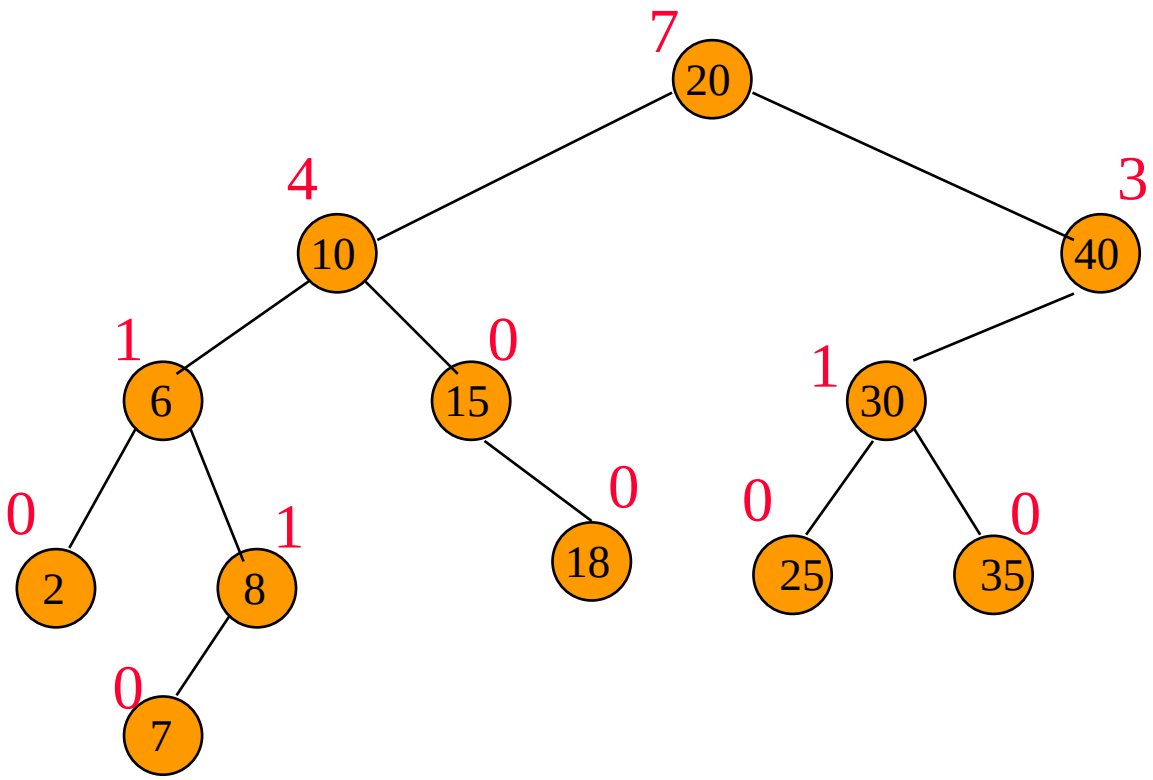
Arborele binar de căutare indexat

1. Definire : Arbore binar de căutare indexat

- arbore binar de căutare
- fiecare nod are încă un câmp `LeftSize` în care se va reține numărul de descendenți din subarborele stâng

```
struct nod {
      nod *st, dr;
      int val, LeftSize;
      }
```

2. Exemplu: Cifrele în roşu reprezintă `LeftSize` pentru fiecare nod al arborelui



3. Indexul unui element este poziția sa la traversarea în inordine a arborelui binar de căutare. Pentru arborele din imagine:

[2, 6, 7, 8, 10, 15, 18, 20, 25, 30, 35, 40]

```
index(2) = 0
index(15) = 5
index(20) = 7
```

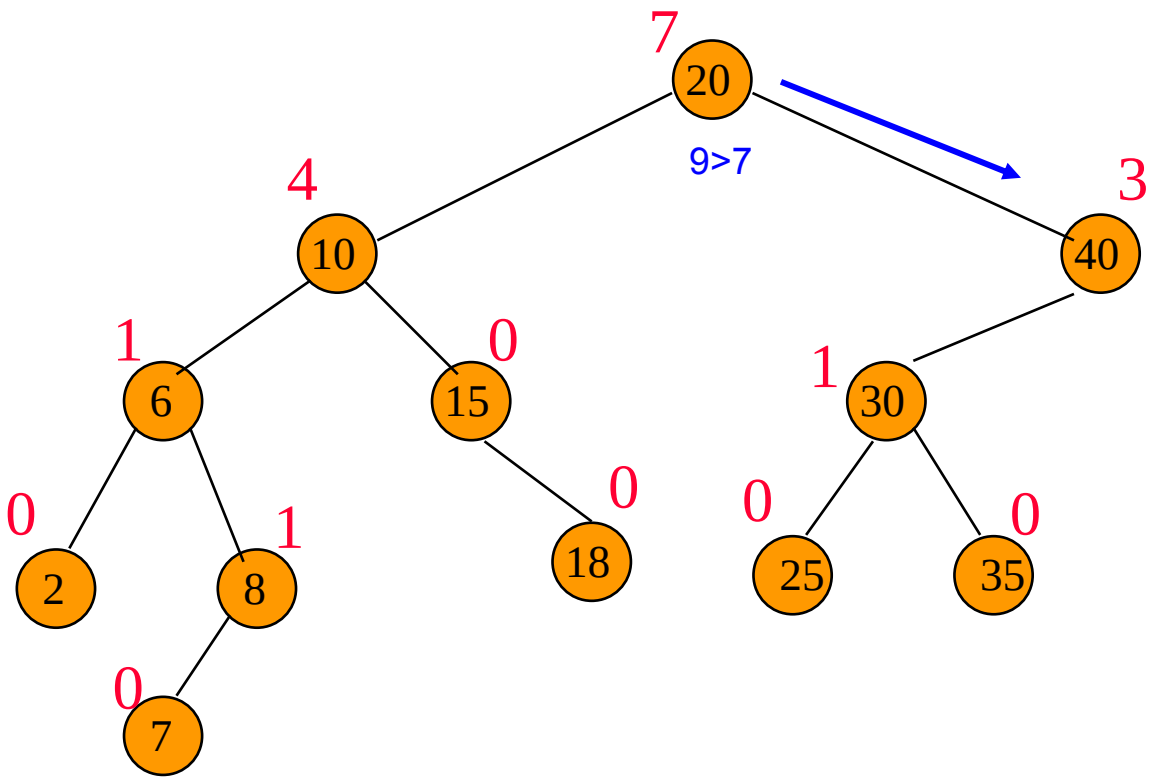
`LeftSize(x) = index(x)`, relativ la elementele din subarborele stâng cu rădăcina în nodul `x` .

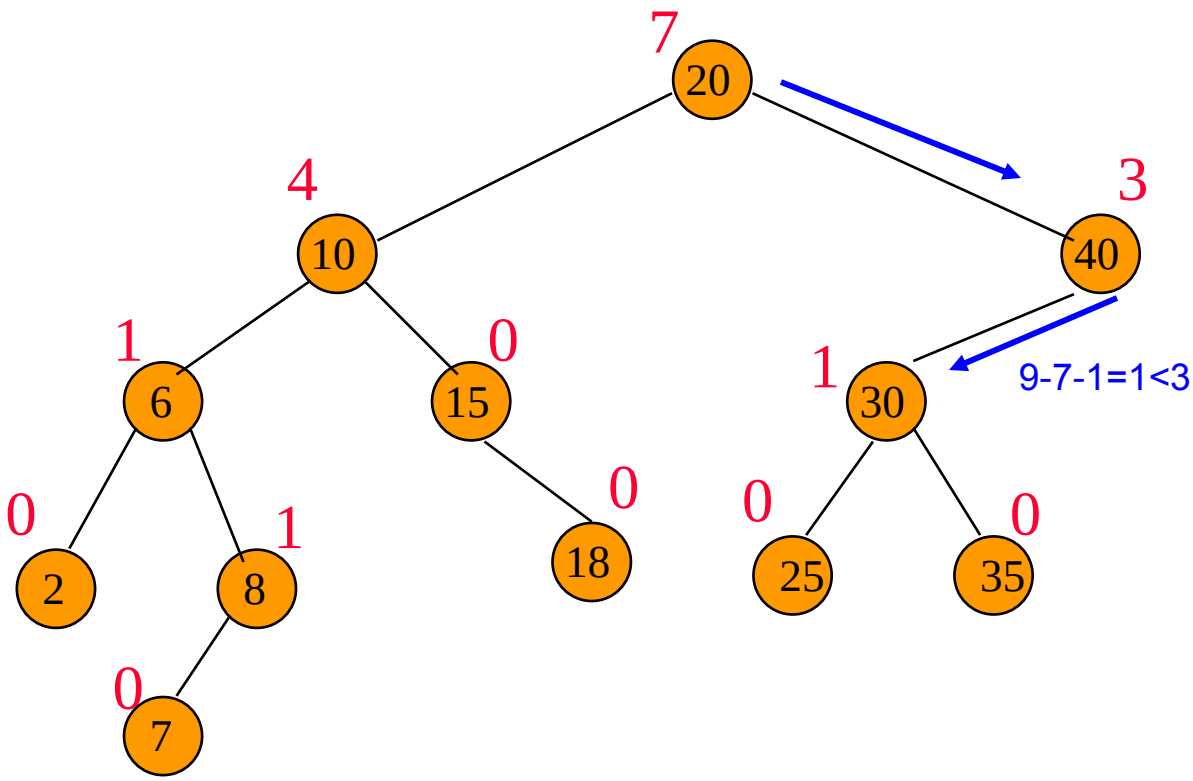
4. Căutarea unui element de index `nr` în arborele binar de căutare

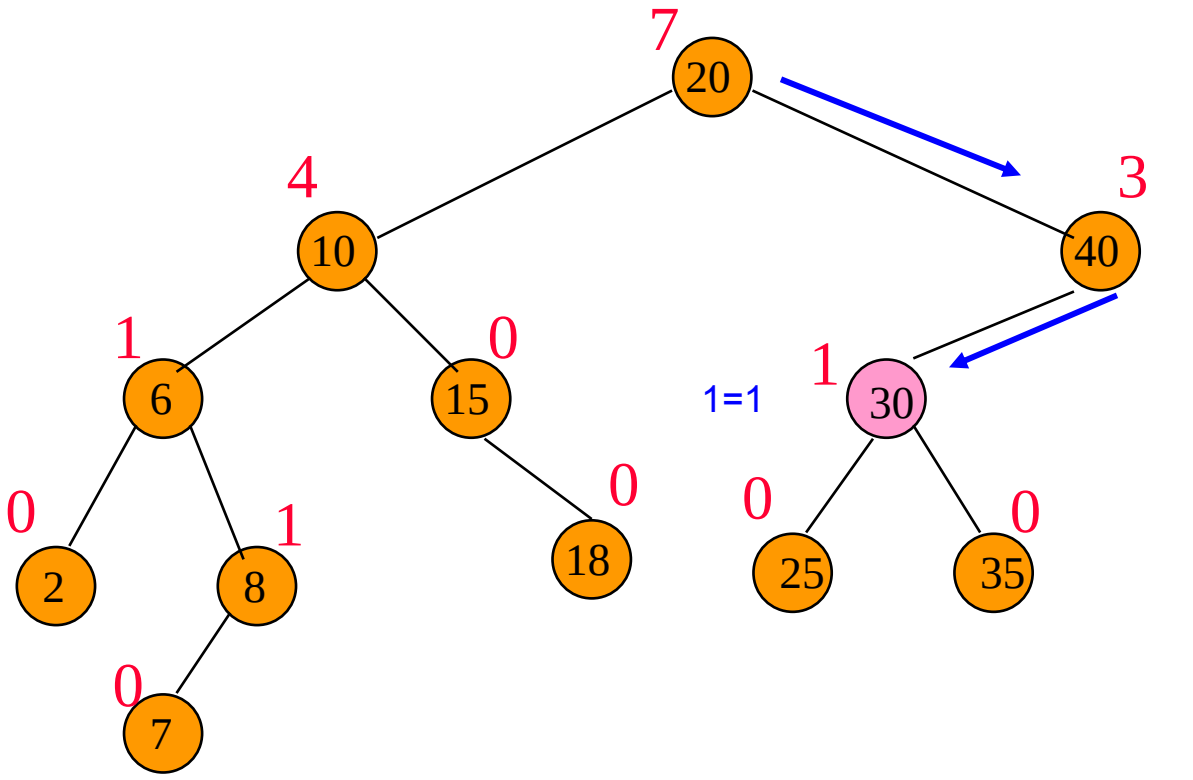
Să presupunem că indexarea începe cu elementul aflat pe poziția 0 (ca în exemplu). Atunci, pentru prelucrarea nodului `x`, avem cazurile :

- dacă `nr=x.LeftSize` atunci elementul căutat are valoarea `x.val`
- dacă `nr<x.LeftSize` atunci elementul căutat este în subarborele stâng al lui `x`, pe poziția `nr` (deci apelăm recursiv căutarea pentru `x.st`)
- dacă `nr>x.LeftSize` atunci elementul căutat se află în subarborele drept al lui `x` şi este pe poziția (`nr - x.LeftSize-1`) în acest subarbore

Exemplu:
`nr=9`







Deci, elementul cu indexul 9 este 30.