Arbori binari

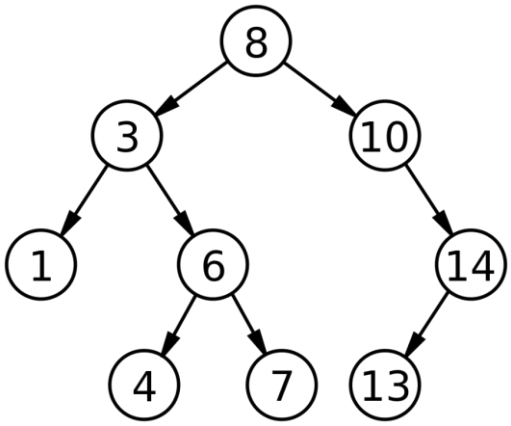
Un **arbore cu rădăcină** este o multime de elemente numite **noduri** sau **vârfuri** pentru care:

1. exista un nod cu destinatie speciala numit **radacina arborelui**;
2. celelalte noduri sunt repartizate în **n** seturi disjuncte A1, A2, ..., An, fiecare set constituind la rândul sau un arbore.

Obs.

- Daca un nod nu are fii, el se numeste nod **terminal** sau **frunză**.
- Rădăcina arborelui se află pe nivelul 0.
- Înălţimea/adâncimea arborelui este data de lungimea drumului de la rădăcină la cea mai îndepărtată frunză (nivelul maxim din arbore).

Un **arbore binar** este un arbore cu rădăcină în care fiecare nod are cel mult doi fii.
Arborii binari au şi ei o definiţie recursivă: un arbore binar este fie vid, fie format dintr-o ***rădăcină R*** şi doi subarbori, numiţi ***subarborele stâng S,*** respectiv ***subarborele drept D***. Se face întotdeauna o distincţie clară între cei doi subarbori. Dacă subarborele stâng ***S*** este nevid, rădăcina lui se numeşte fiul stâng al rădăcinii R. Analog, dacă subarborele drept ***D*** este nevid, rădăcina lui este fiul drept al rădăcinii R.



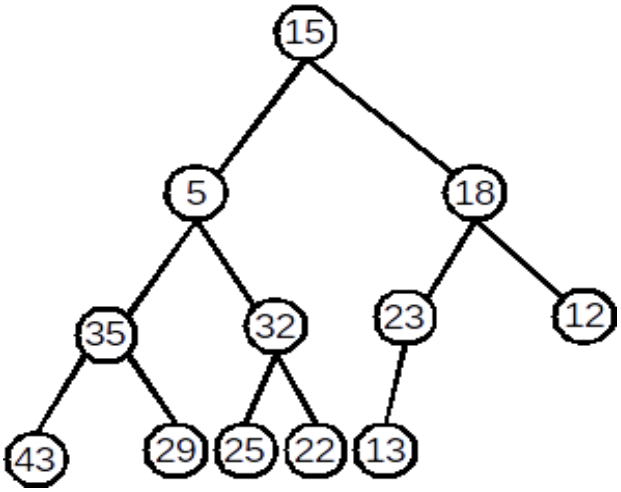
Tipuri speciale de arbori binari

Un **arbore binar** este **echilibrat** dacă şi numai dacă, pentru orice nod, înaltimile celor doi subarbori diferă cu cel mult 1.

Un **arbore binar** se numeşte **strict** dacă orice nod din arbore are exact doi fii sau este terminal
Un **arbore binar** se numeşte **plin** dacă fiecare nivel ***i*** este complet, adică conţine 2^i noduri.

Propoziţie. Un arbore binar strict/plin care are n noduri terminale are 2*n-1 noduri.
Dem. Inducţie după n.

Un **arbore binar** se numeşte **complet** dacă toate nivelurile sunt complete cu excepţia ultimului, unde frunzele sunt plasate de la stânga la dreapta (el se obţine dintr-un arbore binar plin prin eliminarea de la dreapta la stânga a unor noduri de pe ultimul nivel).
Arborele binar de mai jos este complet.



Înălțimea unui arbore binar plin/complet este $[\log_2 n]+1$, unde n este numărul de noduri.

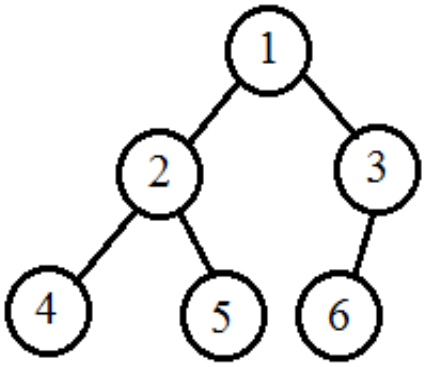
Metode de reprezentare a arborilor binari

1. Cu ajutorul a doi vectori ST şi DR

Vectorul **ST** - conţine descendenţii stângi ai fiecărui nod
Vectorul **DR** - conţine descendentii drepţi ai fiecărui nod.

Lipsa unuia dintre descendenţi se specifică prin valoarea 0.

Exemplu:
ST = (2, 4, 6, 0, 0, 0)
DR = (3, 5, 0, 0, 0, 0)



2. Folosind alocarea dinamică a memoriei

```
struct nod
{
    int inf;
    nod *st, *dr;

} *r; // r = adresa rădăcinii
```

Unde: st = adresa subarborelui stâng
dr = adresa subarborelui drept

Parcurgerea arborilor binari

Parcurgerea unui **arbore binar** înseamnă a vizita fiecare nod al arborelui o singură dată, în scopul prelucrării informației.

Avem două tipuri de parcurgere: **în lățime (Breadth First)** asemănătoare cu cea de la grafuri neorientate şi **în adâncime**.

Există trei metode de parcurgere în **adâncime** a unui arbore binar: în **preordine**, **inordine** şi **postordine**. În toate cele trei tipuri de parcurgere se vizitează prima dată subarborele stâng şi apoi subarborele drept, iar diferenţa constă în momentul în care se vizitează rădăcina.

Parcurgerea în preordine (RSD) – se vizitează mai întâi rădăcina ***R***, apoi se parcurg în preordine subarborele stâng ***S*** şi subarborele drept ***D***.
*Pentru arborele din figură, şirul parcurgerii **RSD** este:* **1 2 4 5 3 6**.

Parcurgerea în inordine (SRD) – se parcurge mai întâi în inordine suarborele stâng ***S***, apoi se vizitează rădăcina ***R***, apoi se parcurge în inordine subarborele drept ***D***.
*Pentru arborele din figură, şirul parcurgerii **SRD** este:* **4 2 5 1 6 3**.

Parcurgerea în postordine (SDR) – se parcurg în postordine subarborele stâng ***S*** şi subarborele drept ***D***, iar apoi se vizitează rădăcina ***R***.
*Pentru arborele din figură, şirul parcurgerii **RSD** este:* **4 5 2 6 3 1**.

Obs. Deoarece definiţia unui arbore binar este **recursivă**, algoritmii folosiți pentru crearea şi prelucrarea **arborilor binari** utilizează **tehnica recursivității** şi **metoda Divide et Impera**: prelucrarea unui nod al arborelui se descompune în două **subprobleme**:
– prelucrarea subarborelui stâng;
– prelucrarea subarborelui drept;
La final are loc **combinarea** rezultatelor prelucrării celor doi subarbori.

Aplicaţii

1. Se citesc din fişierul **arbore.in** de pe prima linie două numere **n** şi **r** reprezentând numărul de noduri şi rădăcina unui arbore binar. De pe a doua şi a treia linie se vor citi câte n elemente reprezentând rădăcinile subarborilor stâng şi respectiv drept al fiecărui nod, sau 0 dacă nu există subarborele. Afişaţi în fişierul **arbore.out** şirurile parcurgerilor RSD, SRD şi SDR ale arborelui citit.

arbore.in
8 1
2 4 7 0 0 0 0 0
3 5 8 0 6 0 0 0
arbore.out
RSD: 1 2 4 5 6 3 7 8
SRD: 4 2 5 6 1 7 3 8
SDR: 4 6 5 2 7 8 3 1

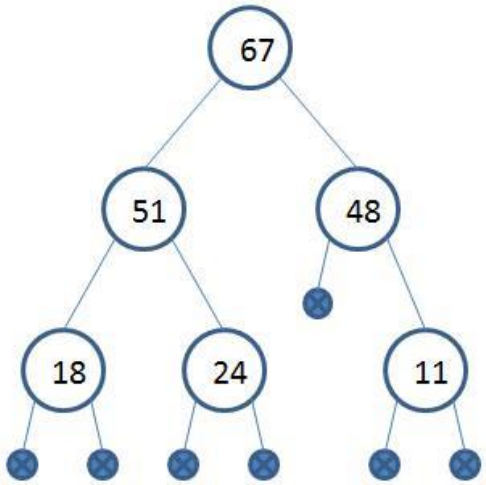
```
void RSD(int i)
{ if(i>0)
  {
      cout<<i<<' ';
      RSD(st[i]);
      RSD(dr[i]);
  }
}
```

Apel: `RSD(r);`

2. Fişierul de intrare **arb.in** conţine pe prima linie lista valorilor memorate în nodurile arborelui, obţinute în urma parcurgerii în preordine. Dacă un nod nu are descendent stâng, în listă va apărea valoarea **0**. Dacă un nod nu are descendent drept, în listă va apărea valoarea **0**. Afişaţi în fişierul **arb.out** şirurile parcurgerilor RSD, SRD şi SDR ale arborelui citit.

Exemplu:

arb.in
67 51 18 0 0 24 0 0 48 0 11 0 0




```
struct nod
{
    int inf;
    nod *st,*dr;
} *r;

void creare(nod *&r)
{
    int x;
    cin>>x; //se citeşte informaţia utilă pentru nod
    if (x==0)r=NULL;
    else
    {
        r=new nod; //se alocă spaţiu pentru noul nod
        r->inf=x; //se completează informaţia utilă

        creare(r->st); //se construieşte subarborele stâng prin autoapel
        creare(r->dr); //se construieşte subarborele drept prin autoapel
    }
}

void RSD(nod *r) //preordine
{
    if (r)
    {
        cout<<r->inf<<" ";
        RSD(r->st);
        RSD(r->dr);
    }
}
```

Apeluri:
```
creare(r);
RSD(r);
```