Arbori

Notiuni introductive

Definitie:

Un **arbore** este o structura dinamica T=(V, E) in care V este o multime de varfuri/noduri (eng. vertices) si E este o multime de muchii/arce (eng. edges).

$$E \subseteq V \times V$$

- Exista un nod radacina
- Multimea nodurilor, in afara de radacina, poate fi partitionata in $n \geq 0$ multimi disjuncte T_1, T_2, …, T_n a.i. fiecare dintre acestea sa fie un arbore. T_1, T_2, …, T_n se numesc *subarbori* ai radacinii.

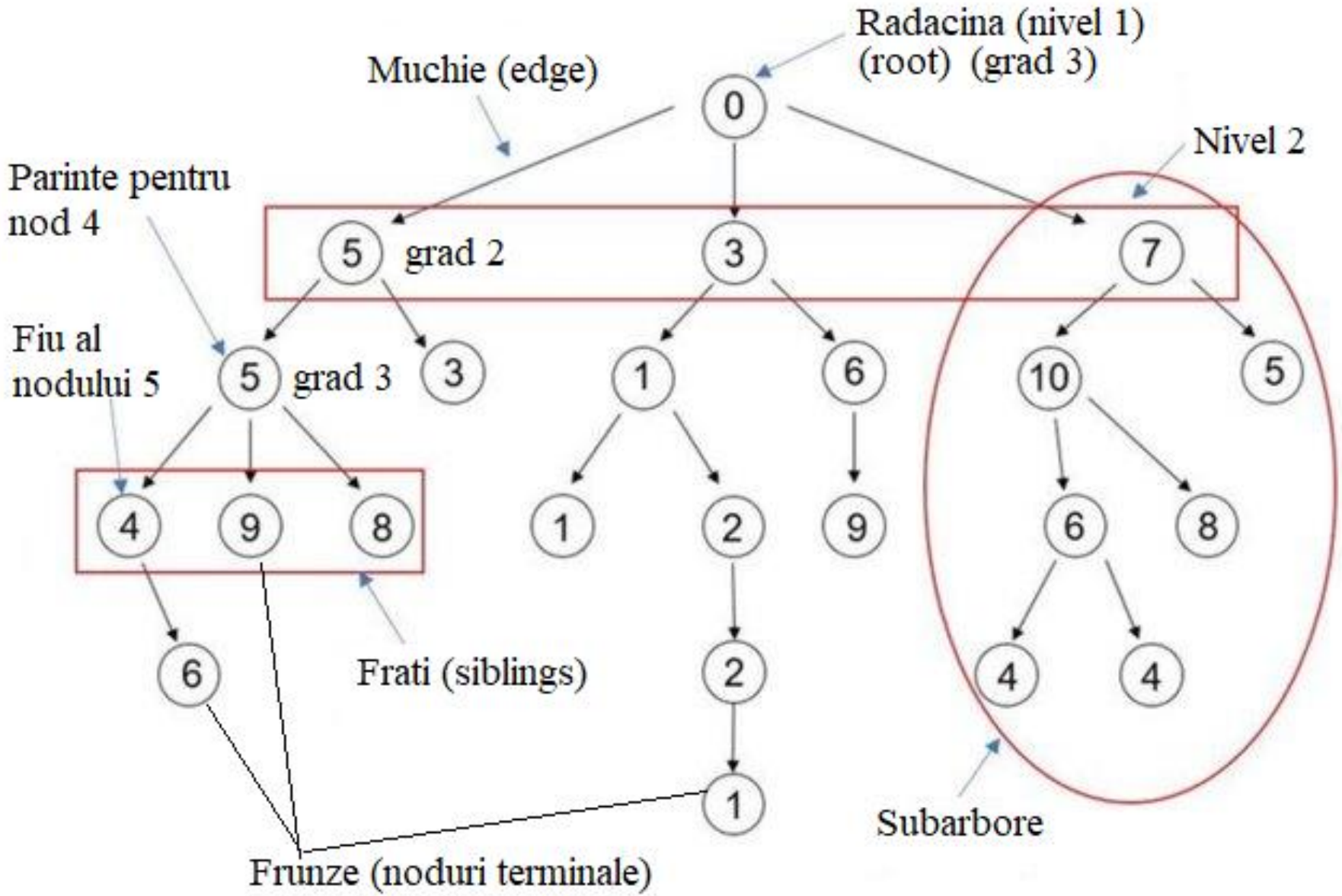
Notiuni introductive

- Numarul de subarbori ai unui nod oarecare *x* se numeste **gradul** acelui nod si se noteaza cu

$$\text{degree}(x)$$

- Nodurile de grad 0 se numesc **frunze** sau **noduri terminale**. Frunzele nu au niciun successor
- Nodurile de grad mai mare sau egal cu 1 se numesc **noduri interne** sau **neterminale**. Orice nod intern are cel putin un sucessor.
- Pentru un nod *x* al unui arbore, radacina unui subarbore al lui *x* se numeste **fiu** al lui *x* (*child(x)*) iar *x* este **parintele** (*father*) tuturor subarborilor sai

Arbore de grad oarecare



Notiuni introductive

- Nodurile care sunt fii ai aceluiasi nod *x* (au acelasi parinte) se numesc **frati**/**noduri inrudite** (*siblings*)
- O **cale** de la un nod *x* la un nod *y* al aceluiasi arbore este definita ca o secventa de noduri $x=n_0,n_1,\ldots,n_k=y$ alese astfel incat nodul n_i sa fie parintele nodului n_{i+1}.
  - Calea de la un nod la el insusi are lungimea 0.
  - Intr-un arbore exista o cale unica de la radacina la un nod oarecare al sau.
  - Lungimea unei cai este egala cu numarul de muchii ale sale
- Revenind, se poate preciza ca o **muchie**/**arc** este o pereche ordonata (*x*,*y*) in care *y* = child(*x*)

Notiuni introductive

- **Stramosii** unui nod x sunt nodurile aflate pe calea de la x la radacina arborelui

  ancestors(x)

- Multimea **succesorilor** unui nod x: successors(x)
- **Nivelul** unui nod x: level(x)
  - 1 in cazul radacinii arborelui
  - n+1 in cazul unui nod fiu care are n stramosi
- **Adancimea** arborelui este nivelul maxim al tuturor nodurilor

  depth(T)

Notiuni introductive

- Un arbore cu ordinul mai mic sau egal cu 2 se numeste **arbore binar**.
- In caz contrar se numeste **arbore multicai**.
- In cazul arborilor binar se face distinctie intre cei doi fii ai sai: stang si drept.

Reprezentarea arborilor

- Reprezentarea **“tata-fii”**
  - Se pleaca de la idea ca, pentru fiecare nod, pe langa informatia de baza, se precizeaza informatia de inlantuire spre fiii acestuia.
  - Un nod x se va reprezenta sub forma

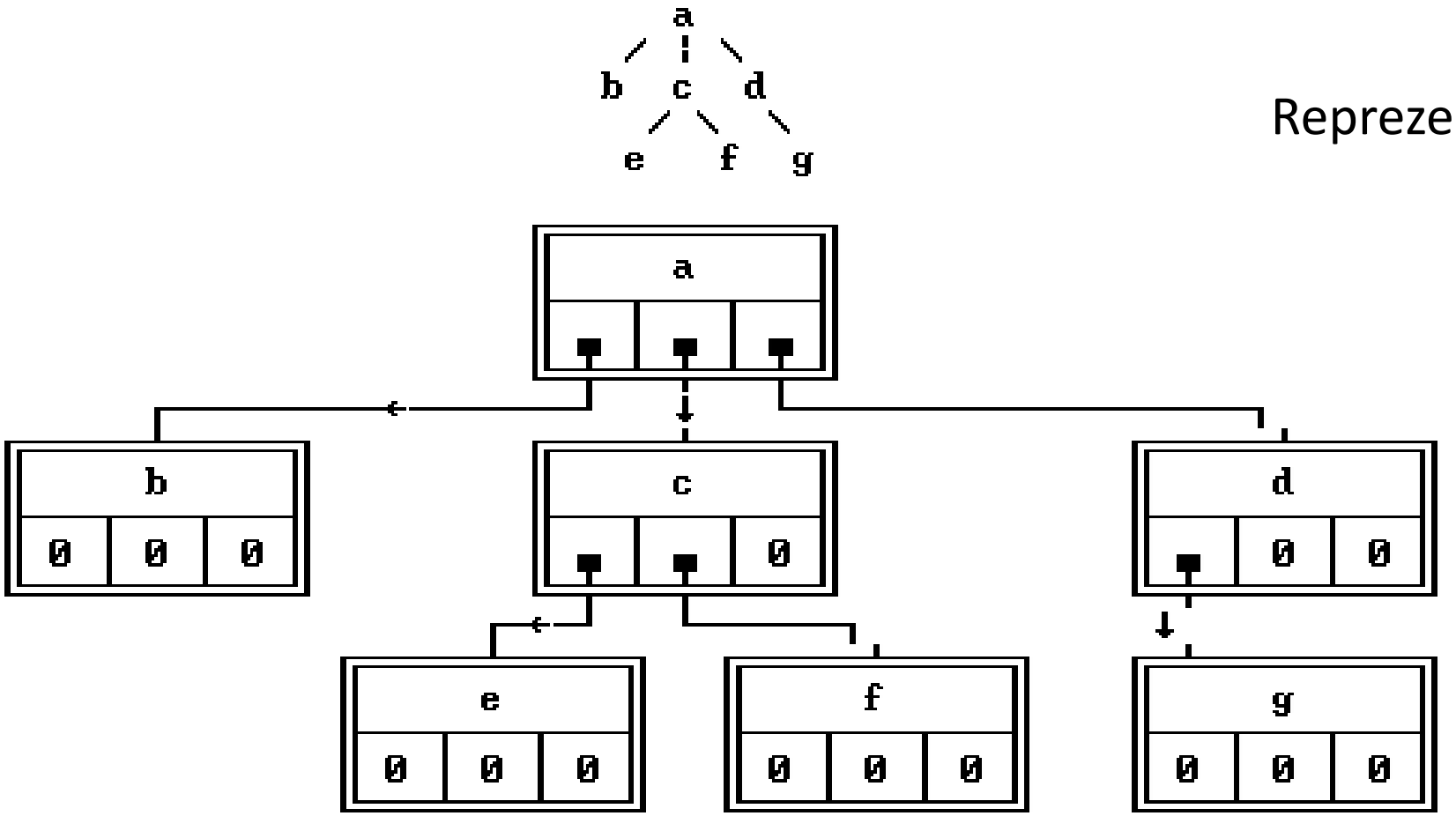
    (val, $child_1$, $child_2$, …, $child_k$), k=degree(T)

```
#define GRMAX 20
typedef Atom …
struct Nod {
    Atom data;
    Nod *v[GRMAX];
};
```

Aceasta reprezentare este utila atunci cand intereseaza ordinea descendentilor

Eficienta doar pentru arbori de grad mic (multi pointeri nuli)

Reprezentarea arborilor



Reprezentarea **“tata-fii”**

Reprezentarea arborilor

- Reprezentarea **“tata-fii”**
  - Se pleaca de la idea ca, pentru fiecare nod, pe langa informatia de baza, se precizeaza informatia de inlantuire spre fiii acestuia.
  - Un nod x se va reprezenta sub forma

    (val, $child_1$, $child_2$, …, $child_k$), k=degree(T)

```
#define GRMAX 20
typedef Atom …
struct Nod {
    Atom data;
    int grad;
    Nod **v;
};
```

Aceasta reprezentare este utila atunci cand intereseaza ordinea descendentilor

Noduri de dimensiune variabila (alocam spatiu doar pt. *grad* pointeri spre fii)

Reprezentarea arborilor

- Reprezentarea **“fiu-tata”**
  - Se utilizeaza atunci cand se doreste accesarea directa dintr-un nod a tuturor stramosilor, similar cu parcurgerea unei liste liniare inlantuite.
  - Informatia de inlantuire se refera la predecesorul nodului, a.i. nodul va fi

(val, father)

  - Pentru o astfel de reprezentare se utilizeaza o structura de memorare tip tablou

```
struct Nod {
        Atom data;
        int father;
};
```

Aceasta reprezentare este utila pentru colectii de multimi disjuncte deoarece permite implementarea eficienta a unor operatii de tip reuniune, sau gasirea unei multimi ce contine o valoare specificata

Reprezentarea arborilor

- Reprezentarea **“fiu stang – frate drept”**
  - Aceasta reprezentare transforma de fapt arborele multicai in unul binar
  - Daca la fiecare nod se adauga o legatura care sa refere nodul parinte, atunci arborele ete parcurs in ambele sensuri

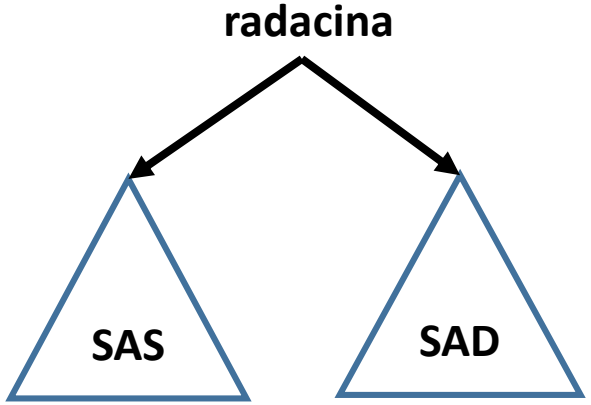
Arbori binari oarecare

Arbori binari oarecare

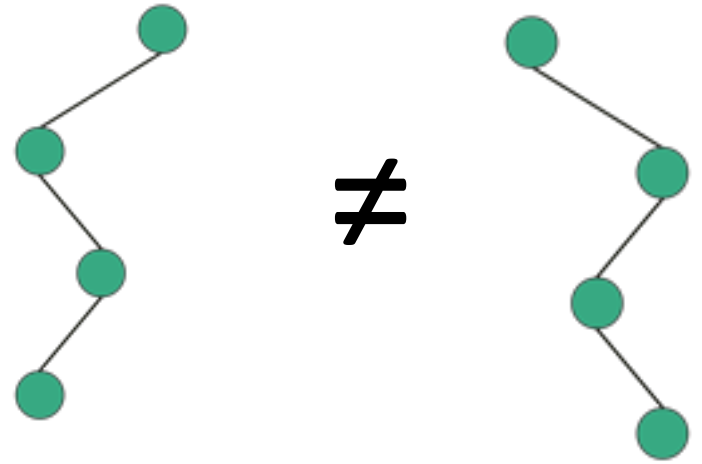
- Un arbore binar are gradul maxim 2. Un nod al unui arbore binar poate avea gradul 0, 1 sau 2.

exemple…

- In general, structura de baza a unui arbore binar se reprezinta

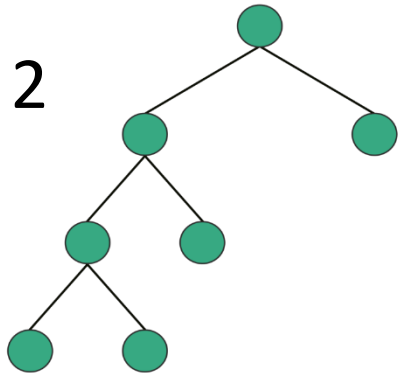


subarbore stang subarbore drept

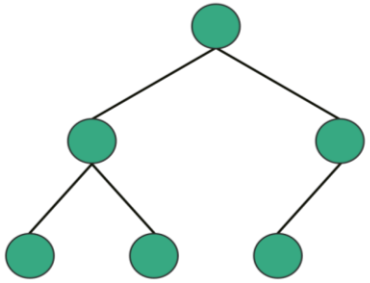


Arbori binari oarecare

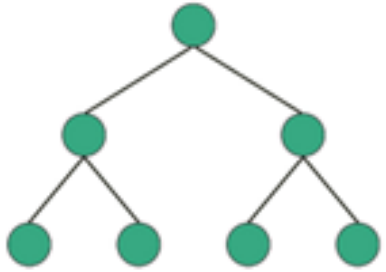
- Un arbore binar **strict** sau plin are noduri numai de grad 0 sau 2



- Un arbore binar **complet** este un arbore plin cu toate nivelurile complete cu exceptia ultimului



- Un arbore binar **perfect** este un arbore binar complet in care toate frunzele apartin ultimului nivel



Arbori binari oarecare

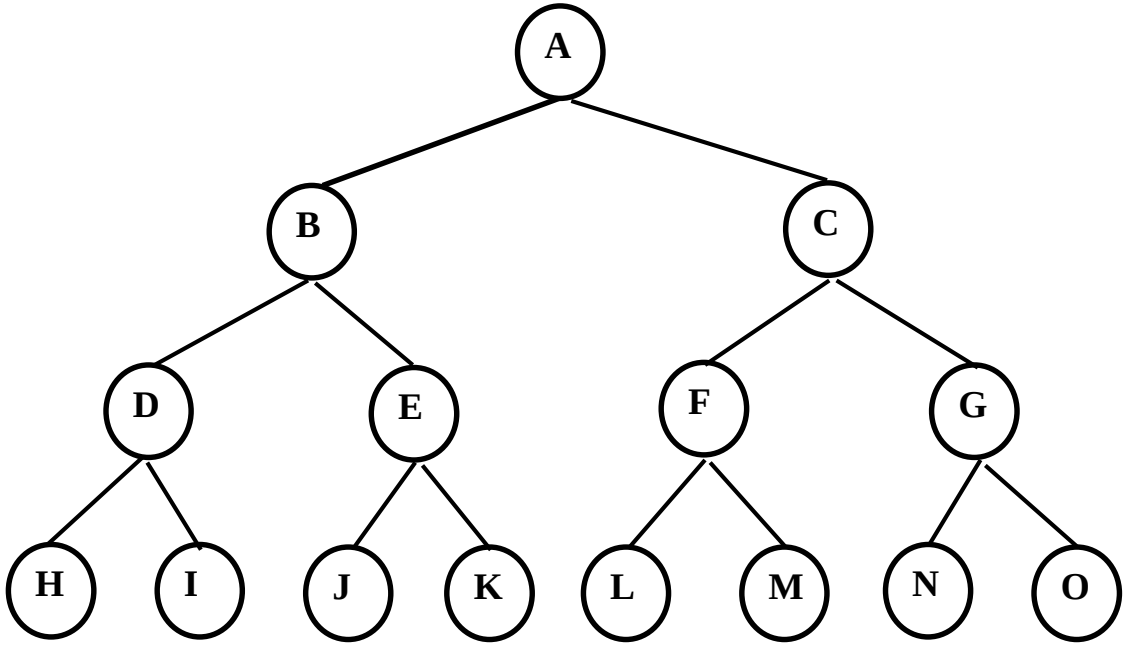
- Numarul maxim de noduri pe nivelul **i** este $\mathbf{2^{i-1}}$
  - Demonstratie (prin inductie):

    nivelul 1: $2^0=1$ (radacina)

    P(n): presupunem adevarat ca pe nivelul n sunt 2^{n-1} noduri

    P(n+1): demonstram ca pe nivelul n+1 sunt 2^n noduri

    P(n+1): $2*P(n) = 2*2^{n-1}$ (fiecare nod are maxim 2 fii)



Arbori binari oarecare

- Numarul maxim de noduri ale unui arbore de adancime **h** este **2^h-1**
  - Demonstratie:

$$nr_{max} = \sum_{i=1}^{h} 2^{i-1} = 2^0 + 2^1 + 2^2 + \cdots + 2^{h-1} = \frac{2^h - 1}{2-1} = 2^h - 1$$

Arbori binari oarecare

- Pentru un arbore binar se noteaza
  - n_2 – nr. de noduri de grad 2
  - n_1 – nr. de noduri de grad 1
  - n_0 – nr. de noduri de grad 0

$$n_0 = n_2 + 1$$

  - Demonstratie:
    - Nr. total de noduri: $n = n_2 + n_1 + n_0$
    - Nr. total de muchii: $E = n - 1$ (fiecare nod mai putin radacina are un parinte)
    - Dar, $E = 2*n_2 + 1*n_1$ (din orice nod de grad 2 pleaca 2 muchii, grad 1 – 1 muchie)

    $\Rightarrow 2*n_2 + n_1 = n_2 + n_1 + n_0 - 1 \Rightarrow n_0 = n_2 + 1$

Arbori binari oarecare

- Fie un arbore de grad **k**, cu **n** noduri de grad **k**. In reprezentarea "tata-fii" sunt **n*(k-1)+1** pointeri nuli.
  - Nr. total de pointeri = n * k
  - Nr. total de pointeri nenuli = n – 1 (nr. de arce)
  - Nr. pointeri nuli = nr. total pointeri – nr. pointeri nenuli

$$= nk - (n-1) = n(k-1) + 1$$

$$\frac{nr.pointeri\ nuli}{nr.total\ pointeri} = \frac{nk-(n-1)}{nk} = 1 - \frac{n-1}{nk}$$

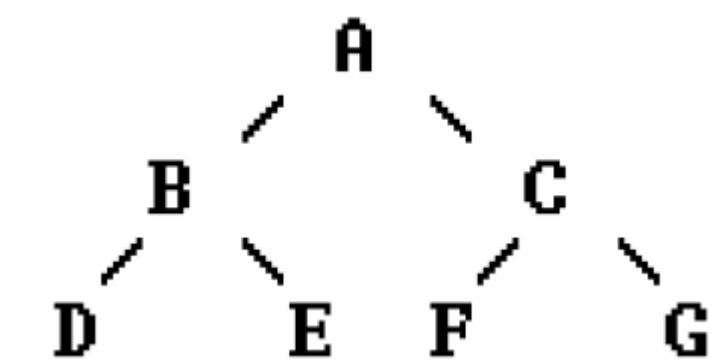
maxim cand k=2

Arborii binari utilizeza cel mai bine resursa de memorie

Arbori binari oarecare

Reprezentarea in memorie

- statica (structura ordonata)



A	B	C	D	E	F	G
1	2	3	4	5	6	7

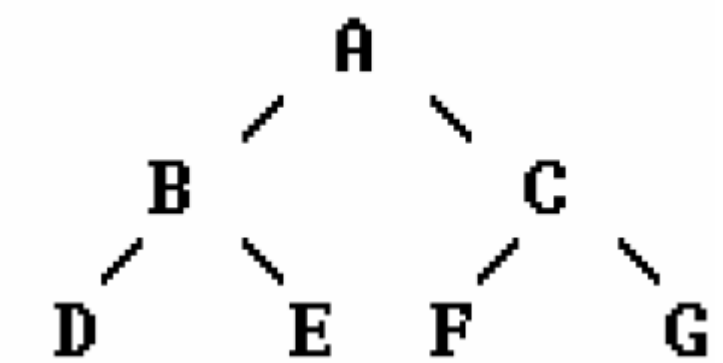
Pozitie nod	Pozitie tata	Pozitie fiu sting	Pozitie fiu drept
i	[i/2]	2*i	2*i+1

Ineficienta pt arbori binari care nu sunt completi !!

Arbori binari oarecare

Reprezentarea in memorie

- dinamica




```
struct Nod {
        Atom data;
        Nod *stg, *drt;
};
```

Arbori binari oarecare

Parcurgerea/traversarea arborelui

- Se viziteaza toate nodurile intr-o ordine bine pecizata astfel incat fiecare nod este vizitat o singura data

Metode de parcurgere:

- PREORDINE: **R-S-D**
  - se parcurge mai intai radacina, apoi subarborele stang, apoi cel drept
  - parcurgere in adancime
- INORDINE: **S-R-D**
  - se parcurge mai intai subarborele stang, apoi radacina, apoi subarborele drept
- POSTORDINE: **S-D-R**
  - se parcurge mai intai subarborele stang, apoi subarborele drept, apoi radacina

recursiv
sau
iterativ

Arbori binari oarecare

Parcurgerea/traversarea arborelui

Metode de parcurgere:

- Pe niveluri (folosind o coada):
  - se adauga radacina in coada
  - se scot noduri din coada pana cand aceasta devine goala
    - pt fiecare nod scos, valoarea acestuia se afiseaza si se adauga in coada fiii sai stang si drept

Arbori binari oarecare

Algoritmi pt operatii cu arbori binari - discutie

- Parcurgeri – impementari recursive vs. iterative
- Determinarea adancimii
- Determinare nod max
- Numararea nodurilor/frunzelor
- Afisare indentata/cu paranteze
- Swap L-R

Arbori binari oarecare

Aplicatie: codificarea expresiilor aritmetice

- Un nod intern al unui astfel de arbore va memora un operator, iar un nod frunza va memora un operand

- Cele trei tipuri de parcurgeri (RSD, SRD, SDR) vor genera cele trei forme ale expresiei aritmetice (prefixata, infixata, postfixata)

  exemplu

- Crearea arborelui din forma postfixata:

  foloseste o stiva de pointeri la noduri si creează (sub)arbori care se combina treptat într-un singur arbore final

```
/////Creare ARBORE BINAR din forma postfixata
Nod * Buildtree ( char * exp)                    // exp= sir postfixat terminat cu 0
{
        Stiva s ; char ch;
        int i=0;                                 // s este o stiva de pointeri Nod*
        Nod* r=0;                                // r= adresa radacina subarbore
        initStack(s);                            // initializare stiva goala
        while (ch=*exp++)                        // repeta pana la sfarsitul expresiei exp
        {
                r=new Nod;                               // construire nod de arbore
                r->data=ch;                              // cu operand sau operator ca date
                if (isdigit(ch))                         // daca ch este operand
                        r->stg=r->drt=0;                         // atunci nodul este o frunzã
                else                                     // daca ch este operator
                {
                        r->drt =pop (s);    // la dreapta un subarbore din stiva
                        r->stg= pop (s);    // la stanga un alt subarb. din stiva
                }
                push (s,r);          // pune radacina noului subarbore in stiva
        }
        return r;                    // radacina arbore creat { return(pop(s));}
}
```