Clasa 11A−Informatică
CAPITOLUL 4 – Structuri de date arborescente
Arbori oarecare. Definiție. Proprietăți

Arbori oarecare (arbori liberi)

Din punct de vedere structural cele mai simple grafuri sunt arborii. Figurativ, o structură de tip arbore arată ca un arbore (copac), doar că este răsturnat.

Definiție

Se numește ***arbore*** A un ***graf neorientat*** care este conex și nu conține cicluri (este aciclic).

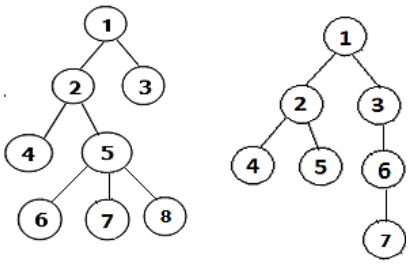


Figura 1 Arbori

Proprietățile arborilor

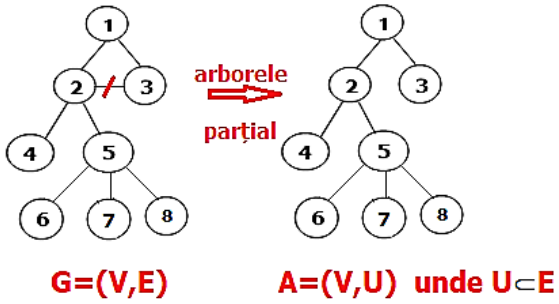
Teorema 1 Pentru un graf neorientat **G= (V, E)** cu **n>=2 noduri** și **m muchii**, următoarele definiții sunt echivalente și caracterizează un arbore:

1. **G** este **conex** și **fără cicluri**
2. **G** este **fără cicluri** și **m=n-1**
3. **G** este **conex** și **m=n-1**
4. **G** este un **graf fără cicluri**, *maximal* adică dacă se *adaugă o muchie* între 2 noduri neadiacente se formează **un ciclu** și **numai unul.**
5. **G** este un **graf conex**, *minimal* adică dacă se *elimină o muchie* oarecare se obține un graf care **NU** mai este **conex**
6. Orice pereche de noduri este legată de un lanț și numai unul

Arbore parțial

Definiție

Fie un graf neorientat G. un graf parțial al său care este arbore se numește arbore parțial al grafului G.



Teorema 1 Un graf neorientat **G= (V, E)** conține un arbore parțial dacă și numai dacă este conex

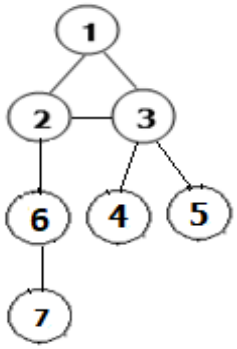
Proprietatea1 Orice arbore A=(V,U) cu **n>=2 noduri** conține *cel puțin* **2 noduri terminale.**

Proprietatea1 Un **graf neorientat** cu **n noduri** este *arbore* dacă și numai dacă are **n-1 muchii** și este **aciclic**.

Aplicații

1. Fie un **graf neorientat conex** pentru care se cunosc **numărul de noduri n** și **matricea de adiacență** – citire din fișierul **arbore.in.** Să se afișeze muchiile unui arbore parțial al său. Afișarea se va realiza în fișierul **arbore.out.**

Ex.



arbore.in	arbore.out
7 0 1 1 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0	Arborele partial contine: 1 2 2 3 3 4 3 5 2 6 6 7

Indicație: Se modifică parcurgerea DF și se afișează pentru fiecare nod de pornire np și fiecare vecin i nevizitat al său etichetele lor: **fout<<np<<" "<<i<<"\n";** După cum se observă inițial **np=1.**

Programul C++ este următorul. Realizați-l pe calculator.

```
#include<fstream>
using namespace std;
ifstream fin("arbore.in");
ofstream fout("arbore.out");

int a[30][30],viz[30],n;

void CitireMatr()
{
int i,j;
   fin>>n;
   for(i=1; i<=n; i++)
      for(j=1; j<=n; j++)
         fin>>a[i][j];
   fin.close();
}
```

```
void DF(int np)
{int i;
viz[np]=1;
for(i=1;i<=n;i++)
        if(a[np][i]==1&&viz[i]==0)
                {
                fout<<np<<" "<<i<<"\n";
                DF(i);
                 }
}
 int main ()
 {
CitireMatr();
fout<<"Arborele partial contine:\n";
DF(1);
fout.close();
return 0;
}
```

2. Se dă un graf neorientat G pentru care se cunosc numărul de noduri n, *numărul de muchii* **m** și *extremitățile muchiilor* – citire din fișierul **date.in**. Să se verifice dacă acesta este arbore. Afișarea se va realiza în fișierul **date.out.**
   Ex. Se consideră graful neorientat de la problema1. Se observă că este conex, dar are cicluri.

date.in	date.out
7 7 1 2 1 3 2 3 2 6 3 4 3 5 6 7	Matricea de adiacenta contine: 0 1 1 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 Este conex dar are cicluri

Rezolvare: Se pornește de la definiția arborelui și se verifică dacă este **conex** și **aciclic**. Vom realiza tot o **parcurgere DF** *modificată*: pentru a se **evita** *tendința de a vizita un nod deja vizitat*, dacă elementul **a[np][i]==1,** *simetricul său* **a[i][np]** va deveni **0**. Dacă totuși apare situația de a vizita un nod deja vizitat, înseamnă că **arborele conține un ciclu**.

```
int gasit;     //gasit este variabila globala deci se va initializa cu 0

void DF(int np)
{int i;
viz[np]=1;
for(i=1;i<=n;i++)
        if(a[np][i]==1)
                {
                a[i][np]=0;  //  pt a nu vizita un nod deja vizitat
                if(viz[i]==0)
                        DF(i);
                  else
                        gasit=1;
                 }
}
```

Avem 3 situaţii:

- **NU este conex** – după **parcurgerea DF** rămân *noduri nevizitate*
- **Este conex** dar **are cicluri**, deci **NU este arbore** când**,** după **parcurgerea DF** toate *nodurile sunt vizitate* și **gasit==1,** deci s-au găsit cicluri în graf.
- **Este arbore** (este **conex** și **fără cicluri**) – după **parcurgerea DF** toate *nodurile sunt vizitate* și **gasit==0**

Programul C++ este:

```
#include<fstream>
using namespace std;
ifstream fin("date.in");
ofstream fout("date.out");

int a[30][30],viz[30],n,m,gasit;
void CreareMatr()
{//citire n si m
  //citire muchii si creare matrice
 }
```

```
    void Afisare()
   {
    //Se afiseaza matricea de adiacenta linie cu linie
    }
   void DF(int np)
   {int i;
    viz[np]=1;
    for(i=1;i<=n;i++)
        if(a[np][i]==1)
                {
                a[i][np]=0;  //  pt a nu vizita un nod deja vizitat
                if(viz[i]==0)
                        DF(i);
                   else
                        gasit=1;
                 }
     }
int main()
{int i, s=0;
CreareMatr();
fout<<" Matricea de adiacenta contine:\n";
Afisare();
DF(1);
for(i=1;i<=n;i++)
          s=s+viz[i];
if(s!=n)
     fout<<"NU este conex";
  else    //s==n deci graful este conex
   {
   if(gasit==1)
        fout<<"Este conex dar are cicluri";
    else
        fout<<"Este arbore";
   }
fout.close();
return 0;
}
```