Graf conex. Componente conexe

Definiție: Un graf neorientat se numește **graf conex** dacă pentru oricare două vârfuri x și y diferite ale sale, există cel puțin un lanț care le leagă (adică x este extremitatea inițială și y este extremitatea finală).

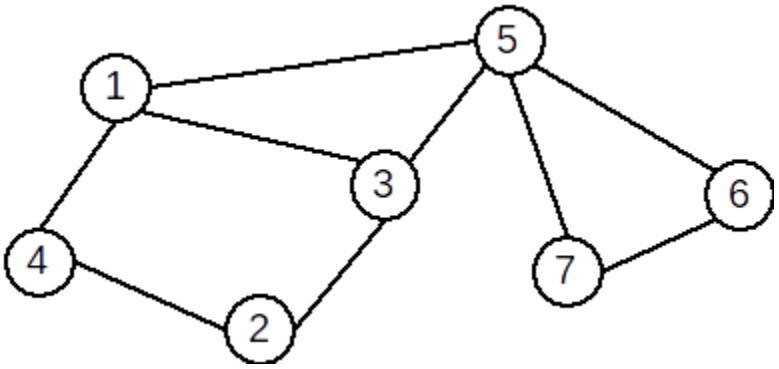
Un graf cu un singur nod este, prin definiție, conex.

Definiție: Se numește **componentă conexă** a unui graf G=(X,U) un subgraf H=(Y,V), conex, al lui G care are proprietatea că nu există nici un lanț în G care să lege un vârf din Y cu un vârf din X – Y. Altfel spus, s**e numește componentă conexă într-un graf neorientat un subgraf conex, maximal cu această proprietate (dacă s-ar mai adăuga un vârf nu ar mai fi conex).**

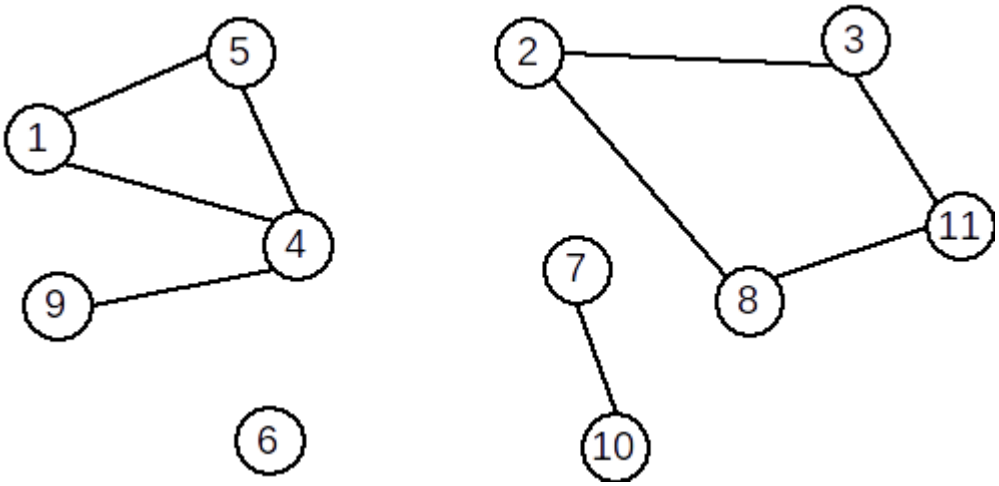
Un graf este conex dacă admite o singură componentă conexă.

Exemple:

Graful următor este conex:



Graful următor nu este conex și are 4 componente conexe.
C1: 1, 4, 5, 9
C2: 2, 3, 8, 11
C3: 7,10
C4: 6



https://www.pbinfo.ro/articole/5513/aplicatii-ale-parcurgerii-grafurilor

Verificarea conexității

Pentru a verifica dacă un graf este conex, putem folosi oricare metodă de parcurgere, astfel:

- Realizăm o parcurgere pornind din primul vârf
- Apoi, verificăm dacă au fost parcurse toate vârfurile, folosind vectorul viz ; daca toate nodurile au fost vizitate graful este conex, altfel graful nu este conex.

```
df(1); // parcurgem graful din nodul 1
ok=1;  // presupunem ca graful este conex
for(int i=1; i<=n; i++)
   if(!viz[i])  // daca mai exista cel putin un varf nevizitat graful nu este conex
    { ok=0;
      break;
    }
if (ok) cout<<"conex";
else cout<<"neconex";
```

Determinarea componentelor conexe

Determinarea componentelor conexe se poate face folosind oricare algoritm de parcurgere. Se parcurg în ordine nodurile de la 1 la n, şi se reia parcurgerea din fiecare vârf nevizitat, deoarece cu orice vârf nevizitat începe o nouă componentă conexă (cele din componentele anterioare fiind deja vizitate). De asemenea, la orice vârf nevizitat creşte numărul de componente conexe cu 1.

```
for(int i=1; i<=n; i++)
   if(!viz[i])
    { df(i); // parcurge si afiseaza componenta conexa din care face parte nodul i
      cout<<endl;
      nrc++; // creste numarul de componente conexe
    }
```

Tema: V1(2), V6(2), V7(4), V8(2), V28(1), V32(2), V34(1), V35(4), V36(3), V39(3), V40(3), V72(3), V78(2), V95(4), V96(4), V99(3)