Parcurgerea grafurilor în adâncime

Foarte mulţi algoritmi de prelucrare a grafurilor necesită examinarea tuturor nodurilor unui graf.Pentru aceasta este necesară definirea unei strategii de traversare a grafului.Se poate vorbi în principal de două tehnici de traversare:

- în adâncime (Depth First)
- în lăţime (Breadth First)

În explicarea modului de funcţionare a primei variante se foloseşte un şir de întregi, VIZITAT, de lungime n cu ajutorul căruia se marchează nodurile deja "vizitate" pentru a evita trecerea de mai multe ori prin acelaşi nod.Cu alte cuvinte VIZITAT[j] = 1 dacă nodul j a fost deja atins şi VIZITAT[j] = 0 în caz contrar.Vom spune despre un nod i că a fost explorat dacă are toate nodurile adiacente vizitate.

Procedura recursivă care asigură parcurgerea unui graf în adâncime începând cu un anumit vârf i:

Procedura *Parcurgere_în_adâncime(i)*
  pentru *toate vârfurile k adiacente cu vârful i* **execută**
    daca *vârful k este neparcurs*
      atunci *se parcurge vârful k*
        apel parcurgere_în_adâncime(k)

Ieşirea din recursivitate se produce în momentul în care nu se mai găsesc vârfuri neparcurse adiacente cu vârfurile parcurse deja. Este posibil ca după un apel al procedurii începând cu un anumit vârf i să rămână în graf vârfuri neparcurse.În această situaţie apelul procedurii se repetă pentru un alt vârf iniţial (dintre vârfurile neparcurse) până la parcurgerea tuturor nodurilor grafului. Programul apelant trebuie să asigure parcurgerea vârfului utilizat în apel.Condiţiile interne care apar în problemele particulare de backtracking pot impune o parcurgere integrală sau numai parţială a grafului.Procedura backtracking(i) este pentru cazul parcurgerii integrale a unui graf în adâncime:

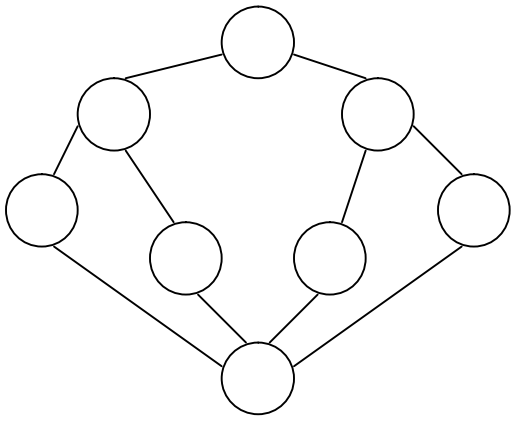
Procedura Backtracking*(i)*
  pentru *toate vârfurile k adiacente cu vârful i* **execută**
    daca *vârful k este neparcurs şi sunt îndeplinite condiţiile de continuare*
      atunci *se parcurge vârful k*

se utilizeaza vârful k în soluţie
dacă s-a ajuns la o soluţie se afişează
apel Backtracking(k)

Folosind această tehnică de traversare ne propunem să răspundem la întrebarea:

Fiind dat un graf neorientat G=(V,E), este acesta un graf conex?

Conform acestei metode explorarea unui nod este suspendată ori de câte ori un nou vârf este vizitat.Pentru graful G daca pornim din vârful 1, vizitarea nodurilor se va face în ordinea: 1,2,4,8,5,6,3,7.



Următoarea funcţie returnează true dacă graful este conex şi false în caz contrar folosind tehnica parcurgerii în adâncime:

Function *Conex: boolean;*

Procedura *adâncime(s)* {parcurge graful in adancime}
VIZITAT[s]:=1;
pentru *fiecare nod w adiacent cu s* **execută**
dacă *VIZITAT[w] = 0* **atunci** *apel adâncime(w)*
sfârşit dacă;
sfârşit pentru;
sfârşit procedura;

pentru *toate nodurile w* **execută**
VIZITAT[w]:=0;
sfârşit pentru;
apel adâncime(1);
Conex:=true;
pentru *toate nodurile w* **execută**

```
                dacă VIZITAT[w] = 0 atunci
                    conex:=false;
                sfârşit dacă;
        sfârşit pentru;
Sfârşit funcţie;
```