Exemple, teorie si cod

Traversarea grafului in adancime

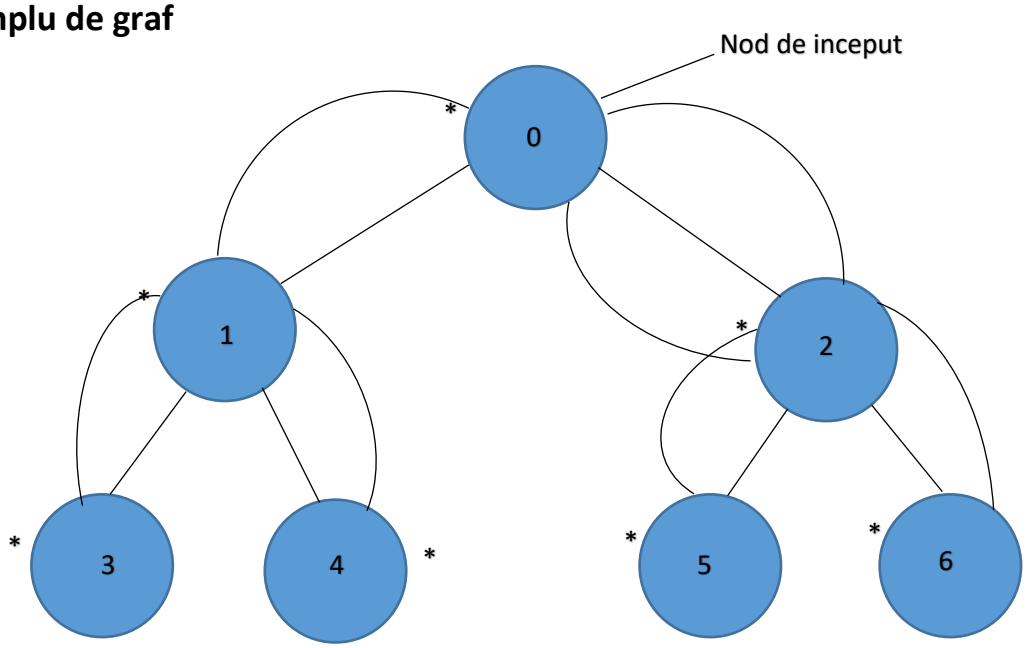
Regula

- Trebuie traversate toate nodurile din graf doar o singura data fara repetitie.

Implementare

- Se implementeaza cu structura de date numita Stiva(**STACK** in engleza).

Exemplu de graf

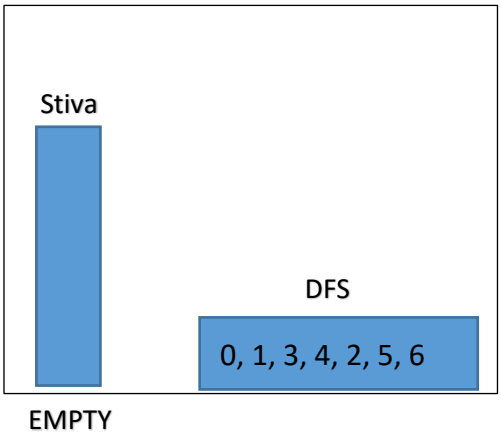


Mod de lucru :

- Marcam 0 ca nod de inceput.
- Traversam de la 0 la 1 [0->1]; il marcam pe 1 ca vizitat.
- Traversam de la 1 la 3 [1->3]; il marcam pe 3 ca vizitat.
- 3 nu are noduri adiacente; ne intoarcem inapoi de la 3 la 1 [3->1].
- Traversam de la 1 la 4 [1->4]; il marcam pe 4 ca vizitat.
- Ne intoarcem inapoi de la 4 la 1 [4->1] de la 1 la 0 [1->0].
- Traversam de la 0 la 2[0->2]; il marcam pe 2 ca vizitat.
- Traversam de la 2 la 5[2->5]; il marcam de 5 ca vizitat.
- 5 nu are noduri adiacente; ne intoarcem de la 5 la 2[5->2].
- Traversam de la 2 la 6[2->6]; il marcam pe 6 ca vizitat.
- Ne intoarcem de la 6 la 2[6->2].
- Ne intoarcem de la 2 la 0[2->0].
- Acum toate nodurile din graf sunt vizitate.
- Rezultat : 0, 1, 3, 4, 2, 5, 6

Exemple, teorie si cod

Implementare :



- PUSH(0)->STACK
- POP(0)
- PUSH(0)->DFS
- PUSH(1,2)->STACK
- POP(1)
- PUSH(1)->DFS
- PUSH(4)->STACK
- PUSH(3)->STACK
- POP(3)
- PUSH(3)->DFS
- POP(4)
- PUSH(4)->DFS
- POP(2)
- PUSH(2)->DFS
- PUSH(6)->STACK
- PUSH(5)->STACK
- POP(5)
- PUSH(5)-DFS
- POP(6)
- PUSH(6)->DFS
- STACK is empty

Reprezantam graful cu matricea de adiacenta

- Creem o matrice patratica cu n linii si n coloane in care toate elementele sunt egale cu zero reprezentand ca nu avem muchii in graf.
- Pentru fiecare muchie a grafului care este intre nodurile i si j avem matricea Mat[i][j]=1 [inseamna ca exista o muchie care leaga cele doua noduri].
- Dupa ce matricea de adiacenta este creata si umpluta cu elemente, apeleaza functia recursiva pentru nodul de inceput iar apoi pentru toate celelate noduri adiacente cu nodul de inceput.
- Avem nevoie si de un array care sa tina evidenta nodurilor vizitate vizitat[i]=true daca nodul este vizitat si vizitat[i]=false daca nodul nu este vizitat. Functia DFS nu va fi apelata pentru nodurile deja vizitate.

Structura programului

- Avem clasa Graf care este alcatuita din : {Numarul de noduri ale grafului, Numarul de muchii ale grafului, matricea de adiacenta, constructorul pentru initializarea matricei de adiacenta, functia de adaugare a nodurilor, functia de traversare DFS}
- Functia pentru a umple matricea de adiacenta goala
- Functia pentru a adauga o muchie la grafului
- Functia pentru traversarea grafului

Implementare in cod

Exemple, teorie si cod

```
#include <bits/stdc++.h>
using namespace std;
class Graf{
private:
   int numar_noduri, numar_muchii;
   int** matrice_adiacenta;
public:
   Graf(int numar_noduri,int numar_muchii);
   void inserare_muchie(int nod_inceput, int numar_muchii);
   void traversare_graf_DFS(int nod_inceput, vector<bool>& vizitat);
   void afisare_matrice_adaicenta();
};
// Matricea de adiaceanta initializata cu 0
Graf::Graf(int numar_noduri,int numar_muchii){

   this->numar_noduri = numar_noduri;
   this->numar_muchii = numar_muchii;
   matrice_adiacenta = new int*[numar_noduri];

   for(int linii=0; linii<numar_noduri;linii++){
      matrice_adiacenta[linii] = new int[numar_noduri];
      for(int coloane=0;coloane<numar_noduri;coloane++){
         matrice_adiacenta[linii][coloane]=0;
      }
   }
}
// Afisare Matrice de adiacenta
void Graf::afisare_matrice_adaicenta(){
   cout<<"Matrice de adiacenta"<<endl;
```

```
   for(int linii=0; linii<numar_noduri;linii++){
      for(int coloane=0;coloane<numar_noduri;coloane++){
         cout<<matrice_adiacenta[linii][coloane]<<" ";
      }
      cout<<"\n";
   }
}
// Adauga un nod nou in graf
void Graf::inserare_muchie(int nod_inceput, int numar_muchii){

   matrice_adiacenta[nod_inceput][numar_muchii]=1;
   matrice_adiacenta[numar_muchii][nod_inceput]=1;

}
// Traversare graf cu DFS
void Graf::traversare_graf_DFS(int nod_inceput, vector<bool>& vizitat){

   // Afiseaza nodul curent
   cout<<nod_inceput<<" ";
   // Marcheaza nodul curent ca vizitat
   vizitat[nod_inceput]=true;

   for(int i=0;i<numar_noduri;i++){
      // Daca este vreun nod adiacent cu nodul curent si nu a fost vizitat inca
      if(matrice_adiacenta[nod_inceput][i]==1 && vizitat[i]==false){
        traversare_graf_DFS(i,vizitat);
      }
   }
}
```

Exemple, teorie si cod

```
int main(){
int numar_noduri, numar_muchii;
numar_noduri = 7;
numar_muchii = 6;
Graf g(numar_noduri,numar_muchii);
g.inserare_muchie(0,1);
g.inserare_muchie(0,2);
g.inserare_muchie(1,3);
g.inserare_muchie(1,4);
g.inserare_muchie(2,5);
g.inserare_muchie(2,6);
vector<bool> vizitat(numar_noduri, false);
g.afisare_matrice_adaicenta();
 cout<<"DFS"<<endl;
g.traversare_graf_DFS(0,vizitat);
   return 0;
}
```

Matricea de adiacenta

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	**0**	**1**	**1**	**0**	**0**	**0**	**0**
1	**0**	0	**0**	**1**	**1**	**0**	**0**
2	**1**	**0**	**0**	**0**	**0**	**1**	**1**
3	**0**	**1**	**0**	**0**	**0**	**0**	**0**
4	**0**	**1**	**0**	**0**	**0**	**0**	**0**
5	**0**	**0**	**1**	**0**	**0**	**0**	**0**
6	**0**	**0**	**1**	**0**	**0**	**0**	**0**

DFS

0, 1, 3, 4, 2, 5, 6

Exemple, teorie si cod

Bibliografie :

DEPTH FIRST SEARCH(DFS) | GRAPH TRAVERSALS - DATA STRUCTURES

- **Teoria:** https://www.youtube.com/watch?v=xwlRWdAHMBg&t=1s
- **Codul:** https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/