Exemple, teorie si cod

Traversarea grafului in latime

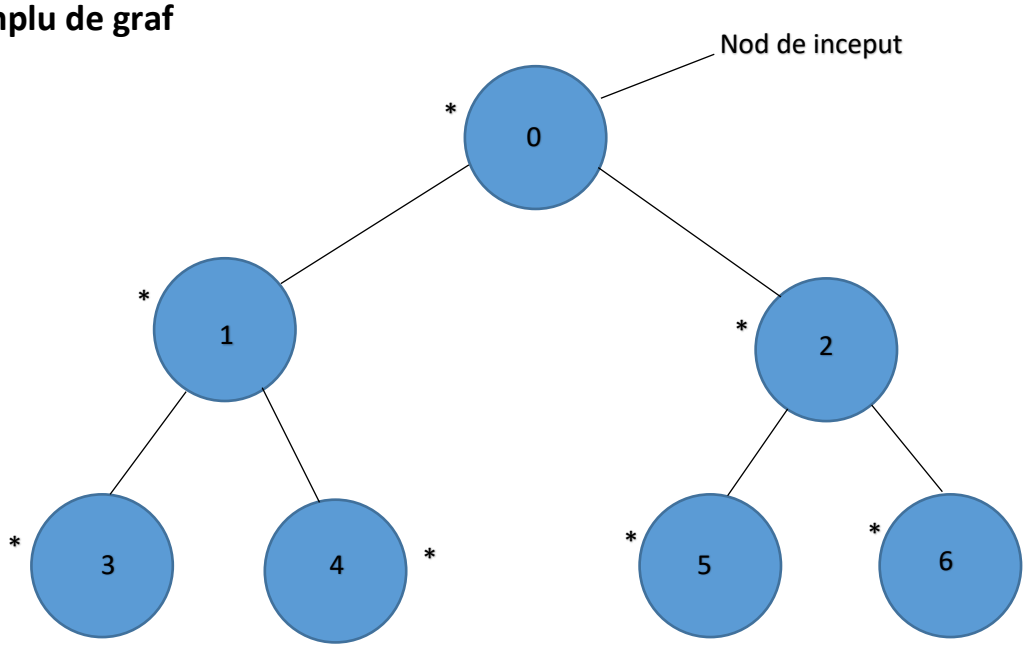
Regula

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

Implementare

- Se implementeaza cu structura de date numita Coada(**Queue** in engleza).

Exemplu de graf

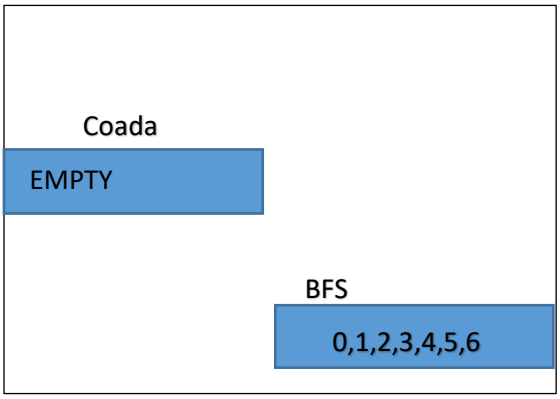


Mod de lucru :

- Marcam primul nod ca vizitat[nodul 0].
- Vizitam nodurile adiacente cu 0, 1 si 2.
- Marcam nodurile 1 si 2 ca vizitate.
- Ne intoarcem inapoi la 1 si ii vizitam toate nodurile adiacente 3 si 4 le marcam ca vizitate
- Ne intoarcem la 2 si ii vizitam toate nodurile adiacente 5 si 6 le marcam ca vizitate
- Acum toate nodurile din graf au fost vizitate
- Rezultat (0, 1, 2, 3, 4 ,5,6)

Exemple, teorie si cod

Implementare :



- Enqueue(0)->Queue
- Dequeue(0)
- Insert(0) -> BFS
- Enqueue(1,2)->Queue
- Dequeue(1)
- Insert(1)->BFS
- Enqueue(3,4)->Queue
- Dequeue(2)
- Insert(2)->BFS
- Enqueue(5,6)->Queue
- Dequeue(3)
- Insert(3)->BFS
- Dequeue(4)
- Insert(4)->BFS
- Dequeue(5)
- Insert(5)->BFS
- Dequeue(6)
- Insert(6)->BFS

Reprezantam graful cu matricea de adiacenta

- Creem o matrice patratica cu n linii si n coloane in care fiecare element este egal cu 0, ceea ce inseamna ca nu avem nici o muchie in graf.
- Pentru fiecare muchie din graf care se afla intre nodurile i si j setam matricea de adiacanta pe 1[Mat[i][j]=1].
- Dupa ce matricea a fost creata si elementele au fost adaugate in matrice, gasim traversarea BFS.

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
- Functie pentru afiasarea matricei de adiacenta

Implementare in 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_BFS(int nod_inceput);
   void afisare_matrice_adiacenta();
};
// 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_adiacenta(){
   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 BFS
void Graf::traversare_graf_BFS(int nod_inceput){

   // Vizitam vectorul astfel incat nodul sa nu fie vizitati mai mult decat odata
   // Initializam nodul pe false oentru ca nici un nod nu este vizitat la inceput
   int nod_curent;
   vector<bool> vizitat(nod_inceput, false);
   vector<int> coada;
   coada.push_back(nod_inceput);
   // Marcheaza nodul de inceput ca vizitat
   vizitat[nod_inceput] = true;
   while(coada.empty()==0){
      nod_curent = coada[0];
```

```
      // Afisare nod curent
      cout<<nod_curent<<" ";
      coada.erase(coada.begin());
      for(int i=0;i<numar_noduri;i++){
         // Daca exita muchie intre noduri si nodul nu este vizitat
         if(matrice_adiacenta[nod_curent][i]==1 && vizitat[i]==0){
            // eliminam elementul vizitat din coada
               coada. push_back(i);
               vizitat[i]= true;
         }
      }
   }
}
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);
g.afisare_matrice_adiacenta();
cout<<"BFS"<<endl;
g.traversare_graf_BFS(0);
   return 0;
}
```

Exemple, teorie si cod

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**

BFS

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

Bibliografie :

BREADTH FIRST SEARCH(BFS) | GRAPH TRAVERSALS - DATA STRUCTURES

- **Teoria:** https://www.youtube.com/watch?v=cMELxr5hKYU&t=578s
- **Codul:** https://www.geeksforgeeks.org/implementation-of-bfs-using-adjacency-matrix/