L12. Grafuri

Exercitiu: Desenati pe hartie graful reprezentat de urmatoarele date:

n=7
m=12
V={1,2,3,4,5,6,7}
E={(1,2), (1,3), (2,4), (2,5), (3,5), (4,1), (4,7), (5,4), (5,6), (5,7), (6,7), (7,5) }

Reprezentarea prin matrice de adiacenta

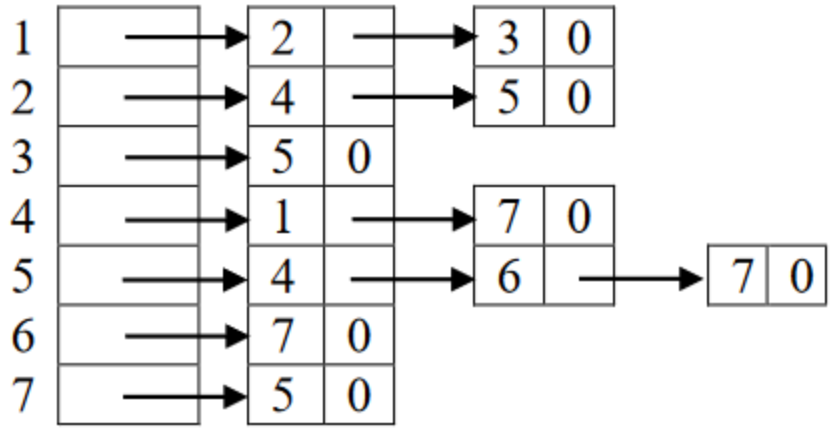
A=

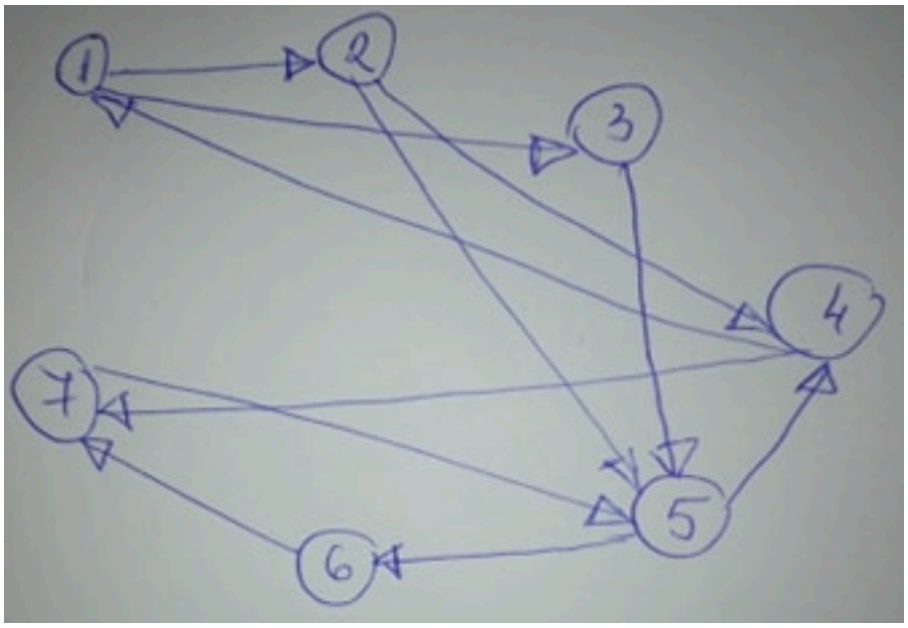
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	1	1	0	0	0	0
2	0	0	0	1	1	0	0
3	0	0	0	0	1	0	0
4	1	0	0	0	0	0	1
5	0	0	0	1	0	1	1
6	0	0	0	0	0	0	1
7	0	0	0	0	1	0	0

Reprezentarea prin liste de adiacenta

1	2,3
2	4,5
3	5
4	1,7
5	4,6,7
6	7
7	5

Pentru exemplul nostru:





Explorarea in adancime (DFS)

```
DFS_iterativ(A, L, M , i)
//A - matricea de adiacenta
//L - lista nodurilor care formeaza parcurgerea din i
//M - vector de etichete vizitat/nevizitat - initial 0
//i - nodul de start
//S - stiva
push(S,i)
while S ≠ empty do
      j := pop(S)
      if M[j] = 0 then
            insert(L, j)
            M[j] := 1
            for k = 1 to n do
                  if A[j,k] = 1 then
                        push(S,k)
                  end-if
            end-for
      end-if
end-while

DFS_recursiv(A, L, M, i)
//A - matricea de adiacenta
//L - lista nodurilor care formeaza parcurgerea din i
//M - vector de etichete vizitat/nevizitat - initial 0
//i - nodul de start
insert(L, i)
M[i] := 1
for k = 1 to n do
      if A[i,k] = 1 then
            if M[k] = 0 then
                  DFS_recursiv(A, L, M, k)
            end-if
      end-if
end-for
```

Explorarea in latime (BFS)

```
BFS_iterativ(A, n, L, M, i)
//A - matricea de adiacenta n*n
//L - lista nodurilor care formeaza parcurgerea din i
//M - vector de etichete vizitat/nevizitat - initial 0
//i - nodul de start
//Q - coada
        InitQueue(Q)
        Put(Q,i)
        While Q!=empty do
                j := Get(Q)
                Insert(L,j) //sau afisare/procesare
                For k:=1 to n do
                        If A[j,k]=1 AND M[k]=0
                                M[k]:=1
                                Put(Q,k)
                        EndIf
                EndFor
        EndWhile
End
```

```
BFS_recursiv(A, n, L, Q, M)
//A - matricea de adiacenta n*n
//L - lista nodurilor care formeaza parcurgerea din i
//M - vector de etichete vizitat/nevizitat - initial 0
//Q - coada
       If Q!=empty do
               j := Get(Q)
               Insert(L,j) //sau afisare/procesare
               For k:=1 to n do
                       If A[j,k]=1 AND M[k]=0
                               M[k]:=1
                               Put(Q,k)
                       EndIf
               EndFor
               BFS_recursiv(A,n,L,Q,M)
       EndIf
End
```

La apel:

```
InitQueue(Q)
Put(Q, i) //i = nod de start
Init(M) //M[k] = 0, pt toate val. lui k
BFS_recursiv(A,n,L,Q,M)
```

Aplicatii

1. Se da un graf orientat ponderat prin urmatoarele informatii :

```
n m
i1 j1 cost_i1_j1
i2 j2 cost_i2_j2
…
```

Sa se construiasca reprezentarea grafului prin matricea de adiacenta.

Indicatii:

- Se citeste `n` (numarul de noduri din graf)
- Se aloca o matrice de dimensiune `n` x `n` si se initializeaza cu valori 0

- Se citesc pe rand cele `m` muchii; pentru fiecare muchie `(i,j)` se scrie valoarea `cost_i_j` de pe pozitia `(i,j)` din matricea de adiacenta

2. Sa se scrie o functie recursiva pentru afisarea nodurilor unui graf in ordinea DFS.

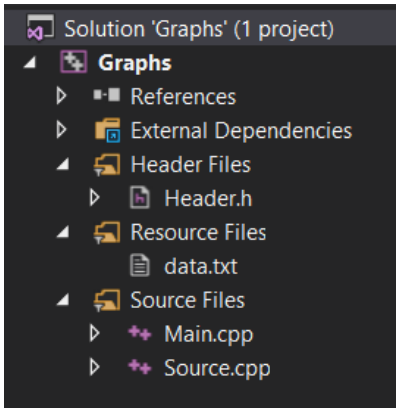
Indicatii:

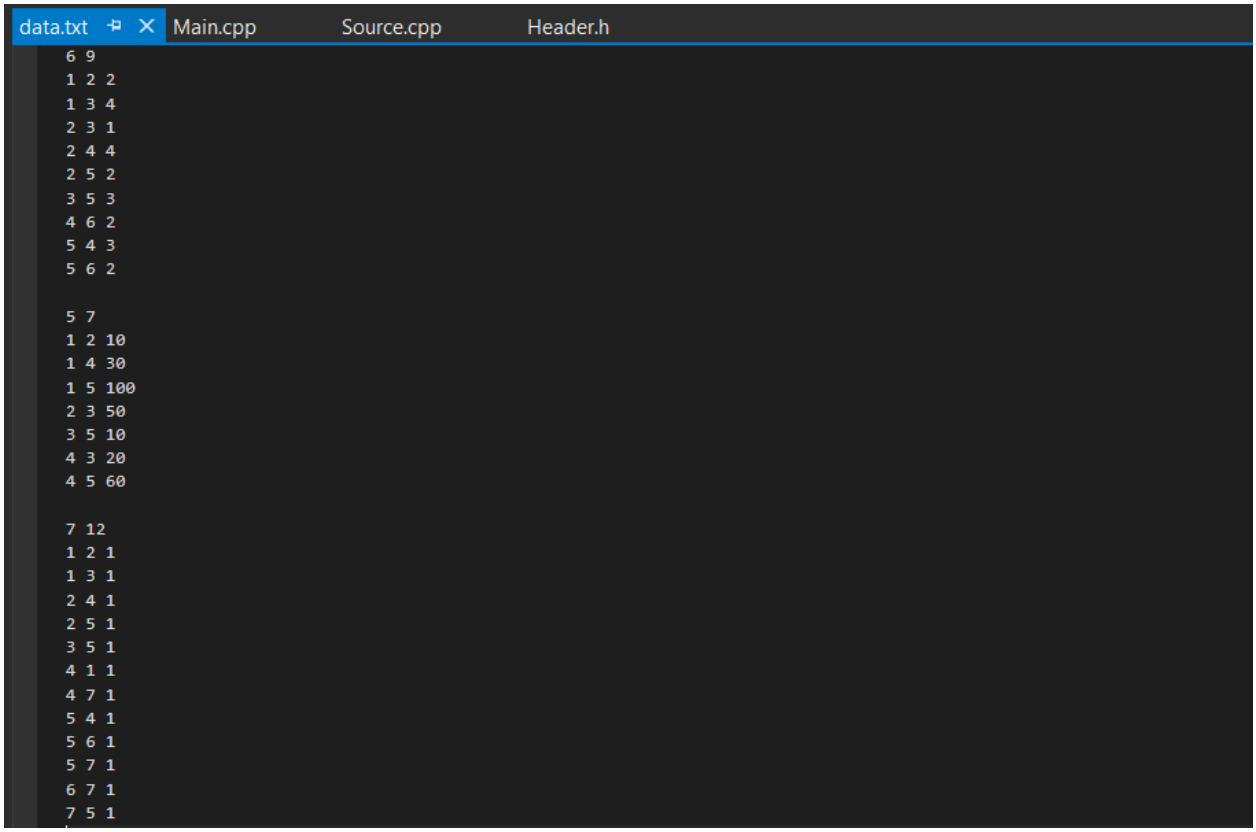
- Functia de explorare DFS va primi ca argumente: matricea de adiacenta ce reprezinta graful, n – numarul de noduri din graf, i – nodul de start, M – vectorul de marcaje, L – lista nodurilor ce formeaza parcurgerea din i

3. Sa se scrie functia iterativa pentru parcurgerea DFS.

4. Sa se scrie functiile pentru parcurgerea BFS, iterativ, respectiv recursiv.

NOTARE
1 – 2p
2 – 2p
3 – 2p
4 – 3p/3p


Solution 'Graphs' (1 project)
Graphs
References
External Dependencies
Header Files
Header.h
Resource Files
data.txt
Source Files
Main.cpp
Source.cpp


data.txt
Main.cpp
Source.cpp
Header.h
6 9
1 2 2
1 3 4
2 3 1
2 4 4
2 5 2
3 5 3
4 6 2
5 4 3
5 6 2
5 7
1 2 10
1 4 30
1 5 100
2 3 50
3 5 10
4 3 20
4 5 60
7 12
1 2 1
1 3 1
2 4 1
2 5 1
3 5 1
4 1 1
4 7 1
5 4 1
5 6 1
5 7 1
6 7 1
7 5 1

data.txt Main.cpp Source.cpp Header.h*

Graphs (Global Scope)

```
#pragma once

#include <iostream>
#include <stack>
#include <queue>

using namespace std;

void DFS(int** a, int n, int i, int*viz, queue<int>& L);
void DFS_it(int** a, int n, int i, int*viz, queue<int>& L);

void BFS_it(int** a, int n, int i, int* viz, queue<int>& L);
void BFS(int** a, int n, int* viz, queue<int>& q, queue<int>& L);
```

data.txt Main.cpp* Source.cpp Header.h

Graphs (Global Scope)

```
#include "Header.h"

void DFS(int** a, int n, int i, int*viz, queue<int>& L)
{
    viz[i] = 1;
    L.push(i);
    for (int k = 0;k < n;k++)
    {
        if (a[i][k] && !viz[k])
            DFS(a, n, k, viz, L);
    }
}
```

data.txt | Main.cpp* | Source.cpp | Header.h

Graphs | (Global Scope)

```cpp
#include "Header.h"

int main()
{
    int **a, * viz, n, m, i,j;
    queue<int> L;
    queue<int> q;
    FILE* f;

    if (fopen_s(&f, "data.txt", "r"))
    {
        fprintf(stderr, " \n Eroare la deschiderea fisierului \n");
        exit(EXIT_FAILURE);
    }
    fscanf_s(f, "%d %d", &n, &m);

    a = new int*[n];
    viz = new int[n];
    for (int k = 0;k < n;k++)
    {
        a[k] = new int[n];
    }
    for (i = 0;i < n;i++)
        for (j = 0;j < n;j++)
            a[i][j] = 0;
    for (int k = 0;k < m;k++)
    {
        fscanf_s(f, "%d %d", &i, &j);
        fscanf_s(f, "%d", &a[i - 1][j - 1]);
    }
    if (fclose(f))
    {
        fprintf(stderr, " \n Eroare la deschiderea fisierului \n");
        exit(EXIT_FAILURE);
    }
```

81 %

```cpp
    cout << "\n Matricea de adiacenta:\n";
    for (int i = 0;i < n;i++)
    {
        for (int j = 0;j < n;j++)
            cout << a[i][j] << "\t";
        cout << endl;
    }
    cout << endl;

    cout << "\n Traversals: ";
    cout << "\n DFS rec: ";
    for (int k = 0;k < n;k++)
        viz[k] = 0;
    DFS(a, n, 0, viz, L);
    while (!L.empty())
    {
        cout << L.front() << " ";
        L.pop();
    }
    cout << endl;
```

pentru simplitate folositi queue si stack din stl

http://www.cplusplus.com/reference/queue/queue/

http://www.cplusplus.com/reference/stack/stack/