Ministerul Educaţiei, Culturii și Cercetării

al Republicii Moldova

Universitatea Tehnică a Moldovei

Departamentul Ingineria Software şi Automatică

RAPORT

Lucrarea de laborator Nr.4

la Matematica specială

Tema : DETERMINAREA FLUXULUI MAXIM ÎNTR-O REŢEA DE TRANSPORT

A efectuat : gr. SI – 201 , Ivanova Evghenia

A verificat : asis. univ. Popovici Nadejda

Chişinău – 2021

Scopul Lucrării :

Ø Studierea noţiunilor de bază leagate de reţelele de transport;

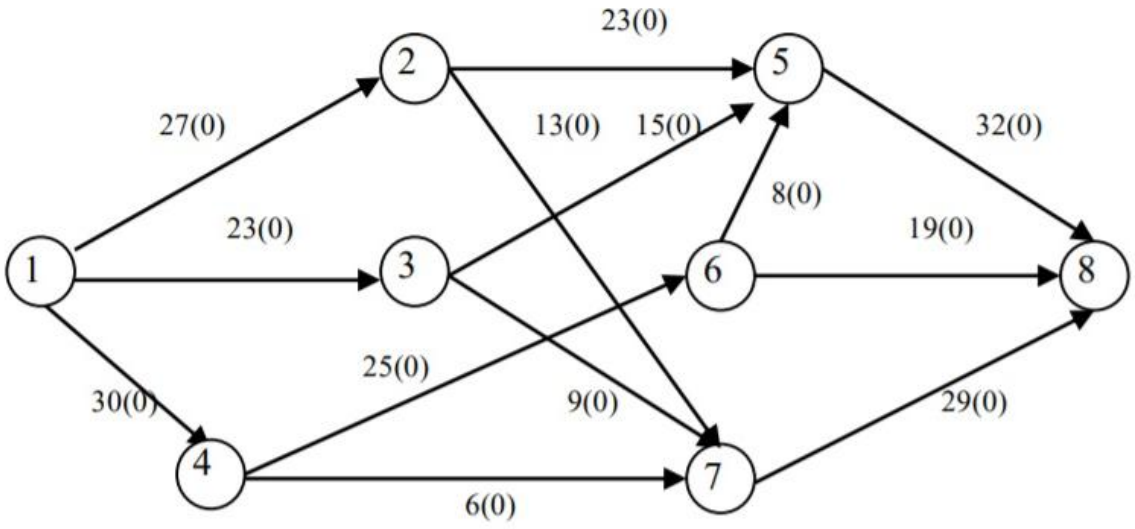
Ø Programarea algoritmului Ford-Fulkerson pentru determinarea fluxului maxim într-o reţea de transport.

Sarcina de bază :

1. Realizaţi procedura introducerii unei reţele de transport cu posibilităţi de verificare a corectitudinii datelor introduse;
2. În conformitate cu algoritmul Ford-Fulkerson elaboraţi procedura determinării fluxului maxim pentru valori întregi ale capacităţilor arcelor;
3. Elaboraţi programul care va permite îndeplinirea următoarelor deziderate:
   - introducerea reţelei de transport în memorie;
   - determinarea fluxului maxim pentru reţeaua concretă;
   - extragerea datelor obţinute (fluxul maxim şi fluxul fiecărui arc) la display şi printer.

EXEMPLU. De determinat valoarea fluxului maximal in rețeaua de trasport **G=<X,U,C>** , unde $X=\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ mulțimea vârfurilor;

U={(1,2), (1,3) , (1,4), (2,5), (2,7), (3,5), (3,7), (4,6), (4,7), (5,8), (6,5), (6,8), (7,8) } – mulțimea arcelor, iar C – capacitățile arcelor: $C(x_1, x_2) = c_{12}$= 27, c_{13}=23, c_{14}=30, c_{25}=23, c_{27}= 13, c_{35}=15, c_{37}=9, c_{46}=25, c_{47}= 6, c_{58}= 32, c_{65}= 6, c_{68}= 19, c_{78}= 29. Reprezentăm graful



Codul programului :

```
#include <iostream>
#include <limits.h>
#include <queue>
#include <string.h>
using namespace std;
#define V 8   // Numărul de vârfuri în graful dat
//Returnează adevăr dacă există o cale de la 's' la 't' în graf. Umple părintele pentru a stoca calea
bool bfs(int rGraph[V][V], int s, int t, int parent[])
{
        // Creăm o matrice vizitată și marcăm toate vârfurile ca nevizitate
        bool visited[V];
        memset(visited, 0, sizeof(visited));
        // Creăm o coadă, eliberăm vârful sursă și marcăm vârful sursă ca vizitat
        queue<int> q;
        q.push(s);
        visited[s] = true;
        parent[s] = -1;
        while(!q.empty()) {
                int u = q.front();
                q.pop();
                for(int v = 0; v < V; v++) {
                        if(visited[v] == false && rGraph[u][v] > 0) {
//Dacă găsim o conexiune,atunci nu există punct în BFS,setăm părintele și returnăm adevăr

                if(v == t) {
```

```
                    parent[v] = u;
                    return true;
                }
                q.push(v);
                parent[v] = u;
                visited[v] = true;
            }
        }
    }
    // Nu am ajuns la chiuveta în BFS pornind de la sursă, așa că returnează false
    return false;
}
int fordFulkerson(int graph[V][V], int s, int t)
{
    int u, v;
    //Creăm un graf rezidual și completăm graful cu capacitățile date în graful original
    int rGraph[V][V]; //căutăm dacă există o margine, dacă rGraph[i][j]=0, atunci nu există
    for (u = 0; u < V; u++)
        for (v = 0; v < V; v++)
            rGraph[u][v] = graph[u][v];
    int max_flow = 0, parent[V]; //pentru a stoca calea
    while (bfs(rGraph, s, t, parent)) {
    //Găsim capacitatea reziduală minimă a marginilor de-a lungul căii umplute de BFS.
        int path_flow = INT_MAX;
        for (v = t; v != s; v = parent[v]) {
```

```
            u = parent[v];
            path_flow = min(path_flow, rGraph[u][v]);
        }
    //actualizați capacitățile reziduale ale marginilor și marginilor inverse de-a lungul traseului
        for (v = t; v != s; v = parent[v]) {
            u = parent[v];
            rGraph[u][v] -= path_flow;
            rGraph[v][u] += path_flow;
        } //Adăugați fluxul de cale la fluxul general
        max_flow += path_flow;
    }
    return max_flow;
}
int main()
{
    int graph[V][V]= { { 0, 27, 23, 30, 0, 0, 0 ,0 },
                       { 0, 0, 0, 0, 23, 0, 13, 0 },
                       { 0, 0, 0, 0, 15, 0, 9, 0 },
                       { 0, 0, 0, 0, 0, 25, 6, 0 },
                       { 0, 0, 0, 0, 0, 0, 0, 32 },
                       { 0, 0, 0, 0, 6, 0, 0, 19 },
                       { 0, 0, 0, 0, 0, 0, 0, 29 },
                       { 0, 0, 0, 0, 6, 0, 0, 0 }};
   printf("Fluxul maxim posibil este %d",fordFulkerson(graph, 0, 7));
}
```

Execuția Programului :

```
The maximum possible flow is 79

...Program finished with exit code 0
Press ENTER to exit console.
```

ÎNTREBĂRI DE CONTROL

1. Ce se numeşte reţea de transport?
2. Formulaţi noţiunile de flux şi capacitate.
3. Ce este un arc saturat? Dar un drum saturat?
4. Ce se numeşte flux complet? Ce este un flux maxim?
5. Definiţi noţiunea de tăietură.
6. Formulaţi teorema Ford-Fulkerson.
7. Descrieţi algoritmul de determinare a fluxului maxim.
8. Demonstraţi că algoritmul se va opri după un număr finit de paşi.

Concluzia :

Un graf orientat poate fi utilizat pentru modelarea unui proces de transport într-o rețea între un producător *s* și un consumator *t*. Destinația nu poate consuma mai mult decât să producă, iar cantitatea trimisă pe o cale nu poate depăși capacitatea sa de transport.

Rețelele de transport pot modela curgerea lichidului în sisteme cu țevi, deplasarea pieselor pe benzi rulante, deplasarea curentului prin rețele electrice, transmiterea informațiilor prin rețele de comunicare etc.

Algoritmul Ford-Fulkerson este o metodă iterativă de găsire a fluxului maxim într-un graf care pleacă de la ideea: cât timp mai există un drum de ameliorare (o cale de la sursă la destinație) pot pompa pe această cale un flux suplimentar egal cu capacitatea reziduală a căii.

Acest algoritm reprezintă mai mult un șablon de rezolvare pentru că nu detaliază modul în care se alege drumul de ameliorare din rețeaua reziduală.