**Виконав:** Герман І.В.

**Група:** КН-404

**Кафедра:** САПР

**Дисципліна:** Дискретні моделі в САПР

# Варіант: 6

**Перевірив:** канд. техн. наук , доц. Кривий Р.З.

**Дата:** 20.03.2021

**GitHub:** https://github.com/Gera-byte/DM\_CAD/tree/main/Lab4

# ЗВІТ

до лабораторної роботи №4 на тему:

Потокові алгоритми

# МЕТА РОБОТИ

Метою даної лабораторної роботи є вивчення потокових алгоритмів

**2. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ**

Останнім часом значно зросла зацікавленість учених та практиків потоковими моделями. Це пов’язано із впровадженням та активним розвитком різноманітних територіально розподілених систем: трубопровідних, транспортних, телекомунікаційних та ін. Основою таких систем є певна мережа (мережа трубопроводів, доріг, каналів зв’язку тощо), в якій циркулюють певні потоки (потоки речовин, транспорту, даних тощо), тому задачі, які доводиться розв’язувати при проектуванні та експлуатації систем з мережною структурою, часто зводяться до розробки математичних моделей розподілу потоків та постановки і розв’язання відповідних оптимізаційних задач. Відомі моделі розподілу потоків у мережах базуються на поняттях теорії графів. Це пов’язано з тим, що граф дає можливість наочно відобразити структуру мережі, а параметри його вузлів і дуг – представити основними числовими характеристиками її елементів. Набір характеристик залежить від природи модельованої системи, а також характеру розв’язуваних задач, однак у потокових моделях їх, як правило, представляють такими параметрами, як зовнішній потік у вузлі, потік по дузі, пропускна здатність дуги, вартість передавання одиниці потоку по дузі тощо. Потокові задачі, як правило, зводяться до пошуку такого розподілу потоків у мережі, при якому б забезпечувався екстремум деякого критерію. При цьому мають враховуватися обмеження, що накладаються умовами збереження потоків у вузлах і неперевищення потоками пропускної здатності дуг. Типовими потоковими задачами є задача про потік мінімальної вартості, про максимальний потік, транспортна задача, задача про призначення та інші. Для їх розв’язання розроблено чимало ефективних алгоритмів, сформувався навіть відповідний напрям обчислювальних методів під назвою потокового програмування

ПОНЯТТЯ ПРО ПОТОКИ

Потік-визначає спосіб пересилання деяких об’єктів з одного пункту в інший. Розв’язання задачі потоку зводиться до таких основних підзадач:

Максимізація сумарного обсягу перевезень

Мінімізація вартості пересилань предметів з одного пункту в інший

Мінімізація часу перевезень в заданій системі

**3. ЗАВДАННЯ**

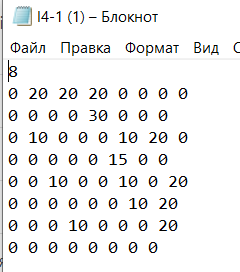
Потокові алгоритми. Реалізувати алгоритм Форда — Фалкерсона  


Рис. 1. Індивідуальне завдання

# 4. Код програми

#include <limits.h>

#include <string.h>

#include <iostream>

#include <queue>

#include <vector>

#include <fstream>

using namespace std;

bool bfs(const vector<vector<int>>& rGraph, int s, int t, vector<int>& parent)

{

vector<bool>visited(rGraph.size());

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 < rGraph.size(); v++) {

if (visited[v] == false && rGraph[u][v] > 0) {

q.push(v);

parent[v] = u;

visited[v] = true;

}

}

}

return (visited[t] == true);

}

int fordFulkerson(const vector<vector<int>>& graph, int s, int t)

{

int u, v;

vector<vector<int>> rGraph(graph.size(), vector<int>(graph.size()));

for (u = 0; u < graph.size(); u++)

for (v = 0; v < graph.size(); v++)

rGraph[u][v] = graph[u][v];

vector<int> parent(graph.size());

int max\_flow = 0;

while (bfs(rGraph, s, t, parent))

{

int path\_flow = INT\_MAX;

for (v = t; v != s; v = parent[v]) {

u = parent[v];

path\_flow = min(path\_flow, rGraph[u][v]);

}

for (v = t; v != s; v = parent[v]) {

u = parent[v];

rGraph[u][v] -= path\_flow;

rGraph[v][u] += path\_flow;

}

max\_flow += path\_flow;

}

return max\_flow;

}

int main()

{

int size;

ifstream input("matrix.txt");

input >> size;

vector<vector<int>> graph(size, vector<int>(size));

for (int i = 0; i < size; i++)

{

for (int j = 0; j < size; j++)

{

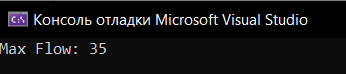
input >> graph[i][j];

}

}

cout << "Max Flow: " << fordFulkerson(graph, 0, 5) << endl;

}

  
Рис. 2. Результат виконання програми

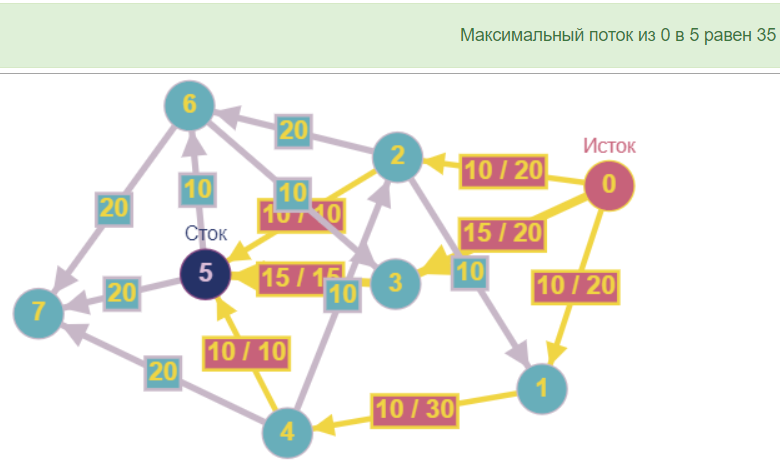


Рис. 3. Візуалізація

## Висновок

На лабораторній роботі вивчено і досліджено алгоритм Форда-Фалкерсона