ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ

УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра Інформатики

**Звіт**

з лабораторної роботи № 3

на тему «Графи»

Виконав Перевірив

ст.гр.ІТІНФ-20-1 ас.каф. Інформатики

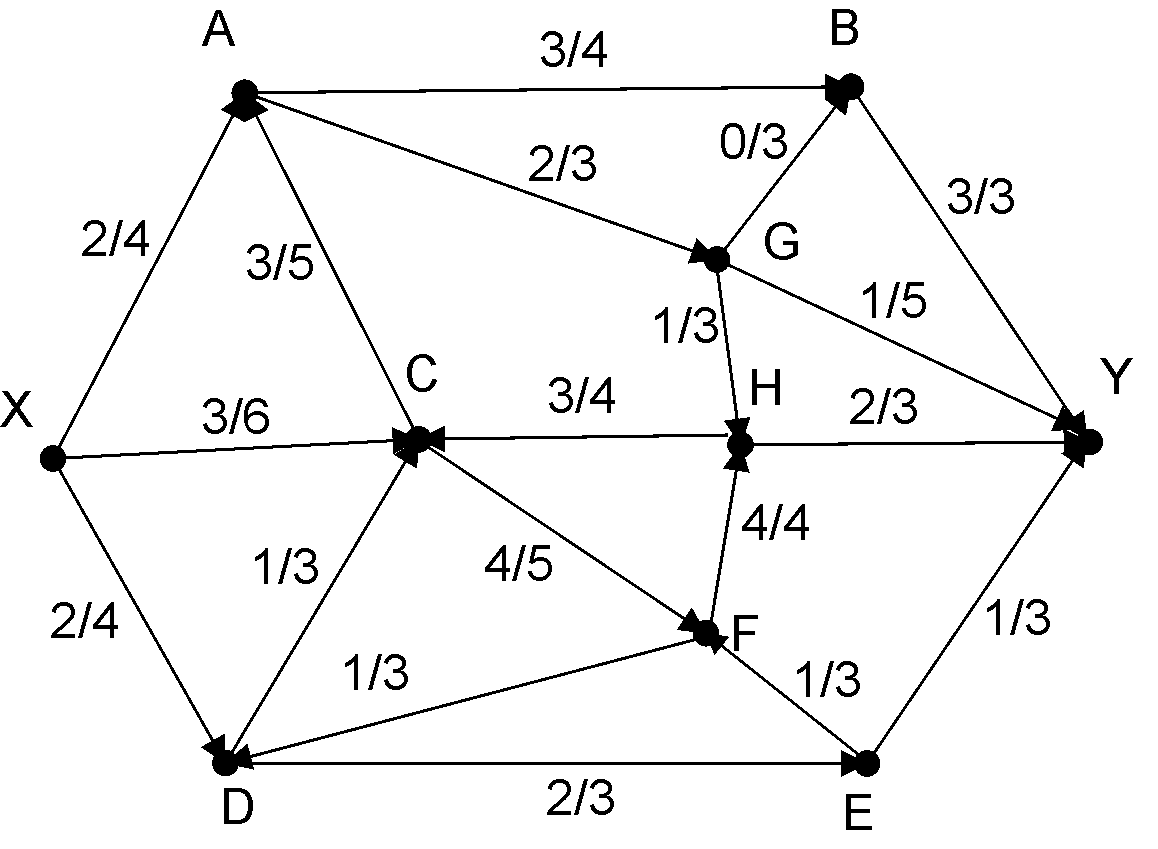
Самченко С.О. Бродецький Ф. А.

Харків 2021

**Мета роботи:** Реалізація алгоритму Форда-Фалкерсона для мереж.

Вершина - початок транспортної мережі, з якої дуги тільки виходять. вершина z - кінець транспортної мережі, в яку дуги тільки входять. На безлічі дуг задана целочисленная функція с (і)> О, де з (і) - пропускна здатність дуги. Потоком по транспортній мережі називається целочисленная функція.

Вариант 21 (9)



Реалызація:

#include <iostream>

using namespace std;

#define GREEN 0

#define YELLOW 1

#define RED 2

#define SIZE 10

int bandwidth[SIZE][SIZE] = {

//X A B C D E F G H Y

/\*X\*/{0,4,0,6,4,0,0,0,0,0},

/\*A\*/{0,0,4,0,0,0,0,3,0,0},

/\*B\*/{0,0,0,0,0,0,0,0,0,3},

/\*C\*/{0,5,0,0,0,0,5,0,0,0},

/\*D\*/{0,0,0,3,0,3,3,0,0,0},

/\*E\*/{0,0,0,0,0,0,3,0,0,3},

/\*F\*/{0,0,0,0,3,0,0,0,4,0},

/\*G\*/{0,0,3,0,0,0,0,0,3,5},

/\*H\*/{0,0,0,4,0,0,0,0,0,3},

/\*Y\*/{0,0,0,0,0,0,0,0,0,0},

};

int flow[SIZE][SIZE] = {

//X A B C D E F G H Y

/\*X\*/{0,2,0,3,2,0,0,0,0,0},

/\*A\*/{0,0,3,0,0,0,0,2,0,0},

/\*B\*/{0,0,0,0,0,0,0,0,0,3},

/\*C\*/{0,3,0,0,0,0,4,0,0,0},

/\*D\*/{0,0,0,1,0,2,1,0,0,0},

/\*E\*/{0,0,0,0,0,0,1,0,0,1},

/\*F\*/{0,0,0,0,1,0,0,0,4,0},

/\*G\*/{0,0,0,0,0,0,0,0,1,1},

/\*H\*/{0,0,0,3,0,0,0,0,0,2},

/\*Y\*/{0,0,0,0,0,0,0,0,0,0},

};

int color[SIZE];

int path[SIZE];

int queue[SIZE + 2];

int first, last;

void add(int x) {

queue[last] = x;

++last;

color[x] = YELLOW;

}

int delite() {

int x = queue[first];

++first;

color[x] = RED;

return x;

}

int wideSearch(int start, int end) {

first = 0, last = 0;

int i, j;

for (i = 0; i < SIZE; ++i) {

color[i] = GREEN;

}

add(start);

path[start] = -1;

while (first != last) {

i = delite();

for (j = 0; j < SIZE; ++j) {

if (color[j] == GREEN && (bandwidth[i][j] - flow[i][j]) > 0) {

add(j);

path[j] = i;

}

}

}

if (color[end] == RED)

return 1;

else return 0;

}

int min(int x, int y) {

if (x < y)

return x;

else

return y;

}

int maxFlow(int source, int stock) {

int maxflow = 0;

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

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

flow[i][j] = 0;

while (wideSearch(source, stock)) {

int counter = 100000;

for (int i = SIZE - 1; path[i] >= 0; i = path[i]) {

counter = min(counter, (bandwidth[path[i]][i] - flow[path[i]][i]));

}

for (int i = SIZE - 1; path[i] >= 0; i = path[i]) {

flow[path[i]][i] += counter;

flow[i][path[i]] -= counter;

}

maxflow += counter;

}

return maxflow;

}

void show(int start[SIZE][SIZE], int flow[SIZE][SIZE], int bandwidth[SIZE][SIZE]) {

const char\* letter = "XABCDEFGHY";

for (int i = 0; i < SIZE; ++i, cout << endl) {

cout << letter[i] << ":";

for (int j = 0; j < SIZE; ++j) {

if (start[i][j] == bandwidth[i][j]) {

if (flow[i][j] != start[i][j] && flow[i][j] > 0) {

if (flow[i][j] - start[i][j] > 0) {

cout << letter[j] << "(+" << flow[i][j] - start[i][j] << ") ";

}

else if (flow[i][j] - start[i][j] < 0) {

cout << letter[j] << "(" << flow[i][j] - start[i][j] << ") ";

}

continue;

}

else continue;

}

if (flow[i][j] - start[i][j] > 0) {

cout << letter[j] << "(+" << flow[i][j] - start[i][j] << ") ";

}

else if (flow[i][j] - start[i][j] < 0) {

cout << letter[j] << "(" << flow[i][j] - start[i][j] << ") ";

}

}

}

}

int main() {

setlocale(LC\_ALL, "");

cout << "Bершин: " << SIZE << endl;

const char\* letter = "XABCDEFGHY";

int start[SIZE][SIZE];

for (int i = 0; i < SIZE; ++i) {

for (int j = 0; j < SIZE; ++j) {

start[i][j] = flow[i][j];

}

}

cout << " X A B C D E F G H Y\n";

cout << "--------------------------------------------------------------------------\n";

for (int i = 0; i < SIZE; ++i, cout << endl) {

cout << letter[i] << "|";

for (int j = 0; j < SIZE; ++j) {

cout << bandwidth[i][j] << "\t";

}

}

cout << endl << "Максимальный поток: " << maxFlow(0, SIZE - 1) << endl;

show(start, flow, bandwidth);

return 0;

}

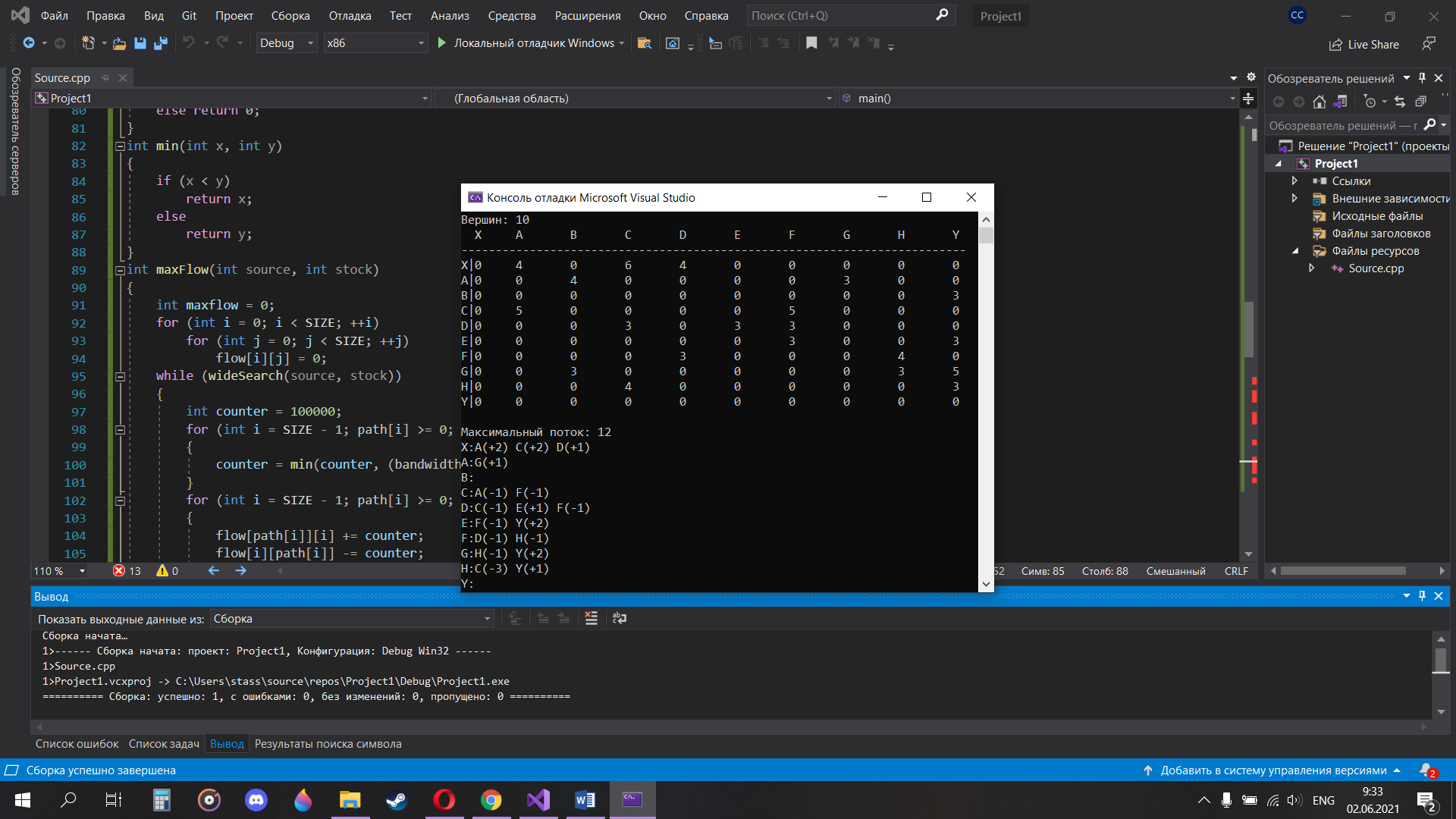


Рисунок 2

**Висновок:** В цій лабораторній роботі, за допомогою мови програмування С++, отримано навички побудови алгоритму Форда-Фалкерсона для мереж. Використовуючи 2 матриці( матрицю максимальної пропускної можливості і матрицю, в якій записана кількість заповненого простору в шляху), я зобразив вагу ребер, після чого перебираючи у циклі всіх можливих шляхів, додав(або віднімав) необхідне число до ребра для їх насичення.