Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра інформаційних управляючих систем

Дисципліна: «Дискретна математика»

**ЛАБОРАТОРНА РОБОТА № 5**

**« РЕАЛІЗАЦІЯ АЛГОРИТМУ ФОРДА-ФАЛКЕРСОНА ДЛЯ МЕРЕЖ »**

|  |  |  |
| --- | --- | --- |
| Виконав:  ст. гр. ІТУ-19-2  Куренков Богдан Михайлович |  | Прийняла:  Васильцова Н.В.  з оцінкою «\_\_»  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2020р. |
|  |  |  |

Харків 2020

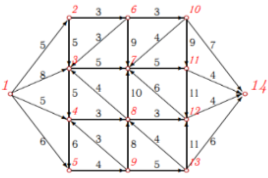
**МЕТА РОБОТИ**

Ознайомлення із загальними мережними моделями дослідження задач обміну інформацією в комп’ютерних мережах, під час планування постачання в транспортних мережах, розподілення матеріальних потоків тощо. Програмна реалізація алгоритму Форда-Фалкерсона, який використовується для знаходження максимальної течії в мережах.

**ПОСТАНОВКА ЗАДАЧІ**

**Розробити програму, яка визначає максимальну течію (мінімальний розріз) заданої транспортної мережі.**

**Варіант 9**



**ХІД РОБОТИ**

#include <iostream>

#include <iomanip>

const int dimension = 14; // число вершин в графе

int f[dimension][dimension]; // f[i][j] - поток, текущий от вершины i к j

int c[dimension][dimension]; // c[i][j] - максимальная величина потока, способная течь по ребру (i,j)

// поиск пути, по которому возможно пустить поток алгоритмом обхода графа в ширину.

// функция ищет путь из истока в сток, по которому еще можно пустить поток,

// считая вместимость ребера (i,j) равной c[i][j] - f[i][j]

int FindPath(int source, int target) // source - исток, target - сток

{

int queue\_pointer = 0; // queue\_pointer - указатель начала очереди

int queue\_count = 1; // queue\_count - число эл-тов в очереди

int queue[dimension] = {}; // очередь для хранения пути

int link[dimension] = {}; // link[i] хранит номер предыдущей вешины на пути i -> исток

int flow[dimension] = {}; // flow - значение потока через данную вершину на данном шаге поиска

queue[0] = source;

link[target] = -1;

int CurVertex;

for (int i = 0; i < dimension; ++i) // обнуляем поток

flow[i] = 0;

flow[source] = INT\_MAX;

// работаем, пока не дойдём до точки-стока, либо не перезаполним очередь (т.е. не дойдём до точки-стока)

while (link[target] == -1 && queue\_pointer < queue\_count)

{

CurVertex = queue[queue\_pointer]; // текущая вершина - последняя, на данный момент, вершина в искомом пути

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

// если у текущей вершины есть дуга с і-той вершиной, для которой поток пуст

if ((c[CurVertex][i] - f[CurVertex][i]) > 0 && flow[i] == 0)

{

queue[queue\_count] = i; ++queue\_count; // новым элементом очереди (пути) записывается очередная дуга

link[i] = CurVertex; // записывает текущую вершину как предыдущую

// записываем наибольшее значение потока для текущей вершины (либо новое, либо прежнее)

if (c[CurVertex][i] - f[CurVertex][i] < flow[CurVertex])

flow[i] = c[CurVertex][i];

else

flow[i] = flow[CurVertex];

}

++queue\_pointer; // смещаем указатель очереди, дабы начать работу со следующим элементом

}

if (link[target] == -1) // если пути до точки-стока нет, возвращаем 0 (код неудачи)

return 0;

CurVertex = target; // текущая вершина - точка-стока

while (CurVertex != source) // работаем, пока не дойдём до точки-истока

{

// для дуги, концом которой является текущая вершина, обновляем значение потока

f[link[CurVertex]][CurVertex] += flow[target];

// после работы с i-той вершиной, переходим к i-1-ой вершине

CurVertex = link[CurVertex];

}

return flow[target]; // возвращаем точки-стока

}

// основная функция поиска максимального потока

int max\_flow(int source, int target) // source - исток, target - сток

{

// если нужно обнулить поток (хранящийся в f), что бы использовать функцию ещё раз

//for (int i = 0; i < dimension; ++i)

// for (int j = 0; j < dimension; ++j)

// f[i][j] = 0;

int max\_flow = 0;

int add\_flow;

do { // сложение потоков всех найденных путей

add\_flow = FindPath(source, target);

max\_flow += add\_flow;

} while (add\_flow > 0);

return max\_flow; // возвращаем сумму потоков (т.е. максимальный поток)

}

int main()

{

setlocale(LC\_ALL, "Russian");

int\*\* C, source, target, dugs;

std::cout << "Введите количество рёбер: ";

std::cin >> dugs;

std::cout << "Введите вершину-исток: ";

std::cin >> source;

std::cout << "Введите вершину-сток: ";

std::cin >> target;

C = new int\* [dimension];

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

C[i] = new int[dimension];

int\* data = new int[dimension \* dimension];

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

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

C[i][j] = 0;

}

}

int a, b, c;

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

std::cout << "Введите начало " << i + 1 << "-ого ребра, его конец и вес: ";

std::cin >> a;

std::cin >> b;

std::cin >> c;

C[a - 1][b - 1] = c;

}

std::cout << std::setw(40) << "Транспортная сеть\n\n\n";

std::cout << std::setw(8);

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

std::cout << i + 1 << std::setw(4);

std::cout << "\n \_ \_ \_ \_ \_ \_ \_ \_ \_ \_\_ \_\_ \_\_ \_\_ \_\_\n\n";

for (int i = 0; i < dimension; ++i, std::cout << "\n\n") {

std::cout << std::setw(2) << i + 1 << "|" << std::setw(5);

for (int j = 0; j < dimension + 1; ++j)

if (c[i][j])

std::cout << c[i][j] << std::setw(4);

else

std::cout << " " << std::setw(4);

std::cout << "\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

}

//------------------------------------------------------------------

// либо через консоль

//------------------------------------------------------------------

//for (int i = 0; i < dimension; ++j)

// for (int j = 0; j < dimension; ++j)

// scanf("%d", &c[i][j]);

//------------------------------------------------------------------

// поиск максимального потока и вывод оного на экран

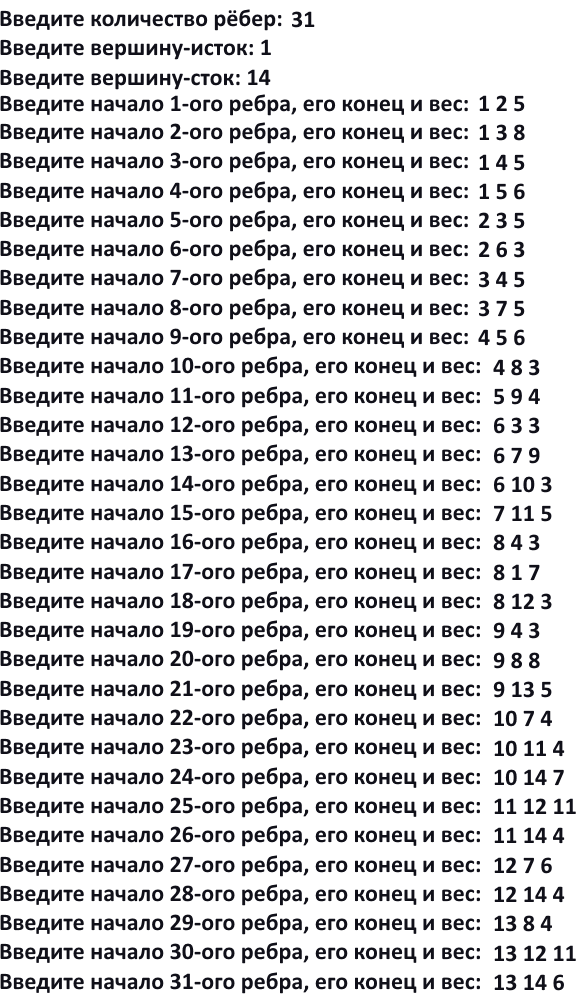
//------------------------------------------------------------------

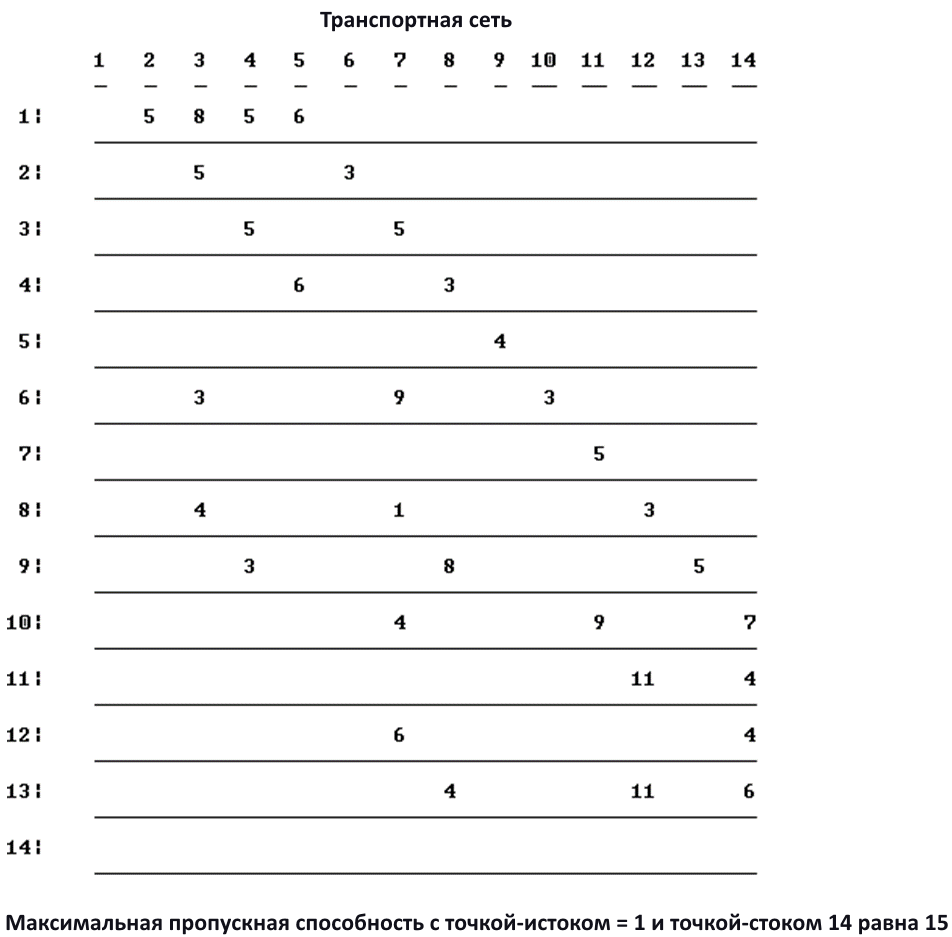
std::cout << "\Максимальная пропускная способность с точкой-истоком = " << source << " и точкой-стоком= " << target << " равна " << max\_flow(source - 1 , target - 1) << "\n\n";

return 0;

}

**ВИКОНАННЯ ЕКСПЕРИМЕНТІВ ПО НАЛАГОДЖЕННЮ ПРОГРАМИ**



****

**ВИСНОВКИ**

Протягом лабораторної була розроблена программа для підрахунку максимальної течії (мінімального розрізу) заданої транспортної мережі за допомогою алгоритму Форда-Фалкерсона для мереж. Програма підраховує та виводить на значення максимальної течії мережі. Граф задається у функції main(), як множина кортежей інциденцій (перша вершина, другая вершина, вага дуги).