МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационные системы

Бариев Эмин Юсуфович

Институт информационных технологий и управления в технических системах

курс 4 группа ИС/б-16-2

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №8

По дисциплине: «ТРСиПВ»

По теме: «Исследование алгоритмов поиска кратчайшего пути на графе»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

ст.преподаватель Дрозин А. Ю.

(должность) (подпись) (инициалы,фамилия)

Севастополь 2019

1 ЦЕЛЬ РАБОТЫ

Программно реализовать и исследовать эффективность алгоритмов поиска кратчайшего пути на графе с использованием функций библиотеки MPI.

2 ПОСТАНОВКА ЗАДАЧИ

Вариант – 1

Выполнить разработку программы, реализующей поиск кратчайшего пути на графе при помощи алгоритма Дейкстры. Применить разработанную процедуру к графу на Рис. 2.1 для поиска кратчайшего пути от вершины 0 к вершине 9.

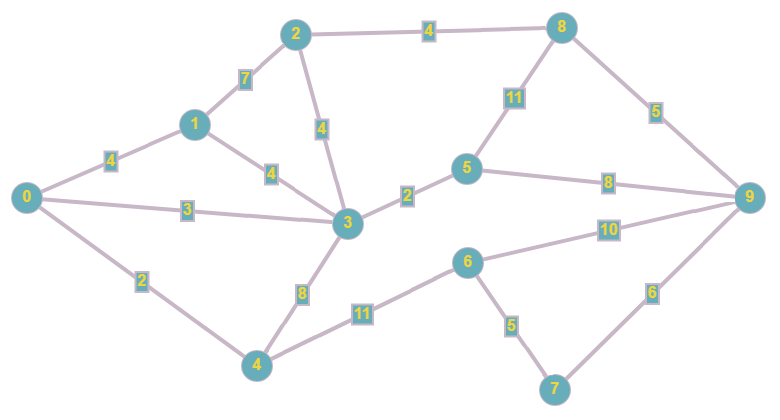


Рисунок 2.1 – Исходный граф для реализации алгоритма Дейкстры

3 ХОД РАБОТЫ

3.1 Код программы на языке С++

#include <mpi.h>

#include <iostream>

**using** **namespace** std;

#define ROOT 0

**const** **int** NodesCount = 10;

**const** **int** INFINITY = 100000;

**static** **int** getMinimumFromArray(**int** \*currentMetricWeight, **int** \*finalElements);

**int** main(**int** argc, **char**\* argv[]){

**int** proccessRank,proccessCount;

MPI\_Status status;

MPI\_Init(&argc, &argv);

MPI\_Comm comm = MPI\_COMM\_WORLD;

MPI\_Comm\_rank(comm, &proccessRank);

MPI\_Comm\_size(comm, &proccessCount);

**int** \*pack = **new** **int** [4];

**int** \*responce = **new** **int** [2];

**if** (proccessRank == ROOT){

**int** arrayA[NodesCount][NodesCount] = {

{0,4,0,3,2,0,0,0,0,0},

{0,0,7,4,0,0,0,0,0,0},

{0,0,0,4,0,0,0,0,4,0},

{0,0,0,0,0,2,0,0,0,0},

{0,0,0,8,0,0,11,0,0,0},

{0,0,0,0,0,0,0,0,11,8},

{0,0,0,0,0,0,0,5,0,10},

{0,0,0,0,0,0,0,0,0,6},

{0,0,0,0,0,0,0,0,0,5},

{0,0,0,0,0,0,0,0,0,0}

};

**int** finalElements[NodesCount] = {-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};

**int** currentMetrics [NodesCount];

currentMetrics[0] = 0;

**for**(**int** i = 1;i<NodesCount;i++)

currentMetrics[i] = INFINITY;

**int** iterationCount = 0;

**int** resourceCounter = 1;

**int** currentElement;

cout << "Iteration"<<proccessRank<<": ";

**for** (**int** i = 0;i < NodesCount;i++){

**if** (currentMetrics[i] == INFINITY){

cout << "-" << " ";

}

**else** cout <<currentMetrics[i] << " ";

}

cout << endl;

**while** (iterationCount < NodesCount){

resourceCounter = 1;

currentElement = getMinimumFromArray(currentMetrics, finalElements);

finalElements[iterationCount] = currentElement;

**for**(**int** i=0;i<NodesCount;i++){

**if** (arrayA[currentElement][i] != 0){

//отправляем 3 величины; [0] - величина от корня до рассматриваемого;

// [1] - величина от корня до имеющего путь;

// [2] - величина от рассматриваемого до имеющего путь

// [3] - индекс элемента до которого имеется путь

pack[0] = currentMetrics[currentElement];

pack[1] = currentMetrics[i];

pack[2] = arrayA[currentElement][i];

pack[3] = i;

MPI\_Send(pack, 4, MPI\_INT, resourceCounter, 99, comm);

resourceCounter++;

}

}

resourceCounter = 1;

**for**(**int** i=0;i<NodesCount;i++){

**if** (arrayA[currentElement][i] != 0){

MPI\_Recv(responce, 2, MPI\_INT, resourceCounter, 99, comm, &status);

currentMetrics[responce[1]] = responce[0];

resourceCounter++;

}

}

cout << "Iteration" << iterationCount+1 << ": ";

**for** (**int** i = 0;i < NodesCount;i++){

**if** (currentMetrics[i] == INFINITY){

cout << "-" << " ";

}

**else** cout <<currentMetrics[i] << " ";

}

cout << endl;

iterationCount++;

}

**for** (**int** i = 1;i<NodesCount;i++){

cout << "From node #0 to #"<< i << " shortest path: " << currentMetrics[i] << endl;

}

pack[0] = -1;

**for** (**int** i = 1;i<proccessCount;i++){

MPI\_Send(pack, 4, MPI\_INT, i, 99, comm);

}

}

**if** (proccessRank > ROOT){

**while**(**true**){

MPI\_Recv(pack, 4, MPI\_INT, ROOT, 99, comm, &status);

**if** (pack[0] != -1){

**if** (pack[0] + pack[2] < pack[1]){

responce[0] = pack[0] + pack[2];

}

**else**

responce[0] = pack[1];

responce[1] = pack[3];

MPI\_Send(responce, 2, MPI\_INT, ROOT, 99, comm);

}

**else** **break**;

}

}

MPI\_Barrier(comm);

**delete** []responce;

**delete** []pack;

MPI\_Finalize();

}

**static** **int** getMinimumFromArray(**int** \*currentMetricWeight, **int** \*finalElements){

**int** min = INFINITY;

**int** minIndex;

**bool** isUnreachable = **true**;

**for**(**int** i = 0;i<NodesCount;i++){

**for**(**int** j = 0;j<NodesCount;j++){

**if** (i == finalElements[j]){

isUnreachable = **false**;

**break**;

}

}

**if** (isUnreachable){

**if** (currentMetricWeight[i] < min){

min = currentMetricWeight[i];

minIndex = i;

}

}

isUnreachable = **true**;

}

**return** minIndex;

}

Результат работы

**Iteration0: 0 - - - - - - - - -**

**Iteration1: 0 4 - 3 2 - - - - -**

**Iteration2: 0 4 - 3 2 - 13 - - -**

**Iteration3: 0 4 - 3 2 5 13 - - -**

**Iteration4: 0 4 11 3 2 5 13 - - -**

**Iteration5: 0 4 11 3 2 5 13 - 16 13**

**Iteration6: 0 4 11 3 2 5 13 - 15 13**

**Iteration7: 0 4 11 3 2 5 13 18 15 13**

**Iteration8: 0 4 11 3 2 5 13 18 15 13**

**Iteration9: 0 4 11 3 2 5 13 18 15 13**

**Iteration10: 0 4 11 3 2 5 13 18 15 13**

**From node #0 to #1 shortest path: 4**

**From node #0 to #2 shortest path: 11**

**From node #0 to #3 shortest path: 3**

**From node #0 to #4 shortest path: 2**

**From node #0 to #5 shortest path: 5**

**From node #0 to #6 shortest path: 13**

**From node #0 to #7 shortest path: 18**

**From node #0 to #8 shortest path: 15**

**From node #0 to #9 shortest path: 13**

**Program ended with exit code: 0**

ВЫВОДЫ

В ходе выполнения лабораторной работы было реализовано и исследована эффективность алгоритмов поиска кратчайшего пути на графе с использованием функций библиотеки MPI.