Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

**Лабораторная работа №6**

**По дисциплине АиСД за 2 семестр**  
**Тема: «Алгоритм Дейкстры»**

**Вариант 4**

**Выполнил:**

Студент группы ПО-6(1)  
 1-го курса

Мартынович Даниил

**Проверила:**

Гирель Т. Н.

**Постановка задачи**

Требуется реализовать алгоритм Дейкстры поиска кратчайшего пути от заданной вершины до всех остальных. Для хранения длин кратчайших путей (массив d[i]) следует использовать бинарную кучу (min-heap). Для хранения графа использовать матрицу смежности.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#define SIZE 6

int main()

{

int a[SIZE][SIZE]; // матрица связей

int d[SIZE]; // минимальное расстояние

int v[SIZE]; // посещенные вершины

int temp, minindex, min;

int begin\_index = 0;

system("chcp 1251");

system("cls");

// Инициализация матрицы связей

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

{

a[i][i] = 0;

for (int j = i + 1; j < SIZE; j++) {

printf("Введите расстояние %d - %d: ", i + 1, j + 1);

scanf("%d", &temp);

a[i][j] = temp;

a[j][i] = temp;

}

}

// Вывод матрицы связей

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

{

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

printf("%5d ", a[i][j]);

printf("\n");

}

//Инициализация вершин и расстояний

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

{

d[i] = 10000;

v[i] = 1;

}

d[begin\_index] = 0;

// Шаг алгоритма

do {

minindex = 10000;

min = 10000;

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

{ // Если вершину ещё не обошли и вес меньше min

if ((v[i] == 1) && (d[i] < min))

{ // Переприсваиваем значения

min = d[i];

minindex = i;

}

}

// Добавляем найденный минимальный вес

// к текущему весу вершины

// и сравниваем с текущим минимальным весом вершины

if (minindex != 10000)

{

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

{

if (a[minindex][i] > 0)

{

temp = min + a[minindex][i];

if (temp < d[i])

{

d[i] = temp;

}

}

}

v[minindex] = 0;

}

} while (minindex < 10000);

// Вывод кратчайших расстояний до вершин

printf("\nКратчайшие расстояния до вершин: \n");

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

printf("%5d ", d[i]);

// Восстановление пути

int ver[SIZE]; // массив посещенных вершин

int end = 4; // индекс конечной вершины = 5 - 1

ver[0] = end + 1; // начальный элемент - конечная вершина

int k = 1; // индекс предыдущей вершины

int weight = d[end]; // вес конечной вершины

while (end != begin\_index) // пока не дошли до начальной вершины

{

for (int i = 0; i < SIZE; i++) // просматриваем все вершины

if (a[i][end] != 0) // если связь есть

{

int temp = weight - a[i][end]; // определяем вес пути из предыдущей вершины

if (temp == d[i]) // если вес совпал с рассчитанным

{ // значит из этой вершины и был переход

weight = temp; // сохраняем новый вес

end = i; // сохраняем предыдущую вершину

ver[k] = i + 1; // и записываем ее в массив

k++;

}

}

}

// Вывод пути (начальная вершина оказалась в конце массива из k элементов)

printf("\nВывод кратчайшего пути\n");

for (int i = k - 1; i >= 0; i--)

printf("%3d ", ver[i]);

getchar(); getchar();

return 0;

}



