#include "stdafx.h"

#include iostream

#include <conio.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

int SIZE;

cout « "Количество вершин: ";

cin » SIZE;

int \*a = new int[SIZE];// матрица связей

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

a[i] = new int[SIZE];

int \*d = new int[SIZE]; // минимальное расстояние

int \*v = new int[SIZE]; // посещенные вершины

int temp;

int minindex, min;

for (int i = 0; i < SIZE; i++) a[i][i] = 0;

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

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

cout « "Расстояние " « i + 1 « " - " « j + 1 « ": ";

cin » a[i][j];

a[j][i] = a[i][j];

}

// Вывод матрицы связей

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

{

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

cout « a[i][j] « "\t";

cout « endl;

}

//Инициализация вершин и расстояний

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

{

d[i] = 10000;

v[i] = 1;

}

d[0] = 0;

// Шаг алгоритма

// 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

cout « "\nКратчайшие расстояния до вершин: \n";

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

cout « d[i] « "\t";

// Восстановление пути

int \*ver = new int[SIZE]; // массив посещенных вершин

int end = SIZE - 1; // индекс конечной вершины = SIZE- 1

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

int k = 1; // индекс предыдущей вершины

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

while (end > 0) // пока не дошли до начальной вершины

{

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

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

{

int temp = weight - a[end][i]; // 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)

cout « "\nВывод кратчайшего пути из первой в последнюю\n";

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

cout « ver[i] « "\t";

\_getch();

return 0;

}