Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

Уфимский государственный авиационный технический университет

Кафедра вычислительной математики и кибернетики

Расчетно-графическая работа

по дисциплине «Структуры и алгоритмы компьютерной обработки данных»

Вариант №18

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ф.И.О. | Дата | Подпись |
| Выполнила студентка группы МО-213 : | Сиротенко Е.И. |  |  |
| Проверила  к.т.н., доц. каф. ВМиК: | Верхотурова Г.Н. |  |  |

**Уфа 2018 г.**

**Задание:**

Задана система двусторонних дорог. N-периферией называется множество городов, расстояние от которых до выделенного города(столица) больше N. Определить N-периферию для заданного N.

Разработать алгоритм решения этой задачи и написать программу.

**Алгоритм нахождения N-периферии:**

Данную задачу можно решить с помощью алгоритма Дейкстры. Алгоритм позволяет найти расстояния от заданной вершины до любой другой. Зная расстояния до вершин, сравним их с заданным N. Те расстояния, которые больше N, и будут являться N-периферией.

Опишем алгоритм Дейкстры:

Алгоритм использует три массива из n = (числу вершин графа) чисел каждый.

Первый массив **a** содержит метки с двумя значениями: 0 (вершина еще не рассмотрена) и 1 (вершина уже рассмотрена);

второй массив **b** содержит расстояния - текущие кратчайшие расстояния от S до соответствующей вершины;

Матрица расстояний Dik задает длины дуг dik; если такой дуги нет, то dik присваивается большое число Б, равное "машинной бесконечности".

1 (инициализация).

В цикле от 1 до n заполнить нулями массив a;

перенести i-ю строку матрицы D в массив b;

a[i]: = 1; {i - номер стартовой вершины}

2 (общий шаг).

Найти минимум среди неотмеченных вершин, т.е. тех k, для которых a[k] = 0;

пусть минимум достигается на индексе j, т.е. bj ≤ bk.

Затем выполняются следующие операции:

a[j]: = 1;

если bk > bj + djk, то bk: = bj + djk;

Это условие означает, что путь vi ... vk длиннее, чем путь vi ... vjvk.

Если все вершины массива a[] отмечены, то все расстояния от заданной вершины до любой другой – найдены.

**Входные и выходные данные:**

textBox1.Text – Задаем вершину

textBox2.Text – Задаем N

matrix – Матрица весов

bool[] a = new bool[V] – Массив, пройденных вершин

a[i]=false – еще не найден кратчайший путь в i-ю вершину,

a[i]=true – кратчайший путь в i-ю вершину уже найден

int[] b = new int[V] – Массив, содержащий длины кратчайших путей

richTextBox1.Text – N-периферия

**Код программы:**

private void button2\_Click(object sender, EventArgs e)

{

richTextBox1.Text = string.Empty;

int start;

int Npf;

int[] b = new int[dataGridView1.RowCount - 1];

if (int.TryParse(textBox1.Text, out start) && (start > 0) && (start < dataGridView1.RowCount)&& int.TryParse(textBox2.Text, out Npf))

{

int N = dataGridView1.RowCount - 1; кол-во вершин

int[,] matrix = new int[N, N];

for (int i = 0; i < N; i++) Заполнение матрицы весов

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

{

if (this.dataGridView1.Rows[i + 1].Cells[j + 1].Value == null)

matrix[i, j] = 1000000;

else

matrix[i, j] = Convert.ToInt32(dataGridView1.Rows[i + 1].Cells[j + 1].Value);

}

b= Deikstra(matrix, start - 1, N); Ищем кратчайшие пути

for (int i = 0; i < b.Length; i++)

{

if (b[i] > Npf && (b[i]!= 1000000))

{

richTextBox1.Text += (i + 1).ToString()+" "; Вывод N-периферии

}

}

}

}

int[] Deikstra(int[,] matrix, int st, int N)

{

bool[] a = new bool[N];

int[] b = new int[N];

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

{

a[i] = false;

b[i] = matrix[st, i];

}

a[st] = true; Начальную вершину помечаем как пройденную

int j = st;

while (!mas(a))

{

int min = 1000000;

for (int i = 0; i < N; i++) Ищем минимальный вес из непомеченных вершин

{

if ((!a[i]) && ((b[i] < min) || (b[i] == min)))

{

min = b[i];

j = i;

}

}

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

{

if (!a[i] && (b[i] > min + matrix[i, j])) Добавляем найденный минимальный вес к текущему весу вершины и сравниваем с текущим минимальным весом вершины

{

b[i] = min + matrix[i, j];

}

}

a[j] = true;

}

return b; Возвращаем массив кратчайших путей от заданной вершины

}

bool mas(bool[] a) Метод определяет все ли вершины графа пройдены

{

for (int i = 0; i < a.Length; i++)

if (!a[i])

return false;

return true;

}

**Оценка сложности:** О(N^2)

**СПИСОК ЛИТЕРАТУРЫ**

1. Вирт Н. Алгоритмы и структуры данных.1989.
2. Б.С. Хусаинов. Структуры и алгоритмы обработки данных. Москва, «Финансы и статистика», 2004
3. Тюкачев Н. А., В. Г. Хлебостарев. C#. Алгоритмы и структуры данных. 2017.