Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

Дисциплина «Математическая логика и дискретная математика»

Лабораторная работа № 4

«**Нахождение путей в графе**»

Выполнил студент

группы ИВТАПбд-12

Матюнин И.С.

Ульяновск, 2022

Оглавление

[Цель Работы 3](#_Toc103946218)

[Ввод данных 4](#_Toc103946219)

[Вывод результата 7](#_Toc103946220)

[Вывод 7](#_Toc103946221)

[Код: 8](#_Toc103946222)

[Список использованной Литературы 10](#_Toc103946223)

# **Цель Работы**

Написать программу, которая будет искать все кротчайшие пути в неориентированном крафе.

**Описание метода работы**

Прежде всего, необходимо узнать, какими графы могут быть:

* **Конечный граф** – граф с конечным количеством рёбер и вершин.
* **Бесконечный граф** – граф, конец которого в определённом направлении(ях) простирается до бесконечности.
* **Неориентированный граф** – граф, рёбра которого не имеют определённого направления.
* **Ориентированный граф** – граф, рёбра которого имеют определённое направление.
* **Связный граф** – граф, в котором отсутствуют недостижимые вершины (вершины, не связанные с остальными).
* **Несвязный граф** – граф, в котором существуют недостижимые вершины.

Графы могут быть ориентированными и неориентированными.

По заданию я работаю с неориентированным графом. Как в Лабораторной работе №2 я буду строить таблицу с названиями вершин и на их пересечении выводится длина ребра между ними. После таблица будет пересчитываться и на пересечении вершин будут помещены кратчайшие пути между вершинами.

## **Ввод данных**

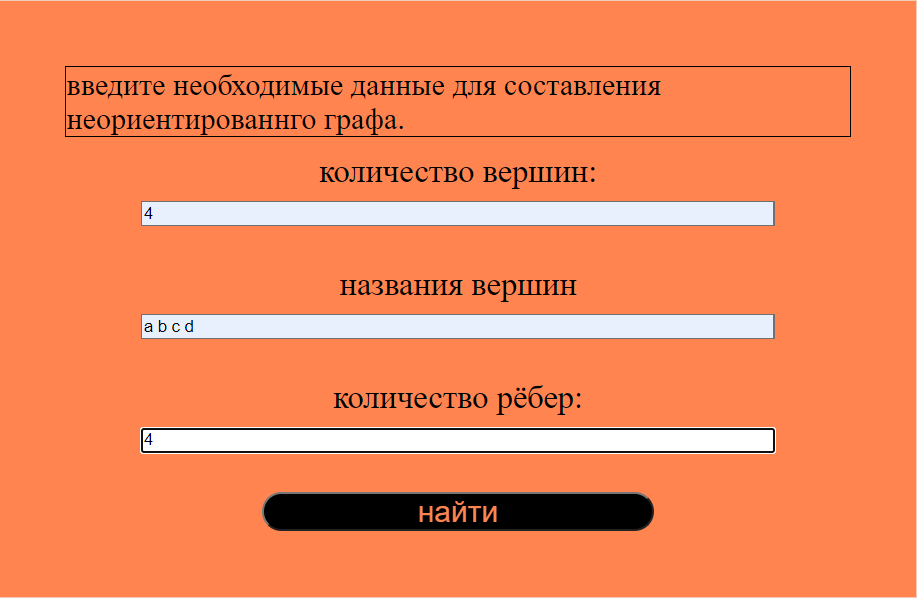


Рис. 1 – форма ввода данных.

Сначала пользователь вводит количество вершин, их названия и количество рёбер. Эти данные необходимы для дальнейшей работы.

Для примера будем работать с таким графом:

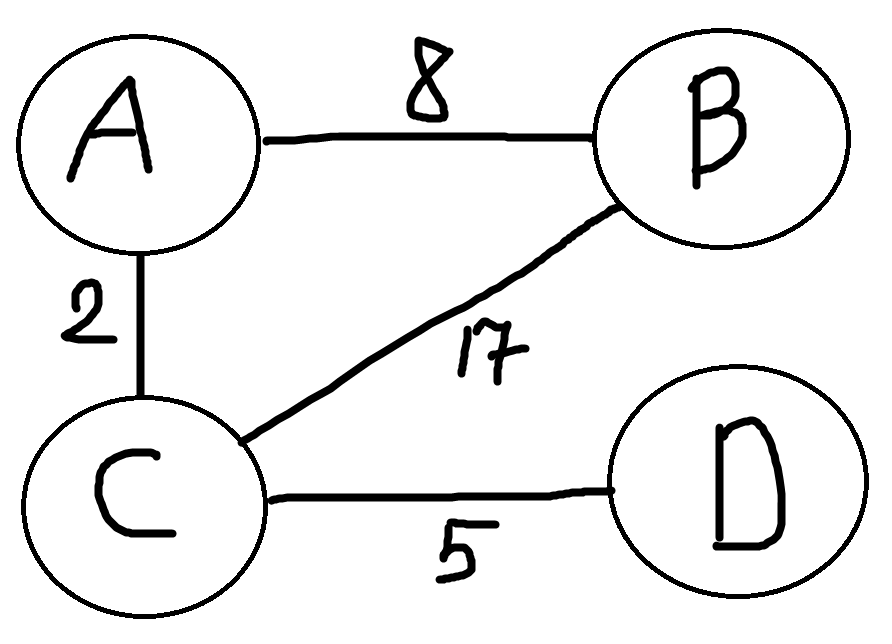
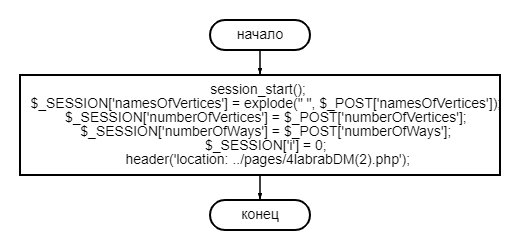


Рис. 2 – пример неориентированного графа

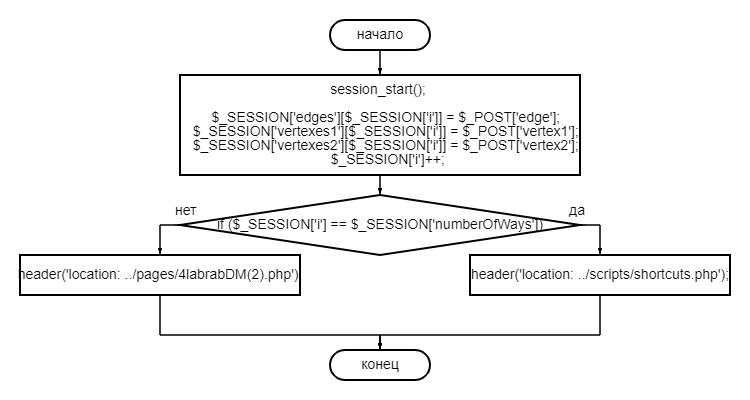
После ввода названия вершин, их количества и количества рёбер последовательно пользователь вводит существующее ребро и его длину.



Рис. 3 – ввод рёбер графа

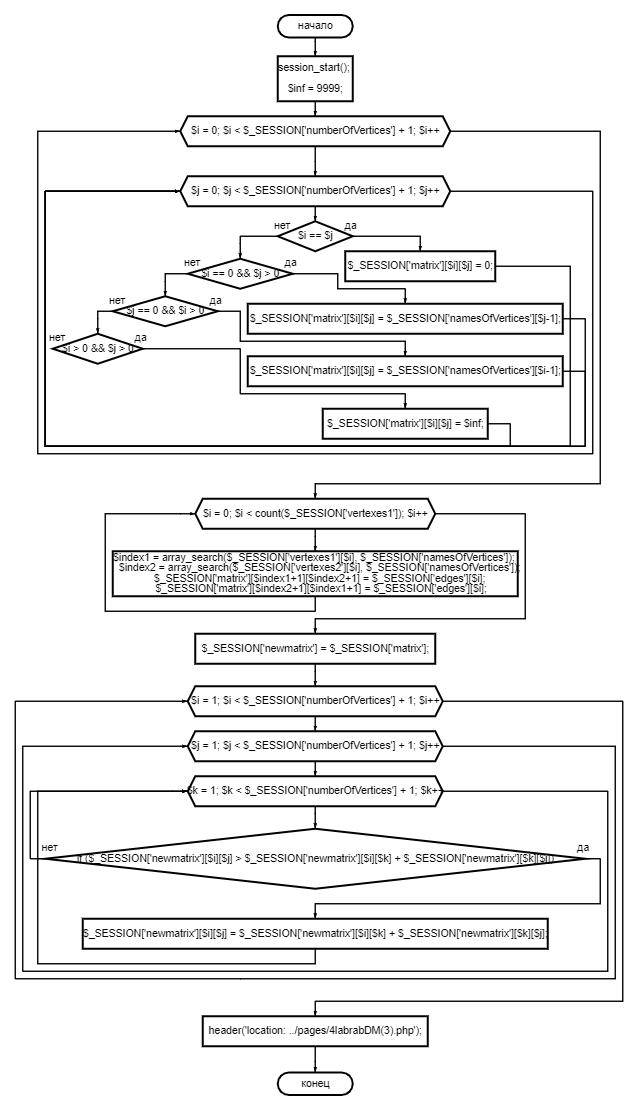


Блок-схема 1 – сохранение основы графа



Блок-схема 2 – сохранение рёбер графа.

После ввода всех рёбер программа строит матрицу доступных путей, а после ищет минимальны пути от одной вершины к каждой другой. Так как граф неориентированный матрица будет симметричная и нерефлективная.



Блок-схема 3 – построение матрицы кротчайших путей.

## **Вывод результата**



Рис. 3 – результат

## **Вывод**

Я написал программу, работающую с неориентированным графом и определяющую кротчайшие пути в этом графе. Так же была изучена теория о связанная с графами. Так же были изучены возможности PHP.

# **Код:**

**saveVetices.php**

<?php

    session\_start();

    $\_SESSION['namesOfVertices'] = explode(" ", $\_POST['namesOfVertices']);

    $\_SESSION['numberOfVertices'] = $\_POST['numberOfVertices'];

    $\_SESSION['numberOfWays'] = $\_POST['numberOfWays'];

    $\_SESSION['i'] = 0;

    header('location: ../pages/4labrabDM(2).php');

**saveWays.php**

<?php

    session\_start();

    $\_SESSION['edges'][$\_SESSION['i']] = $\_POST['edge'];

    $\_SESSION['vertexes1'][$\_SESSION['i']] = $\_POST['vertex1'];

    $\_SESSION['vertexes2'][$\_SESSION['i']] = $\_POST['vertex2'];

    $\_SESSION['i']++;

    if ($\_SESSION['i'] == $\_SESSION['numberOfWays']) {

        header('location: ../scripts/shortcuts.php');

    } else {

        header('location: ../pages/4labrabDM(2).php');

    }

?>

**shortcuts.php**

<?php

session\_start();

$inf = 9999;

for ($i = 0; $i < $\_SESSION['numberOfVertices'] + 1; $i++) {

    for ($j = 0; $j < $\_SESSION['numberOfVertices'] + 1; $j++) {

        if ($i == $j) {

            $\_SESSION['matrix'][$i][$j] = 0;

        } else {

            if ($i == 0 && $j > 0) {

                $\_SESSION['matrix'][$i][$j] = $\_SESSION['namesOfVertices'][$j-1];

            }

            if ($j == 0 && $i > 0) {

                $\_SESSION['matrix'][$i][$j] = $\_SESSION['namesOfVertices'][$i-1];

            }

            if ($i > 0 && $j > 0) {

                $\_SESSION['matrix'][$i][$j] = $inf;

            }

        }

    }

}

for ($i = 0; $i < count($\_SESSION['vertexes1']); $i++) {

    $index1 = array\_search($\_SESSION['vertexes1'][$i], $\_SESSION['namesOfVertices']);

    $index2 = array\_search($\_SESSION['vertexes2'][$i], $\_SESSION['namesOfVertices']);

    $\_SESSION['matrix'][$index1+1][$index2+1] = $\_SESSION['edges'][$i];

    $\_SESSION['matrix'][$index2+1][$index1+1] = $\_SESSION['edges'][$i];

}

$\_SESSION['newmatrix'] = $\_SESSION['matrix'];

for ($i = 1; $i < $\_SESSION['numberOfVertices'] + 1; $i++) {

    for ($j = 1; $j < $\_SESSION['numberOfVertices'] + 1; $j++) {

        for ($k = 1; $k < $\_SESSION['numberOfVertices'] + 1; $k++) {

            if ($\_SESSION['newmatrix'][$i][$j] > $\_SESSION['newmatrix'][$i][$k] + $\_SESSION['newmatrix'][$k][$j]) {

                $\_SESSION['newmatrix'][$i][$j] = $\_SESSION['newmatrix'][$i][$k] + $\_SESSION['newmatrix'][$k][$j];

            }

        }

    }

}

header('location: ../pages/4labrabDM(3).php');

# **Список использованной Литературы**

* proglib «Иллюстративное введение в теорию графов и её применение» URL: https://proglib.io/p/graph-theory
* php documentation URL: https://www.php.net/docs.php
* хабр «Базовые алгоритмы нахождения кратчайших путей во взвешенных графах» URL: https://habr.com/ru/post/119158/