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](#_Toc104034128)

[**Ввод данных** 4](#_Toc104034129)

[**Вывод результата** 6](#_Toc104034130)

[**Вывод** 6](#_Toc104034131)

[**Код:** 7](#_Toc104034132)

[**Список использованной Литературы** 10](#_Toc104034133)

# **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**

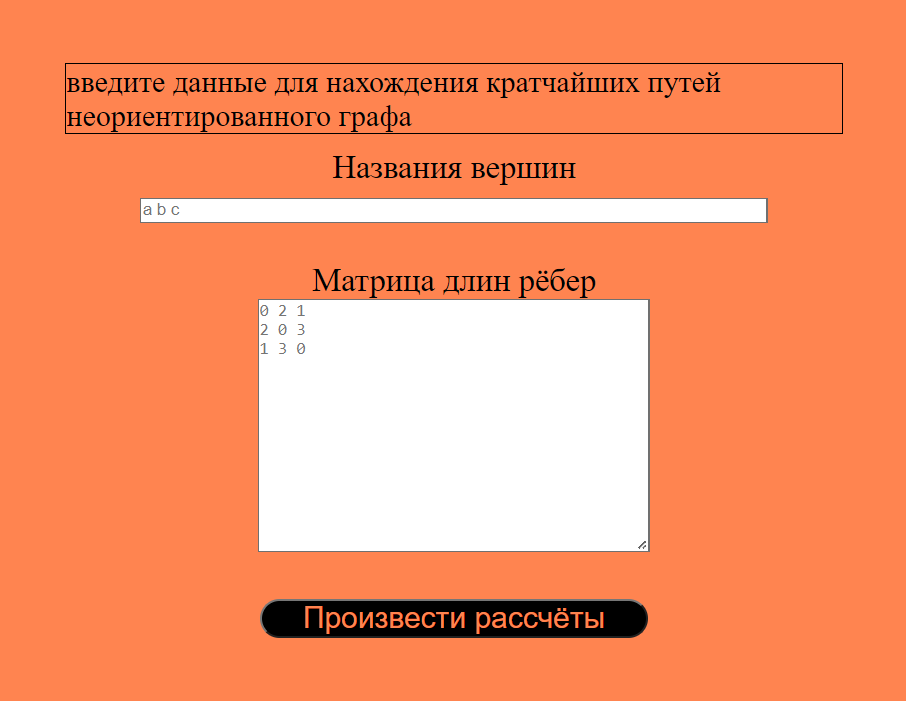
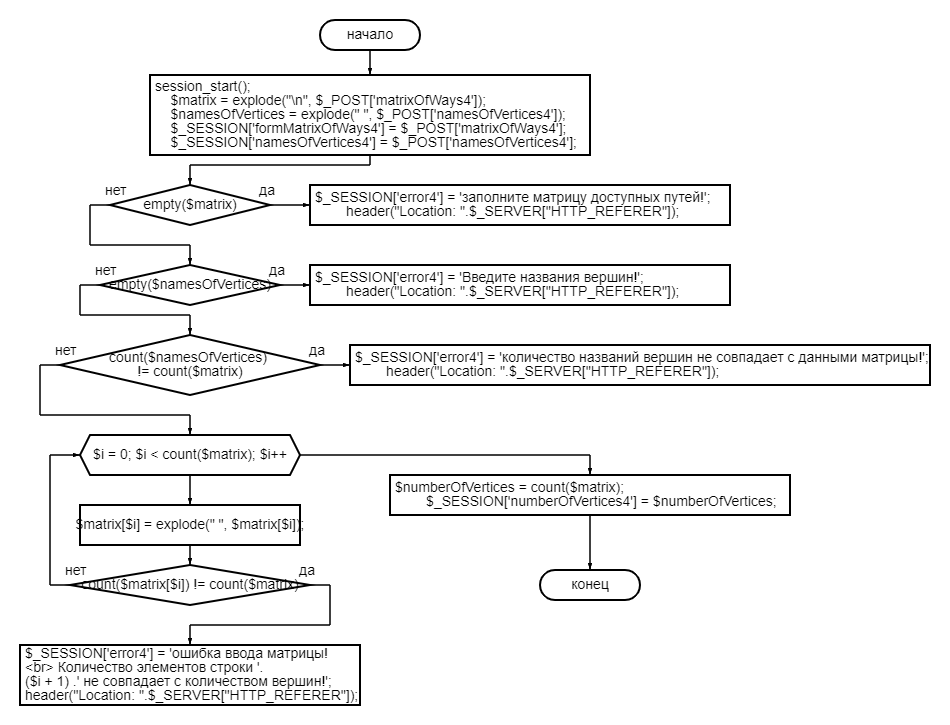
****

Рис. 1 – форма ввода данных

Сначала пользователь вводит имена вершин и матрицу длин доступных путей. Важно, что матрица должна быть нерефлексивной и симметричной, так как граф неориентированный.

Первым делом валидация введённых данных.



Болк-схема 1 – проверка ввода данных

Для примера будем работать с таким графом:

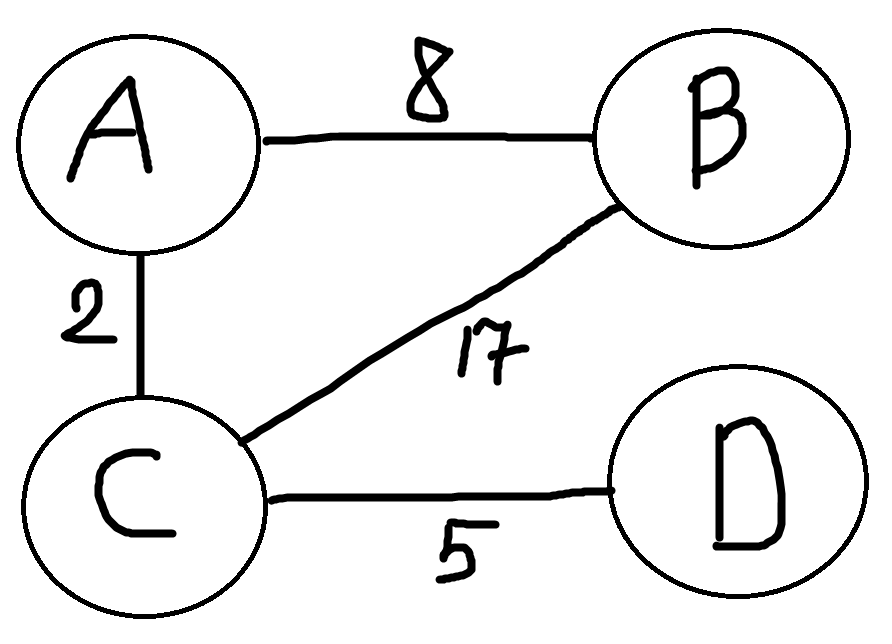
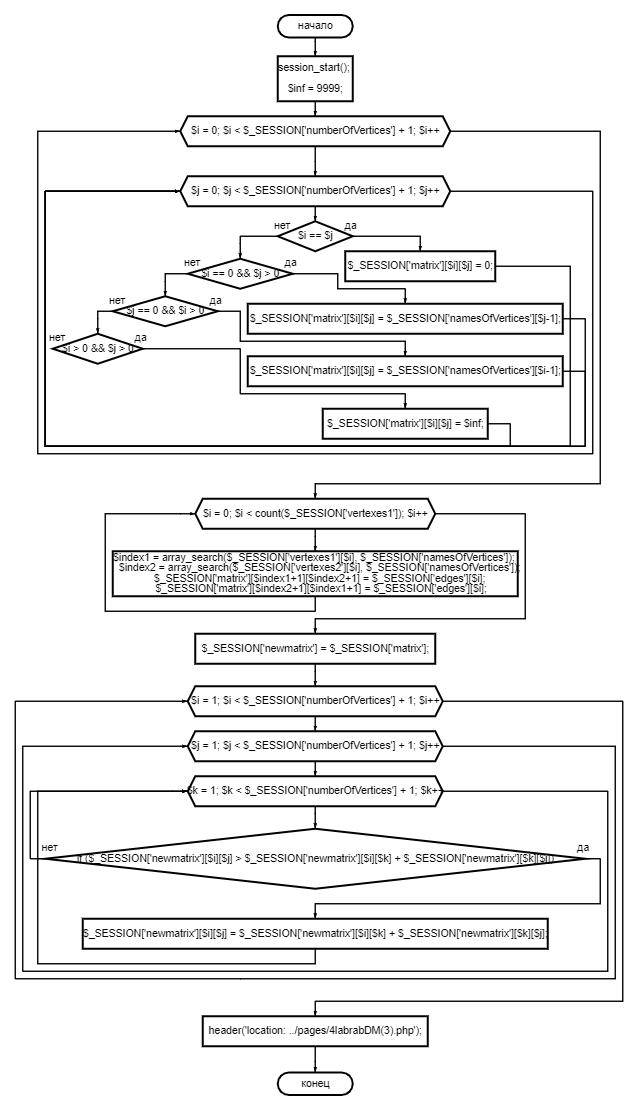


Рис. 2 – пример неориентированного графа

После ввода названия вершин и длин рёбер программа рассчитывает кратчайшие пути.



Блок-схема 2 – 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:**

<?php

    session\_start();

    $matrix = explode("\n", $\_POST['matrixOfWays4']);

    $namesOfVertices = explode(" ", $\_POST['namesOfVertices4']);

    $\_SESSION['formMatrixOfWays4'] = $\_POST['matrixOfWays4'];

    $\_SESSION['namesOfVertices4'] = $\_POST['namesOfVertices4'];

    if (empty($matrix)) {

        $\_SESSION['error4'] = 'заполните матрицу доступных путей!';

        header("Location: ".$\_SERVER["HTTP\_REFERER"]);

    }

    if (empty($namesOfVertices)) {

        $\_SESSION['error4'] = 'Введите названия вершин!';

        header("Location: ".$\_SERVER["HTTP\_REFERER"]);

    }

    if (count($namesOfVertices) != count($matrix)) {

        $\_SESSION['error4'] = 'количество названий вершин не совпадает с данными матрицы!';

        header("Location: ".$\_SERVER["HTTP\_REFERER"]);

    } else {

        for ($i = 0; $i < count($matrix); $i++) {

            $matrix[$i] = explode(" ", $matrix[$i]);

            if (count($matrix[$i]) != count($matrix)) {

                $\_SESSION['error4'] = 'ошибка ввода матрицы!

                <br> Количество элементов строки '.

                ($i + 1) .' не совпадает с количеством вершин!';

                header("Location: ".$\_SERVER["HTTP\_REFERER"]);

            }

        }

        $numberOfVertices = count($matrix);

        $\_SESSION['numberOfVertices4'] = $numberOfVertices;

    }

    $Matrix = explode("\n", $\_POST['matrixOfWays4']);

    for ($i = 0; $i < count($Matrix); $i++) {

        $Matrix[$i] = explode(" ", $Matrix[$i]);

        for ($j = 0; $j < count($Matrix[$i]); $j++) {

            print\_r($Matrix[$i][$j]);

        }

        echo "<br>";

    }

    echo "<br>";

    $len = count($Matrix);

    for ($i = 0; $i < $len; $i++) {

        for ($j = 0; $j < $len; $j++) {

            if ((int) $Matrix[$i][$j] != (int) $Matrix[$j][$i] || ($i == $j && (int) $Matrix[$i][$j] != 0)) {

                $\_SESSION['error4'] = 'Матрица должна быть нерефлексивной и симмтричной, так как граф неориентированный!';

                header("Location: ".$\_SERVER["HTTP\_REFERER"]);

            }

        }

    }

    echo "<br>";

    $inf = 9999;

    for ($i = 0; $i < $len; $i++) {

        for ($j = 0; $j < $len; $j++) {

            if ($i != $j && $Matrix[$i][$j] == 0) {

                $Matrix[$i][$j] = $inf;

            }

        }

    }

    $\_SESSION['matrix4'][0][0] = 0;

    for ($i = 0; $i <= $numberOfVertices + 1; $i++) {

        for ($j = 0; $j <= $numberOfVertices + 1; $j++) {

            if ($i == 0 && $j > 0) {

                $\_SESSION['matrix4'][$i][$j] = $namesOfVertices[$j - 1];

            }

            if ($i > 0 && $j == 0) {

                $\_SESSION['matrix4'][$i][$j] = $namesOfVertices[$i - 1];

            }

            if ($i > 0 && $j > 0) {

                $\_SESSION['matrix4'][$i][$j] = $Matrix[$i-1][$j-1];

                if ($\_SESSION['matrix4'][$i][$j] == $inf) {

                    $\_SESSION['matrix4'][$i][$j] = 'n';

                }

            }

            print\_r($\_SESSION['matrix4'][$i][$j]);

            echo " ";

        }

        echo "<br>";

    }

    echo "<br><br>";

    $newmatrix = $Matrix;

    for ($i = 0; $i < $len; $i++) {

        for ($j = 0; $j < $len; $j++) {

            for ($k = 0; $k < $len; $k++) {

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

                    $newmatrix[$i][$j] = $newmatrix[$i][$k] + $newmatrix[$k][$j];

                }

            }

        }

    }

    for ($i = 0; $i < $len; $i++) {

        for ($j = 0; $j < $len; $j++) {

            print\_r($newmatrix[$i][$j]);

            echo " ";

        }

        echo "<br>";

    }

    for ($i = 0; $i <= $len; $i++) {

        for ($j = 0; $j <= $len; $j++) {

            if ($i == 0 && $j > 0) {

                $\_SESSION['newMatrix4'][$i][$j] = $namesOfVertices[$j - 1];

            }

            if ($i > 0 && $j == 0) {

                $\_SESSION['newMatrix4'][$i][$j] = $namesOfVertices[$i - 1];

            }

            if ($i > 0 && $j > 0) {

                $\_SESSION['newMatrix4'][$i][$j] = $newmatrix[$i-1][$j-1];

                if ($\_SESSION['newMatrix4'][$i][$j] == $inf) {

                    $\_SESSION['newMatrix4'][$i][$j] = 'n';

                }

            }

        }

    }

    unset($\_SESSION['formMatrixOfWays4']);

    unset($\_SESSION['namesOfVertices4']);

    header('location: ../pages/4labrabDM.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/