Министерство образования и науки Российской Федерации

ФГБОУ ВО Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Лабораторная работа №4

«Графы. Определение кратчайшего пути»

Выполнил студент

группы ИВТАСбд-11

Абросимов К. С.

Ульяновск, 2022

Содержание

[1. Цель работы 3](#_Toc103887715)

[2. Описание метода выполнения 3](#_Toc103887716)

[2.1 Теория 3](#_Toc103887717)

[2.2 Файл «index» 4](#_Toc103887718)

[2.3 Файл «mldmLab4.php» 5](#_Toc103887719)

[2.4 Файл «scriptslab4.php» 7](#_Toc103887720)

[3. Выводы 15](#_Toc103887721)

[4. Список литературы 15](#_Toc103887722)

1. **Цель работы**

Необходимо написать программу, которая будет находить кратчайший путь в орграфе. Пользователь вводит матрицу смежности графа n\*n, начальную и конечную точки. При компиляции программа должна проверить корректность ввода данных пользователя.

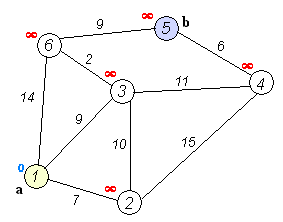
1. **Описание метода выполнения**

## **2.1 Теория**

Граф - это множество упорядоченных пар G := (V, E), где V – множество вершин графа, а E – множество ребер графа. Оба множества непустые. Графы бывают ориентированные и неориентированные. Их отличие заключается в том, что в первом случае проход по ребру ориентированного графа (дуга) возможен только в одном направлении, а во втором – в обоих. Матрица смежности – квадратная матрица представления отношения вершин и ребер, где значение элемента говорит о весе ребра из i точки в j. Вес ребра — значение, поставленное в соответствие данному ребру. Кратчайший путь в графе – последовательность вершин, соединенных ребрами или дугами, у которых сумма весов этих ребер или дуг наименьшая.

Основная работа такова. Алгоритм начинает свою работу с конечной точки, продвигаясь к начальной. Для того, чтобы определить кратчайшее расстояние от А до В, к примеру, нужно перебрать все узлы, которые входят в B, взять кратчайшие расстояние до них от точки A и сложить с длиной ребра, соединяющего данную точку с точкой B. Из полученных расстояний выбрать минимальное.

Однако, кратчайшее расстояние до точек, которые соединяются с узлом B, неизвестно, но его можно получить с помощью использования этой же функции, которая определяет кратчайшее расстояние до точки B. В итоге получается вызов функции самой себя или рекурсия.



Код проводимых операций написан на PHP. PHP - это язык программирования, специально разработанный для написания web-приложений (сценариев), исполняющихся на Web-сервере. PHP взаимодействует с html.

## **2.2 Файл «index»**

Первый файл «index», является html страницей, которая имеет ссылки (переходы) на другие страницы – лабораторные работы.

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Лабораторные работы</title> - основной заголовок  
 <script type="text/javascript" src="/scripts/scripts.js"></script> - ссылка на папку со скриптами  
</head>  
<body>  
 <h1> Мои лабораторные работы </h1>   
 <a href ="mldmLab1.php">Первая лабораторная работа по МЛиДМ</a><br>  
 <a href ="mldmLab2.php">Вторая лабораторная работа по МЛиДМ</a><br> - ссылки к другим лабораторным работам

<a href ="mldmLab3.php">Третья лабораторная работа по МЛиДМ</a><br>  
<a href ="mldmLab4.php">Четвёртая лабораторная работа по МЛиДМ</a><br>  
</body>  
</html>

## **2.3 Файл «mldmLab4.php»**

Для того, чтобы пользователь мог вводить данные необходимо написать небольшую html-структуру, состоящую из двух полей для ввода (input) начальной и конечной вершины и поля для ввода матрицы смежности (textarea). Также кнопки, к которой привязан код поиска кратчайшего пути. Если раньше можно было легко передать переменные, на файл javascript, то теперь, чтобы передать файлы на php, нужно воспользоваться ajax запросом.

Как только пользователь введёт матрицу смежности графа (Пример ввода также помещён в поле для ввода), начальную и конечную точки и нажмёт кнопку «Сделать расчёт», то программа выполнит функцию «sendmass». Эта функция возьмёт все введённые значения и, с помощью функции ajax запросов, передаст их в scriptslab4.php. Причём теперь не придётся каждый раз обновлять страницу, с помощью этого запроса всё будет происходить автоматически. Также, чтобы реализовать ajax запрос, легче всего подключить библиотеку JQuery. Это словно передача сперва в файл js, а затем в php.

<html>  
<head>  
 <title>Лабораторная работа 4</title>  
 <script type="text/javascript" src="/scripts/scriptslab4.php" ></script>  
 <script  
 src=<https://code.jquery.com/jquery-3.6.0.js> – включаем библиотеку, чтобы воспользоваться ajax (jquery)  
 integrity="sha256-H+K7U5CnXl1h5ywQfKtSj8PCmoN9aaq30gDh27Xc0jk="  
 crossorigin="anonymous"></script>  
</head>  
  
<h1> Лабораторная работа №4 </h1>  
  
<body>  
<form>  
 <table>  
<tr>  
 <td> Введите матрицу смежности графа (n\*n) </td>  
 <td> <textarea id="mass" value="" rows = "10" cols = "20" size=""  
 placeholder="Пример ввода:   
\* 7 \*  
\* \* 3  
1 \* \*"></textarea></td>  
 </tr>

<tr>  
 <td>Начальная точка <input type="text" id="begin" value="" size="1"/>  
 Конечная точка <input type="text" id="end" value="" size="1"/></td>  
 </tr>  
 <tr>  
 <td colspan="2"> <input type="button" value="Сделать расчёт" onclick="sendmass();"/></td> - при нажатии кнопки переходим в функцию sendmass  
 </tr>  
 </table>  
  
</form>  
<script>  
 function sendmass() { - функция, берёт введённые значения, выполняет ajax запрос и переводит в функцию php файла  
 let text = ***document***.querySelector('textarea').value;  
 let starttext = ***document***.getElementById('begin').value;  
 let endtext = ***document***.getElementById('end').value; - «извлекаем» значения, введённые пользователем  
 $.ajax({  
 type: "POST",  
 url: '/scripts/scriptslab4.php', - путь к скрипту  
 data: ({  
 'mass': text, слева – переносные значения, справа значения в этой функции  
 'begin': starttext,  
 'end': endtext  
 }),  
 success: function(data){ - при успешном выполнении запроса возвращаем то, что вообще либо выводит php файл, в который мы значения и передали и сразу же выводим на страницу  
 ***document***.getElementById("test").innerHTML = data;  
 }  
 })  
 }  
 </script>  
<div id ="test"></div>  
</body>  
</html>

## **2.4 Файл «scriptslab4.php»**

Первоначально объявляем некоторые переменные. Затем создаём матрицу, пользуясь сперва функцией preg\_split для деления значения, а затем explode для деления массива ещё по мелким элементам. Затем проверка корректности ввода (функция validate – позже), при которой либо выведется текст с ошибкой (не окно), либо полученный результат, а именно кратчайший путь и его траектория.

$error\_text = "";  
$message = $\_POST['mass'];  
$start = $\_POST['begin']; - берём значения с ajax запроса  
$end = $\_POST['end'];  
  
$matrix = preg\_split('/[\n]/', $message);  
$size = count($matrix);  
  
for ($x = 0; $x < $size; $x++) {  
 $matrix[$x] = explode(' ', $matrix[$x]); - создаём массив (разбиваем нынешний по элементам)  
}  
$error\_text = Validate($matrix, $start, $end); - функция проверки правильности ввода  
if($error\_text == "") {

В случае, если всё корректно введено, то происходят дальнейшие действия. А именно по функции findShort находится массив с кратчайшим путём. В данном массиве после всех операций только одно значение положительное, поэтому, как только нашли его, сразу же выводим как кратчайший путь. Затем выполняется функция поиска траектории с его выводом там же.

$minway = findShort($matrix, $start, $end); - функция определения кратчайшего пути графа  
 $countBreak = 0;   
 for ($x = 0; $x < $size; $x++) {  
 if ($minway[$x] > 0) {  
 $countBreak++;  
 }  
 }  
 if ($countBreak > 0) {  
echo "Кратчайший путь: ", $minway[$end];

echo "<br>Траектория кратчайшего пути: ";

findWay($matrix, $minway, $start, $end); функция вывода траектории кратчайшего пути  
 } else {  
 echo "Невозможно найти путь.";  
 }  
} else {  
 echo $error\_text; - вывод ошибки  
}

Функция Validate проверяет несколько условий:

* Правильный ли формат элементов матрицы (либо звезды, как «нет пути», либо цифр, как путь)
* Правильный ли формат самой матрицы (всегда n\*n)
* Правильный ли формат начальной и конечной точки (только цифры)

Сначала функция циклично проверяет, является ли квадратной матрицей, циклично сравнивая длины строк и столбцов, в ошибочном случае возвращает текст ошибки (сразу выводит на страницу текст). Затем он проверяет ошибочное условие (если элемент и не звездочка, и не число, и не типа integer, то возвращается текст ошибки). А третье условие проверяет, являются ли вообще числом начальная и конечная точка.

function Validate ($matrix, $start, $end) {

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

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

$error\_text = "Матрица должна быть квадратной.";

return $error\_text;

}

}

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

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

if ($matrix[$x][$y] != '\*' && !is\_numeric($matrix[$x][$y]) && !is\_int($matrix[$x][$y])) {

$error\_text = "Не верный формат данных. Таблица должна состоять из звёздочек и натуральных чисел.";

return $error\_text;

}

}

}

if (!is\_numeric($start) || !is\_numeric($end)) {

$error\_text = "Неверный ввод начальной и конечной точки";

return $error\_text;

}

}

Функция findShort подготавливает специальный массив для определения кратчайшего пути и переводит на следующую, самую главную функцию – findShortWay. Этот массив будет отображать все кратчайшие пути от конечной точки до всех остальных точек, но только путь для нужной точки будет выводиться, ведь он специально будет положительным, а остальные – отрицательными.

function findShort($matrix, $start, $end)  
{  
 $size = count($matrix); - длина массива идентична длине строки (столбца) матрицы  
 $minway = array($size);  
 for ($x = 0; $x < $size; $x++) { - создание массива  
 $minway[$x] = -1;  
 }  
 return findShortWay($matrix, $start, $end, $minway)[1];  
}

Функция findShortWay начинает поиск короткого маршрута с конечной точки и ищет кратчайшие пути к каждой точке рекурсивно (что было сказано во вкладке теория). На каждом шаге обрабатывается ближайшая ещё не обработанная вершина (расстояние до неё уже известно). При её обработке все ещё не посещённые соседи добавляются в очередь для посещения с приоритетом (расстояние до каждой из них рассчитывается как расстояние до текущей вершины + длина ребра). При этом рекурсивная функция должна вернуть значение 0, как только она доберется до точки старта, так как расстояние от точка старта до точки старта равна нулю. Также, чтобы не просчитывать кратчайшие расстояние до отдельных узлов повторно, мы и записываем уже просчитанные расстояние в массив minway - именно с помощью него можно определить узлы, составляющие минимальный путь, а затем вывести эту последовательность узлов на страницу сайта.

function findShortWay($matrix, $start, $end, $minway)  
{  
 if($start == $end) {  
 return [0, $minway]; - если начальная и конечная точки совпадают, возвращается 0 и короткий путь  
 }  
  
 $minway[$end] = -2;   
 $size = count($matrix);  
  
 $isFind = false; - данное значение будет True только в одной из рекурсивных функций – нужный кратчайший путь  
 $min = 99999999; - необязательное значение, чтобы проще её было изменить  
 for($x = 0; $x < $size; $x++)  
 {  
 if($matrix[$x][$end] != '\*') {  
 if($minway[$x] != -2) {  
 if ($minway[$x] == -1) {  
 $result = findShortWay($matrix, $start, $x, $minway);  
 $minway = $result[1];  
 $minway[$x] = $result[0];  
 }  
 $path = $matrix[$x][$end] + $minway[$x]; - путь (подбор путей к некоторой точке  
  
 if ($path < $min) $min = $path; - если путь найден минимальный, записываем его в специальный массив  
 {  
 $isFind = true; - как только значение нашлось, функции постепенно возвращают свои значения  
 }  
 }  
 }  
 }  
  
 if($isFind) {   
 $minway[$end] = $min; - присваивание минимального значение   
 return [$min, $minway]; - и его возвращение  
 } else {  
 return [-1, $minway];  
 }  
}

Функция findWay на основе уже найденного кратчайшего пути выводит его траекторию, начиная с конца и постепенно переходя к началу. Сперва находится idMin – то есть предыдущая точка кратчайшего пути в матрице, а затем end присваивает этот idMin, как бы показывая себя как нынешняя окончательная точка. Причём массив way присваивает себе последнюю конечную точку, тем самым в конце образов путь от конца до начала. А последние циклы просто выводят этот массив в обратном порядке

function findWay($matrix, $minway, $start, $end)  
{  
 $size = count($minway);  
  
 $way = array($size);  
 $sizeWay = 0; - подготовка массива траектории  
  
 while($end != $start) {  
 $min = 999999;  
 $idMin = -1;  
 for ($x = 0; $x < $size; $x++) {  
 if ($matrix[$x][$end] != '\*' && $minway[$x] >= 0) {  
 $path = $matrix[$x][$end] + $minway[$x];  
 if ($path < $min) {  
 $min = $path;  
 $idMin = $x; - нахождение предыдущей последней точки  
 }  
 }  
 }  
  
 $way[$sizeWay] = $end; - присваивание последней конечной точке   
 $sizeWay++; - и увеличение массива  
  
 $end = $idMin; - присваивание новой конечной точки  
 }  
  
 $way[$sizeWay] = $end;  
 $sizeWay++;  
  
 for($x = $sizeWay - 1; $x >= 0; $x--){ - конечная точка находится в начале массива, а начальная – в конце, поэтому массив с конца выводим  
 echo $way[$x];  
 if($x != 0)  
 echo "⇢";  
 }  
}

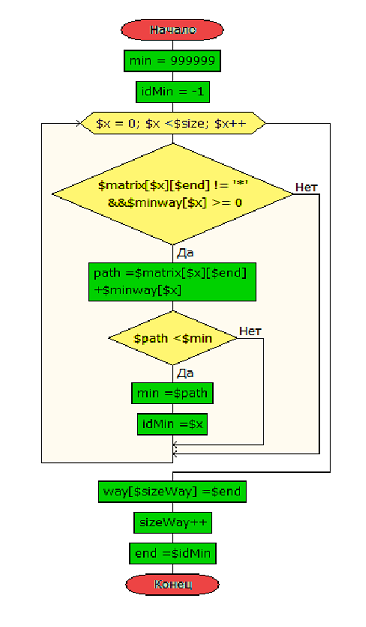


Рис. 1 - функция findWay

1. **Выводы**

В результате проведения работы были изучены:

* Ajax запросы
* JQuery
* Отладка php файла с помощью XDebug
* Изменение работы сервера с помощью изменения файла php.ini
* Основы работы с PHP
* Ориентированные графы
* Алгоритм Дейкстры (переделанный в рекурсию)

# **Список литературы**

(б.д.).

*jQuery CDN – Latest Stable Versions*. (б.д.). Получено из jquery: https://releases.jquery.com/

*Как сохранить настройки php.ini в OpenServer?* (б.д.). Получено из Хабр Q&A: https://qna.habr.com/q/417546

*Лекция 9: Алгоритм Дейкстра поиска кратчайших путей в графе*. (б.д.). Получено из ИНТУИТ: https://intuit.ru/studies/courses/1033/241/lecture/6224?page=1

*Ориентированный граф*. (б.д.). Получено из Академик: https://dic.academic.ru/dic.nsf/ruwiki/1077749

*Примеры отправки AJAX JQuery*. (б.д.). Получено из snipp.ru: https://snipp.ru/jquery/ajax-jquery