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

**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.

**Описание метода выполнения**

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.

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.

Воспользуемся уже созданным файлом с прошлой работы, чтобы установить ссылку на другую лабораторную работу.

<!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>  
</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>

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.

<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=""></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>  
 </tr>  
 </table>  
  
</form>  
<script>  
 function sendmass() { - функция, которая 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;  
 $.ajax({  
 type: "POST",  
 url: '/scripts/scriptslab4.php',  
 data: ({  
 'mass': text,  
 'begin': starttext,  
 'end': endtext  
 }),  
 success: function(data){  
 ***document***.getElementById("test").innerHTML = data;  
 }  
 })  
 }  
  
 </script>  
<div id ="test"></div>  
</body>  
</html>

Скрипт php файла начинается с следующих строк. В ходе компиляции код обращается к функциям.

$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 == "") {  
  
 $minway = findShort($matrix, $start, $end); - функция определения кратчайшего пути графа  
 $countBreak = 0; - переменная, отвечающая за то, есть ли вообще путь от одной заданной вершины к другой  
 for ($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)

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.

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

Эта функция начинает поиск короткого маршрута с конечной точки и ищет кратчайшие пути рекурсивно.

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;  
 $min = 99999999;  
 for($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];  
  
 if ($path < $min) $min = $path;  
 {  
 $isFind = true; - как только значение нашлось, функции постепенно возвращают свои значения  
 }  
 }  
 }  
 }  
  
 if($isFind) {  
 $minway[$end] = $min;  
 return [$min, $minway];  
 } else {  
 return [-1, $minway];  
 }  
}

Функция findWay на основе уже найденного кратчайшего пути выводит его траекторию.

function findWay($matrix, $minway, $start, $end)  
{  
 $size = count($minway);  
  
 $way = array($size);  
 $sizeWay = 0;  
  
 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;  
 }  
 }  
 }  
  
 $way[$sizeWay] = $end;  
 $sizeWay++;  
  
 $end = $idMin;  
 }  
  
 $way[$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 "⇢";  
 }  
}

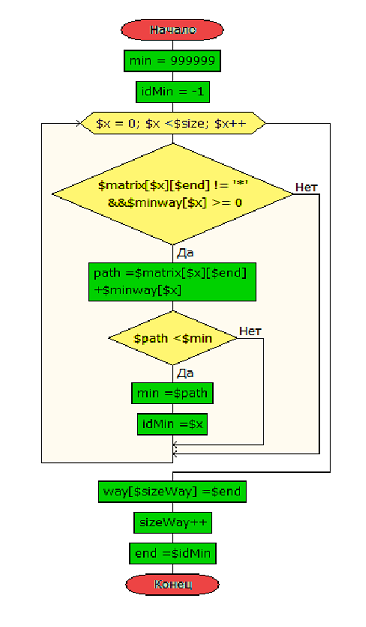


Рис. 1 - функция findWay

**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