Министерство образования и науки Российской Федерации

ФГБОУ ВО Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Лабораторная работа №4

«Кратчайший путь в ориентированном графе»

Выполнил студент

группы ИВТАСбд-11

Зюзин Г.А

Проверил Игонин А.Г.

Ульяновск, 2022 г.

# Оглавление.

[Оглавление. 2](#_Toc1)

[Цель работы. 3](#_Toc2)

[Метод выполнения.](#_Toc3)

[Листинг кода. 8](#_Toc4)

[Выводы. 12](#_Toc5)

[Список литературы.](#_Toc6)

# Цель работы.

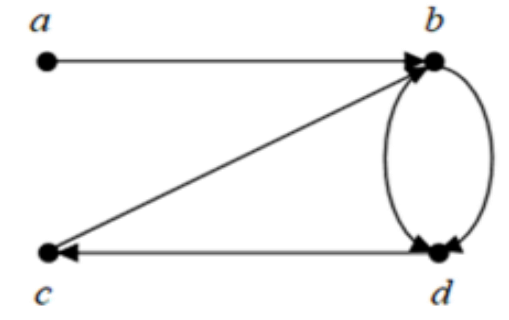
Разработать программу на языке программирования PHP нахождения кратчайшего пути в ориентированном графе с использованием AJAX запроса. Реализовать интерфейс с вводом матрицы и с вводом начала и конца пути. При выводе программа должна показать весь маршрут, путем перечисления вершин и суммарная стоимость пути.

# Метод выполнения.

Ориентированный граф – это граф, рёбра которого имеют направление, то есть ориентированные ребра или *дуги*. Орграф представляет собой пару G = (V, E), где V – конечное множество вершин, а E – отношение на V.

На Рис. 1 приведён пример орграфа с множество вершин V = {a, b, c ,d} и множеством дуг E = {ab, bd, cb, db, dc}.

Рис. 1



Граф имеет матрицу смежности. Матрица смежности – это квадратная логическая матрица отношения на множестве вершин графа, которое задаётся его рёбрами, является одним из способов представить граф. Таблица имеет два значение 1 и 0 (или И и Л, соответственно), которые обозначают отношения между вершинами графа.

Для работы программа используется AJAX запросы (Листинг 1). Программа представляет из себя следующее: страничка с тремя полями ввода (ввод матрицы, начальный и конечный путь), поле вывода и кнопка, которая передаёт данные с помощью AJAX запроса (Рис. 2).

Листинг 1

$.ajax({

type: "POST",

url: '/scripts/script4.php',

data: `matrix=${matrix}&start=${startPosition}&end=${endPosition}`,

success: function (data){

document.getElementById('output').innerHTML = data;

},

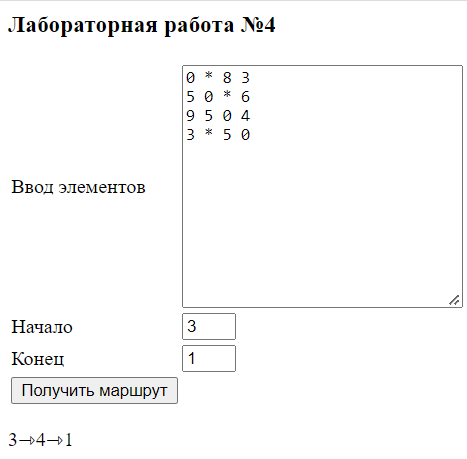
error: function (){

alert("Ошибка передачи данных");

}

});

Рис. 2



В программе прописаны следующие проверки: проверку на корректность ввода, проверка на наличия конца и начала.

Для нахождения кротчайшего пути используется алгоритм Флойда. Алгоритм Флойда делает N итераций, после i-й итерации матрица А будет содержать длины кратчайших путей между любыми двумя парами вершин при условии, что эти пути проходят через вершины от первой до i-й. На каждой итерации перебираются все пары вершин и путь между ними сокращается при помощи i-й вершины (Рис. 3, Листинг 1).

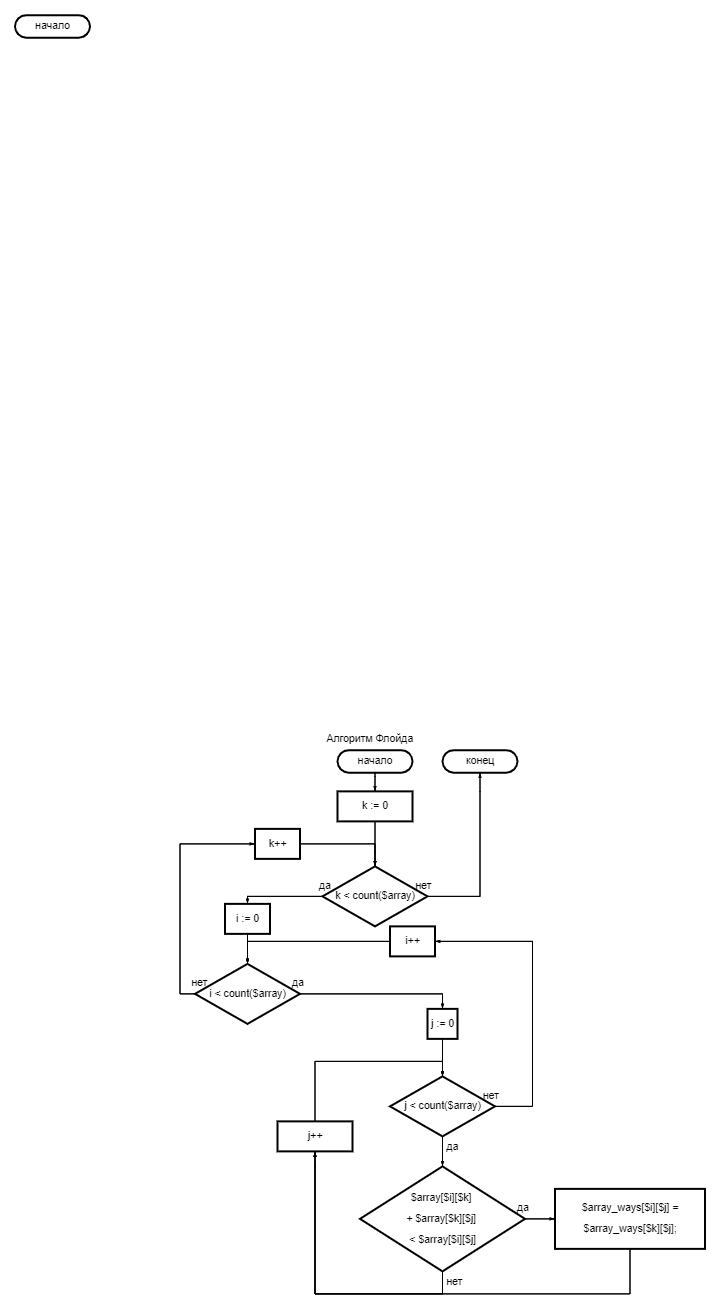


Рис. 3

Листинг 2

for($k = 0; $k < count($array); $k++){

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

for($j = 0; $j < count($array); $j++){

if($array[$i][$k] + $array[$k][$j] < $array[$i][$j]){

$array[$i][$j] = $array[$i][$k] + $array[$k][$j]; // укорачивает в матрице расстояний

$array\_ways[$i][$j] = $array\_ways[$k][$j]; //укорачивает путь в матрице путей

}

}

}

}

Перед работой алгоритма матрица А заполняется длинами рёбер графа. Результатом работы программы является путь, кротчайший между заданными вершинами (Рис. 4).

Рис. 4



На вход программа получает текст, поэтому необходимо инициализировать матрицу из этого текста. Текст разбить на строки, а строки разбить на элементы (Листинг 3).

Листинг 3

for($i = 0; $i < count($row\_matrix);$i++){

$column\_row\_matrix = explode(' ', trim($row\_matrix[$i]));

$array\_row = [];

for($j = 0; $j < count($column\_row\_matrix); $j++){

if(count($column\_row\_matrix) == 1 && $column\_row\_matrix[$j] == '')

die("Ошибка: лишние переносы!");

if($column\_row\_matrix[$j] != "\*" && !is\_numeric($column\_row\_matrix[$j])){

die('Ошибка: в матрице должно содержаться цифра или "\*"!');

}

if($column\_row\_matrix[$j] === '\*'){

array\_push($array\_row, INF);

} else{

array\_push($array\_row, $column\_row\_matrix[$j]);

}

}

array\_push($array, $array\_row);

}

# Листинг кода.

AJAX запрос (ajax\_script4.js):

function sendData(){

let matrix = document.getElementById('matrix').value;

let startPosition = document.getElementById('startPosition').value;

let endPosition = document.getElementById('endPosition').value;

let array = [];

let rows\_array = matrix.split("\n");

for(let i = 0; i < rows\_array.length; i++){

let column\_rows\_array = rows\_array[i].split(" ");

let array\_row = [];

for(let j = 0; j < column\_rows\_array.length; j++){

array\_row.push(column\_rows\_array[j]);

}

array.push(array\_row);

}

for(let i = 0; i < array.length; i++){

if(array[i][i] !== '0'){

alert("Алгоритм Флойда игнорирует петли!");

break;

}

}

$.ajax({

type: "POST",

url: '/scripts/script4.php',

data: `matrix=${matrix}&start=${startPosition}&end=${endPosition}`,

success: function (data){

document.getElementById('output').innerHTML = data;

},

error: function (){

alert("Ошибка передачи данных");

}

});

}

Алгоритм поиска кротчайшего пути (script4.php):

<?php

$matrix = $\_POST['matrix'];

$start = $\_POST['start'];

$end = $\_POST['end'];

$row\_matrix = explode("\n", $matrix);

$array = [];

$array\_ways = []; // матрица путей

for($i = 0; $i < count($row\_matrix);$i++){

$column\_row\_matrix = explode(' ', trim($row\_matrix[$i]));

$array\_row = [];

for($j = 0; $j < count($column\_row\_matrix); $j++){

if(count($column\_row\_matrix) == 1 && $column\_row\_matrix[$j] == '')

die("Ошибка: лишние переносы!");

if($column\_row\_matrix[$j] != "\*" && !is\_numeric($column\_row\_matrix[$j])){

die('Ошибка: в матрице должно содержаться цифра или "\*"!');

}

if($column\_row\_matrix[$j] === '\*'){

array\_push($array\_row, INF);

} else{

array\_push($array\_row, $column\_row\_matrix[$j]);

}

}

array\_push($array, $array\_row);

}

if($start > count($array) || $end > count($array) || $start <= 0 || $end <= 0)

die("Ошибка: нет такой вершины! Подсказка: число должно быть больше 0 и меньше ". (count($array)+1) .".");

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

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

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

die("Ошибка: матрица должна быть квадратной!");

}

}

}

// инициализация матрицы путей

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

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

$array\_ways[$i][$j] = $i; //задаёт путь до вершины напрямую, если есть

}

$array\_ways[$i][$i] = -1;

}

//Алгоритм Флойда

//Матрица расстояний Ak[i,j] = min(Ak-1[i, j], Ak-1[i,k] + Ak-1[k, j])

for($k = 0; $k < count($array); $k++){

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

for($j = 0; $j < count($array); $j++){

if($array[$i][$k] + $array[$k][$j] < $array[$i][$j]){

$array[$i][$j] = $array[$i][$k] + $array[$k][$j]; // укорачивает в матрице расстояний

$array\_ways[$i][$j] = $array\_ways[$k][$j]; //укорачивает путь в матрице путей

}

}

}

}

//Последовательность вершин

$array\_node = [];

$start--;

$end--;

//Проверка наличия путей из вершины

if($array[$start][$end] == INF)

die("Нет такого пути! Подсказка: возможно из какой-то вершины рёбра не идут.");

while($end!=-1){

array\_push($array\_node, $end+1);

$end = $array\_ways[$start][$end];

}

$array\_node = array\_reverse($array\_node);

for($i = 0; $i < count($array\_node); $i++){

if($i === 0){

echo $array\_node[$i];

} else{

echo '⇾'.$array\_node[$i];

}

}

?>

# Выводы.

Во время выполнения лабораторной работы, были изучены ориентированный граф и способы нахождения кротчайшего пути в орграфе, разработана программа для нахождения кротчайшего пути между вершинами. Для разработки был использован язык программирования PHP и JS, а именно AJAX запросы.

# Список литературы.

1. Р. Хаггарти, Дискретная математика для программистов / Техносфера, г. Москва, 2005 г. - 400с.
2. Справочник по JavaScript - <https://javascript.ru/manual>
3. Справочник по PHP - <https://www.php.net/manual/ru/langref.php>