Минестерство 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

Долгов А. П.

Ульяновск, 2022

**Цель работы:**

Разработать программу на php нахождения кратчайшего пути в графе. Граф должен быть неориентированным.

**Описание метода выполнения.**

Граф - это множество упорядоченных пар G := (V, E), где V – множество вершин графа, а E – множество рёбер 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 рёбер, 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, соединённых рёбрами или дугами, у которых сумма весов этих рёбер или дуг наименьшая.

Граф указывается через матрицу смежности. Ввод матрицы смежности осуществляется через специальное поле textarea. Ниже имеются два поля, предназначенные для указания вершины откуда будет искаться кратчайший путь и вершины, куда и этот путь ведёт. Под полями ввода находится кнопка «Get shortest path», на нажатию на которую будет производится поиск кратчайшего пути вы вывод соответствующего результата. В случае некорректности введённых данных будет выведена ошибка.

В js файле реализован метод передачи сообщения в php файл:

function send(message)  
{  
 $.ajax({  
 url: 'php/shortestPath.php',  
 type: 'POST',  
 data: message,  
 success: function(data){  
 document.getElementById('divForOutput').innerHTML = data;  
 //('p.output').text('Data sent!');  
 }  
 })  
}

Само сообщение задаётся в html структуре в onclick поле кнопки.

<button id = "getShortestPath" onclick = "  
let matrix\_adjacency = document.getElementById('matrix\_adjacency').value;  
let source = document.getElementById('input\_source').value;  
let destination = document.getElementById('input\_destination').value;  
  
let message = {  
 'matrix\_adjacency': matrix\_adjacency,  
 'source' : source,  
 'destination' : destination  
}  
  
send(message);  
">Get shortest path</button><br><br>

В PHP файле первым делом происходит считывание данных:

$message = $\_POST['matrix\_adjacency'];  
$source = $\_POST['source'];  
$dest = $\_POST['destination'];

Затем производится инициализация матрицы:

$matrix\_adjacencyElements = preg\_split('/[ \n]/', $message);  
$sizeMatrix = sqrt(count($matrix\_adjacencyElements));

В следующей части кода находятся проверки корректности введённых данных, инициализация матрицы как двумерного массива и, собственно, само нахождение кратчайшего пути и вывод данных о нём.

if(($dest >= 0 && $dest < $sizeMatrix) && ($source >= 0 && $source < $sizeMatrix) ) {  
 $isFormat = true;  
 for ($x = 0; $x < $sizeMatrix \* $sizeMatrix; $x++) {  
 if ($matrix\_adjacencyElements[$x] != '0' && !ctype\_digit($matrix\_adjacencyElements[$x]) && !is\_int($matrix\_adjacencyElements[$x])) {  
  
 $isFormat = false;  
 break;  
 }  
 }  
  
 if ($isFormat) {  
 if ($sizeMatrix - (int)$sizeMatrix == 0) {  
 $matrix\_adjacency = array($sizeMatrix);  
 for ($x = 0; $x < $sizeMatrix; $x++) {  
 $matrix\_adjacency[$x] = array($sizeMatrix);  
 for ($y = 0; $y < $sizeMatrix; $y++)  
 $matrix\_adjacency[$x][$y] = $matrix\_adjacencyElements[$x + $y \* $sizeMatrix];  
 }  
  
 $short = findShort($matrix\_adjacency, $source, $dest);  
 $countBreak = 0;  
 for ($x = 0; $x < $sizeMatrix; $x++)  
 if ($short[$x] > 0)  
 $countBreak++;  
  
 if ($countBreak > 0) {  
 echo "Shortest path length: " . $short[$dest];  
 echo "<br> Path: ";  
 findPath($matrix\_adjacency, $short, $source, $dest);  
 } else {  
 echo "It is not possible to find the path";  
 }  
 } else {  
 echo "The matrix must be squared";  
 }  
 } else {  
 echo "Invalid data format. The table must consist of zeros and natural digits";  
 }  
} else {  
 echo "The numbers of starting or ending points is entered wrong <br> They must be more or equals to zero and less than the size of matrix";  
}

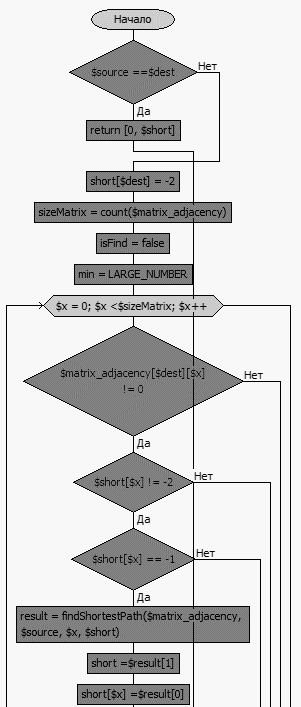
Нахождение кратчайшего пути реализовано в функции findShort.

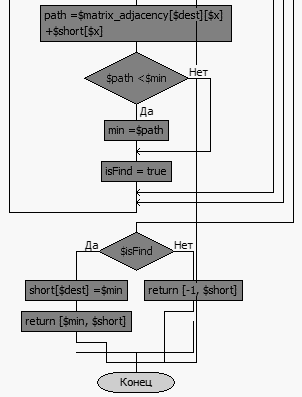
function findShort($matrix\_adjacency, $source, $dest)  
{  
 $sizeMatrix = count($matrix\_adjacency);  
 $short = array($sizeMatrix);  
 for($x = 0; $x < $sizeMatrix; $x++)  
 $short[$x] = -1;  
 return findShortestPath($matrix\_adjacency, $source, $dest, $short)[1];  
}

Эта функция фактически является вспомогательной, в которой лишь производится создание и заполнение переменной short, которая предназначена для указания кратчайшего пути до каждого элемента от A и состояния элементов (путь не вычислен, путь рассчитывается).

Основная же работа происходит в функции findShortestPath.

function findShortestPath($matrix\_adjacency, $source, $dest, $short)  
{  
 if($source == $dest)  
 return [0, $short];  
  
 $short[$dest] = -2;  
 $sizeMatrix = count($matrix\_adjacency);  
  
 $isFind = false;  
 $min = LARGE\_NUMBER;  
 for($x = 0; $x < $sizeMatrix; $x++)  
 {  
 if($matrix\_adjacency[$dest][$x] != 0) {  
 if($short[$x] != -2) {  
 if ($short[$x] == -1) {  
 $result = findShortestPath($matrix\_adjacency, $source, $x, $short);  
 $short = $result[1];  
 $short[$x] = $result[0];  
 }  
 $path = $matrix\_adjacency[$dest][$x] + $short[$x];  
  
 if ($path < $min) $min = $path;  
 $isFind = true;  
 }  
 }  
 }  
  
 if($isFind) {  
 $short[$dest] = $min;  
 return [$min, $short];  
 } else{  
 return [-1, $short];  
 }  
}

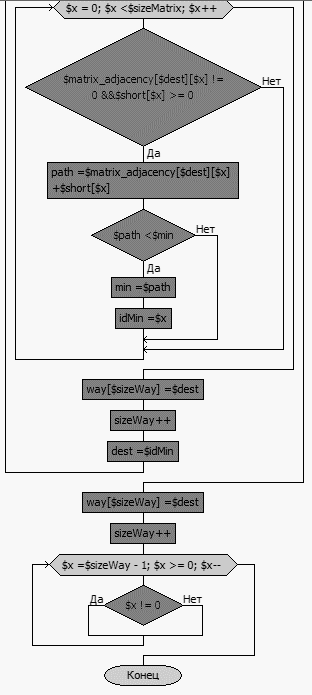




Это рекурсивная функция, которая прокладывает путь с конца, до начала. Всё начинается с последнего элемента. Первым делом определяется до какого из всех элементов, соединённых с последним, ближе дойти до начала. Делается это путём вызова той же самой рекурсивной функции для каждого из инцедентных элементов. Элементы, которые запросили у другого элемента кратчайшее растояние до начала находятся в режиме рассчитования, а поэтому дочерные вызовы не могут запрашивать у таких элементов кратчайший путь (если это не сделать, образовывались бы зацикливания программы). Каждый вызов рекурсивно вызывает ещё функции запроса кратчайшего пути для инцедентных с вершиной, из которой был произведён вызов, вершин. Так продолжается до того момента, пока рекурсия не дойдёт до вершины начала пути, который имеет кротчайшее расстояние до себя же, равное 0. После чего функции начинают друг за другом завершаться, пока функции, вызванные из конечного элемента не вернут результат, из которого не будет выбран кротчайший.

Функция findPath производит вывод кротчайшего пути.

function findPath($matrix\_adjacency, $short, $source, $dest)  
{  
 $sizeMatrix = count($short);  
  
 $way = array($sizeMatrix);  
 $sizeWay = 0;  
  
 while($dest != $source) {  
 $min = LARGE\_NUMBER;  
 $idMin = -1;  
 for ($x = 0; $x < $sizeMatrix; $x++) {  
 if ($matrix\_adjacency[$dest][$x] != 0 && $short[$x] >= 0) {  
 $path = $matrix\_adjacency[$dest][$x] + $short[$x];  
 if ($path < $min) {  
 $min = $path;  
 $idMin = $x;  
 }  
 }  
 }  
  
 $way[$sizeWay] = $dest;  
 $sizeWay++;  
  
 $dest = $idMin;  
 }  
  
 $way[$sizeWay] = $dest;  
 $sizeWay++;  
  
 for($x = $sizeWay - 1; $x >= 0; $x--){  
 echo $way[$x];  
 if($x != 0)  
 echo "->";  
 }  
}



Делается это благодаря массиву short. С помощью него функция как бы идёт по пути от конечного до начального элемента. Из соединённых с конечной вершиной элементов выбирается тот элемент, из которого всех ближе до начала (это известно из массива short). Затем, из выбранного элемента так же выбирается тот, от которого ближе до начала, и так до конца, параллельно записывая данные о пути в массив way. Получив путь, он, вместе с кротчайшем расстоянием выводится на экран.

**Вывод**

Данная программа протестирована и успешно работает. Программа позволяет ввести граф с помощью матрицы смежности и определить путь, а также кротчайшее расстояние от первой выбранной вершины до второй.

**Список источников**

1. Видео с youtube. Графы, вершины, ребра, инцидентность, смежность. URL: <https://www.youtube.com/watch?v=fCfPjm8u89U>
2. Видео с youtube. Что такое граф? | Дискретная математика | Умскул. URL: <https://www.youtube.com/watch?v=CsLszXfaQE4>
3. Видео с youtube. Алгоритм Дейкстры или как навигатор определяет оптимальный маршрут. URL: <https://www.youtube.com/watch?v=7oUt0zhv2sA&t=364s>