**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 2 з дисципліни

«Проектування алгоритмів»

**«Неінформативний, інформативний та локальний пошук»**

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*Боборикін Артем ІП-12*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.М.*

Київ 2022

# Мета лабораторної роботи

Мета роботи – розглянути та дослідити алгоритми неінформативного, інформативного та локального пошуку. Провести порівняльний аналіз ефективності використання алгоритмів.

**Варіант 5**

1. Псевдокод алгоритму

## Програмна реалізація алгоритму

### Вихідний код

Файл app.js – файл створення Desktop Application

const {***app***, BrowserWindow} = ***require***('electron')  
const url = ***require***('url')  
const path = ***require***('path')  
  
  
let win;  
function createWindow(){  
 win = new BrowserWindow({width: 700, height: 500,  
 webPreferences: {  
 nodeIntegration: true,  
 contextIsolation: false,  
 enableRemoteModule: true,  
 }})  
 win.loadURL(url.format({  
 pathname: path.join(***\_\_dirname***, 'index.html'),  
 protocol: 'file:',  
 slashes: true  
 }))  
 win.webContents.openDevTools()  
 win.on('closed', ()=>{  
 win=null  
 })  
}  
  
***app***.on('ready', createWindow)

Файл index.js – файл реалізації алгоритмів, та обробник .html файлу візуалізації

const generator=***require***('generate-maze')  
const {SearchTree} = ***require***('./tree.js')  
  
const MAZE\_SIZE = 5  
let finish = {  
 x: MAZE\_SIZE-1,  
 y: MAZE\_SIZE-1,  
}  
const seed = ***Math***.floor(***Math***.random()\****Math***.pow(10, 10)+***Math***.pow(10,9))  
const maze = generator(MAZE\_SIZE, MAZE\_SIZE, true, seed)  
showMaze()  
let tree = new SearchTree(maze)  
***console***.log(tree)  
  
***document***.querySelector('#ids').onclick = ids  
***document***.querySelector('#reset').onclick = showMaze  
***document***.querySelector('#a').onclick = aip  
  
//FUNCTIONS  
function showMaze(){  
 let html = `<div id="mazeContainer" style="display:grid; grid-template-columns: repeat(${MAZE\_SIZE}, 1fr); background-color:whitesmoke; margin: auto; width: min-content">`  
  
 for(let row of maze){  
 for(let cell of row){  
 html+=`<div style="width: 20px; height: 20px; ${cell.top?'border-top: 1px solid black;':'border-top: 1px solid whitesmoke;'}${cell.left?'border-left: 1px solid black;':'border-left: 1px solid whitesmoke;'}${cell.right?'border-right: 1px solid black;':'border-right: 1px solid whitesmoke;'}${cell.bottom?'border-bottom: 1px solid black;':'border-bottom: 1px solid whitesmoke;'}"></div>`  
 }  
 }  
  
 ***document***.querySelector('#workArea').innerHTML=html  
}  
function ids(){  
 let n = 0  
 tree.moveFull(tree.root, (node)=>{  
 n++  
 if(node.left===null&&node.top===null&&node.right===null&&node.bottom===null){  
 if(node.x===finish.x&&node.y===finish.y){  
  
 return true  
 }  
 else{  
 tree.init(node)  
 return false  
 }  
 }  
 })  
 ***console***.log(n)  
 drawWay()  
}  
function drawWay(node){  
 let arr = ***document***.querySelector('#mazeContainer').children  
 let currentNode = node?node:tree.aim  
  
 while(currentNode){  
 arr[currentNode.y\*MAZE\_SIZE+currentNode.x].style.backgroundColor = 'red'  
  
 currentNode = currentNode.parent  
 }  
}  
function aip(){  
 let arr = [tree.root]  
 let finalNode  
 let it = 0  
 let stan = 0  
 let kut = 0  
  
 while(true){  
 finalNode = arr[0]  
 let finalIndex = 0  
  
 for(let i=0; i<arr.length; i++){  
 it++  
 if(checkDistance(arr[i])<checkDistance(finalNode)){  
 finalNode = arr[i]  
 finalIndex = i  
 }  
 }  
 if(finalNode.x === MAZE\_SIZE-1 && finalNode.y === MAZE\_SIZE-1)break  
 else{  
 it++  
 stan++  
 tree.init(finalNode)  
  
 if(finalNode.left&&finalNode.top&&finalNode.right&&finalNode.bottom){  
 kut++  
 }  
  
 if(finalNode.left)arr.push(finalNode.left)  
 if(finalNode.top)arr.push(finalNode.top)  
 if(finalNode.right)arr.push(finalNode.right)  
 if(finalNode.bottom)arr.push(finalNode.bottom)  
  
 arr.splice(finalIndex, 1)  
 }  
 }  
 ***console***.log(it)  
 ***console***.log(stan)  
 ***console***.log(kut)  
 ***console***.log(arr.length)  
 drawWay(finalNode)  
}  
function checkDistance(node){  
 return node.wayL + ***Math***.abs(MAZE\_SIZE-1-node.x)+***Math***.abs(MAZE\_SIZE-1-node.y)  
}

Файл tree.js – файл, який описує клас пошукового дерева SearchTree та клас вузла дерева Node. Використовується під час реалізації алгоритмів у index.js

class Node {  
 constructor(obj, parent = null, pDir = null) {  
 this.x = obj.x;  
 this.y = obj.y;  
 this.left = null;  
 this.top = null;  
 this.right = null;  
 this.bottom = null;  
 this.parent = parent;  
 this.pDir = pDir;  
 this.wayL = parent?parent.wayL+1:0;  
 }  
}  
  
class SearchTree {  
 constructor(maze) {  
 this.root = new Node(maze[0][0]);  
 this.maze = maze;  
 this.aim = null;  
 }  
  
 init(node){  
 if(!this.maze[node.y][node.x].left&&node.pDir!=='left'){  
 node.left= new Node(this.maze[node.y][node.x-1], node, 'right')  
 }  
 if(!this.maze[node.y][node.x].top&&node.pDir!=='top'){  
 node.top= new Node(this.maze[node.y-1][node.x], node, 'bottom')  
 }  
 if(!this.maze[node.y][node.x].right&&node.pDir!=='right'){  
 node.right= new Node(this.maze[node.y][node.x+1], node, 'left')  
 }  
 if(!this.maze[node.y][node.x].bottom&&node.pDir!=='bottom'){  
 node.bottom= new Node(this.maze[node.y+1][node.x], node, 'top')  
 }  
 }  
  
 moveFull(node, callback) {  
 if (node != null) {  
 if(callback(node)){  
 this.aim = node  
 return true  
 }  
 return this.moveFull(node.left, callback) || this.moveFull(node.top, callback) || this.moveFull(node.right, callback) || this.moveFull(node.bottom, callback);  
 }  
 }  
}  
  
***module***.exports = {SearchTree, Node}

Файл index.html – візуалізація контенту

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Document</title>  
  
 <style>  
 p{  
 margin: 0;  
 font-size: 25px;  
 }  
 .tdiv{  
 margin: 0 5px;  
 border-radius: 5px;  
 background-color: aqua;  
 padding: 10px 20px;  
 cursor: pointer;  
 }  
 .tdiv:hover{  
 opacity: 0.7;  
 }  
 </style>  
</head>  
<body>  
  
<div id="workArea" style="max-width: 100%"></div>  
<div style="display: flex; flex-direction: row; justify-content: center; width: 100%; margin-top: 10px">  
 <div id="ids" class="tdiv">  
 <p>IDS</p>  
 </div>  
 <div id="reset" class="tdiv">  
 <p>Reset</p>  
 </div>  
 <div id="a" class="tdiv">  
 <p>A\*</p>  
 </div>  
</div>  
  
<script>  
 ***require***('./index')  
</script>  
</body>  
</html>

Характеристика оцінювання алгоритму пошуку A\*

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Ітерації | Всього станів | Всього станів у пам'яті |
| Стан 1 (5) | 19 | 19 | 4 |
| Стан 2 (5) | 20 | 20 | 5 |
| Стан 3 (10) | 34 | 34 | 7 |
| Стан 4 (10) | 25 | 25 | 8 |
| Стан 5 (20) | 134 | 134 | 7 |
| Стан 6 (20) | 292 | 292 | 8 |
| Стан 7 (30) | 498 | 498 | 12 |
| Стан 8 (30) | 760 | 760 | 13 |
| Стан 9 (40) | 607 | 607 | 16 |
| Стан 10 (40) | 481 | 481 | 21 |
| Стан 11 (50) | 1554 | 1554 | 33 |
| Стан 12 (50) | 1399 | 1399 | 26 |
| Стан 13 (60) | 2245 | 2245 | 35 |
| Стан 14 (60) | 2790 | 2790 | 22 |
| Стан 15 (70) | 3059 | 3059 | 31 |
| Стан 16 (70) | 2333 | 2333 | 14 |
| Стан 17 (80) | 3620 | 3620 | 21 |
| Стан 18 (80) | 3014 | 3014 | 22 |
| Стан 19 (100) | 5595 | 5595 | 25 |
| Стан 20 (100) | 6217 | 6217 | 35 |

Під час експериментів використовувалися алгоритми з різною розмірністю. З кожним наступним експериментом розмірність збільшувалася

Характеристика оцінювання алгоритму пошуку IDS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Ітерації | Всього станів | Всього станів у пам'яті | Глухі кути |
| Стан 1 (5) | 17 | 17 | 5 | 2 |
| Стан 2 (5) | 23 | 23 | 5 | 1 |
| Стан 3 (10) | 72 | 72 | 10 | 12 |
| Стан 4 (10) | 57 | 57 | 8 | 13 |
| Стан 5 (20) | 319 | 319 | 11 | 73 |
| Стан 6 (20) | 317 | 317 | 12 | 76 |
| Стан 7 (30) | 823 | 823 | 16 | 231 |
| Стан 8 (30) | 774 | 774 | 21 | 196 |
| Стан 9 (40) | 1410 | 1410 | 25 | 395 |
| Стан 10 (40) | 1175 | 1175 | 41 | 316 |
| Стан 11 (50) | 1813 | 1813 | 65 | 487 |
| Стан 12 (50) | 1233 | 1233 | 50 | 322 |
| Стан 13 (60) | 3074 | 3074 | 56 | 887 |
| Стан 14 (60) | 2789 | 2789 | 77 | 759 |
| Стан 15 (70) | 3961 | 3961 | 66 | 1046 |
| Стан 16 (70) | 3904 | 3904 | 73 | 1054 |
| Стан 17 (80) | 4368 | 4368 | 162 | 1131 |
| Стан 18 (80) | 5393 | 5393 | 50 | 1502 |
| Стан 19 (100) | 9657 | 9657 | 77 | 2678 |
| Стан 20 (100) | 8638 | 8638 | 126 | 2388 |

**Висновок**

У цій лабораторній роботі був розроблений алгоритм пошуку IDS та A\* для задачі пошуку шляху у лабіринті. Проведено 20 експериментів з кожним алгоритмом, які показали, що у масштабі алгоритм A\* набагато ефективніший, бо потребує менше ітерацій і пам’яті. Його ефективність збільшується у геометричній прогресії при збільшені масштабу лабіринту