LAPORAN TUGAS KECIL 3 IF2211

Penentuan Lintasan Terpendek

dengan Algoritma UCS dan A*

Ditujukan untuk memenuhi salah satu tugas kecil mata kuliah IF2211 Strategi Algoritma pada Semester II Tahun Akademik 2022/2023

Disusun oleh:

Fakih Anugerah Pratama	**13521091**
Edia Zaki Naufal Ilman	**13521141**

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

1. DESKRIPSI PERSOALAN

Peta suatu daerah dapat direpresentasikan menggunakan graf dengan node melambangan suatu lokasi dengan edge merepresentasikan jalan antar dua lokasi. Dengan banyaknya permutasi jalur dari dua titik, mencari jalur terpendek merupakan suatu dambaan. Untuk mencari lintasan terpendek antara dua titik terdapat berbagai pilihan algoritma: Dijkstra, BFS, Floyd-Warshall, Bellman Ford, UCS, dan A*. Dalam tugas kecil ini, mahasiswa diminta untuk membuat algoritma UCS dan A* untuk mencari lintasan terpendek antara dua titik tertentu.

2. SOURCE CODE

2.1. Script.js

```
const prompt = require('prompt-sync')();

// Initialize information matrices

var matrix = getMatrix('test.txt');

var nodeNames = getNames('test.txt');

var nodeCoordinates = [];

var eucToGoal = {};



// Insert coordinates of all nodes

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

    var coorX = prompt(`Coordinate X of ${nodeNames[i]}: `);
    var coorY = prompt(`Coordinate Y of ${nodeNames[i]}: `);

    nodeCoordinates.push({name: nodeNames[i], X: Number(coorX), Y:
Number(coorY)});

}



// Get begin and goal nodes

var begin = prompt("Start node: ");

var goal = prompt('Goal node: ');



// Get array of euclidean distances to goal

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

    eucToGoal[nodeNames[i]] =
euclidean(nodeCoordinates[nodeNames.indexOf(nodeNames[i])],
nodeCoordinates[nodeNames.indexOf(goal)]);

    //eucToGoal.push({name: nodeNames[i], euclidean:
euclidean(nodeCoordinates[nodeNames.indexOf(nodeNames[i])],
nodeCoordinates[nodeNames.indexOf(goal)])});

}

//var goalCoor = nodeCoordinates[nodeNames.indexOf(goal)];



var chosenMethod = prompt("What method do you want to use? ");

compute(begin, goal, matrix, createGraphMap, chosenMethod);



function euclidean(start, end){

    let Xelement = Math.pow(Math.abs(start.X - end.X), 2);

    let Yelement = Math.pow(Math.abs(start.Y - end.Y), 2);
```

```
        return (Math.sqrt(Xelement + Yelement));
    }


    function getMatrix(filepath){
        const fs = require('fs');
        var matrix = [];
        //console.log(matrix[0]);
        const allFileContents = fs.readFileSync(filepath, 'utf-8');
        var i = 0;
        var nameArray = [];
        var item = 0;
        //console.log(allFileContents);
        allFileContents.split(/\r?\n/).forEach(line =>  {
            var textLine = line.split(' ');
            console.log(`Line from file: ${textLine}`);
            for(let j = 0; j < textLine.length; j++){
                var Insert = [];
                if(i == 0){
                    nameArray[j] = textLine[j];
                    //console.log(nameArray);
                }else if(j != i - 1 && textLine[j] > 0){
                    Insert.push(nameArray[j]);
                    Insert.push(nameArray[i - 1]);
                    Insert.push(textLine[j]);
                    matrix.push(Insert);
                    item++;
                    //console.log(matrix);
                }
            }
            i++;
        });

        return matrix;
    }

    function getNames(filepath){
        const fs = require('fs');
        var matrix = [];
        //console.log(matrix[0]);
        const allFileContents = fs.readFileSync(filepath, 'utf-8');
        var i = 0;
        var nameArray = [];
        var item = 0;
        //console.log(allFileContents);
        allFileContents.split(/\r?\n/).forEach(line =>  {
            var textLine = line.split(' ');
            //console.log(`Line from file: ${textLine}`);
            for(let j = 0; j < textLine.length; j++){
                var Insert = [];

                if(i == 0){

                    nameArray[j] = textLine[j];

                    // console.log(nameArray);

                }

            }

            i++;

        });


        return nameArray;

    }


    function createPath(parent, current, value) {

        return {

            parent : parent,

            current: current,

            value: Number(value)

        };

    };


    function isExist(history, node){

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

            if(history[i] == node.current) return true;

        }
```

```
        return false;

    }


    function createGraphMap(graph){

        var graphMap = {};

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

            var current = graph[i];

            // First

            if(graphMap[String(current[0])] == null){

                graphMap[String(current[0])] = {};

            }

            if(graphMap[String(current[1])] == null){

                graphMap[String(current[1])] = {};

            }

            graphMap[String(current[0])][String(current[1])] =
    Number(current[2]);

            graphMap[String(current[1])][String(current[0])] =
    Number(current[2]);

        }

        // console.log("GRAPH MAP");

        // console.log(graphMap);

        return graphMap;

    }


    function result(start, end, graphMap, func, method){
```

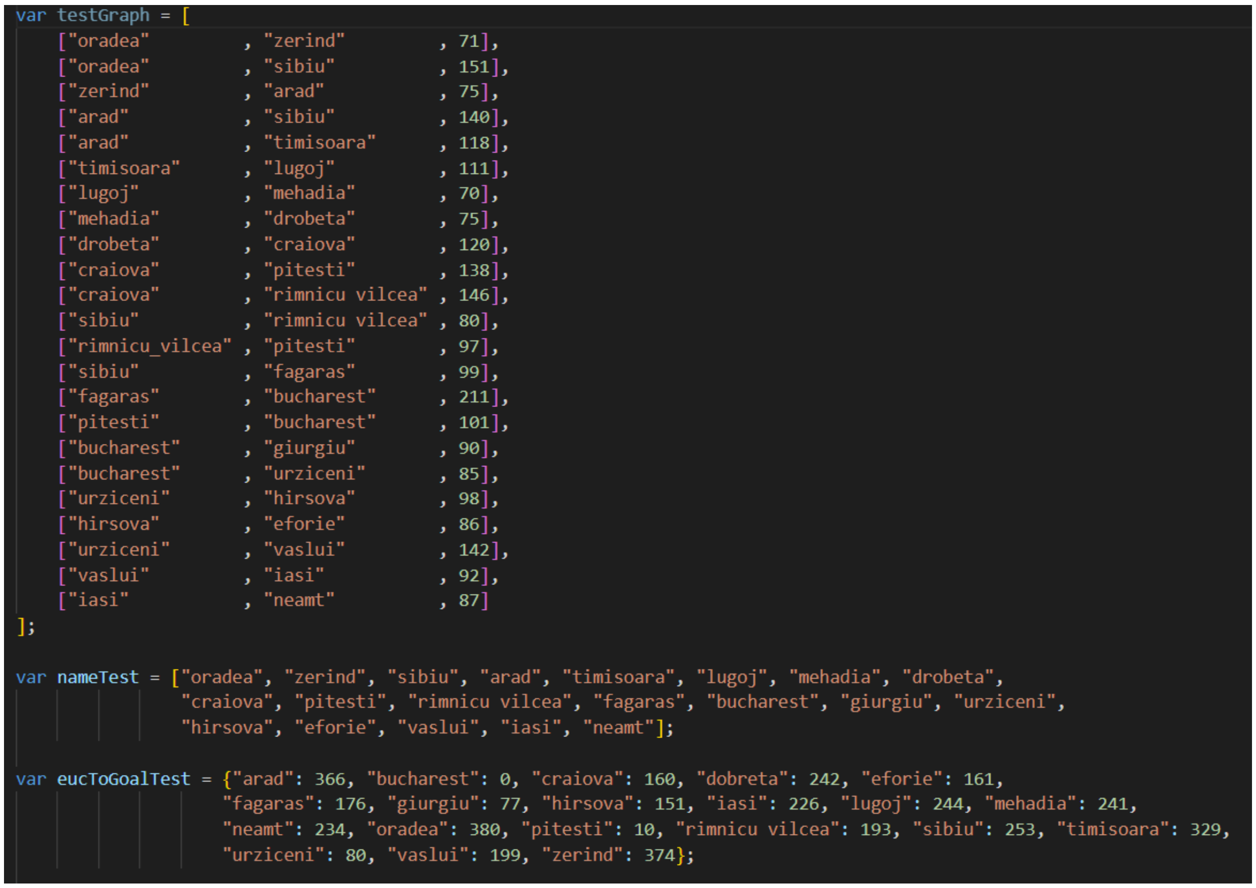
```
    function writeResult(resultPath){

        if(resultPath.parent != null){

            writeResult(resultPath.parent);

            console.log(" => " + resultPath.current + "(" +
resultPath.value + ")");

        }else{

            console.log(resultPath.current + "(" +
resultPath.value + ")");

        }

    }


    writeResult(func(graphMap, start, end, method));

}


function compute(start, end, graph, func, method){

    function search(graphMap, start, end, method){

        function dequeue(array){

            var element = array[0];


            for(let i = 0; i < array.length - 1; i++){

                array[i] = array[i + 1];

            }

            array.pop();


            return element;
        }


        function enqueue(array, element){

            if(array.length == 0){

                array.push(element);

            }else{

                var pos = -1;

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

                    if(chosenMethod == "UCS" && array[i].value >
element.value){

                        pos = i;

                        break;

                    }



                    // Diganti euclidean

                    else if(chosenMethod == "A*" && array[i].value
+ eucToGoal[array[i].current] > element.value +
eucToGoal[element.current]){



                        //console.log(`${element.current} is more
little than ${array[i].current} because`);

                        //console.log(`${element.value +
eucToGoal[element.current]} < ${array[i].value +
eucToGoal[array[i].current]}`);

                        pos = i;

                        break;
                    }
                }

                // If pos is at the end or not
                if(pos == -1){
                    array.push(element);
                }else{
                    for(let i = array.length; i > pos; i--){
                        if(i == array.length){
                            array.push(array[array.length - 1]);
                        }else{
                            array[i] = array[i - 1];
                        }
                    }
                }

                // Insert element at pos
                array[pos] = element;
            }
        }

        function possibleNodes(graphMap, currentNode){
            var possibilities = [];
            var node = graphMap[String(currentNode)];
```

```
            for(var i in node){

                possibilities.push({current: i, value: node[i]});

            }

            return possibilities;

        }

        var queue = [];

        queue.push(createPath(null, start, 0));

        var head = 0;

        // console.log("START QUEUE");

        // console.log(queue);

        var history = []

        //console.log(queue);

        while(queue.length > 0){

            var active = queue[0];

            // console.log("CURRENTLY ACTIVE");

            // console.log(active.current);

            queue.splice(0, 1);

            //head++;


            //head++;

            // console.log(active.current);

            // console.log(head);

            history.push(active.current);
            // if(head == 5){

            //     console.log(queue);

            // }


            if(active.current == end){

                return active;

            }else{

                var nextNodes = possibleNodes(graphMap,
active.current);

                // console.log("POSSIBLE NEXT");

                // console.log(nextNodes);

                //console.log(history);

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

                    if(!isExist(history, nextNodes[i])){



                        enqueue(queue, createPath(active,
nextNodes[i].current, Number(nextNodes[i].value + active.value)));

                    }

                    else{

                        //console.log(`${active.current} is
already in history!`);

                    }



                }

            }
```

```
                    // console.log("QUEUE");

                    // console.log(queue);

               }

          }

          result(start, end, func(graph), search, method);

     }
```

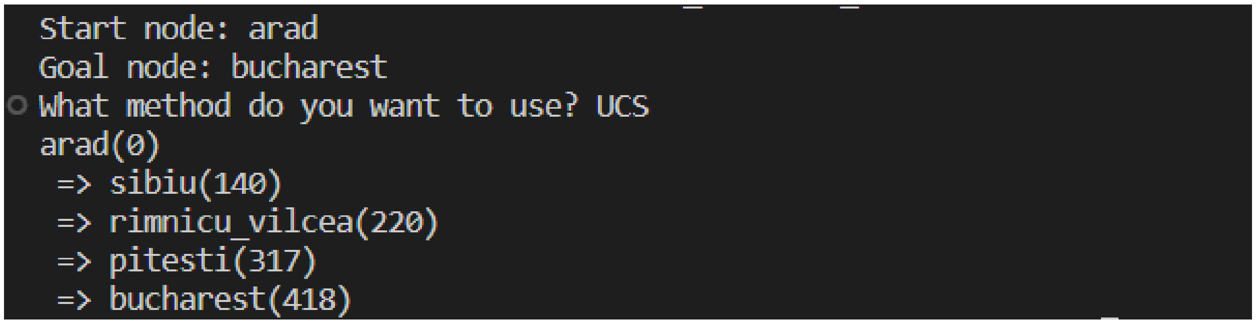
3. EKSPERIMEN

3.1 TestCase 1



```
var testGraph = [
    ["oradea"         , "zerind"         , 71],
    ["oradea"         , "sibiu"          , 151],
    ["zerind"         , "arad"           , 75],
    ["arad"           , "sibiu"          , 140],
    ["arad"           , "timisoara"      , 118],
    ["timisoara"      , "lugoj"          , 111],
    ["lugoj"          , "mehadia"        , 70],
    ["mehadia"        , "drobeta"        , 75],
    ["drobeta"        , "craiova"        , 120],
    ["craiova"        , "pitesti"        , 138],
    ["craiova"        , "rimnicu vilcea" , 146],
    ["sibiu"          , "rimnicu vilcea" , 80],
    ["rimnicu_vilcea" , "pitesti"        , 97],
    ["sibiu"          , "fagaras"        , 99],
    ["fagaras"        , "bucharest"      , 211],
    ["pitesti"        , "bucharest"      , 101],
    ["bucharest"      , "giurgiu"        , 90],
    ["bucharest"      , "urziceni"       , 85],
    ["urziceni"       , "hirsova"        , 98],
    ["hirsova"        , "eforie"         , 86],
    ["urziceni"       , "vaslui"         , 142],
    ["vaslui"         , "iasi"           , 92],
    ["iasi"           , "neamt"          , 87]
];

var nameTest = ["oradea", "zerind", "sibiu", "arad", "timisoara", "lugoj", "mehadia", "drobeta",
                "craiova", "pitesti", "rimnicu vilcea", "fagaras", "bucharest", "giurgiu", "urziceni",
                "hirsova", "eforie", "vaslui", "iasi", "neamt"];

var eucToGoalTest = {"arad": 366, "bucharest": 0, "craiova": 160, "dobreta": 242, "eforie": 161,
                    "fagaras": 176, "giurgiu": 77, "hirsova": 151, "iasi": 226, "lugoj": 244, "mehadia": 241,
                    "neamt": 234, "oradea": 380, "pitesti": 10, "rimnicu vilcea": 193, "sibiu": 253, "timisoara": 329,
                    "urziceni": 80, "vaslui": 199, "zerind": 374};
```

Data sampel 1



```
Start node: arad
Goal node: bucharest
What method do you want to use? UCS
arad(0)
 => sibiu(140)
 => rimnicu_vilcea(220)
 => pitesti(317)
 => bucharest(418)
```

4. KESIMPULAN

Terdapat banyak cara untuk dapat menyelesaikan masalah pencarian arah dan rute, pada kehidupan sehari-hari dan salah satu caranya adalah dengan menggunakan algoritma UCS dan A*. Algoritma UCS dan A* memberikan hasil yang optimal jika dibandingkan algoritma pencarian lainnya karena terdapat perhitungan heuristic yang membuat kalkulasi pengambilan keputusan lebih akurat dan efisien.

5. LAMPIRAN

Link Github: https://github.com/Ezaaan/Tucil3_13521091_13521141

		Centang (√) jika ya
1	Program dapat menerima input graf	
2	Program dapat menghitung lintasan terpendek dengan UCS	✓
3	Program dapat menghitung lintasan terpendek dengan A*	✓
4	Program dapat menampilkan lintasan terpendek serta jaraknya	✓
5	Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta serta lintasan terpendek pada peta	✓