Laporan Tugas Kecil 3 IF2211 Strategi Algoritma

Semester II Tahun 2022/2023

Implementasi Algoritma UCS dan A* untuk Menentukan Lintasan Terpendek

Disusun Oleh :

Michael Utama (13521137)

Johanes Lee (13521148)

Kelas 2

Program Studi S1 Teknik Informatika

Sekolah Teknik Elektro dan Informatika

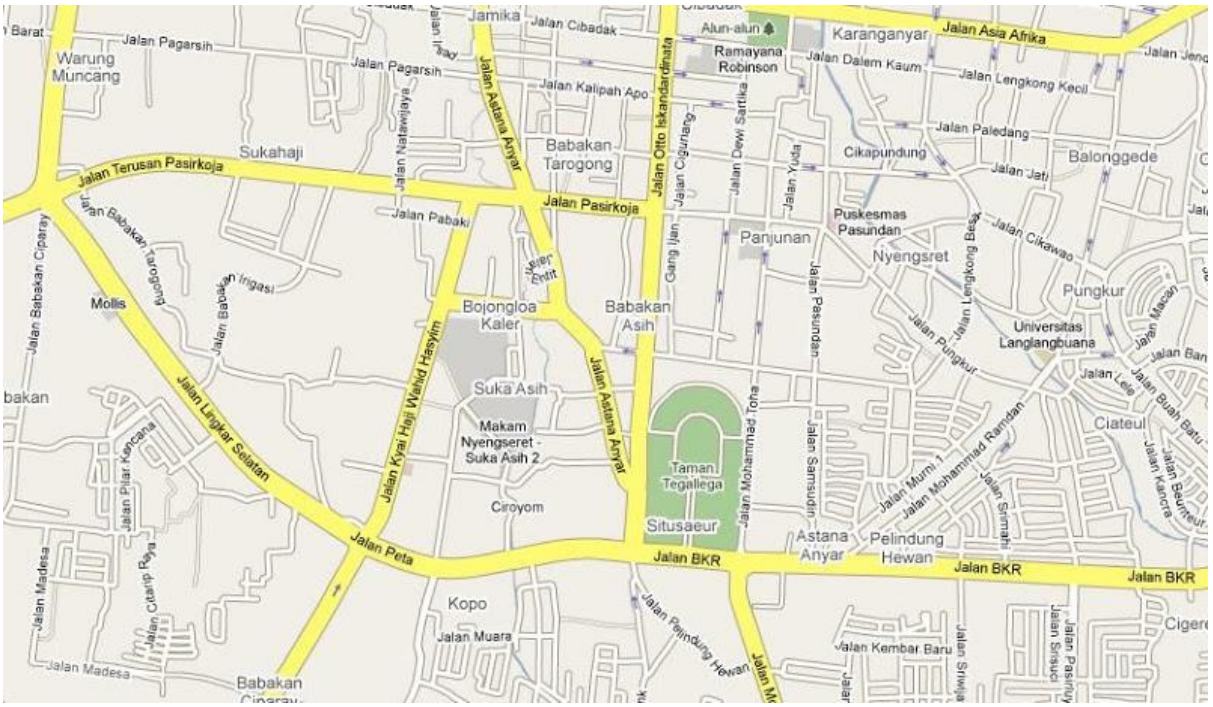
Institut Teknologi Bandung

2023

Bab I

Deskripsi Persoalan

Algoritma UCS (Uniform cost search) dan A* (atau A star) dapat digunakan untuk menentukan lintasan terpendek dari suatu titik ke titik lain. Pada tugas kecil 3 ini, anda diminta menentukan lintasan terpendek berdasarkan peta Google Map jalan-jalan di kota Bandung. Dari ruas-ruas jalan di peta dibentuk graf. Simpul menyatakan persilangan jalan (simpang 3, 4 atau 5) atau ujung jalan. Asumsikan jalan dapat dilalui dari dua arah. Bobot graf menyatakan jarak (m atau km) antar simpul. Jarak antar dua simpul dapat dihitung dari koordinat kedua simpul menggunakan rumus jarak Euclidean (berdasarkan koordinat) atau dapat menggunakan ruler di Google Map, atau cara lainnya yang disediakan oleh Google Map.



Langkah pertama di dalam program ini adalah membuat graf yang merepresentasikan peta (di area tertentu, misalnya di sekitar Bandung Utara/Dago). Berdasarkan graf yang dibentuk, lalu program menerima input simpul asal dan simpul tujuan, lalu menentukan lintasan terpendek antara keduanya menggunakan algoritma UCS dan A*. Lintasan terpendek dapat ditampilkan pada peta/graf (misalnya jalan-jalan yang menyatakan lintasan terpendek diberi warna merah). Nilai heuristik yang dipakai adalah jarak garis lurus dari suatu titik ke tujuan.

Bab II

Implementasi Program

Berikut adalah implementasi program untuk mencari solusi lintasan terpendek dari suatu titik ke titik lain dalam suatu graf menggunakan algoritma UCS dan A*.

2.1PathFinding.js

```js
const UniformCostSearch = (adjMatrix, source, dest) => {
    source = parseInt(source);
    dest = parseInt(dest);

    var pq = new PriorityQueue({compareFunction: (a, b) => b[0] - a[0]});
    var vis = new Array(adjMatrix.length).fill(false);
    var prev = new Array(adjMatrix.length).fill(-1);

    // Insert source node
    // Tuple definition: [f(n), n, previous_node]
    pq.push([0, source, -2]);
    var finalDistance;

    while (!pq.isEmpty()) {
        var top = pq.top();
        pq.pop();
        if (vis[top[1]]) continue;
        vis[top[1]] = true;
        prev[top[1]] = top[2];

        if (top[1] === dest) {
            finalDistance = top[0];
            break;
        }

        for (var i = 0; i < adjMatrix[top[1]].length; i++) {
            if (vis[i] || adjMatrix[top[1]][i].toUpperCase() === 'X') continue;
            pq.push([top[0] + parseInt(adjMatrix[top[1]][i]), i, top[1]]);
        }
    }

    if (prev[dest] === -1)
        return {
            found: false,
            solution: [],
            distance: -1,
        }

    // Backtrack
    var path = [];
    var pos = dest;
    while(pos >= 0) {
        path.push(pos);
        pos = prev[pos];
    }
    path.reverse();

    return {
        found: true,
        solution: path,
        distance: finalDistance,
    }
}
```

```
const AyStar = (adjMatrix, source, dest, distOption) => {
    if (distOption.length > 1) {
        throw Error("Cannot use more than one option!");
    }
    source = parseInt(source);
    dest = parseInt(dest);

    var distanceToDest;
    if (distOption['distanceToDest']) {
        // copy distanceToDest
        distanceToDest = [...distOption['distanceToDest']];
    }
    else {
        throw Error("Fill distOption with either 'coordinates' or 'distanceToDest'");
    }

    // calculate distance with A*
    var pq = new PriorityQueue({compareFunction: (a, b) => b[0] - a[0]});
    var vis = new Array(adjMatrix.length).fill(false);
    var prev = new Array(adjMatrix.length).fill(-1);

    // Insert source node
    // Tuple definition: [f(n), n, previous_node, g(n)]
    pq.push([distanceToDest[source], source, -2, 0]);
    var finalDistance;

    while (!pq.isEmpty()) {
        var top = pq.top();
        pq.pop();
        if (vis[top[1]]) continue;
        vis[top[1]] = true;
        prev[top[1]] = top[2];

        if (top[1] === dest) {
            finalDistance = top[0];
            break;
        }

        for (var i = 0; i < adjMatrix[top[1]].length; i++) {
            if (vis[i] || adjMatrix[top[1]][i].toUpperCase() === 'X') continue;
            pq.push([top[3] + parseInt(adjMatrix[top[1]][i]) + distanceToDest[i],
                    i,
                    top[1],
                    top[3] + parseInt(adjMatrix[top[1]][i])]);
        }
    }

    if (prev[dest] === -1)
        return {
            found: false,
            solution: [],
            distance: -1,
        }

    // Backtrack
    var path = [];
    var pos = dest;
    while(pos >= 0) {
        path.push(pos);
        pos = prev[pos];
    }
    path.reverse();

    console.log(distanceToDest);
    return {
        found: true,
        solution: path,
        distance: finalDistance,
    }

}
```

2.2 PriorityQueue.js

```
class PriorityQueue {
    // Representasi data dengan heap
    // contoh fungsi compareFunction:
    //      (a, b) => a - b         berarti PriorityQueue menggunakan max heap
    //                              dengan priority = value
    //      (a, b) => b - a         berarti PriorityQueue menggunakan min heap
    //                              dengan priority = value
    //      (a, b) => a[0] - b[0]   berarti PriorityQueue menggunakan max heap
    //                              dengan priority = value[0]
    constructor(options = {}) {
        this.heap = options.heap || [];
        this.compareFunction = options.compareFunction || defaults.compareFunction;
    }

    push(element) {
        var index = this.heap.length;
        this.heap.push(element);

        while(index) {
            var par = this.#getParent(index);
            if (this.compareFunction(this.heap[index], this.heap[par]) <= 0) break;
            [this.heap[index], this.heap[par]] = [this.heap[par], this.heap[index]];
            index = par;
        }
    }

    top() {
        return this.heap[0];
    }

    pop() {
        [this.heap[0], this.heap[this.heap.length-1]] =
            [this.heap[this.heap.length-1], this.heap[0]];
        this.heap.pop();
        this.#heapify(0);
    }

    isEmpty() {
        return this.heap.length === 0;
    }

    #getParent(index) {
        return Math.floor((index - 1) / 2);
    }

    #getLeftChild(index) {
        return index * 2 + 1;
    }

    #getRightChild(index) {
        return index * 2 + 2;
    }

    #heapify(index) {
        if (index >= this.heap.length) return;

        var maxIdx = index;
        if (this.#getLeftChild(index) < this.heap.length &&
            this.compareFunction(this.heap[this.#getLeftChild(index)], this.heap[maxIdx]) > 0 ) {
            maxIdx = this.#getLeftChild(index);
        }
        if (this.#getRightChild(index) < this.heap.length &&
            this.compareFunction(this.heap[this.#getRightChild(index)], this.heap[maxIdx]) > 0 ) {
            maxIdx = this.#getRightChild(index);
        }

        [this.heap[index], this.heap[maxIdx]] = [this.heap[maxIdx], this.heap[index]];
        if (maxIdx !== index) {
            this.#heapify(maxIdx);
        }
    }

}
```

2.3 ConfigMap.js

```
import Graph from "react-graph-vis";
import { useState, useMemo, useEffect } from "react";
import { v4 as uuid } from 'uuid';


const ConfigMap = ({adjacencyMatrix, names, solution, isDirected}) => {

    const [windowWidth, setWidth] = useState(window.innerWidth);

    useEffect(() => {
        const updateWidth = () => {
          setWidth(window.innerWidth)
        }

        window.addEventListener('resize', updateWidth);

        return(() => {
            window.removeEventListener('resize', updateWidth);
        })

      }, [windowWidth])

    const nodeCount = adjacencyMatrix.length;

    const [graph, setGraph] = useState({
        nodes: [],
        edges: []
    })
```

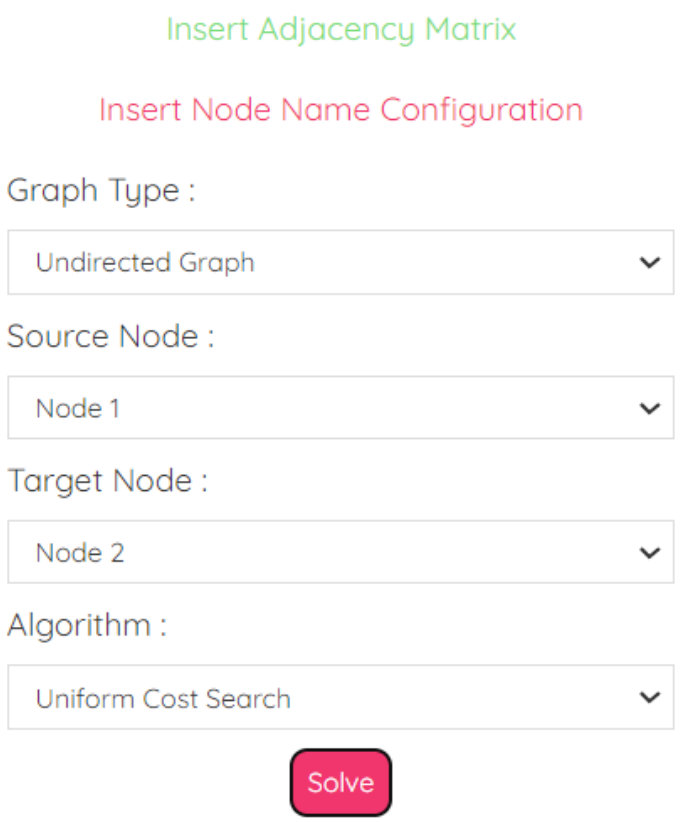
```
useEffect(() => {

    const font = {
        color: "#333",
        face: "Quicksand",
    }

    const tempGraph = {
        nodes: [],
        edges: []
    }

    for (var i = 0; i < nodeCount; i++)
    {
        var firstSolutionIndex = -1;
        var isSolutionPart = false;

        if (solution)
        {
            firstSolutionIndex = solution.indexOf(i);
            isSolutionPart = (firstSolutionIndex !== -1) && (firstSolutionIndex < solution.length);
        }

        tempGraph.nodes.push({
            id: i,
            label: names? names[i] : "Node " + (i+1),
            color: {
                background: 'white',
                border: "#f1356d",
                highlight: "#f1356d"
            },
            font: font,
            labelHighlightBold: false,
            shape: "circle",
        })
        for (var j = 0; j < nodeCount; j++)
        {
            if (i === j) continue;
            if (!isDirected && j < i) continue;


            if (adjacencyMatrix[i][j].toUpperCase() !== "X")
            {
                const tempEdge = {
                    from: i,
                    to: j,
                    arrows: {
                        to: isDirected,
                    },
                    label: adjacencyMatrix[i][j].toString() + " m",
                    physics: false,
                    color: {
                        color: "#f1356d",
                        highlight: "#f1356d"
                    },
                    font: font,
                    labelHighlightBold: false,
                    selectionWidth: 0,
                    smooth: {enabled: isDirected? true : false,  type: 'curvedCW', roundness: 0.2}
                }

                var secondSolutionIndex = -1;
                var isSolutionPartFull = false;

                if (solution)
                {
                    secondSolutionIndex = solution.indexOf(j);

                    isSolutionPartFull = isSolutionPart && (secondSolutionIndex !== -1 && secondSolutionIndex < solution.length);
                }


                if (isSolutionPartFull)
                {
                    if (solution[firstSolutionIndex+1] === j)
                    {
                        tempEdge.arrows.to = true;
                        tempEdge.color.color = "#16a52d";
                    }

                    else if (!isDirected && solution[secondSolutionIndex+1] === i)
                    {
                        tempEdge.arrows.from = true;
                        tempEdge.color.color = "#16a52d";
                    }
                }
                tempGraph.edges.push(tempEdge)
            }
        }

        setGraph(tempGraph)
    }
}, [adjacencyMatrix, nodeCount, names, solution, windowWidth, isDirected])

```

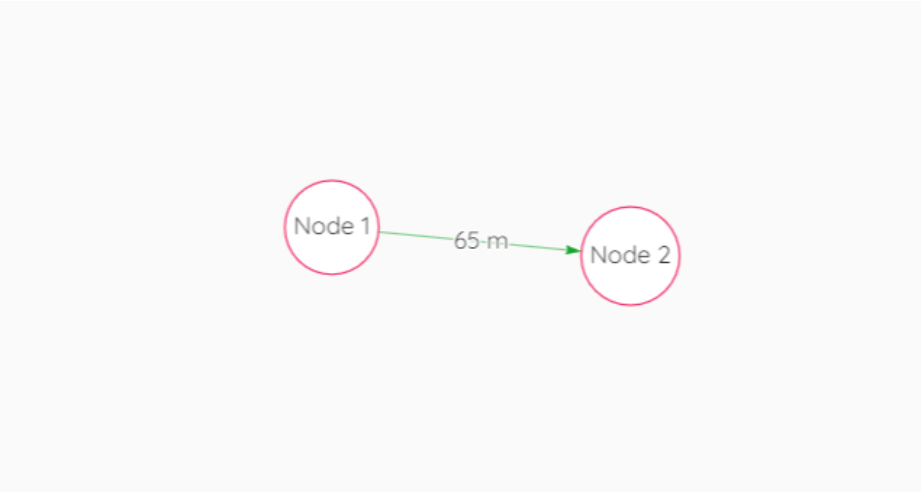
BAB III

Eksperimen

Berikut adalah hasil percobaan beberapa kasus uji terhadap program yang telah dibuat. (bab 3.1 dan 3.2 saling berhubungan)

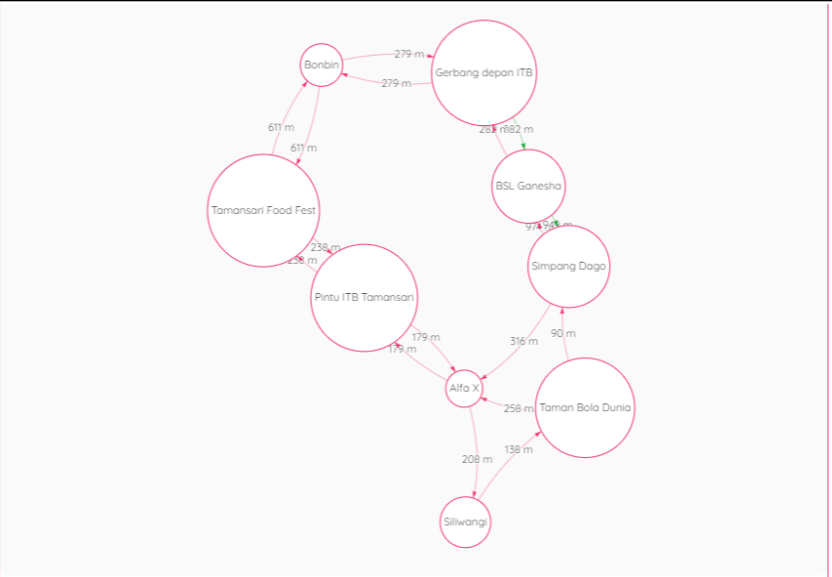
3.1 File Configuration App

No.	*Input/ Output*	Gambar
1	*Input Matrix*	0 65 65 0
	Input Node Names	–
	Input Coordinates	–
	Other Inputs	

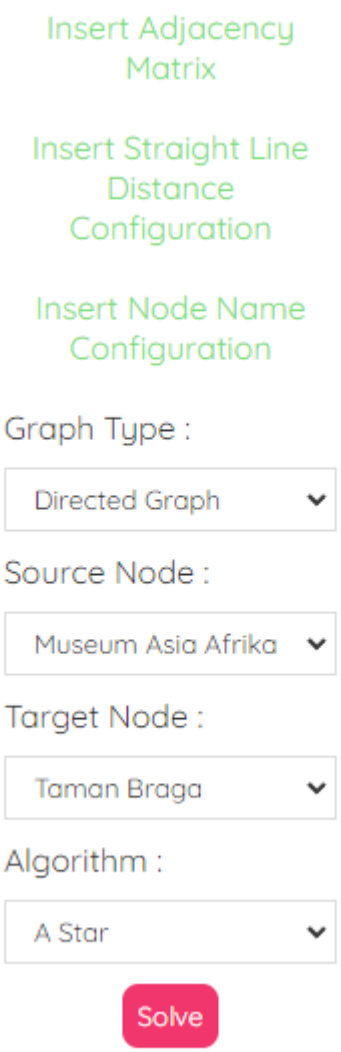
	Output	
2	*Input Matrix*	0 279 X X X X X X 282 279 0 X 611 X X X X X X X 0 238 179 X X X X X 611 238 0 X X X X X X X 179 X 0 X 208 X X X X X X 316 0 X X 974 X X X X X X 0 138 X X X X X 258 90 X 0 X 282 X X X X 949 X X 0
	Input Node Names	Gerbang depan ITB Bonbin Pintu ITB Tamansari Tamansari Food Fest Alfa X Simpang Dago Siliwangi Taman Bola Dunia BSL Ganesha
	Input Coordinates	-6.893182905867662 107.61044267557261 -6.893353326381835 107.608350552541 -6.887679457495552 107.61034611604808 -6.887920240847769 107.60825386315514 -6.8867791762278205 107.61155999711362 -6.885197698469967 107.61366991359704 -6.884904783135887 107.61160461265557 -6.885075206624985 107.61283306438439 -6.893710629359909 107.61292591112024

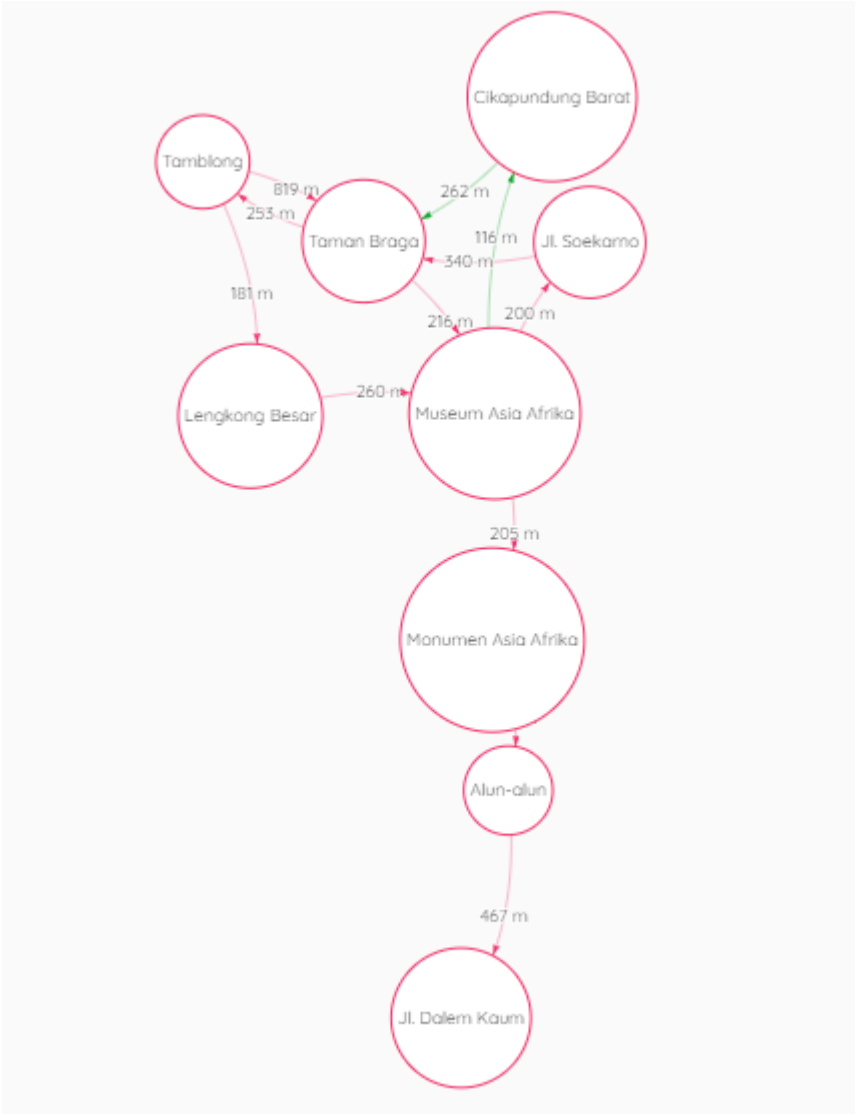
<table>
<tr><td></td><td>Other Inputs</td><td>Insert Adjacency Matrix

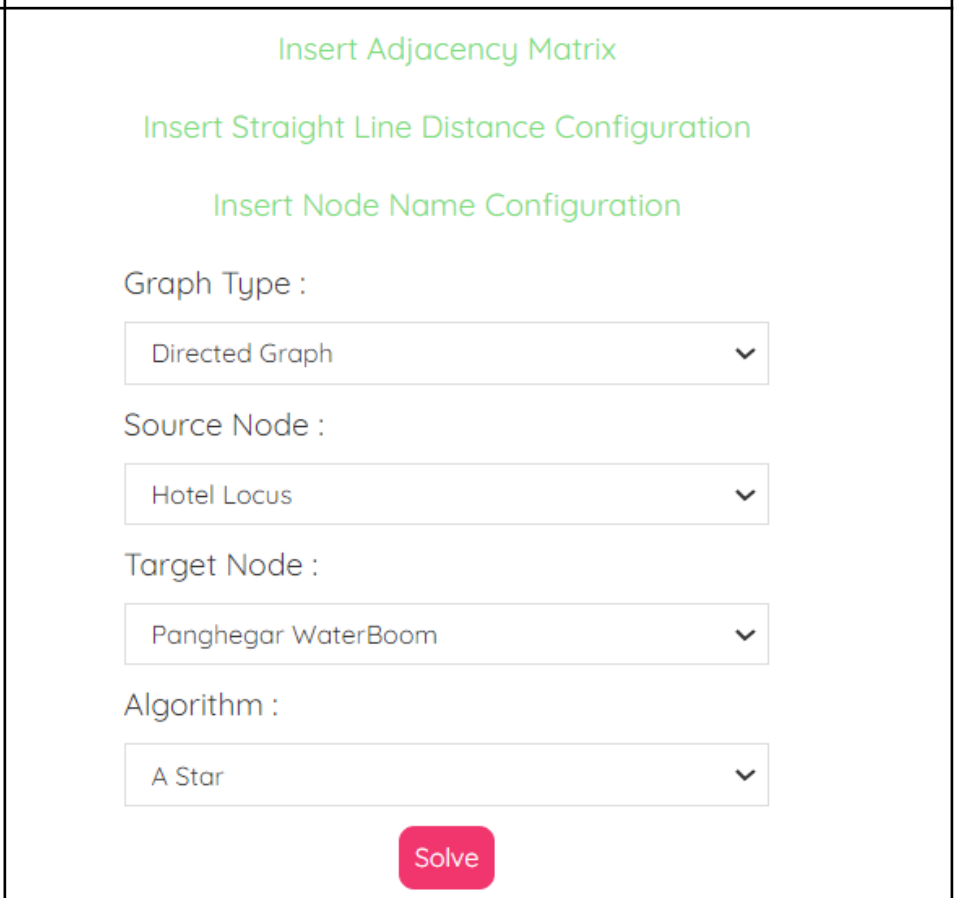
Insert Node Name Configuration

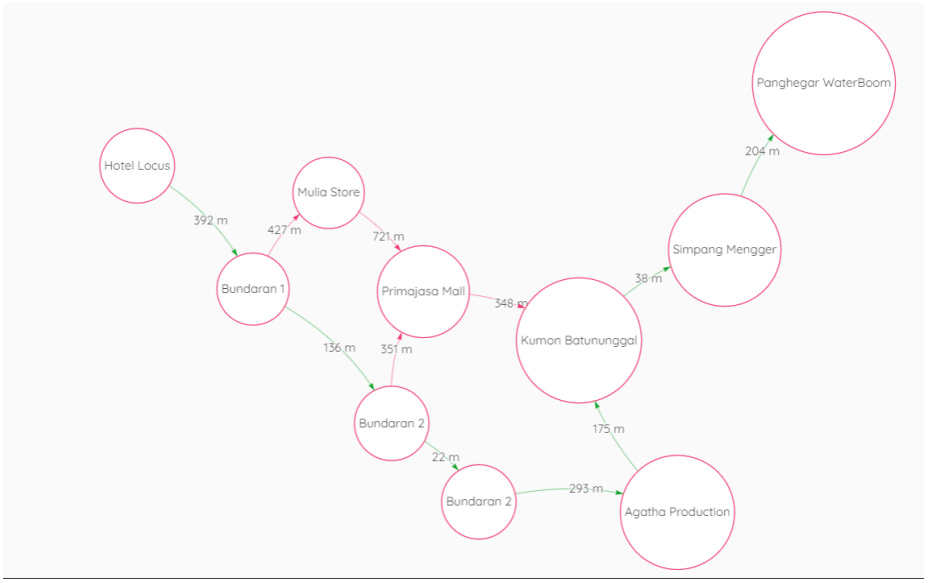
Graph Type :
Directed Graph
Source Node :
Gerbang depan ITB
Target Node :
Simpang Dago
Algorithm :
Uniform Cost Search
Solve</td></tr>
<tr><td></td><td>Output</td><td> 
</td></tr>
<tr><td>3</td><td>Input Matrix</td><td>0 200 116 X X X 205 X X
X 0 X 340 X X X X X
X X 0 262 X X X X X
216 X X 0 X 253 X X X
260 X X X 0 X X X X
X X X 819 181 0 X X X
X X X X X X 0 147 X
X X X X X X X 0 467
X X X X X X X X 0</td></tr>
</table>

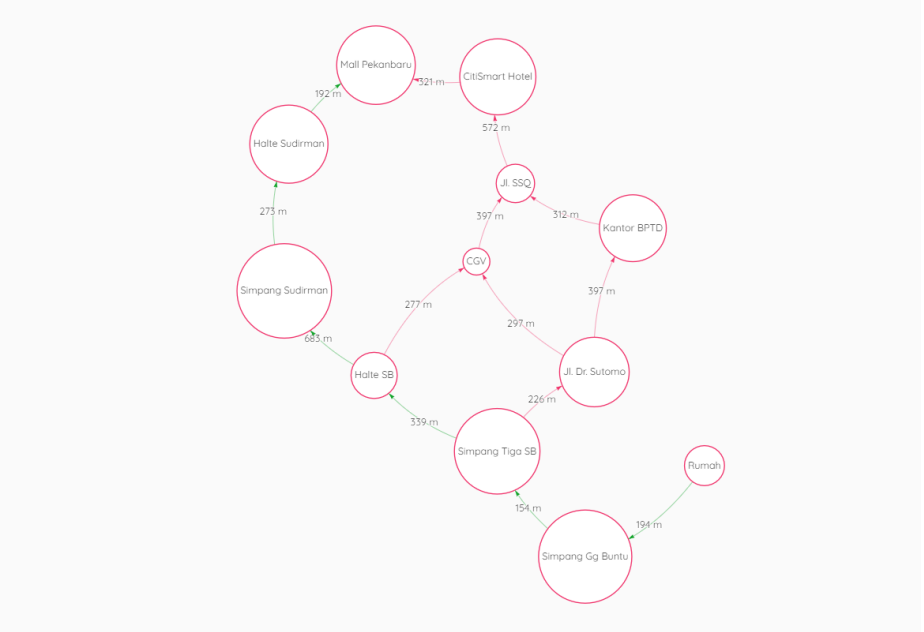
	Input Node Names	Museum Asia Afrika Jl. Soekarno Cikapundung Barat Taman Braga Lengkong Besar Tamblong Monumen Asia Afrika Alun-alun Jl. Dalem Kaum
	Input Coordinates	-6.921411330438687 107.60959834669556 -6.921034959350296 107.60804868081522 -6.921141465906744 107.60879969933939 -6.919729496854814 107.60992300161459 -6.921732041429313 107.61193206349856 -6.920134012931102 107.61217591049912 -6.9212351165169235 107.60775508320624 -6.922545144935525 107.60764779484565 -6.9230066074838374 107.6118528837299

	Other Inputs	

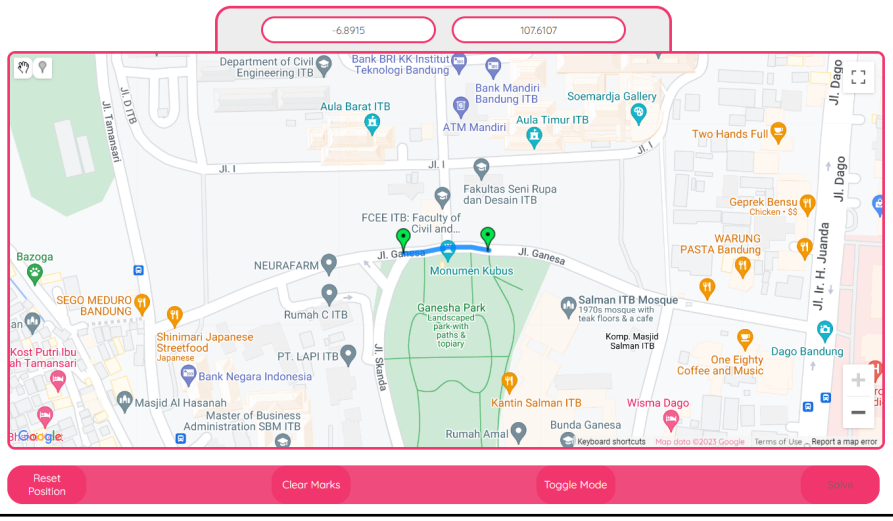
	Output	
4	*Input Matrix*	0 204 X X X X X X X X X 0 X X X X X X X X X X 0 X X X X X X 348 X X 721 0 X X X X X X X X X 427 0 X 136 X X X X X X X 392 0 X X X X X X 351 X X X 0 X 22 X X X X X X X X 0 X 175 X X X X X X X 293 0 X 38 X X X X X X X X 0
	Input Node Names	Simpang Mengger Panghegar WaterBoom Primajasa Mall Mulia Store Bundaran 1 Hotel Locus Bundaran 2 Agatha Production Bundaran 2 Kumon Batununggal

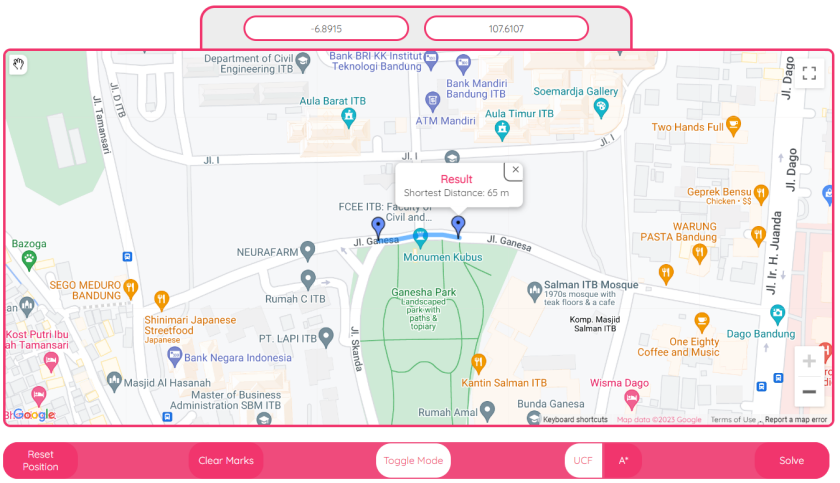
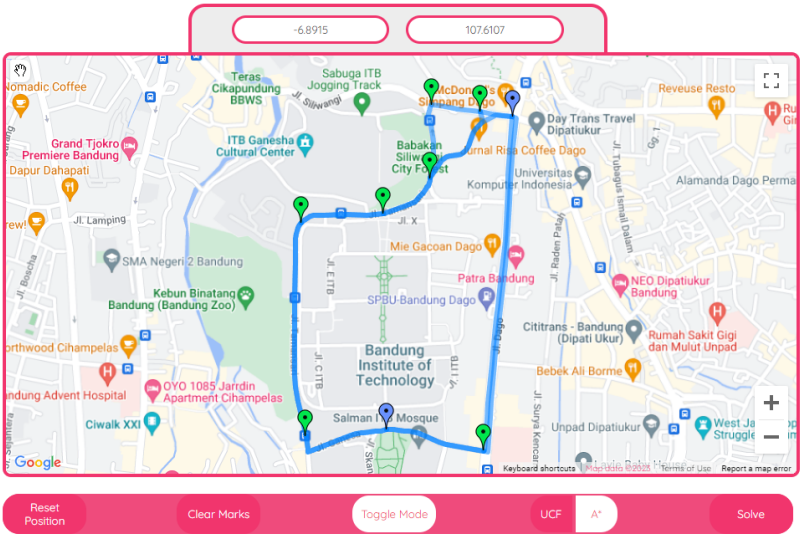
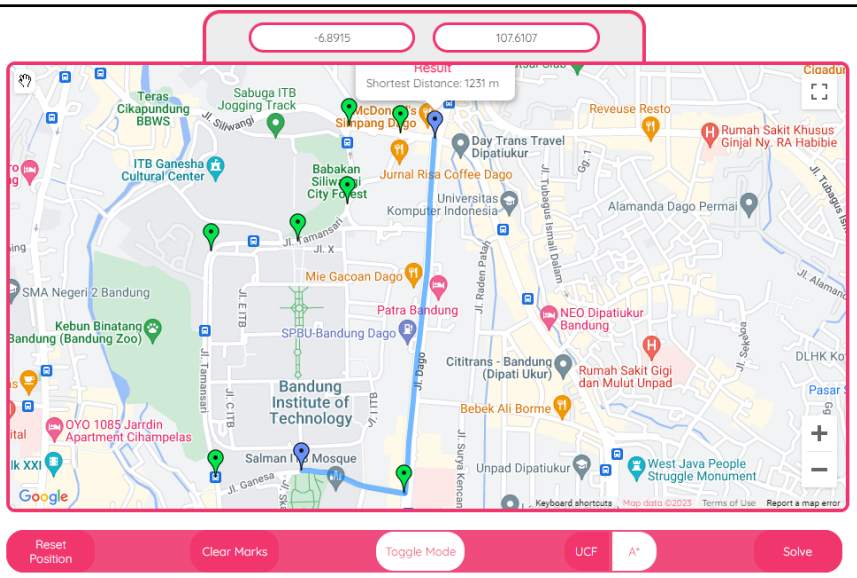
	Input Coordinates	-6.959456577059039 107.62309571476733 -6.961267037441487 107.62306352825915 -6.961810174194228 107.62565990658557 -6.962906338851718 107.62694736691272 -6.9591043772627845 107.627719843109 -6.955742581299274 107.62844582615632 -6.958886129309951 107.6264477876394 -6.9582471715212595 107.62366746042615 -6.958832910977207 107.62627993200665 -6.959644661871995 107.62338864804913
	Other Inputs	

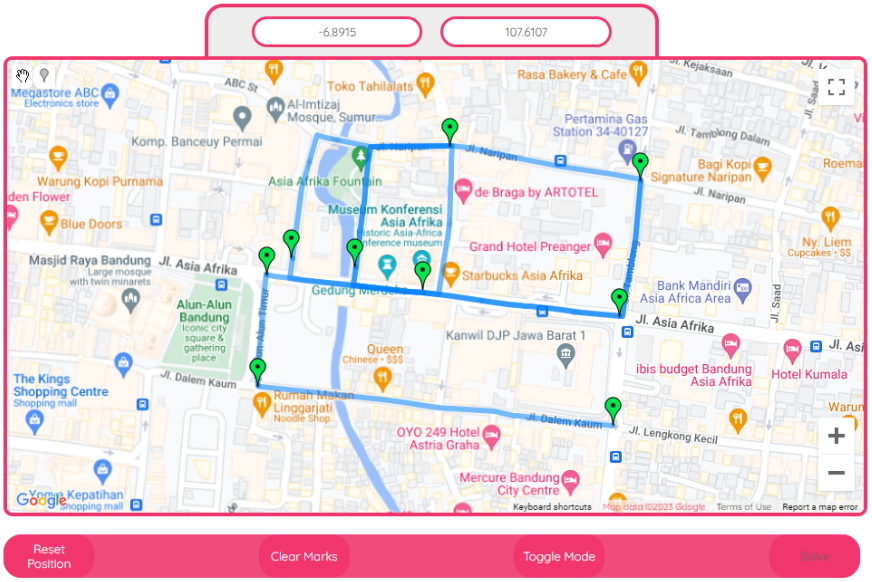
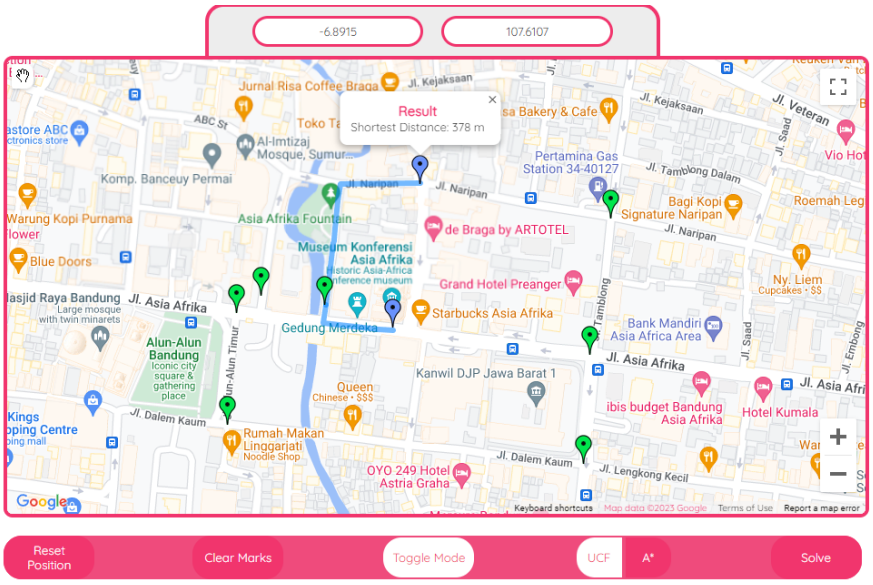
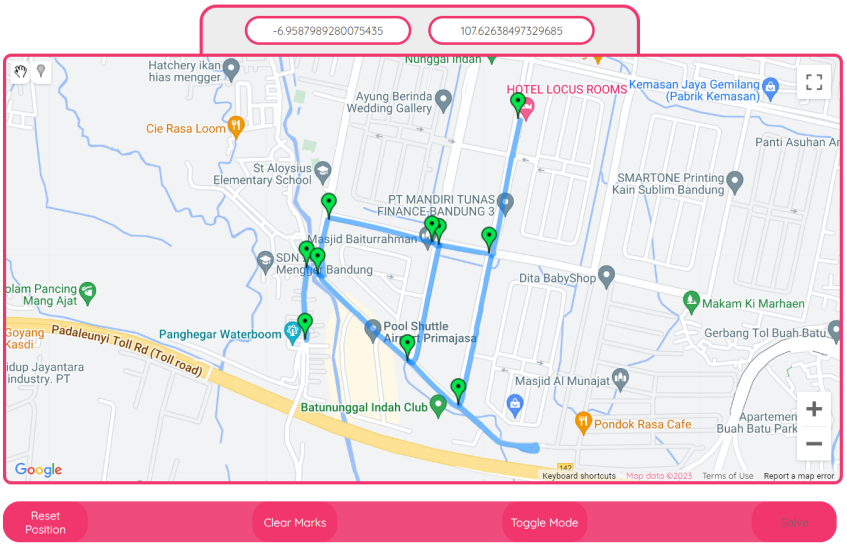
	Output	
5	*Input Matrix*	0 194 X X X X X X X X X X 0 154 X X X X X X X X X X 0 X 339 226 X X X X X X X X X 0 X X X X X X X X X X X 0 X 277 X 683 X X X X X X X X 0 297 397 X X X X X X X X X X 0 X X X X 397 X X X X X X X 0 X X X 312 X X X X X X X X 0 273 X X X X X 192 X X X X X 0 X X X X X 321 X X X X X X 0 X X X X X X X X X X 572 0
	Input Node Names	Rumah Simpang Gg Buntu Simpang Tiga SB Mall Pekanbaru Halte SB Jl. Dr. Sutomo CGV Kantor BPTD Simpang Sudirman Halte Sudirman CitiSmart Hotel Jl. SSQ
	Input Coordinates	-

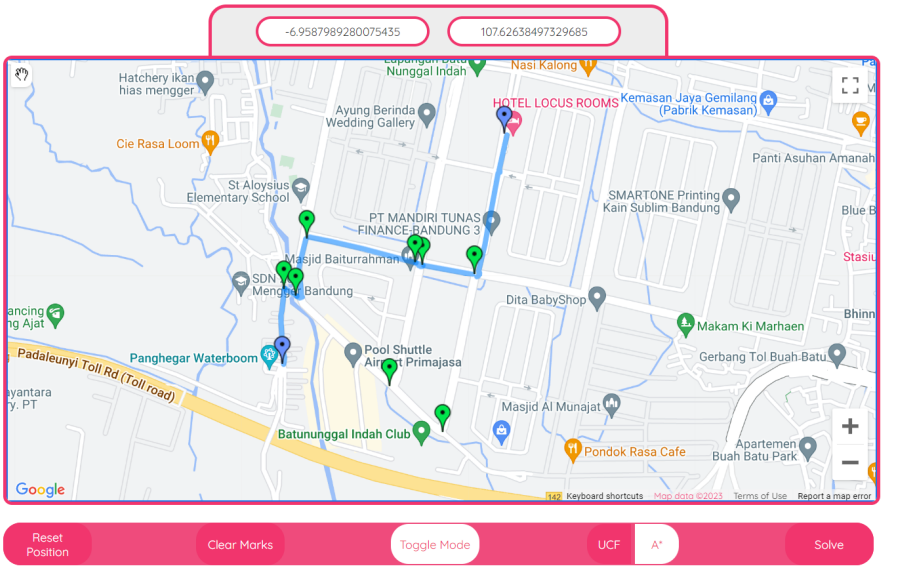
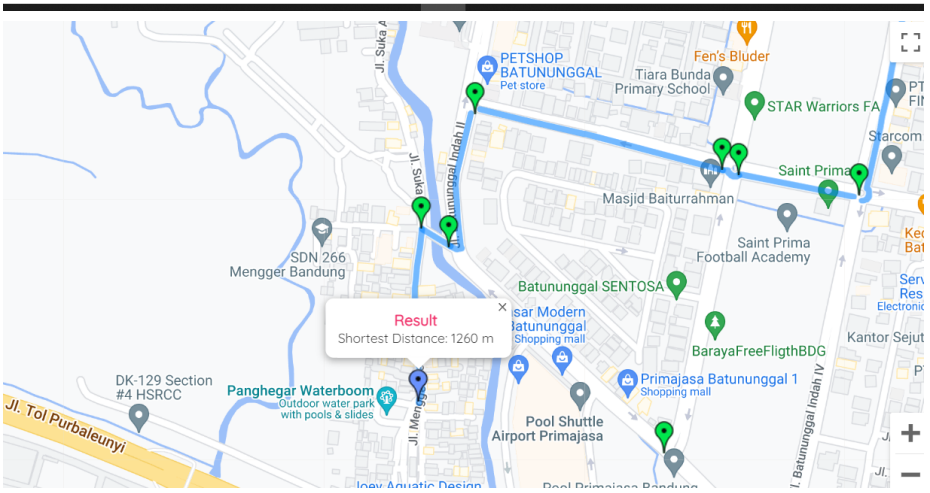
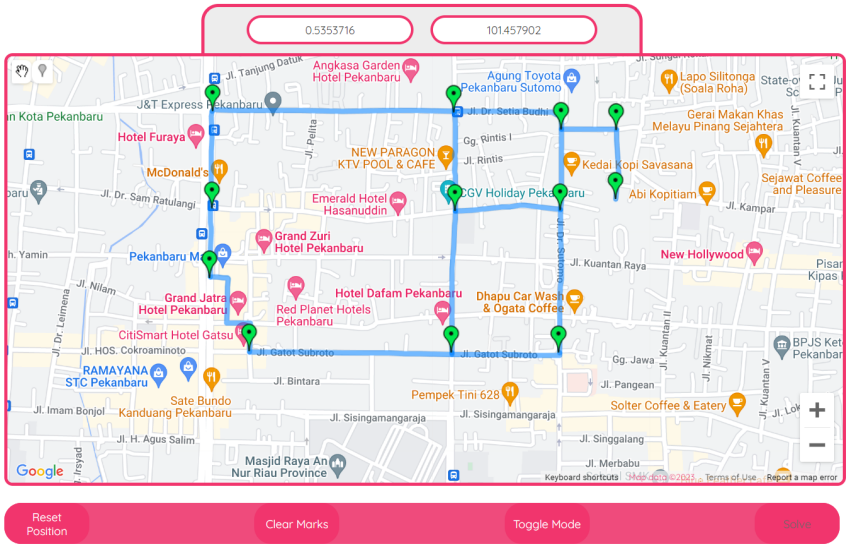
	Other Inputs	Insert Adjacency Matrix Insert Node Name Configuration Graph Type : Directed Graph Source Node : Rumah Target Node : Mall Pekanbaru Algorithm : Uniform Cost Search Solve
	Output	

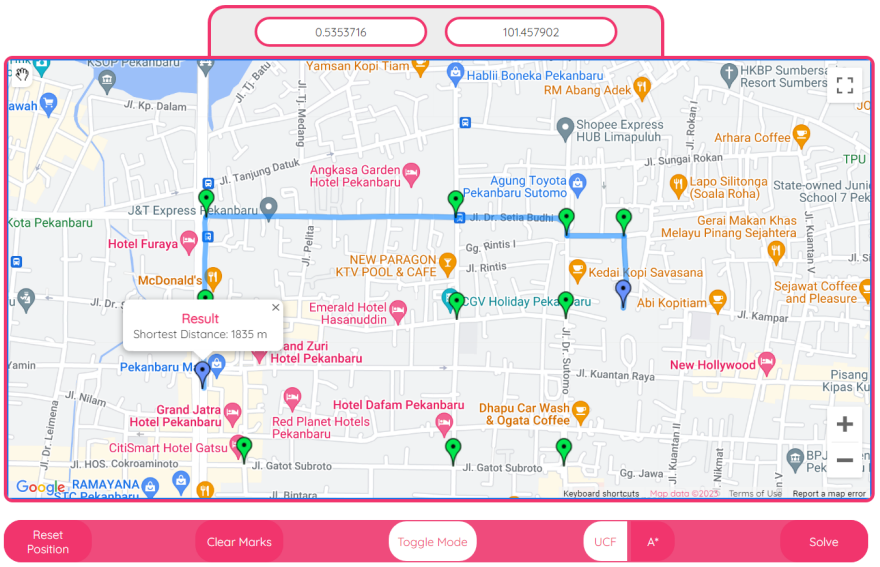
3.1 Google Map App

No.	*Input/ Output*	Gambar
1	*Input Map*	

	Output	
2	*Input Map*	
	Output	

3	*Input Map*	
	Output	
4	*Input Map*	

	Output	 
5	*Input Map*	

	Output	 0.5353716 101.457902 Result Shortest Distance: 1835 m Reset Position Clear Marks Toggle Mode UCF A* Solve

BAB IV

Penutup

4.1 Kesimpulan

Penentuan lintasan terpendek pada peta dapat dilakukan oleh algoritma *uniform cost search* (UCS) dan A*. Penentuan yang dilakukan algoritma UCS merupakan *blind search* karena program tidak mengetahui sejauh apa simpul sekarang dengan tujuan, sedangkan A* melakukan *informed search* dengan informasi jarak euclidean dari simpul tujuan. Algoritma UCS pasti mendapatkan hasil optimal (rute dengan jarak terpendek), sedangkan A* dipastikan mendapatkan hasil optimal hanya jika jarak euclidean yang diberikan *valid* atau *admissible*.

4.2 Komentar

Tugas besar ini telah menjadi sarana kelompok meningkatkan wawasan tentang algoritma UCS dan A* serta keterampilan dan kreativitas dalam pembangunan perangkat lunak akibat kebebasan penggunaan *framework* oleh kelompok. Selain itu, Spesifikasi bonus membuat kelompok memiliki kesempatan untuk melakukan eksplorasi penggunaan Google Maps API. Dengan eksplorasi tersebut, kelompok menjadi lebih paham tentang berbagai layanan Google Maps API serta pemanfaatannya dalam pembangunan perangkat lunak.

Lampiran

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	✓

Link repository Github : https://github.com/Enliven26/Tucil3_13521137_13521148