

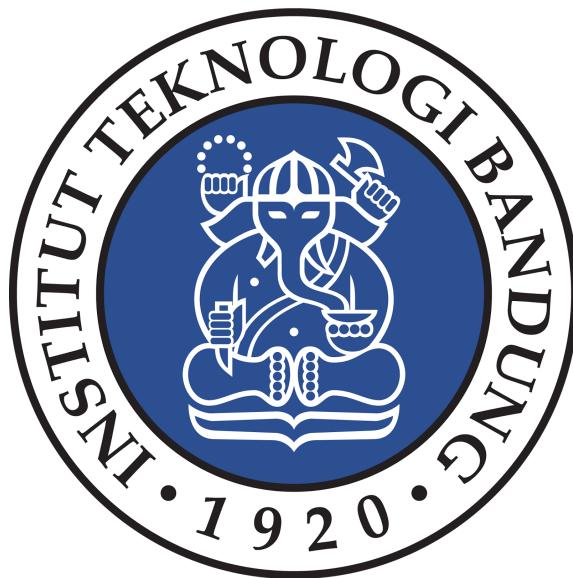
Tugas Kecil 3 IF2211 Strategi Algoritma

**Implementasi Algoritma UCS dan A* untuk Menentukan Lintasan
Terpendek**

Disusun oleh:

13521077 – Husnia Munzayana

13521156 – Brigita Tri Carolina



INSTITUT TEKNOLOGI BANDUNG
TEKNIK INFORMATIKA
2023

DAFTAR ISI

DAFTAR ISI.....	2
BAB 1.....	4
1.1 Deskripsi Masalah.....	4
1.2 Algoritma UCS.....	5
1.3 Algoritma A*.....	6
1.4 Algoritma Penyelesaian Menentukan Lintasan Terpendek dengan Pendekatan UCS.....	6
1.5 Algoritma Penyelesaian Menentukan Lintasan Terpendek dengan Pendekatan A*.....	7
BAB 2.....	10
2.1 Console.....	10
2.1.1 mainProgram.js.....	10
2.1.2 Astar.js.....	10
2.1.3 UCS.js.....	11
2.1.4 input.js.....	11
2.1.5 output.js.....	11
2.1.6 operation.js.....	12
2.1.7 PriorityQueue.js.....	12
2.2 Web.....	12
2.2.1 app.js.....	12
2.2.2 Astar.js.....	12
2.2.3 UCS.js.....	13
2.2.4 input.js.....	13
2.2.5 PriorityQueue.js.....	14
2.2.6 map.js.....	14
2.2.7 index.html.....	15
2.3 Ketentuan Format Input File.....	15
BAB 3.....	17
3.1 Console.....	17
3.1.1 mainProgram.js.....	17
3.1.2 Astar.js.....	19
3.1.3 UCS.js.....	20
3.1.4 input.js.....	21
3.1.5 output.js.....	23
3.1.6 operation.js.....	24
3.1.7 PriorityQueue.js.....	24
3.2 Web.....	25
3.2.1 app.js.....	25
3.2.2 Astar.js.....	26

3.2.3 UCS.js.....	28
3.2.4 input.js.....	29
3.2.5 PriorityQueue.js.....	32
3.2.6 map.js.....	33
3.2.7 index.html.....	37
BAB 4.....	42
4.1 Masukkan dan Keluaran Program.....	42
4.2 Analisis Hasil Keluaran Program.....	45
BAB 5.....	47
5.1. Kesimpulan.....	47
5.2. Saran.....	47
REFERENSI.....	48
LAMPIRAN.....	49

BAB 1

DESKRIPSI MASALAH DAN ALGORITMA

1.1 Deskripsi Masalah

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.

Spesifikasi program:

- a. Program menerima input file graf (direpresentasikan sebagai matriks ketetanggaan berbobot), jumlah simpul minimal 8 buah.
- b. Program dapat menampilkan peta/graf
- c. Program menerima input simpul asal dan simpul tujuan.
- d. Program dapat menampilkan lintasan terpendek beserta jaraknya antara simpul asal dan simpul tujuan.
- e. Antarmuka program bebas, apakah pakai GUI atau command line saja.

Bonus: Bonus nilai diberikan jika dapat menggunakan Google Map API untuk menampilkan peta, membentuk graf dari peta, dan menampilkan lintasan terpendek di peta (berupa jalan yang diberi warna). Simpul graf diperoleh dari peta (menggunakan API Google Map) dengan mengklik ujung jalan atau persimpangan jalan, lalu jarak antara kedua simpul dihitung langsung dengan rumus Euclidean.

Berkas yang dikumpulkan adalah laporan format PDF yang berisi:

- a. Deskripsi persoalan
- b. Kode program
- c. Peta/graf input, output screenshoot peta yang memperlihatkan lintasan terpendek untuk sepasang simpul, tampilkan hasil untuk beberapa lintasan terpendek di kota Bandung atau kota lainnya yang kalian suka.
- d. Alamat github/google drive tempat kode sumber program diletakkan jika perlu dieksekusi oleh asisten.
- e. Kesimpulan, komentar, dll

Peta jalan yang digunakan sebagai kasus uji adalah:

- a. Peta jalan sekitar kampus ITB/Dago/Bandung Utara
- b. Peta jalan sekitar Alun-alun Bandung
- c. Peta jalan sekitar Buahbatu atau Bandung Selatan
- d. Peta jalan sebuah kawasan di kota asalmu

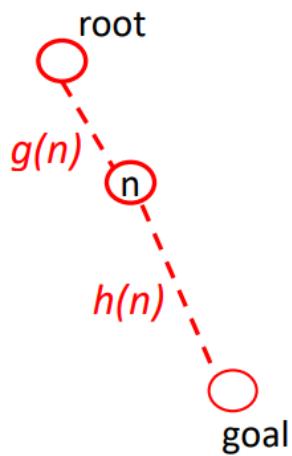
1.2 Algoritma UCS

Uniform Cost Search adalah algoritma pencarian untuk menentukan lintasan paling optimum dari sebuah graf. Algoritma pencarian ini termasuk dalam algoritma uninformed search atau disebut juga blind search dimana pencarian goal state dilakukan tanpa adanya informasi apakah state yang saat ini dikunjungi sudah mendekati goal. Proses penelusuran dengan algoritma UCS dilakukan dengan memprioritaskan penelusuran pada jalur dengan biaya(cost) paling minimum. Dalam penerapannya, digunakan Priority Queue untuk menentukan jalur paling minimum yang selanjutnya akan di ekspansi. Penentuan nilai priority pada setiap jalur yang telah diekspansi dilakukan dengan memanfaatkan fungsi evaluasi berikut:

$$g(n) = \text{biaya yang dibutuhkan dari simpul akar menuju simpul } n$$

Dengan biaya atau cost yang dimaksud dapat berupa jarak dari simpul akar menuju simpul n.

1.3 Algoritma A*



Gambar 1.3.1 Algoritma A*

Algoritma A* (baca: A star) merupakan algoritma yang umum digunakan dalam bidang informatika untuk menentukan lintasan terpendek antara dua simpul. Pencarian goal node dengan algoritma ini termasuk dalam informed search dimana informasi seberapa dekat node yang sedang di ekspansi ke simpul tujuan telah diketahui sebelumnya. Algoritma A* adalah salah satu algoritma terbaik dalam menentukan lintasan terpendek, karena dalam proses pencarian *goal* algoritma A* menggunakan informasi heuristik, yaitu pemecahan masalah berbasis pendekatan yang *logical*. Ide dari algoritma A* adalah untuk menghindari jalur dengan *cost* yang mahal. Fungsi evaluasi dari algoritma A* adalah sebagai berikut.

$$f(n) = g(n) + h(n)$$

$g(n)$ → biaya untuk mencapai n

$h(n)$ → estimasi biaya dari n menuju *goal*

$f(n)$ → estimasi biaya menuju *goal* dengan melewati n

1.4 Algoritma Penyelesaian Menentukan Lintasan Terpendek dengan Pendekatan UCS

Penentuan lintasan terpendek dapat dilakukan dengan pendekatan UCS, dengan prioritas *cost* berupa lintasan dengan jarak terpendek dari simpul akar. Penentuan lintasan terpendek tersebut dapat dirumuskan dalam langkah-langkah berikut:

- a. Tambahkan simpul akar ke priority queue dengan cost 0. Prioritas tertinggi priority queue adalah lintasan dengan cost paling rendah atau memiliki jarak terdekat dengan simpul akar.

- b. Ambil node terdepan yang memiliki cost terendah (prioritas tertinggi). Kita misalkan simpul yang sedang diekspansi tersebut sebagai currentNode.
- c. Keluarkan currentNode tersebut dari queue dan beri tanda bahwa node sudah pernah diekspansi.
- d. Sisipkan semua simpul yang bertetangga dengan currentNode yang belum pernah di ekspansi ke dalam priority queue.
- e. Ulangi langkah b-d hingga ditemukan currentNode adalah goal node.
- f. Jika goal node berhasil ditemukan, maka penelusuran berhasil. Namun, jika goal node tidak ditemukan dan tidak ada lagi simpul yang dapat di ekspansi (priority queue kosong) maka goal node gagal ditelusuri.

Secara umum, berikut adalah penerapan algoritma UCS dalam pseudocode:

```

function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node  $\leftarrow$  a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier  $\leftarrow$  a priority queue ordered by PATH-COST, with node as the only element
  explored  $\leftarrow$  an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node  $\leftarrow$  POP(frontier) /* chooses the lowest-cost node in frontier */
    if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
    add node.STATE to explored
    for each action in problem.ACTIONS(node.STATE) do
      child  $\leftarrow$  CHILD-NODE(problem, node, action)
      if child.STATE is not in explored or frontier then
        frontier  $\leftarrow$  INSERT(child, frontier)
      else if child.STATE is in frontier with higher PATH-COST then
        replace that frontier node with child
  
```

Pseudocode UCS (sumber : Artificial Intelligence : a modern approach <https://aima.cs.berkeley.edu/>)

1.5 Algoritma Penyelesaian Menentukan Lintasan Terpendek dengan Pendekatan A*

Penentuan lintasan terpendek dapat dilakukan dengan pendekatan A*, dengan prioritas utama fungsi dihitung berdasarkan biaya yang dibutuhkan dari akar menuju suatu simpul, kemudian biaya estimasi (*straight line distance*) dari simpul tersebut menuju simpul tujuan. Berikut adalah langkah-langkah penyelesaian menggunakan algoritma A*:

- a. Inisialisasi suatu set yang akan menyimpan simpul yang telah di-expand, inisialisasi juga node awal berupa index awal dengan simpul parent-nya (dalam hal ini masih null karena simpul awal tidak punya *parent* / dianggap sebagai simpul akar), dan inisialisasi sebuah queue dengan isi node awal tadi.

- b. Inisialisasi nilai awal $g(n)$ dari node awal yaitu 0 dan inisialisasi nilai awal dari $f(n)$ berupa nilai heuristik dari node awal ke node tujuan. Nilai heuristik dihitung dengan menggunakan jarak *Euclidian* dari node sekarang ke node tujuan.
- c. Ambil node pertama dari *queue* yang telah di-*sort* berdasarkan nilai $f(n)$ kemudian periksa apakah *current node* merupakan simpul tujuan. Jika iya, maka program akan me-*return* dan menghasilkan jalur berdasarkan *parent-parent* dari simpul.
- d. Jika tidak, maka program akan memeriksa tetangga dari *current node*. Tetangga dari *current node* tidak akan dimasukan pada *queue* ketika simpul tetangga tersebut sudah pernah di-expand sebelumnya ataupun sudah dimasukan pada *queue*.
- e. Ketika menambahkan simpul tetangga pada *queue*, maka dicatat juga nilai $g(n)$ dan $f(n)$ dari simpul tetangga tersebut. Nilai $g(n)$ disimpan untuk membantu perhitungan $f(n)$.
- f. Ulangi langkah b-e hingga simpul yang sedang diperiksa merupakan simpul tujuan atau hingga panjang *queue* sama dengan 0, artinya jalur menuju simpul tujuan tidak ditemukan.

Secara umum, berikut adalah penerapan algoritma A* pada pseudocode:

The goal node is denoted by `node_goal` and the source node is denoted by `node_start`

We maintain two lists: **OPEN** and **CLOSE**:

OPEN consists on nodes that have been visited but not expanded (meaning that successors have not been explored yet). This is the list of pending tasks.

CLOSE consists on nodes that have been visited *and* expanded (successors have been explored already and included in the open list, if this was the case).

```
1 Put node_start in the OPEN list with  $f(\text{node\_start}) = h(\text{node\_start})$  (initialization)
2 while the OPEN list is not empty {
3     Take from the open list the node node_current with the lowest
4          $f(\text{node\_current}) = g(\text{node\_current}) + h(\text{node\_current})$ 
5     if node_current is node_goal we have found the solution; break
6     Generate each state node_successor that come after node_current
7     for each node_successor of node_current {
8         Set successor_current_cost =  $g(\text{node\_current}) + w(\text{node\_current}, \text{node\_successor})$ 
9         if node_successor is in the OPEN list {
10             if  $g(\text{node\_successor}) \leq \text{successor\_current\_cost}$  continue (to line 20)
11         } else if node_successor is in the CLOSED list {
12             if  $g(\text{node\_successor}) \leq \text{successor\_current\_cost}$  continue (to line 20)
13             Move node_successor from the CLOSED list to the OPEN list
14         } else {
15             Add node_successor to the OPEN list
16             Set  $h(\text{node\_successor})$  to be the heuristic distance to node_goal
17         }
18         Set  $g(\text{node\_successor}) = \text{successor\_current\_cost}$ 
19         Set the parent of node_successor to node_current
20     }
21     Add node_current to the CLOSED list
22 }
23 if(node_current != node_goal) exit with error (the OPEN list is empty)
```

Sumber : <https://mat.uab.cat/~alseda/MasterOpt/AStar-Algorithm.pdf>

BAB 2

IMPLEMENTASI ALGORITMA DALAM BAHASA JAVASCRIPT

Program ini dibuat dalam bahasa pemrograman JavaScript dan HTML asdfgfdhdgj. Adapun struktur dari program terdiri atas dua folder yaitu *Console* dan *Web*. Folder *Console* berisi program berbasis command line, *source code* terdapat pada folder *src* yang terdiri dari *Astar.js*, *input.js*, *mainProgram.js* yang merupakan driver utama program, *map.js*, *operation.js*, *PriorityQueue.js*, dan *UCS.js*. Folder *WEb* berisi program yang berbasis web terdiri atas *app.js* yang merupakan program utama untuk menjalankan web, *Astar.js*, *index.html*, *input.js*, *map.js* berisi inisialisasi map menggunakan *Google Maps API*, *PriorityQueue.js*, *UCS.js* dan *package.json* serta *package-lock.json* berisi *dependency* program..

2.1 Console

2.1.1 mainProgram.js

File merupakan driver utama program *console* berisi pemanggilan fungsi-fungsi untuk menyelesaikan permasalahan utama.

2.1.2 Astar.js

File berisi fungsi-fungsi yang menyusun algoritma A* dalam penyelesaian untuk menentukan lintasan terpendek dari suatu simpul menuju simpul lain.

Fungsi	Deskripsi
<i>Astar(mapAdjMatrix, arrayOfCoordinates, start, end)</i>	Fungsi mengembalikan array dengan elemen pertama bernilai true dan elemen kedua berisi jalur lintasan terpendek jika jalur dari start node ke goal node ditemukan. Mengembalikan array dengan elemen pertama bernilai false dan elemen kedua berisi queue kosong jika jalur dari start node ke goal node tidak ditemukan.
<i>getNeighbour(mapAdjMatrix, point)</i>	Fungsi mengembalikan <i>array of index</i> dari <i>neighbour</i> point.
<i>class Node</i>	Kelas <i>Node</i> merupakan kelas <i>custom</i> yang digunakan untuk menyimpan titik pada queue dalam algoritma A*.
<i>heuristic(start, end)</i>	Fungsi mengembalikan nilai heuristik dari node start ke node end.

2.1.3 UCS.js

File berisi fungsi-fungsi yang menyusun algoritma UCS dalam penyelesaian untuk menentukan lintasan terpendek dari suatu simpul menuju simpul lain.

Fungsi	Deskripsi
function UCS(mapAdjMatrix, startNode, goalNode)	Fungsi mengembalikan path lintasan terpendek berupa <i>array of index</i> dari node. Mengembalikan string kosong jika jalur lintasan terpendek dari <i>start node</i> ke <i>goal node</i> tidak ditemukan.

2.1.4 input.js

File berisi fungsi-fungsi yang menangani input matriks ketetanggan dan koordinat dari sebuah file.txt.

Fungsi	Deskripsi
readFile(filePath)	Fungsi mengembalikan matriks ketetanggan yang dibaca melalui file input txt.
readFileCoordinate(filePath)	Fungsi mengembalikan <i>array of coordinate</i> yang telah dibaca dari file input txt.
validCoordinate(arrayOfCoordinates)	Fungsi mengembalikan <i>false</i> jika koordinat yang dimasukan pada file txt tidak valid, <i>true</i> jika sudah valid.
validMap(adjMatrix)	Fungsi mengembalikan <i>false</i> jika matriks ketetanggan yang dimasukan pada file txt tidak valid, <i>true</i> jika sudah valid.
validNode(inputNode, adjMatrix)	Fungsi mengembalikan <i>false</i> node yang dimasukan user pada console tidak valid (di luar <i>range</i> index node yang ada), <i>true</i> jika sudah valid.

2.1.5 output.js

File berisi fungsi-fungsi yang menangani *output* pada *console*.

Fungsi	Deskripsi
displayMatrix (mapAdjMatrix)	Fungsi menampilkan matriks ketetanggan pada <i>console</i> .

displayCoordinate (array)	Fungsi menampilkan <i>array of coordinate</i> pada <i>console</i>
---------------------------	---

2.1.6 operation.js

File berisi operasi matematika yang digunakan pada program.

Fungsi	Deskripsi
getEuclidianDistance (coordinate1, coordinate2)	Fungsi mengembalikan jarak <i>Euclidian</i> antara kedua simpul (<i>straight line distance</i>).
distance(path, mapAdjMatrix)	Fungsi mengembalikan jarak asli dari jalur lintasan terpendek yang dihasilkan baik dengan algoritma A* maupun UCS.

2.1.7 PriorityQueue.js

File berisi definisi dari kelas *custom* PriorityQueue yang digunakan pada algoritma UCS.

Fungsi	Deskripsi
class PriorityQueue	Kelas PriorityQueue merupakan kelas <i>custom</i> yang digunakan pada algoritma UCS. Prioritas queue didasarkan pada jarak/cost yang paling kecil.

2.2 Web

2.2.1 app.js

File berisi program utama untuk menjalankan website. Berisi pemanggilan source file yang digunakan untuk website juga penyambungan source file ke *website*. File ini digunakan untuk menjalankan website dengan mengetikkan *command* node app.js pada command line di directory yang berisi file ini juga file-file dependency dari file ini.

2.2.2 Astar.js

File berisi fungsi-fungsi yang menyusun algoritma A* dalam penyelesaian untuk menentukan lintasan terpendek dari suatu simpul menuju simpul lain.

Fungsi	Deskripsi
Astar(mapAdjMatrix, arrayOfCoordinates,	Fungsi mengembalikan array dengan elemen

start, end)	pertama bernilai true dan elemen kedua berisi jalur lintasan terpendek jika jalur dari start node ke goal node ditemukan. Mengembalikan array dengan elemen pertama bernilai false dan elemen kedua berisi queue kosong jika jalur dari start node ke goal node tidak ditemukan.
getNeighbour(mapAdjMatrix, point)	Fungsi mengembalikan <i>array of index</i> dari <i>neighbour</i> point.
class Node	Kelas Node merupakan kelas <i>custom</i> yang digunakan untuk menyimpan titik pada queue dalam algoritma A*.

2.2.3 UCS.js

File berisi fungsi-fungsi yang menyusun algoritma UCS dalam penyelesaian untuk menentukan lintasan terpendek dari suatu simpul menuju simpul lain.

Fungsi	Deskripsi
function UCS(mapAdjMatrix, startNode, goalNode)	Fungsi mengembalikan path lintasan terpendek berupa <i>array of index</i> dari node. Mengembalikan string kosong jika jalur lintasan terpendek dari <i>start node</i> ke <i>goal node</i> tidak ditemukan.

2.2.4 input.js

File berisi fungsi-fungsi yang berisi penanganan *input* pada *website*.

Fungsi	Deskripsi
read()	Fungsi melakukan pembacaan pada file yang telah di- <i>input</i> dari web. Kemudian fungsi sekaligus memvalidasi isi dari file input, selanjutnya memanggil initMap(center, marker, adjmatrix) untuk menginisialisasi map berdasarkan koordinat masukan <i>user</i> .
validateJumlah(marker, adjmatrix)	Fungsi melakukan validasi banyaknya marker atau simpul yang terdapat pada graf untuk dibandingkan dengan banyak kolom dan baris pada matriks ketetanggaan. Selain itu pada fungsi ini dilakukan pula validasi

	terhadap simpul kosong.
validateMarker(marker)	Fungsi melakukan validasi pada elemen simpul atau marker. Posisi marker perlu dipastikan berupa angka serta marker perlu dipastikan tidak kosong.
validateMatrix(Matrix)	Fungsi melakukan validasi terhadap matriks ketetanggan dimana matriks ketetanggan tidak boleh kosong serta isi elemen matriks harus berupa angka.

2.2.5 PriorityQueue.js

File berisi definisi dari kelas *custom* PriorityQueue yang digunakan pada algoritma UCS.

Fungsi	Deskripsi
class PriorityQueue	Kelas PriorityQueue merupakan kelas <i>custom</i> yang digunakan pada algoritma UCS. Prioritas queue didasarkan pada jarak/cost yang paling kecil.

2.2.6 map.js

File berisi fungsi-fungsi yang berkesinambungan untuk menampilkan map pada website.

Fungsi	Deskripsi
initMapBlank()	Fungsi menampilkan map kosong (belum ada marker) dengan center pada ITB.
initMap(center, markers, adjmatrix)	Fungsi menampilkan map sesuai center, marker yang telah diinput user.
addMarker(location, map, adj, markers, id)	Fungsi ini membantu initMap untuk menampilkan marker pada map. Fungsi menerima input marker dari file yang dimasukan user juga menerima input marker dari event click yang dilakukan user pada marker di yang telah di-display di map. Ketika event click sudah berjumlah dua (artinya user sudah memilih start node dan goal node) maka fungsi memanggil fungsi calculateAndDisplayRoute(waypoints, id).

calculateAndDisplayRoute(waypoints, id)	Fungsi melakukan perhitungan jalur terpendek dengan memanggil fungsi UCS dan A*. Kemudian di dalam fungsi ini juga dilakukan <i>convert</i> dari hasil <i>array of index</i> menjadi list of title marker untuk display jalur pada web.
distance(path, mapAdjMatrix)	Fungsi menghitung jarak dari jalur terpendek yang dihasilkan.

2.2.7 index.html

File berisi pengaturan display map pada website. Dalam file ini file-file.js di-include menggunakan *script src*. *Styling* pada web juga terdapat pada file ini.

2.3 Ketentuan Format Input File

- Input koordinat simpul maupun matriks ketetanggaan disimpan dalam satu file dengan format .txt.
- Susunan isi pada file masukkan adalah sebagai berikut :

```

10          → Baris 1: banyak simpul pada peta(n)
-6.954288460437817 107.63966112160792 → Baris 2: Koordinat pusat tampilan peta
-6.956113961860843 107.63782236524717
Jalan Masuk Batununggal
-6.9420043901332695 107.627507039967
Simpang ISBI Bandung
-6.961920345201296 107.63858084483809
Pertigaan Tol Padaleunyi
-6.954410789607977 107.63925940982327
Simpang Padar Kordon
-6.948158120745616 107.63353651464993
Lampu Merah Buah Batu
-6.954072622048981 107.64019909420975
Tugu Kordon
-6.945923405214491 107.64183839759487
Simpang Titik Kumpul SuTa
-6.9550779790108415 107.64792687272197
Margacinta Park
-6.965368431841134 107.63795441275899
Simpang Terusan Buah Batu
-6.9494939888335 107.62596010345533
Simpang Cijagra-Soekarno Hatta
0 0 0 850 0 0 0 0 0 0
0 0 0 0 1000 0 0 0 0 850
0 0 0 850 0 0 0 0 400 0
850 0 850 0 950 110 0 0 0 0
0 1000 0 950 0 0 1000 0 0 850
0 0 0 110 0 0 900 850 0 0
0 0 0 0 1000 900 0 0 0 0
0 0 0 0 850 0 0 0 0 0
0 0 400 0 0 0 0 0 0 0
0 850 0 0 850 0 0 0 0 0
} n baris terakhir : matriks ketetanggaan

```

Baris 1 menyatakan banyak simpul pada peta

Baris 2 menyatakan koordinat pusat tampilan peta

Baris 3, 5, 7, dan seterusnya menyatakan koordinat setiap simpul pada peta

Baris 4, 6, 8, dan seterusnya menyatakan nama atau deskripsi setiap simpul

n baris terakhir menyatakan matriks ketetanggaan

- Banyak simpul(n), koordinat, dan elemen dari matriks ketetanggaan berupa angka.
- Setiap koordinat terdiri atas koordinat Latitude (garis lintang) dan Longitude (garis bujur) yang dipisahkan dengan sebuah spasi dan diasumsikan selalu benar atau berada pada peta.
- Matriks ketetanggaan merepresentasikan jarak sesungguhnya antar simpul dalam satuan meter, dengan 0 merepresentasikan bahwa antar simpul tersebut tidak bertetangga.
- Setiap sisi merepresentasikan jalan dua arah sehingga matriks ketetanggaan merepresentasikan graf tidak berarah dengan jarak A ke B sama dengan B ke A

BAB 3

SOURCE CODE PROGRAM

3.1 Console

3.1.1 mainProgram.js

```
import { UCS, distance } from "./UCS.js"
import { Astar } from "./Astar.js";
import { readFile, readFileCoordinate, validCoordinate, validMap,
validNode } from "./input.js";
import promptSync from 'prompt-sync';
const prompt = promptSync();
import { displayMatrix } from "./operation.js";
import { displayCoordinate } from "./operation.js";
import path from 'path';
import { fileURLToPath } from "url";

// Main Program
console.log("Entering program...\n");

// Arranging file path
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
const parentDir = path.resolve(__dirname, '..');

// Input file
var arrayOfCoordinates;
var fileName = prompt('Input coordinate file name (without the .txt
extension): ');
var valid;
do {
    var filePath = path.join(parentDir, 'test', fileName + '.txt');
    valid = true;
    try {
        arrayOfCoordinates = readFileCoordinate(filePath);
    } catch (err) {
        console.log(err);
        fileName = prompt('Input coordinate file name (without the
.txt extension): ');
        valid = false;
    }
} while (!valid);

while (!validCoordinate(arrayOfCoordinates)) {
    var filePath = path.join(parentDir, 'test', fileName + '.txt');
    fileName = prompt("Input coordinate file name (without the .txt
```

```

extension) : ";
    arrayOfCoordinates = readFileCoordinate(filePath);
}

displayCoordinate(arrayOfCoordinates);

// Input file and input file validation
var fileName = prompt('Input adjacency map file name (without the
.txt extension): ');
var mapAdjMatrix;
var valid;
do {
    var filePath = path.join(parentDir, 'test', fileName + '.txt');
    valid = true;
    try {
        mapAdjMatrix = readFile(filePath);
    } catch (err) {
        console.log("File not found");
        console.log("Make sure the .txt file is stored in the test
folder");
        fileName = prompt('Input adjacency map file name (without
the .txt extension): ');
        valid = false;
    }
} while (!valid);

while (!validMap(mapAdjMatrix)) {
    var filePath = path.join(parentDir, 'test', fileName + '.txt');
    fileName = prompt("Input adjacency map file name (without the
.txt extension): ");
    mapAdjMatrix = readFile(filePath);
}
displayMatrix(mapAdjMatrix);

console.log("There is", mapAdjMatrix.length, "node : 0 -",
mapAdjMatrix[0].length - 1);
var startNode = prompt("Input start node : ");
while (!validNode(startNode, mapAdjMatrix)) {
    startNode = prompt("Input start node : ");
}
var goalNode = prompt("Input goal node : ");
while (!validNode(goalNode, mapAdjMatrix)) {
    goalNode = prompt("Input goal node : ");
}

const resultPath = UCS(mapAdjMatrix, startNode, goalNode);
const pathAstar = Astar(mapAdjMatrix, arrayOfCoordinates,
startNode, goalNode);

if (resultPath == "") console.log("No path found\n");

```

```

else {
    console.log(" ===== UCS ===== ");
    const resultDistance = distance(resultPath, mapAdjMatrix);
    console.log("Path found with total distance " +
resultDistance);
    console.log("Shortest path : " +
Array.from(resultPath.values()));
}

console.log("\n===== A STAR ===== ");
const AsDistance = distance(pathAstar, mapAdjMatrix);
console.log("Path found with total distance " + AsDistance);
console.log("Shortest path : " + path);

```

3.1.2 Astar.js

```

import {getEuclidianDistance} from './operation.js';
export function Astar(mapAdjMatrix, arrayOfCoordinates, start, end)
{
    // calculating heuristic score by euclidian distance
    function heuristic(start, end) {
        return getEuclidianDistance(arrayOfCoordinates[start],
arrayOfCoordinates[end]);
    }

    // initialize the has been expand node set
    const hasBeenExpand = new Set(); // close-set

    // Initialize the queue to expand the path
    const nodeStart = new Node(start, null);
    const queue = [nodeStart] // openset

    // Initialize the f(n) and g(n)
    const gScore = new Map();
    gScore.set(nodeStart, 0);
    const fScore = new Map();
    fScore.set(nodeStart, heuristic(start,end));

    while (queue.length > 0) {
        // finding the lowest fScore by sorting it
        queue.sort((node1, node2) => fScore.get(node1) -
fScore.get(node2));
        // dequeue the queue, get vertex to expand
        const current = queue.shift();

        // checking if current node is the goal
        if (current.index == end) {
            const path = [current.index]
            let point = new Node(current.index, current.cameFrom);

```

```

        while (point.index != start) {
            path.unshift(point.cameFrom.index);
            point = point.cameFrom;
        }
        return path
    }

    // adding the current node to sign it has been expand
    hasBeenExpand.add(current);

    // get the current neighbor
    const neighbours = getNeighbour(mapAdjMatrix, current);
    for (const neighbour of neighbours) {
        if (hasBeenExpand.has(neighbour)) {
            continue;
        }
        if (!queue.includes(neighbour)) {
            queue.push(neighbour)
        }
        gScore.set(neighbour, gScore.get(current) +
mapAdjMatrix[current.index][neighbour.index]);
        fScore.set(neighbour, heuristic(neighbour.index, end) +
gScore.get(neighbour));
    }
}

// custom node class
class Node {
    constructor(index, cameFrom = null) {
        this.index = index;
        this.cameFrom = cameFrom;
    }
}

// getting the neighbour of current expanded node
function getNeighbour(mapAdjMatrix, point) {
    const neighbors = [];
    for (let i = 0; i < mapAdjMatrix[0].length; i++) {
        if (mapAdjMatrix[point.index][i] != 0 &&
mapAdjMatrix[point.index][i] != -1) {
            neighbors.push(new Node(i, point));
        }
    }
    return neighbors;
}

```

3.1.3 UCS.js

```

import { PriorityQueue } from "./PriorityQueue.js";

export function UCS(mapAdjMatrix, startNode, goalNode) {
    let livingNode = new PriorityQueue();
    let expandNode = new Set(); // Unique list node
    yang telah di expand
    let found = false;

    let currentExpand;
    let newPath;
    let newCost;
    let isThereAdj = false;

    livingNode.enqueue([0, startNode, [startNode]]);

    // Start Searching
    while (!found && !livingNode.isEmpty()) {
        currentExpand = livingNode.dequeue();
        expandNode.add(currentExpand[1]);

        if (currentExpand[1] == goalNode) {
            found = true;
        }

        if (!found) {
            for (let i = 0; i < Object.keys(mapAdjMatrix).length;
i++) {
                if (mapAdjMatrix[currentExpand[1]][i] > 0 &&
!expandNode.has(i)) {
                    isThereAdj = true;
                    newCost = currentExpand[0] +
mapAdjMatrix[currentExpand[1]][i];
                    newPath = currentExpand[2].slice();
                    newPath.push(i);
                    livingNode.enqueue([newCost, i, newPath]);
                }
            }
        }
    }

    if (!found) return "";
    else return currentExpand[2];
}

```

3.1.4 input.js

```

import fs from 'fs';

```

```

export function readFile(filePath) {
    const result = [];
    const contents = fs.readFileSync(filePath, 'utf-8');
    // Read per line
    const lines = contents.trim().split('\n');
    // Split lines
    for (let i = 0; i < lines.length; i++) {
        const splitted = lines[i].trim().split(/\s+/).map(Number);
        result.push(splitted);
    }

    return result;
}

export function readFileCoordinate(filePath) {
    const result = [];
    let contents;
    try {
        contents = fs.readFileSync(filePath, 'utf-8');
    } catch (err) {
        throw new Error("File not found\nMake sure the .txt file is stored in the test folder")
    }
    // Read per line
    const lines = contents.trim().split('\n');
    // Split lines
    for (let i = 0; i < lines.length; i++) {
        const splitted = lines[i].trim().split(/\s+/).map(Number);
        result.push({x: splitted[0], y: splitted[1]});
    }
    return result;
}

export function validCoordinate(arrayOfCoordinates) {
    for (let i = 0; i < arrayOfCoordinates.length; i++) {
        if (Number.isNaN(arrayOfCoordinates[i].x) ||
            Number.isNaN(arrayOfCoordinates[i].y)) {
            return false;
        }
    }
    return true;
}

export function validMap(adjMatrix) {
    for (let i = 0; i < adjMatrix.length; i++) {
        for (let j = 0; j < adjMatrix[0].length; j++) {
            if (Number.isNaN(adjMatrix[i][j]) || adjMatrix[i][j] < -1) {
                console.log("Invalid element in (" + i + ", " + j + ")");
                console.log("Element must be integer. The minimum

```

```

        value of -1 denotes a non-neighboring node");
            return false;
        }
    }
}
if (adjMatrix.length == 0) {
    console.log("Empty Adjacency Matrix. Try Again!");
    return false;
}
if (adjMatrix.length != adjMatrix[0].length) {
    console.log("Invalid Map AdjacencyMatrix. Try Again!");
    console.log("Adjacency Matrix must be square and separated
by spaces between their columns.");
    return false;
}
return true;
}

export function validNode(inputNode, adjMatrix) {
    if (inputNode < 0 || inputNode >= adjMatrix.length) {
        console.log("Invalid input! Choose node 0 -",
adjMatrix[0].length - 1);
        return false;
    } else return true;
}

```

3.1.5 output.js

```

export function displayMatrix (mapAdjMatrix) {
    console.log(" ===== Matrix of Adjacency
===== ");
    const matrix = mapAdjMatrix;
    for (let i = 0; i < matrix[1].length; i++) {
        for (let j = 0; j < matrix[0].length; j++) {
            process.stdout.write(matrix[i][j].toString() + " ");
        }
        console.log();
    }
    console.log();
}

export function displayCoordinate (array) {
    console.log(" ===== Array of Coordinates
===== ");
    for (let i = 0; i < array.length; i++) {
        console.log(i + ":" + array[i].x.toString() + ", " +
array[i].y.toString());
    }
    console.log();
}

```

```
}
```

3.1.6 operation.js

```
export function getEuclidianDistance (coordinate1, coordinate2) {
    const x1 = coordinate1.x;
    const y1 = coordinate1.y;
    const x2 = coordinate2.x;
    const y2 = coordinate2.y;
    const deltax = x1 - x2;
    const deltay = y1 - y2;
    const distance = Math.sqrt(deltax * deltax + deltay * deltay);
    return distance;
}

export function distance(path, mapAdjMatrix) {
    let resDist = 0;
    for (let i = 0; i < path.length - 1; i++) {
        resDist += mapAdjMatrix[path[i]][path[i + 1]];
    }
    return resDist;
}
```

3.1.7 PriorityQueue.js

```
export class PriorityQueue {
    // queue[i][0] menyatakan priority element ke-i yaitu
    // jarak/cost dari start node
    // dengan prioritas tertinggi pada nilai
    // terkecil
    // queue[i][1] menyatakan currentNode
    // queue[i][2] menyatakan path untuk mencapai currentNode dari
    // startNode

    constructor() {
        this.queue = [];
    }

    // Insert new element
    enqueue(newElement) {
        let done = false;
        for (let i = 0; !done && i < this.queue.length; i++) {
            if (newElement[0] <= this.queue[i][0]) {
                // insert element at index i
                this.queue.splice(i, 0, newElement);
                done = true;
            }
        }
    }
}
```

```

    }

    if (!done) {
        // Insert at the end of element
        this.queue.push(newElement);
    }
}

// Remove leading element
dequeue() {
    return this.queue.shift();
}

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

// Print Queue
print() {
    if (this.queue.length == 0) console.log("Empty Queue\n");
    else {
        console.log("{}");
        for (let i = 0; i < this.queue.length; i++) {
            console.log("0 = " + this.queue[i][0] + ", 1 = " +
this.queue[i][1] + ", 2 = " + this.queue[i][2] + ")");
            if (i == this.queue.length - 1) console.log("}\n");
            else console.log(", ");
        }
    }
}
}

```

3.2 Web

3.2.1 app.js

```

const express = require('express');
const app = express();

app.get('/', (req, res) => {
    res.sendFile(__dirname + '/index.html');
});
app.get('/PriorityQueue.js', (req, res) => {
    res.set('Content-Type', 'application/javascript');
    res.sendFile(__dirname + '/PriorityQueue.js');
});
app.get('/UCS.js', (req, res) => {
    res.set('Content-Type', 'application/javascript');
}

```

```

        res.sendFile(__dirname + '/UCS.js');
    });
app.get('/Astar.js', (req, res) => {
    res.set('Content-Type', 'application/javascript');
    res.sendFile(__dirname + '/Astar.js');
});
app.get('/graph.js', (req, res) => {
    res.set('Content-Type', 'application/javascript');
    res.sendFile(__dirname + '/graph.js');
});
app.get('/input.js', (req, res) => {
    res.set('Content-Type', 'application/javascript');
    res.sendFile(__dirname + '/input.js');
});
app.get('/map.js', (req, res) => {
    res.set('Content-Type', 'application/javascript');
    res.sendFile(__dirname + '/map.js');
});

app.listen(3000, () => {
    console.log('Server is running on port 3000');
});

```

3.2.2 Astar.js

```

function Astar(mapAdjMatrix, arrayOfCoordinates, start, end) {
    // calculating heuristic score by euclidian distance
    function heuristic(start, end) {
        return getEuclidianDistance(arrayOfCoordinates[start],
arrayOfCoordinates[end]);
    }

    // initialize the has been expand node set
    const hasBeenExpand = new Set(); // close-set

    // Initialize the queue to expand the path
    const nodeStart = new Node(start, null);
    const queue = [nodeStart] // openset

    // Initialize the f(n) and g(n)
    const gScore = new Map();
    gScore.set(nodeStart, 0);
    const fScore = new Map();
    fScore.set(nodeStart, heuristic(start,end));

    while (queue.length > 0) {

```

```

        // finding the lowest fScore by sorting it
        queue.sort((node1, node2) => fScore.get(node1) -
fScore.get(node2));
        // dequeue the queue, get vertex to expand
        const current = queue.shift();
        // checking if current node is the goal
        if (current.index == end) {
            const path = [current.index]
            let point = new Node(current.index, current.cameFrom);
            while (point.index != start) {
                var parent = point.cameFrom
                path.unshift(parent.index);
                point = point.cameFrom;
            }
            return [true, path]
        }

        // adding the current node to sign it has been expand
        hasBeenExpand.add(current.index);

        // get the current neighbor
        const neighbours = getNeighbour(mapAdjMatrix, current);
        for (const neighbour of neighbours) {
            if (hasBeenExpand.has(neighbour.index)) {
                continue;
            }
            if (!queue.includes(neighbour)) {
                queue.push(neighbour)
            }
            gScore.set(neighbour, gScore.get(current) +
mapAdjMatrix[current.index][neighbour.index]);
            fScore.set(neighbour, heuristic(neighbour.index, end) +
gScore.get(neighbour));
        }
    }
    return [false, queue]
}

// custom node class
class Node {
    constructor(index, cameFrom = null) {
        this.index = index;
        this.cameFrom = cameFrom;
    }
}

// getting the neighbour of current expanded node
function getNeighbour(mapAdjMatrix, point) {
    const neighbors = [];
    for (let i = 0; i < mapAdjMatrix[0].length; i++) {

```

```

        if (mapAdjMatrix[point.index][i] != 0 &&
mapAdjMatrix[point.index][i] != -1) {
            neighbors.push(new Node(i, point));
        }
    }
    return neighbors;
}

function getEuclidianDistance (coordinate1, coordinate2) {
    const x1 = coordinate1.x;
    const y1 = coordinate1.y;
    const x2 = coordinate2.x;
    const y2 = coordinate2.y;
    const deltax = x1 - x2;
    const deltay = y1 - y2;
    const distance = Math.sqrt(deltax * deltax + deltay * deltay);
    return distance;
}

```

3.2.3 UCS.js

```

function UCS(mapAdjMatrix, startNode, goalNode) {
    let livingNode = new PriorityQueue();
    let expandNode = new Set(); // Unique list node yang telah di expand
    let found = false;

    let currentExpand;
    let newPath;
    let newCost;
    let isThereAdj = false;

    livingNode.enqueue([0, startNode, [startNode]]);

    // Start Searching
    while (!found && !livingNode.isEmpty()) {
        currentExpand = livingNode.dequeue();
        expandNode.add(currentExpand[1]);

        if (currentExpand[1] == goalNode) {
            found = true;
        }

        if (!found) {
            for (let i = 0; i < Object.keys(mapAdjMatrix).length;
i++) {
                if (mapAdjMatrix[currentExpand[1]][i] > 0 &&

```

```

!expandNode.has(i)) {
    isThereAdj = true;
    newCost = currentExpand[0] +
mapAdjMatrix[currentExpand[1]][i];
    newPath = currentExpand[2].slice();
    newPath.push(i);
    livingNode.enqueue([newCost, i, newPath]);
}
}

if (!found) return "";
else return currentExpand[2];
}

function distance(path, mapAdjMatrix) {
let resDist = 0;
for (let i = 0; i < path.length - 1; i++) {
    resDist += mapAdjMatrix[path[i]][path[i + 1]];
}
return resDist;
}

```

3.2.4 input.js

```

const fileInput = document.getElementById('file-input');
const fileContents = document.getElementById('file-contents');
const center = [];
const marker = [];
const adjmatrix = [];

function read() {
    initMapBlank();
    center.length = 0;
    marker.length = 0;
    fileInput.addEventListener('change', (event) => {
        const file = event.target.files[0];
        const reader = new FileReader();
        reader.addEventListener('load', (event) => {
            const contents = event.target.result;
            let splitted;
            const lines = contents.trim().split('\n');
            let simpul;
            let positions;

```

```

for (let i = 0; i < lines.length; i++) {
    if (i == 0) {
        splitted = lines[i].trim().split(/\s+/).map(Number);
        simpul = splitted[0];
    } else if (i == 1) {
        splitted = lines[i].trim().split(/\s+/).map(Number);
        centers = {
            lat: splitted[0],
            lng: splitted[1]
        }
        center.push(centers);
    } else if (i <= simpul * 2 + 1) {
        if (i % 2 == 0) {
            splitted =
lines[i].trim().split(/\s+/).map(Number);
            positions = {
                lat: splitted[0],
                lng: splitted[1]
            }
        } else {
            splitted = lines[i].trim().split().map(String);
            marker.push({
                position: positions,
                title: splitted[0]
            });
        }
    } else {
        splitted = lines[i].trim().split(/\s+/).map(Number);
        adjmatrix.push(splitted);
    }
}
if (!validateJumlah(marker, adjmatrix)) {
    Swal.fire({
        title: 'Oops!',
        text: 'Jumlah koordinat simpul yang dimasukkan tidak
sesuai dengan jumlah simpul pada matriks ketetanggaan!',
        icon: 'error',
        confirmButtonText: 'OK',
        customClass: {
            confirmButton: 'custombutton',
        }
    })
    return
} else if (!validateMarker(marker) ||
!validateMatrix(adjmatrix)) {
    Swal.fire({
        title: 'Oops!',
        text: 'Masukan korrdinat hanya berupa angka!',
        icon: 'error',
        confirmButtonText: 'OK',
        customClass: {

```

```
        confirmButton: 'custombutton',
    }
}
return
}
initMap(center, marker, adjmatrix);
});
reader.readAsText(file);
);
}

function validateJumlah(marker, adjmatrix) {
    if (marker.length == 0) {
        return false;
    }
    if (marker.length != adjmatrix.length) {
        return false;
    }
    return true
}

function validateMarker(marker) {
    if (marker.length < 2) {
        return false;
    }
    for (let i = 0; i < marker; i++) {
        if (Number.isNaN(marker[i].position.lat || Number.isNaN(marker[i].position.lng))) {
            return false;
        }
    }
    return true;
}

function validateMatrix(Matrix) {
    if (Matrix.length == 0) {
        return false;
    } else if (Matrix[1].length != Matrix[0].length) {
        return false;
    }
    for (let i = 0; i < Matrix[1].length; i++) {
        for (let j = 0; j < Matrix[0].length; j++) {
            if (Number.isNaN(Matrix[i][j])) {
                return false;
            }
        }
    }
    return true;
}
```

3.2.5 PriorityQueue.js

```
class PriorityQueue {
    // queue[i][0] menyatakan priority element ke-i yaitu
    // jarak/cost dari start node
    //           dengan prioritas tertinggi pada nilai
    // terkecil
    // queue[i][1] menyatakan currentNode
    // queue[i][2] menyatakan path untuk mencapai currentNode dari
    // startNode

    constructor() {
        this.queue = [];
    }

    // Insert new element
    enqueue(newElement) {
        let done = false;
        for (let i = 0; !done && i < this.queue.length; i++) {
            if (newElement[0] <= this.queue[i][0]) {
                this.queue.splice(i, 0, newElement);
                done = true;
            }
        }

        if (!done) {
            this.queue.push(newElement);
        }
    }

    // Remove leading element
    dequeue() {
        return this.queue.shift();
    }

    isEmpty() {
        return this.queue.length == 0;
    }
    // Print Queue
    print() {
        if (this.queue.length == 0) console.log("Empty Queue\n");
        else {
            console.log("{}");
            for (let i = 0; i < this.queue.length; i++) {
                console.log("0 = " + this.queue[i][0] + ", 1 = " +
                    this.queue[i][1] + ", 2 = " + this.queue[i][2]);
            }
        }
    }
}
```

```

        this.queue[i][1] + ", 2 = " + this.queue[i][2] + ")");
            if (i == this.queue.length - 1) console.log("}\n");
            else console.log(", ");
        }
    }
}

```

3.2.6 map.js

```

var routeMarkers1 = [];
var routeMarkers2 = [];
var directionsService;
var directionsRenderer;
var waypoints = [];
var waypoints2 = [];
var routeIdx1 = [];
var routeIdx2 = [];
var string1 = [];
var string2 = [];
let path;
let UCSPPath;

function initMapBlank() {
    var map11 = new google.maps.Map(document.getElementById("map"), {
        zoom: 15,
        center: new google.maps.LatLng(-6.89067290133392,
107.61002829330324),
        mapTypeId: google.maps.MapTypeId.ROADMAP,
    });
    var map22 = new google.maps.Map(document.getElementById("map2"),
{
    zoom: 15,
    center: new google.maps.LatLng(-6.89067290133392,
107.61002829330324),
    mapTypeId: google.maps.MapTypeId.ROADMAP,
});
}

function initMap(center, markers, adjmatrix) {
    var map1 = new google.maps.Map(document.getElementById("map"), {
        zoom: 15,
        center: new google.maps.LatLng(center[0].lat, center[0].lng),
        mapTypeId: google.maps.MapTypeId.ROADMAP,
    });
    var map2 = new google.maps.Map(document.getElementById("map2"), {

```

```
    zoom: 15,
    center: new google.maps.LatLng(center[0].lat, center[0].lng),
    mapTypeId: google.maps.MapTypeId.ROADMAP,
  });

directionsService1 = new google.maps.DirectionsService();
directionsService2 = new google.maps.DirectionsService();

directionsRenderer1 = new google.maps.DirectionsRenderer({
  map: map1,
  suppressMarkers: true
});

directionsRenderer2 = new google.maps.DirectionsRenderer({
  map: map2,
  suppressMarkers: true
});

// Create markers for each location.
markers.forEach((marker) => {
  const newMarker1 = new google.maps.Marker({
    position: marker.position,
    map: map1,
    title: marker.title,
  });

  // Add accessibility text to marker.
  newMarker1.addListener("click", () => {
    routeIdx1.push(markers.indexOf(marker));
    addMarker(marker.position, map1, adjmatrix, markers, 1);
    new google.maps.InfoWindow({
      content: marker.title,
    }).open(map1, newMarker1);
  });
});

markers.forEach((marker) => {
  const newMarker2 = new google.maps.Marker({
    position: marker.position,
    map: map2,
    title: marker.title,
  });

  // Add accessibility text to marker.
  newMarker2.addListener("click", () => {
    routeIdx2.push(markers.indexOf(marker));
    addMarker(marker.position, map2, adjmatrix, markers, 2);
    new google.maps.InfoWindow({
      content: marker.title,
    }).open(map2, newMarker2);
  });
});
```

```

        });
    });
}

function addMarker(location, map, adj, markers, id) {
    var marker = new google.maps.Marker({
        position: location,
        map: map
    });
    if (id == 1) {
        routeMarkers1.push(marker);
    } else {
        routeMarkers2.push(marker);
    }
    const button = document.getElementById("visualizegraph")
    button.disabled = true;
    const array = []
    for (let i = 0; i < markers.length; i++) {
        array.push({ x: markers[i].position.lat, y:
            markers[i].position.lng })
    }
    if (routeMarkers1.length == 2) {
        UCSPath = UCS(adj, routeIdx1[0], routeIdx1[1]);
        if (UCSPath == "") {
            Swal.fire({
                title: 'Error!',
                text: 'No path found!',
                icon: 'error',
                confirmButtonText: 'OK',
                customClass: {
                    confirmButton: 'custombutton',
                }
            })
        } else {
            var distanceUCS = distance(UCSPath, adj)
            document.getElementById("distanceUCS").textContent = "UCS
Distance: " + distanceUCS + " m"
            for (var i = 0; i < UCSPath.length; i++) {
                waypoints.push({
                    location: markers[UCSPath[i]].position,
                    stopover: true
                });
                string1.push(markers[UCSPath[i]].title);
            }
            var content1 = ""
            for (let i = 0; i < string1.length; i++) {
                content1 += string1[i]
                if (i != string1.length - 1) {
                    content1 += " > "
                }
            }
        }
    }
}

```

```

        document.getElementById("string1").textContent = content1
        calculateAndDisplayRoute(waypoints, 1);
    }
    routeMarkers1.length = 0;
} else if (routeMarkers2.length == 2) {
    console.log("halo");
    let AstarPath = Astar(adj, array, routeIdx2[0], routeIdx2[1]);
    if (!AstarPath[0]) {
        Swal.fire({
            title: 'Error!',
            text: 'No path found!',
            icon: 'error',
            confirmButtonText: 'OK',
            customClass: {
                confirmButton: 'custombutton',
            }
        })
    } else {
        var distanceAstar = distance(AstarPath[1], adj)
        path = AstarPath[1]
        document.getElementById("distanceAstar").textContent = "AStar
Distance: " + distanceAstar + " m"
        for (var i = 0; i < path.length; i++) {
            waypoints2.push({
                location: markers[path[i]].position,
                stopover: true
            });
            string2.push(markers[path[i]].title);
        }
        var content2 = ""
        for (let i = 0; i < string2.length; i++) {
            content2 += string2[i]
            if (i != string2.length - 1) {
                content2 += " > "
            }
        }
        document.getElementById("string2").textContent = content2
        routeMarkers2.length = 0;
        calculateAndDisplayRoute(waypoints2, 2);
    }
}
button.disabled = false;
button.addEventListener('click', function () {
    displayGraph(markers, adj, UCSPPath, path) })
function displayGraph(markers, adj, UCSPPath, path) {
    graphVisual(markers, adj, UCSPPath, 1)
    graphVisual(markers, adj, path, 2)
}

```

```

}

function calculateAndDisplayRoute(waypoints, id) {
  console.log(id)
  if (id == 1) {
    directionsService1.route({
      origin: waypoints[0].location,
      destination: waypoints[waypoints.length - 1].location,
      waypoints: waypoints.slice(1, -1),
      optimizeWaypoints: true,
      travelMode: 'WALKING'
    }, function (response, status) {
      if (status === 'OK') {
        directionsRenderer1.setDirections(response);
      } else {
        window.alert('Directions request failed due to ' + status);
      }
    });
  } else {
    directionsService2.route({
      origin: waypoints[0].location,
      destination: waypoints[waypoints.length - 1].location,
      waypoints: waypoints.slice(1, -1),
      optimizeWaypoints: true,
      travelMode: 'WALKING'
    }, function (response, status) {
      if (status === 'OK') {
        directionsRenderer2.setDirections(response);
      } else {
        window.alert('Directions request failed due to ' + status);
      }
    });
  }
}

function distance(path, mapAdjMatrix) {
  let resDist = 0;
  for (let i = 0; i < path.length - 1; i++) {
    resDist += mapAdjMatrix[path[i]][path[i + 1]];
  }
  return resDist;
}

```

3.2.7 index.html

```

<!DOCTYPE html>
<html>

```

```
<head>
  <title>UCS AStar</title>
  <style>
    .custom-file-input:hover {
      background-color: #4F6846;
      color: #98B594;
    }

    .custom-file-input {
      position: absolute;
      left: 20px;
      width: 127px;
      height: 29px;
      font-family: 'Inria Serif';
      font-style: normal;
      font-weight: 700;
      font-size: 24px;
      line-height: 29px;
      text-align: center;
      color: #4F6846;
      background: #B2C5AB;
      box-shadow: 0px 4px 4px rgba(0, 0, 0, 0.25);
      border-radius: 50px;
      padding: 12px 21px;
      white-space: nowrap;
    }

    .custom-graph-button {
      position: absolute;
      left: 2100px;
      font-family: 'Inria Serif';
      font-style: normal;
      font-weight: 700;
      font-size: 24px;
      line-height: 29px;
      text-align: center;
      color: #4F6846;
      background: #B2C5AB;
      box-shadow: 0px 4px 4px rgba(0, 0, 0, 0.25);
      border-radius: 50px;
      padding: 12px 21px;
      white-space: nowrap;
    }

    .custom-graph-button:hover {
      background-color: #4F6846;
      color: #98B594;
    }

    .custom-graph-button:disabled {
```

```

        color: #999 !important;
        background-color: #ccc !important;
        cursor: not-allowed !important;

    }

.custombutton {
    padding: 12px 21px;
    border-radius: 50px;
    box-shadow: 0px 4px 4px rgba(0, 0, 0, 0.25);
    background-color: #B2C5AB;
    color: #4F6846;
}

#visualizegraph:disabled {
    color: #999;
    background-color: #ccc;
    cursor: not-allowed;
}
</style>
</head>

<body style="background-color:#98B594;">
    <link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/sweetalert2@11.1.5/dist/sweetalert2.min.css">
    <script
    src="https://cdn.jsdelivr.net/npm/sweetalert2@11.1.5/dist/sweetalert2.all.min.js"></script>
    <h1 style="text-shadow: 0px 4px 4px rgba(0, 0, 0, 0.25); display: flex; position: relative; width: 1142px; height: 115px; top: 0px; font-family: 'Inria Serif'; font-style: normal; font-weight: 700; font-size: 96px; line-height: 115px; left: 600px; color: #4F6846;"> Finding the Shortest Path </h1>
    <div style="position:absolute; display: flex; top: 180px;">
        <label class="custom-file-input" for="file-input">Choose File</label>
        <div id="wrong input" style="color: #4F6846; display: inline-block;"></div>
        <input type="file" id="file-input" style="display: none;" accept=".txt">
        <pre id="file-contents"></pre>
        <label for="visualizegraph" class="custom-graph-button">Show Graph</label>
    </div>
    <p style="position:absolute; display: flex; top: 190px; width: 500px; height: 20px; font-family: 'Inria Serif'; font-style: normal; font-weight: 700; font-size: 20px; line-height: 10px; left: 220px; color: #4F6846;"> Refresh the page before starting a new search! </p>
    <button id="visualizegraph" style="display: none;"></button>

```

```
<script type="text/javascript"
src="https://unpkg.com/vis-network/standalone/umd/vis-network.min.js"></script>
<script type="application/javascript" src="input.js"></script>
<script type="application/javascript"
src="PriorityQueue.js"></script>
<script type="application/javascript" src="UCS.js"></script>
<script type="application/javascript" src="Astar.js"></script>
<script type="application/javascript" src="graph.js"></script>
<div class="map container" style=" display: flex;
justify-content: space-between; ">
    <div id="map" style=" height: 600px; width: 1300px; display:
inline-block; margin: 0 1%; "></div>
    <div id="map2" style=" height: 600px; width: 1300px; display:
inline-block; margin: 0 1%; "></div>
</div>
<script type="application/javascript" src="map.js"></script>
<script async defer

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCV0w7APS8qmo
0d7eAT3ju9kuhzN_r4Wjg&callback=read"></script>
<div class="string" style="display: flex; justify-content:
space-between;">
    <p
        style=" text-shadow: 0px 4px 4px rgba(0, 0, 0, 0.25);
white-space:nowrap; position: absolute; display: inline-block;
width: auto; height: 43px; left: 40px; top: 830px; font-family:
'Inria Serif'; font-style: normal; font-weight: 700; font-size:
36px; line-height: 43px; text-align: center; color: #4F6846;">
        UCS Path</p>
    <p
        style=" text-shadow: 0px 4px 4px rgba(0, 0, 0, 0.25);
white-space:nowrap; position: absolute; display: inline-block;
width: auto; height: 43px; left: 1180px; top: 830px; font-family:
'Inria Serif'; font-style: normal; font-weight: 700; font-size:
36px; line-height: 43px; text-align: center; color: #4F6846;">
        Astar Path</p>
</div>
<div style="position: absolute; top: 890px; display: flex;">
    <p id="distanceUCS"
        style=" display:inline-block; position: absolute;
white-space:nowrap; width: 161px; height: 43px; left: 40px;
font-family: 'Inria Serif'; font-style: normal; font-weight: 700;
font-size: 20px; line-height: 43px; text-align: center; color:
#4F6846;">
    </p>
    <p id="distanceAstar"
        style=" position: absolute; white-space:nowrap; width:
161px; height: 43px; left: 1180px; font-family: 'Inria Serif';
font-style: normal; font-weight: 700; font-size: 20px; line-height:
43px; text-align: center; color: #4F6846;">
```

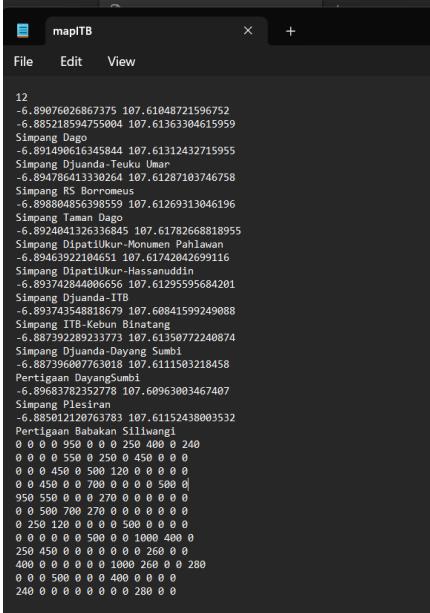
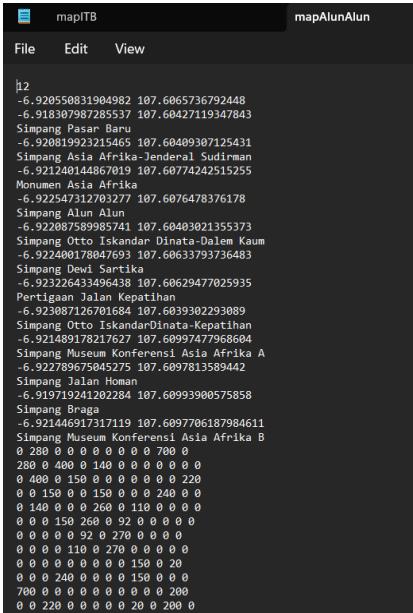
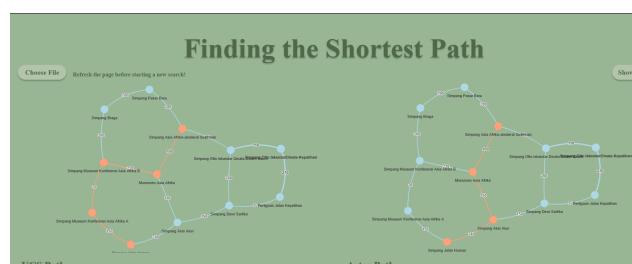
```
</p>
</div>
<div style="position: relative; top:80px; display: flex;
white-space: pre-line">
<p id="string1"
   style=" display: inline-block; position: absolute;
word-wrap: break-word; width: 900px; height: 35px; left: 20px;
font-family: 'Inria Serif'; font-style: normal; font-weight: 700;
font-size: 20px; line-height: 43px; text-align: center; color:
#4F6846;">
</p>
<p id="string2"
   style=" position: absolute; word-wrap: break-word; width:
900px; height: 35px; left: 1170px; font-family: 'Inria Serif';
font-style: normal; font-weight: 700; font-size: 20px; line-height:
43px; text-align: center; color: #4F6846;">
</p>
</div>
</body>

</html>
```

BAB 4

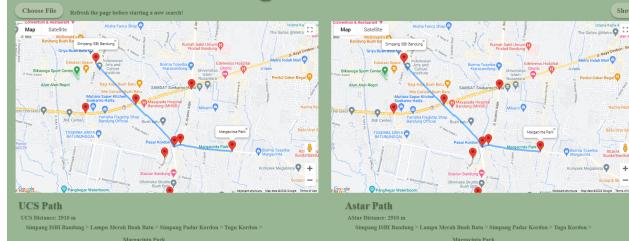
PENGUJIAN PROGRAM

4.1 Masukkan dan Keluaran Program

Input	Output
<p style="text-align: center;">mapITB.txt</p>  <pre> File Edit View 12 -6.89076026867375 107.61048721596752 -6.8852185947558004 107.61363304615959 Simpang Dago -6.89149616345844 107.61312432715955 Simpang Djanda-Teku Umar -6.894786413330264 107.61287103746758 Simpang RS Borromeus -6.898804856398559 107.61269313046196 Simpang Tuan Dago -6.890213255525 107.6170268818955 Simpang Dipatiukun-Monumen Palimanan -6.89463922184651 107.61742042699116 Simpang Dipatiukun-Hasanuddin -6.89374284086656 107.61295595684201 Simpang Djanda-ITB -6.893741548818679 107.60841599249088 Simpang ITB-Kebut Binatang -6.887392289233773 107.61358772240874 Simpang Djanda-Dayang Sumbi -6.887396087763018 107.6111503218458 Pertigaan DayangSumbi -6.89683782352778 107.60963003467407 Simpang Plesiran -6.88501210763783 107.61152438003532 Pertigaan Babakan Sillwangi 0 0 0 0 958 0 0 0 250 400 0 240 0 0 0 0 558 0 250 0 458 0 0 0 0 0 450 0 500 120 0 0 0 0 0 0 0 450 0 700 0 0 0 0 500 500 500 0 0 270 0 0 0 0 0 0 0 0 500 700 270 0 0 0 0 0 0 0 250 120 0 0 0 500 0 0 0 0 0 0 0 0 0 500 0 0 1000 400 0 250 450 0 0 0 0 0 0 260 0 0 400 0 0 0 0 0 1000 200 0 280 0 0 0 500 0 0 0 400 0 0 0 0 240 0 0 0 0 0 0 0 280 0 0 </pre>	<p style="text-align: center;">Finding the Shortest Path</p>  <p>Chosen File: Refresh the page before starting a new search!</p> <p>Map Satellite</p> <p>UCS Path</p> <p>Astar Path</p> <p>Show Go</p> <p style="text-align: center;">Finding the Shortest Path</p>  <p>Chosen File: Refresh the page before starting a new search!</p> <p>Map Satellite</p> <p>UCS Path</p> <p>Astar Path</p> <p>Show Go</p>
<p style="text-align: center;">mapAlunAlun.txt</p>  <pre> File Edit View 12 -6.920550831904982 107.6065736792448 -6.918307987285537 107.604027119347843 Simpang Pasar Baru -6.920819923215465 107.60409307125431 Simpang Asia Afrika-Jenderal Sudirman -6.921240144867019 107.6077424251525 Monumen Asia Afrika -6.922547312703277 107.6076478376178 Simpang Alun Alun -6.92208758985743 107.60403021355373 Simpang Otto Iskandardinata-Dalem Kaum -6.922400178047693 107.60633793736483 Simpang Dewi Kartika -6.923226433496438 107.60629477025935 Pertigaan Jalan Kepatihan -6.923807126701684 107.6039302293089 Simpang Otto Iskandardinata-Kepatihan -6.921489178217627 107.60997477968604 Simpang Museum Konferensi Asia Afrika A -6.922789675045275 107.6097813589442 Simpang Jalan Honan -6.919719241202284 107.60993900575858 Simpang Braga -6.921446917317119 107.6099706187984611 Simpang Museum Konferensi Asia Afrika B 0 280 0 0 0 0 0 0 700 0 280 0 400 0 140 0 0 0 0 0 0 0 400 0 150 0 0 0 0 0 0 220 0 0 150 0 0 150 0 0 0 240 0 0 0 140 0 0 0 260 0 118 0 0 0 0 0 0 0 150 260 0 92 0 0 0 0 0 0 0 0 92 0 270 0 0 0 0 0 0 0 0 110 0 270 0 0 0 0 0 0 0 0 0 0 0 156 0 20 0 0 0 240 0 0 0 0 156 0 0 0 700 0 0 0 0 0 0 0 0 0 200 0 0 220 0 0 0 0 0 20 0 200 0 </pre>	<p style="text-align: center;">Finding the Shortest Path</p>  <p>Chosen File: Refresh the page before starting a new search!</p> <p>Map Satellite</p> <p>UCS Path</p> <p>Astar Path</p> <p>Show Go</p> <p style="text-align: center;">Finding the Shortest Path</p>  <p>Chosen File: Refresh the page before starting a new search!</p> <p>Map Satellite</p> <p>UCS Path</p> <p>Astar Path</p> <p>Show Go</p>

mapBuahBatu.txt

Finding the Shortest Path

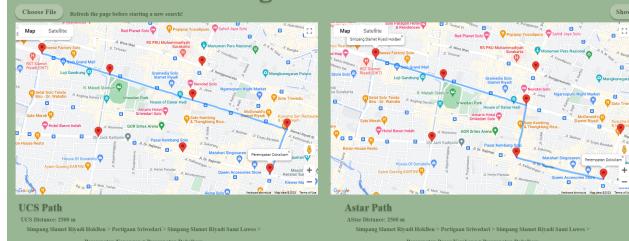


Finding the Shortest Path



mapSolo.txt

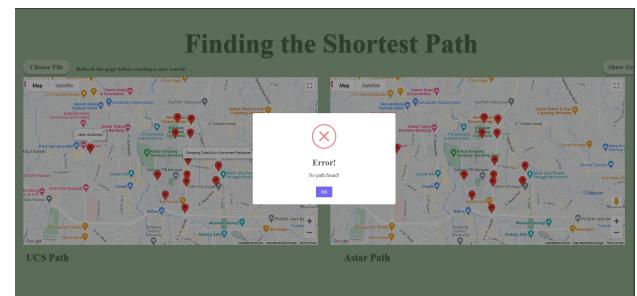
Finding the Shortest Path



Finding the Shortest Path



mapWrongInput.txt

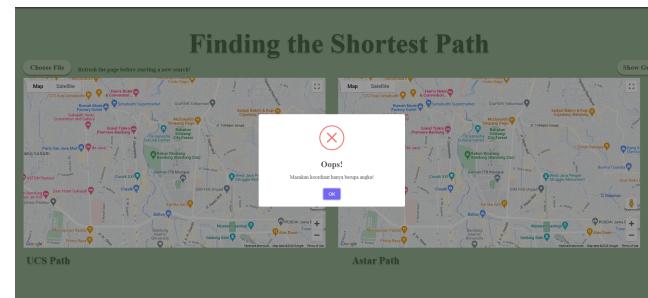


mapWrongInput2.txt

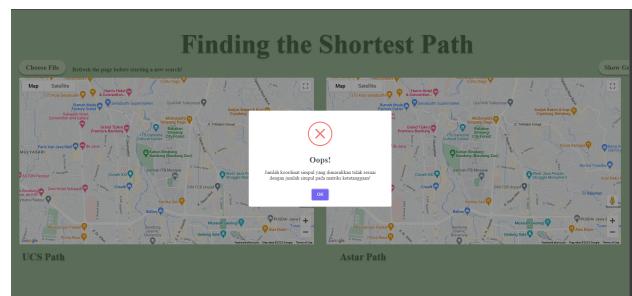
```
mapWrongInput X mapWrongInput mapSolo

File Edit View

13
-6.8907604807375 107.61809721506752
-6.885218584755904 107.6136338615959
Siapang Tago
-6.891496161454849 107.6112423715955
Tanjung Selor
-6.8947861330364 107.6187397637658
Siapang Rg Borromeu
-6.8863850398593 107.61269311046196
Siapang Rg Bintangor
-6.892484126338645 107.6178266811895
Siapang Upietuk-Monaten Palawau
-6.893742844086516 107.6159552693916
Siapang Upietuk-Husandutti
-6.897142844086516 107.619559684261
Siapang Upietuk-Sabah
-6.893749548018679 107.6048159924908
Siapang ITB Kebun Binatang
-6.88709280323793 107.61507772248874
Siapang Upietuk-Sabah
-6.887396087763018 107.6115231254856
Pertigaan DunguSemb
-6.887396087763018 107.61508308047687
Siapang Pleiwer
-6.88512102767312 107.61152438001512
Pertigaan DunguSemb
-6.8896415131929 107.5988988718349
Jalan Sukamaju
```



mapWrongInput3.txt



4.2 Analisis Hasil Keluaran Program

Dari pengujian yang telah dilakukan, didapatkan bahwa program dapat berjalan dengan baik dan berhasil menentukan lintasan terpendek dari simpul akar hingga simpul tujuan dengan tepat.

Pada pengujian mapITB.txt, dilakukan pengujian untuk menentukan lintasan terpendek dari Pertigaan DayangSumbi ke Simpang DipatiUkur-Hassanuddin. Percobaan tersebut memberikan hasil yang sama antara kedua algoritma, dengan rute terpendek adalah Pertigaan Dayang Sumbi → Simpang Djuanda-Dayang Sumbi → Simpang Djuanda-Teruku Umar → Simpang DipatiUkur - Monumen Pahlawan → Simpang DipatiUkur - Hassanuddin dengan total jarak 1530 meter.

Pada pengujian mapAlunAlun.txt, dilakukan pengujian untuk menentukan lintasan terpendek dari Simpang Asia Afrika - Jenderal Sudirman menuju Simpang Jalan Homan. Pada percobaan tersebut, penentuan lintasan terpendek dengan algoritma UCS memberikan hasil lintasan Simpang Asia Afrika - Jenderal Sudirman → Monumen Asia Afrika → Simpang Museum Konferensi Asia Afrika B → Simpang Museum Konferensi Asia Afrika A → Simpang Jalan Homan. Sedangkan pada algoritma A* memberikan hasil lintasan Simpang Asia Afrika - Jenderal Sudirman → Monumen Asia Afrika → Simpang Alun Alun → Simpang Jalan Homan. Walaupun lintasan yang dihasilkan berbeda, jarak dari kedua lintasan tersebut sama, yaitu sejauh 790 meter, sehingga didapatkan bahwa memang terdapat lebih dari satu alternatif jalan dengan jarak paling minimum dan hasil dari kedua algoritma tersebut valid atau benar.

Pada pengujian mapBuahBatu.txt, dilakukan pengujian untuk menentukan lintasan terpendek dari Simpang ISBI Bandung menuju Margacinta Park. Percobaan tersebut memberikan hasil yang sama antara kedua algoritma, dengan rute terpendek adalah Simpang ISBI Bandung → Lampu Merah Buah Batu → Simpang Padar Kordon → Tuju Kordon → margacinta Park dengan total jarak 2910 meter.

Pada pengujian mapSolo.txt, dilakukan pengujian untuk menentukan lintasan terpendek dari Simpang Slamet Riyadi Hokben menuju Perempatan Doksikam. Pada percobaan tersebut, penentuan lintasan terpendek dengan algoritma UCS memberikan hasil lintasan Simpang Slamet Riyadi Hokben → Pertigaan Sriwedari → Simpang Slamet Riyadi Sami Luwes → Perempatan Nonongan → Perempatan Doksikam. Sedangkan pada algoritma A* memberikan hasil lintasan Simpang Slamet Riyadi Hokben → Pertigaan Sriwedari → Simpang Slamet Riyadi Sami Luwes → Perempatan Pasar Kembang → Perempatan Doksikam. Walaupun lintasan yang dihasilkan berbeda, jarak dari kedua lintasan tersebut sama, yaitu sejauh 2500 meter, sehingga didapatkan bahwa memang terdapat lebih dari satu alternatif jalan dengan jarak paling minimum dan hasil dari kedua algoritma tersebut valid atau benar.

Pengujian terhadap jalur yang tidak ditemukan berhasil dilakukan pada pengujian file mapWrongInput.txt dimana terdapat sebuah simpul Jalan Sudirman yang tidak terhubung dengan simpul manapun. Pengujian ini juga berhasil memunculkan pesan kesalahan “No path found”.

Pengujian terhadap masukkan yang salah dilakukan dengan memasukkan file mapWrongInput2.txt serta mapWrongInput3.txt. Pengujian dengan kedua file tersebut berhasil memunculkan pesan kesalahan yang sesuai. Pada mapWrongInput2.txt, kesalahan terdapat pada masukkan elemen adjacency matriks yang tidak bertipe numerik, sedangkan kesalahan pada mapWrongInput2.txt dikarenakan ketidaksesuaian jumlah simpul masukkan dengan dimensi matriks ketetanggaan yang terdapat pada file tersebut.

BAB 5

PENUTUP

5.1. Kesimpulan

Algoritma Uniform Cost Search (UCS) dan algoritma A* adalah algoritma yang sering digunakan dalam pemecahan permasalahan pencarian rute pada graf. UCS menggunakan pendekatan breadth-first-search dengan mempertimbangkan biaya atau cost. Dalam pencarian rute terdekat, cost atau biaya yang diperhatikan merupakan jarak antara simpul yang akan ekspansi dengan simpul asal. Simpul dengan jarak terdekat akan diprioritaskan untuk ditelusuri terlebih dahulu. Algoritma A* menggabungkan algoritma pencarian heuristik dan uniform cost search.

Kedua algoritma tersebut memiliki kelebihan dan kekurangan masing-masing, di mana UCS cenderung sederhana namun relatif lambat tanpa adanya batasan cost atau biaya dalam proses pencarian, sedangkan A* cenderung lebih efisien namun memerlukan penentuan fungsi heuristik yang tepat. Namun secara umum penggunaan algoritma UCS dan A* sudah cukup efektif dan memberikan hasil yang optimum pada pemecahan masalah pencarian jalur terpendek atau optimal.

5.2. Saran

Setelah mengimplementasikan algoritma UCS dan A* pada Tugas Besar ini, kami menyarankan agar program dapat dibuat lebih rapi lagi. Selain itu, penting untuk memahami secara detail terkait algoritma UCS dan A* sebelum mengimplementasikannya. Pengujian terhadap berbagai jenis test case juga diperlukan sehingga kita dapat memahami efisiensi penggunaan kedua algoritma tersebut dalam pemecahan masalah pencarian jalur terpendek atau optimal.

REFERENSI

Hasan, Fatima. (n.d.). *What is uniform-cost search?* Educative: Interactive Courses for Software Developers. Retrieved April 10, 2023, from

<https://www.educative.io/answers/what-is-uniform-cost-search>

Searching: Uniform Cost Search. (2017, August 24). School of Computer Science.

Retrieved April 10, 2023, from

<https://socs.binus.ac.id/2017/08/24/searching-uniform-cost-search/>

Munir, Rinaldi (2021). Bagian 1: BFS, DFS, UCS, Greedy Best First Search. Diakses pada 08 April 2023 dari sumber

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf>

Munir, Rinaldi (2021). Penentuan Rute Bagian 2: Algoritma A*. Diakses pada 08 April 2023 dari sumber

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf>

Stuart J Russell & Peter Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Prentice-Hall International, Inc, 2010, Textbook. Site: <http://aima.cs.berkeley.edu/> (2nd edition)

LAMPIRAN

Link Repository Github

https://github.com/BrigitaCarolina/Tucil3_13521077_13521156.git

Checklist Fitur

Poin	Ya	Tidak
Program dapat menerima input graf	✓	
Program dapat menghitung lintasan terpendek dengan UCS	✓	
Program dapat menghitung lintasan terpendek dengan A*	✓	
Program dapat menampilkan lintasan terpendek serta jaraknya	✓	
Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta serta lintasan terpendek pada peta	✓	