TUGAS KECIL III IF2211 STRATEGI ALGORITMA

SEMESTER II TAHUN 2022/2023

Implementasi Algoritma UCS dan A* untuk Menentukan Lintasan Terpendek

Disusun oleh:

Dhanika Novlisariyanti 13521132

I Putu Bakta Hari Sudewa 13521150

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

1. 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. Spesifikasi program: 1. Program menerima input file graf (direpresentasikan sebagai matriks ketetanggaan berbobot), jumlah simpul minimal 8 buah. 2. Program dapat menampilkan peta/graf 3. Program menerima input simpul asal dan simpul tujuan. 4. Program dapat menampilkan lintasan terpendek beserta jaraknya antara simpul asal dan simpul tujuan. 5. Antarmuka program bebas, apakah pakai GUI atau command line saja

A. UCS

UCS atau Uniform Cost Search adalah algoritma yang mencari rute tanpa informasi menggunakan *cost* terendah untuk menemukan rute dari node sumber ke node tujuan. Algoritma ini beroperasi di sekitar ruang pencarian berbobot terarah untuk berpindah dari node awal ke salah satu node akhir dengan *cost* minimum.

Langkah-langkah menggunakan UCS:

a. Masukan *node root* ke dalam *priority queue*
b. Hapus elemen dari prioritas tertinggi
c. Cek apakah *node* tersebut sama dengan goal atau tidak. Jika tidak masukkan *node* yang bertetangga dengan *node* tersebut ke dalam *priority queue.*
d. Jika node yang dihapus adalah node tujuan, cek *cost* dan hentikan algoritma

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the only element
  explored ← an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node ← 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 ← CHILD-NODE(problem, node, action)
      if child.STATE is not in explored or frontier then
        frontier ← INSERT(child, frontier)
      else if child.STATE is in frontier with higher PATH-COST then
        replace that frontier node with child
```

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

B. A*

Algoritma A* adalah salah satu algoritma yang digunakan dalam mencari rute dan graf traversal. Pada setiap langkah, algoritma A* memilih *node* berdasarkan *f-value* dimana *f-value* adalah *g-value ditambah h-value. G-value* merupakan total weight dari matriks ketetanggaan dan *h-value* dimana adalah nilai heuristik atau nilai straight line dari node awal ke node tersebut. Cara kerja algoritma A* adalah sebagai berikut:

```
make an openlist containing only the starting node
   make an empty closed list
   while (the destination node has not been reached):
       consider the node with the lowest f score in the open list
       if (this node is our destination node) :
           we are finished
       if not:
           put the current node in the closed list and look at all
of its neighbors
           for (each neighbor of the current node):
               if (neighbor has lower g value than current and is
in the closed list) :
                   replace the neighbor with the new, lower, g
value
                   current node is now the neighbor's parent
               else if (current g value is lower and this neighbor
is in the open list ) :
                   replace the neighbor with the new, lower, g
value
                   change the neighbor's parent to our current
node

               else if this neighbor is not in both lists:
```

```
                    add it to the open list and set its g
```

Sumber: https://brilliant.org/wiki/a-star-search/

2. Kode Program

A*.ts

```
import calculate_distance, {
    Connections,
    ElementAStar,
    Nodes,
    Path,
    PriorityQueueAStar,
    QueueElementAStar,
} from "../lib/utils";

export default function astar(
    nodes: Nodes,
    connections: Connections,
    start: string,
    destination: string
) {
    let elements: ElementAStar = {};
    for (let label in nodes) {
        elements[label] = {
            g_score: Infinity,
            f_score: Infinity,
        };
    }
    elements[start].g_score = 0;
    elements[start].f_score = calculate_distance(
        nodes[start].latitude,
        nodes[start].longitude,
        nodes[destination].latitude,
        nodes[destination].longitude
    );

    let open_nodes = new PriorityQueueAStar();
    let node: QueueElementAStar = {
        heuristic_score: elements[start].f_score,
        final_score: elements[start].f_score,
        node: start,
    };
    open_nodes.enqueue(node);
```

```
    let raw_path: Path = {};
    let total_cost = 0;
    while (!open_nodes.empty()) {
        let current_node = open_nodes.dequeu()?.node;
        if (current_node === destination) {
            total_cost = elements[current_node].g_score;
            break;
        }

        for (let neighbor_node of connections[current_node!]) {
            let temp_g_score =
                elements[current_node!].g_score +
                calculate_distance(
                    nodes[current_node!].latitude,
                    nodes[current_node!].longitude,
                    nodes[neighbor_node].latitude,
                    nodes[neighbor_node].longitude
                );

            let heuristic_child_node = calculate_distance(
                nodes[neighbor_node].latitude,
                nodes[neighbor_node].longitude,
                nodes[destination].latitude,
                nodes[destination].longitude
            );

            let temp_f_score = temp_g_score +
heuristic_child_node;

            if (temp_f_score < elements[neighbor_node].f_score) {
                elements[neighbor_node].g_score = temp_g_score;
                elements[neighbor_node].f_score = temp_f_score;

                let child_node: QueueElementAStar = {
                    heuristic_score: heuristic_child_node,
                    final_score: temp_f_score,
                    node: neighbor_node,
                };

                open_nodes.enqueue(child_node);

                raw_path[neighbor_node] = current_node!;
            }
        }
    }
```

```
    let raw_final_path: Path = {};
    let final_path: string[] = [];
    let reviewed_node: string = destination;

    while (reviewed_node !== start) {
        raw_final_path[raw_path[reviewed_node]] = reviewed_node;
        final_path.splice(0, 0, nodes[reviewed_node].name);
        reviewed_node = raw_path[reviewed_node];
    }
    final_path.splice(0, 0, nodes[start].name);

    return {
        raw_path: raw_final_path,
        path: final_path,
        cost: total_cost,
    };
}

export function astarToString(path: string[]) {
    let beautified_path = "";
    for (let i = 0; i < path.length; i++) {
        if (i != 0) {
            beautified_path += " -> ";
        }
        beautified_path += path[i];
    }
    return beautified_path;
}
```

UCS.ts

```
import calculate_distance, { Nodes, Path, Connections, ElementUCS,
QueueElementUCS, PrioQueueUCS} from '../lib/utils';

export default function ucs(
  nodes: Nodes,
  connections: Connections,
  start: string,
  goal: string
) {
  let queueUCS = new PrioQueueUCS();
  let node: QueueElementUCS = {
    final_score: 0,
    node: start
  };
```

```
  queueUCS.enqueue(node);

  let raw_path: Path = {};
  let costSoFar: ElementUCS = {};
  costSoFar[start] = { f_score: 0 };
  let explored: Set<string> = new Set();
  while (!queueUCS.isEmpty()) {
    let currentElement = queueUCS.dequeue();
    let current = currentElement!.node;

    if (current === goal) {
      break;
    }

    explored.add(current);

    let neighbors = connections[current];
    if (neighbors) {
      for (const neighbor of neighbors) {
        if (!explored.has(neighbor)) {
          let newCost = costSoFar[current].f_score +
calculate_distance(nodes[current].latitude,
nodes[current].longitude, nodes[neighbor].latitude,
nodes[neighbor].longitude);
          if (!costSoFar[neighbor] || newCost <
costSoFar[neighbor].f_score) {
            costSoFar[neighbor] = { f_score: newCost };
            let priority = newCost;
            let newElement: QueueElementUCS = {
              final_score: priority,
              node: neighbor
            };
            queueUCS.enqueue(newElement);
            raw_path[neighbor] = current;
          }
        }
      }
    }
  }

  let raw_final_path: Path = {};
  let final_path: string[] = [];
  let reviewed_node: string = goal;
  while (reviewed_node !== start) {
    raw_final_path[raw_path[reviewed_node]] = reviewed_node;
    final_path.splice(0, 0, nodes[reviewed_node].name);
```

```
    reviewed_node = raw_path[reviewed_node];
  }
  final_path.splice(0, 0, nodes[start].name);

  return {
    raw_path: raw_final_path,
    path: final_path,
    cost: costSoFar[goal] ? costSoFar[goal].f_score : undefined
  };
}

export function ucsToString(path: string[]) {
  let beautified_path = "";
  for (let i = 0; i < path.length; i++) {
    if (i != 0) {
      beautified_path += " -> ";
    }
    beautified_path += path[i];
  }
  return beautified_path;
}
```

Utils.ts

```
"use client";

export interface Nodes {
  [key: string]: {
    name: string;
    latitude: number;
    longitude: number;
  };
}

export interface Connections {
  [key: string]: string[];
}

export interface ElementUCS {
  [key: string]: {
    f_score: number;
  };
```

```
}
export interface QueueElementUCS {
  final_score: number;
  node: string;
}

export interface Edges {
  lat_init: number;
  lng_init: number;
  lat_target: number;
  lng_target: number;
  name: string;
  node_init: number;
  node_target: number;
}

export interface QueueElementAStar {
  final_score: number;
  heuristic_score: number;
  node: string;
}

export interface ElementAStar {
  [key: string]: {
    g_score: number;
    f_score: number;
  };
}

export interface Path {
  [key: string]: string;
}

export default function calculate_distance(
  lat1: number,
  lon1: number,
  lat2: number,
  lon2: number
) {
```

```
  const radlat1 = (Math.PI * lat1) / 180;
  const radlat2 = (Math.PI * lat2) / 180;
  const theta = lon1 - lon2;
  const radtheta = (Math.PI * theta) / 180;
  let dist =
    Math.sin(radlat1) * Math.sin(radlat2) +
    Math.cos(radlat1) * Math.cos(radlat2) * Math.cos(radtheta);
  if (dist > 1) dist = 1;
  dist = Math.acos(dist);
  dist = (dist * 180) / Math.PI;
  dist = dist * 60 * 1.1515;
  dist = dist * 1.609344;
  return dist * 1000;
}

export class QElement {
  element: QueueElementAStar;
  constructor(element: QueueElementAStar) {
    this.element = element;
  }
}

export class QElementUCS {
  elementUCS: QueueElementUCS;
  constructor(elementUCS: QueueElementUCS) {
    this.elementUCS = elementUCS;
  }
}

export class PriorityQueueAStar {
  items: QueueElementAStar[];
  constructor() {
    this.items = [];
  }

  enqueue(element: QueueElementAStar) {
    let qElement = new QElement(element);
    let contain = false;
```

```
    for (let i = 0; i < this.items.length; i++) {
      if (this.items[i].final_score >
qElement.element.final_score) {
        this.items.splice(i, 0, qElement.element);
        contain = true;
        break;
      }
      if (this.items[i].heuristic_score >
qElement.element.heuristic_score) {
        this.items.splice(i, 0, qElement.element);
        contain = true;
        break;
      }
    }

    if (!contain) {
      this.items.push(qElement.element);
    }
  }

  dequeu() {
    return this.items.shift();
  }

  empty() {
    return this.items.length === 0;
  }
}

export class PrioQueueUCS {
  queueUCS: QueueElementUCS[];

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

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

```
  size() {
    return this.queueUCS.length;
  }

  isFull() {
    return this.queueUCS.length == this.size();
  }

  enqueue(elementUCS: QueueElementUCS) {
    let qelementUCS = new QElementUCS(elementUCS);
    this.queueUCS.push(qelementUCS.elementUCS);

    for (let i = 0; i < this.size(); i++) {
      for (let j = 0; j < this.size() - i - 1; j++) {
        if (this.queueUCS[j].final_score > this.queueUCS[j +
1].final_score) {
          const temp = this.queueUCS[j];
          this.queueUCS[j] = this.queueUCS[j + 1];
          this.queueUCS[j + 1] = temp;
        }
      }
    }
  }

  dequeue() {
    return this.queueUCS.shift();
  }
}
```

Page.tsx

```
"use client";

import astar, {astarToString} from "./algorithms/astar";
import ucs, {ucsToString} from "./algorithms/ucs";
import Dropdown from "./components/dropdown";
import Map from "./components/map";
```

```
import {Connections, Edges, Nodes, Path} from "./lib/utils";
import {ChangeEvent, use, useState} from "react";
import {ToastContainer, toast} from "react-toastify";
import GraphComponent from "./graph/page";
import "react-toastify/dist/ReactToastify.css";
import L from "leaflet";
import SwitchFile from "./components/switch_file";
import SwitchNewNode from "@/app/components/switch_new_node";
import DropdownNewEdge from "@/app/components/dropdown_new_edge";

interface FormElements extends HTMLFormControlsCollection {
    input_file: HTMLInputElement;
}

interface FileInputFormElement extends HTMLFormElement {
    readonly elements: FormElements;
}

export default function Home() {

    const [nodes, setNodes] = useState<Nodes>({});
    const [connections, setConnections] =
useState<Connections>({});
    const [initial, setInitial] = useState<string>();
    const [target, setTarget] = useState<string>();
    const [path, setPath] = useState<Path>();
    const [algorithm, setAlgorithm] = useState<string>("astar");
    const [displayRoute, setdisplayRoute] = useState<String>();
    const [displayCost, setdisplayCost] = useState<Number>();
    const [displayTime, setdisplayTime] = useState<String>();
    const [markers, setMarkers] = useState<L.Marker[]>([]);
    const [polylines, setPolylines] = useState<L.Polyline[]>([]);
    const [edges, setEdges] = useState<Edges[]>([]);
    const [mode, setMode] = useState<string>("file");
    const [removeAll, setRemoveAll] = useState<boolean>(false);
    const [isNewNode, setIsNewNode] = useState<boolean>(true);
    const [selectedInitialNode, setSelectedInitialNode] =
useState<string>("0");
    const [selectedTargetNode, setSelectedTargetNode] =
```

```
useState<string>();
    const [enabled, setEnabled] = useState(false);

    const [initialRemove, setInitialRemove] = useState<number>();
    const [targetRemove, setTargetRemove] = useState<number>();

    const handleSetPolyline = (polylines: L.Polyline[]) => {
        setPolylines(polylines);
    };

    const handleSetEdges = (edges: Edges[]) => {
        setEdges(edges);
    };

    const handleInitialButton = (value: string) => {
        setInitial(value);
    };

    const handleTargetButton = (value: string) => {
        setTarget(value);
    };

    const handleStartButton = () => {
        if (nodes && connections && initial && target) {
            if (algorithm === "astar") {
                let startTime = performance.now();
                const result = astar(nodes, connections, initial,
target);
                let endTime = performance.now();
                setPath(result.raw_path);
                let resultString = astarToString(result.path);
                setdisplayRoute(resultString);
                setdisplayTime((endTime -
startTime).toFixed(4).toString());
                setdisplayCost(result.cost);
            }
            if (algorithm === "ucs") {
                let startTime = performance.now();
                let final_path = ucs(nodes, connections, initial,
```

```
target);
                let endTime = performance.now();
                let stringPath = ucsToString(final_path.path);
                let cost = (final_path.cost);
                setPath(final_path.raw_path);
                setdisplayRoute(stringPath);
                setdisplayTime((endTime -
startTime).toFixed(4).toString());
                setdisplayCost(cost);
            }
        } else if (nodes && connections && !initial && !target) {
            toast.error("Initial and target has not been
decided!", {
                position: toast.POSITION.TOP_CENTER,
            });
        } else if (nodes && connections && initial && !target) {
            toast.error("Target has not been decided!", {
                position: toast.POSITION.TOP_CENTER,
            });
        } else if (nodes && connections && !initial && target) {
            toast.error("Start node has not been decided!", {
                position: toast.POSITION.TOP_CENTER,
            });
        } else {
            toast.error("No Input File!", {
                position: toast.POSITION.TOP_CENTER,
            });
        }
    };


    const handleInputFile = (event: ChangeEvent<HTMLInputElement>)
=> {
        setInitial(undefined);
        setTarget(undefined);
        if (event.currentTarget.files!.length > 0) {
            const reader = new FileReader();
            reader.onload = async (e) => {
                const text = e.target?.result as string;
```

```
                const raw_nodes: Nodes = {};
                const rows = text.split("\n");
                const data_amount = parseInt(rows[0]);

                for (let i = 1; i <= data_amount; i++) {
                    if (rows[i].trim().split(" ").length != 2) {
                        toast.error("Invalid input. Invalid node",
{
                            position: toast.POSITION.TOP_CENTER,
                        });
                        return;
                    }
                    const label = rows[i].trim().split(" ")[0];
                    const coors = rows[i].trim().split("
")[1].slice(1, -1);
                    const latitude =
parseFloat(coors.split(",")[0]);
                    const longitude =
parseFloat(coors.split(",")[1]);

                    if (isNaN(latitude) || isNaN(longitude)) {
                        toast.error("Invalid input. Invalid
coordinates", {
                            position: toast.POSITION.TOP_CENTER,
                        });
                        return;
                    }

                    if (!enabled && (latitude < -180 || latitude >
180 || longitude < -180 || longitude > 180)) {
                        toast.error("Invalid input. Invalid
latitude or longitude", {
                            position: toast.POSITION.TOP_CENTER,
                        });
                        return;
                    }

                    raw_nodes[i - 1] = {
                        name: label,
```

```
                        latitude,
                        longitude,
                    };
                }

                const raw_connections: Connections = {};
                for (let i = data_amount + 1; i <= 2 *
data_amount; i++) {
                    const cols = rows[i].trim().split(" ");
                    if (cols.length < data_amount) {
                        toast.error("Invalid input. Not complete
adj matrix", {
                            position: toast.POSITION.TOP_CENTER,
                        });
                        return;
                    }
                    for (let j = 0; j < data_amount; j++) {
                        if (cols[j] !== "1" && cols[j] !== "0") {
                            toast.error("Invalid input. Invalid
adj matrix element", {
                                position:
toast.POSITION.TOP_CENTER,
                            });
                            return;
                        }
                        if (cols[j] === "1") {
                            raw_connections[i - data_amount - 1]
                                ? raw_connections[i - data_amount
- 1].push(j.toString())
                                : (raw_connections[i - data_amount
- 1] = [j.toString()]);
                        }
                    }
                }

                setNodes(raw_nodes);
                setConnections(raw_connections);
                setPath(undefined);
                setdisplayCost(undefined);
```

```
                setdisplayRoute(undefined);
                setdisplayTime(undefined);
            };
            reader.readAsText(event.currentTarget.files![0]);
            event.currentTarget.value = "";
        }
    };

    const handleSubmit = async (event:
React.FormEvent<FileInputFormElement>) => {
        event.preventDefault();
    };

    return (
        <div className="text-biru h-screen">
            <div className="fixed right-3 top-3 z-[100000]
bg-card-color p-2">
                <div className="flex flex-col items-start">
                    <SwitchFile
                        setFile={(value) => {
                            setRemoveAll(true);
                            setEdges([]);
                            setNodes({});
                            setConnections({});
                            setInitial(undefined);
                            setTarget(undefined);
                            setdisplayRoute(undefined);
                            setdisplayTime(undefined);
                            setPath(undefined)
                            value ? setMode("file") :
setMode("manual");
                        }}
                    />
                    <SwitchNewNode setNewNode={(value) =>
setIsNewNode(value)}/>
                    {!isNewNode &&
                        <DropdownNewEdge nodes={nodes}
currentNode={selectedInitialNode!} connections={connections}
                                         setConnections={(value)
```

```
=> setConnections(value)}
                                            setEdges={(value) =>
setEdges(value)}
                                            edges={edges}
                        />}
                </div>
            </div>

            {enabled === true ? (
                <GraphComponent nodes={nodes!}
connections={connections!} path={path!}/>
            ) : (
                <Map
                    mode={mode}
                    nodes={nodes!}
                    connections={connections!}
                    path={path}
                    removeAll={removeAll}
                    markers={markers}
                    polylines={polylines!}
                    edges={edges!}
                    isNewNode={isNewNode}
                    initial={initialRemove}
                    target={targetRemove}
                    setInitialRemove={(value) =>
setInitialRemove(value)}
                    setTargetRemove={(value) =>
setTargetRemove(value)}
                    handleSetPolyline={(value) =>
handleSetPolyline(value!)}
                    handleSetEdges={(value) =>
handleSetEdges(value!)}
                    handleSetConnections={(value) =>
setConnections(value!)}
                    handleSetNodes={(value) => setNodes(value!)}
                    handleSetMarkers={(value) =>
setMarkers(value)}
                    handleSetRemoveAll={(value) =>
setRemoveAll((value))}
```

```
                    setSelectedInitialNode={(value) =>
setSelectedInitialNode(value)}
                />
            )}


            <ToastContainer/>


            <div className="fixed left-14 top-3 z-[100000]">
                <div className="block max-w-md p-7 bg-card-color">
                    <img src="/findroute.svg" className="mb-8"/>
                    <div>
                        <label className="inline-flex relative
items-center cursor-pointer">
                            <input
                                type="checkbox"
                                className="sr-only peer"
                                checked={enabled}
                                readOnly
                            />
                            <div
                                onClick={() => {
                                    setEnabled(!enabled);
                                    setNodes({});
                                    setConnections({});
                                }}
                                className="w-11 h-6 bg-gray-200
rounded-full peer  peer-focus:ring-green-300
peer-checked:after:translate-x-full
peer-checked:after:border-white after:content-[''] after:absolute
after:top-0.5 after:left-[2px] after:bg-white
after:border-gray-300 after:border after:rounded-full after:h-5
after:w-5 after:transition-all peer-checked:bg-blue-600"
                            ></div>
                            <span className="ml-2 text-sm
font-medium text-gray-900">
                                Graph
                            </span>
```

```
                        </label>
                    </div>
                    <div className="flex space-x-14">
                        <div className="max-w-lg space-y-4">
                            <div className="max-w-xs">
                                <form className="space-y-6"
onSubmit={handleSubmit}>
                                    <div
                                        className={`relative w-32
text-white ${
                                            mode === "file"
                                                ? "bg-blue-300
hover:bg-blue-800"
                                                : "bg-zinc-300"
                                        } font-medium rounded-lg
text-sm px-4 py-2.5 text-center inline-flex items-center`}
                                    >
                                        <input
                                            className="absolute
z-[-1] w-full"
                                            type="file"
                                            accept=".txt"
                                            id="input_file"
                                            onChange={(e) => {

handleInputFile(e);
                                            }}
                                            disabled={mode ===
"manual"}
                                        />
                                        <svg
                                            className="relative
w-5 h-5"
                                            fill="currentColor"

xmlns="http://www.w3.org/2000/svg"
                                            viewBox="0 0 512 512"
                                        >
                                            <path
```

```
                                                    d="M288
109.3V352c0 17.7-14.3 32-32 32s-32-14.3-32-32V109.3l-73.4
73.4c-12.5 12.5-32.8 12.5-45.3 0s-12.5-32.8
0-45.3l128-128c12.5-12.5 32.8-12.5 45.3 0l128 128c12.5 12.5 12.5
32.8 0 45.3s-32.8 12.5-45.3 0L288 109.3zM64 352H192c0 35.3 28.7 64
64 64s64-28.7 64-64H448c35.3 0 64 28.7 64 64v32c0 35.3-28.7 64-64
64H64c-35.3 0-64-28.7-64-64V416c0-35.3 28.7-64 64-64zM432 456a24
24 0 1 0 0-48 24 24 0 1 0 0 48z"/>
                                            </svg>
                                            <label
htmlFor="input_file">Upload File</label>
                                        </div>
                                    </form>
                                </div>
                                <div className="max-w-xs">
                                    <p
className="text-blue">Algorithm</p>
                                    <div className="flex items-center
mb-4">
                                        <input
                                            id="default-radio-1"
                                            type="radio"
                                            value=""
                                            name="default-radio"
                                            className="w-4 h-4
text-blue-600 bg-gray-100 border-gray-300 focus:ring-blue-500
focus:ring-2"
                                            onChange={() =>
setAlgorithm("ucs")}
                                        />
                                        <label
                                            htmlFor="default-radio-1"
                                            className="ml-2 text-sm
font-medium text-blue"
                                        >
                                            UCS
                                        </label>
                                    </div>
                                    <div className="flex
```

```
items-center">
                                    <input
                                        defaultChecked
                                        id="default-radio-2"
                                        type="radio"
                                        value=""
                                        name="default-radio"
                                        className="w-4 h-4
text-blue-600 bg-gray-100 border-gray-300 focus:ring-blue-500
focus:ring-2"
                                        onChange={() =>
setAlgorithm("astar")}
                                    />
                                    <label
                                        htmlFor="default-radio-2"
                                        className="ml-2 text-sm
font-medium text-blue"
                                    >
                                        A*
                                    </label>
                                </div>
                            </div>
                        </div>
                        <div className="text-biru space-y-5">
                            <div className="flex space-x-6
text-blue">
                                <Dropdown
                                    label={initial && nodes ?
nodes[initial].name : "Init"}
                                    nodes={nodes}
                                    setNode={(value) =>
handleInitialButton(value)}
                                />
                                <Dropdown
                                    label={target && nodes ?
nodes[target].name : "Target"}
                                    nodes={nodes}
                                    setNode={(value) =>
handleTargetButton(value)}
```

```
                                />
                            </div>
                            <button
                                type="button"
                                className="text-white w-full h-10
bg-blue-300 hover:bg-blue-800 font-medium rounded-lg text-sm px-5
py-2.5 mr-2 mb-2 focus:outline-none shadow-lg shadow-blue-500/50"
                                onClick={handleStartButton}
                            >
                                Start
                            </button>
                            <div className="space-x-3">
                                <p onChange={handleStartButton}
className="text-blue">
                                    Execution Time: {displayTime}
                                </p>
                            </div>
                        </div>
                    </div>
                    <div className="space-y-24">
                        <div className="space-y-2">
                            <p className="text-blue">Route</p>
                            <p className="text-blue"
onChange={handleStartButton}>
                                {displayRoute}
                            </p>
                        </div>
                        <div className="space-y-5">
                            <p className="text-blue">Distance
{displayCost}</p>
                        </div>
                    </div>
                    {mode === 'manual' &&
                        <div>
                            <div className='border border-zinc-300
my-2'></div>
                            <p className='font-bold my-2'>Edges
List</p>
                            <p className='text-sm mb-2'>Click
```

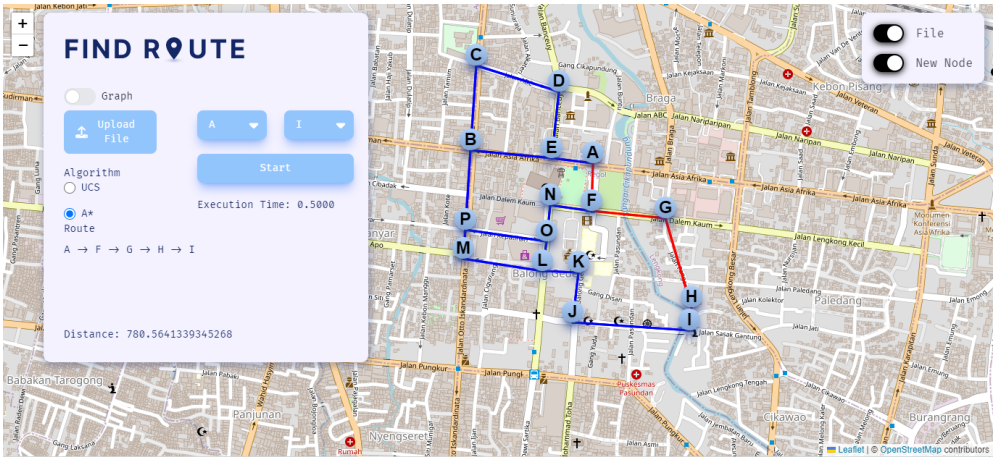
```
buttons below to remove the edge</p>
                            <div className='max-h-44
overflow-auto'>
                                {edges &&
                                    edges.map((edge, i) => (
                                        <div key={i}>
                                            <button
                                                onClick={() => {

setInitialRemove(edge.node_init)

setTargetRemove(edge.node_target)

edges.splice(i, 1);
                                                }}

className='hover:bg-blue-800 text-white justify-center w-full flex
bg-blue-300 my-2 rounded-md px-4 py-2 items-center'>
                                                {edge.node_init} -
{edge.node_target}
                                            </button>
                                        </div>
                                    ))}
                            </div>
                        </div>
                    }
                </div>
            </div>
        </div>
    );
}
```

3. Contoh masukan dan luaran(skrinsut)

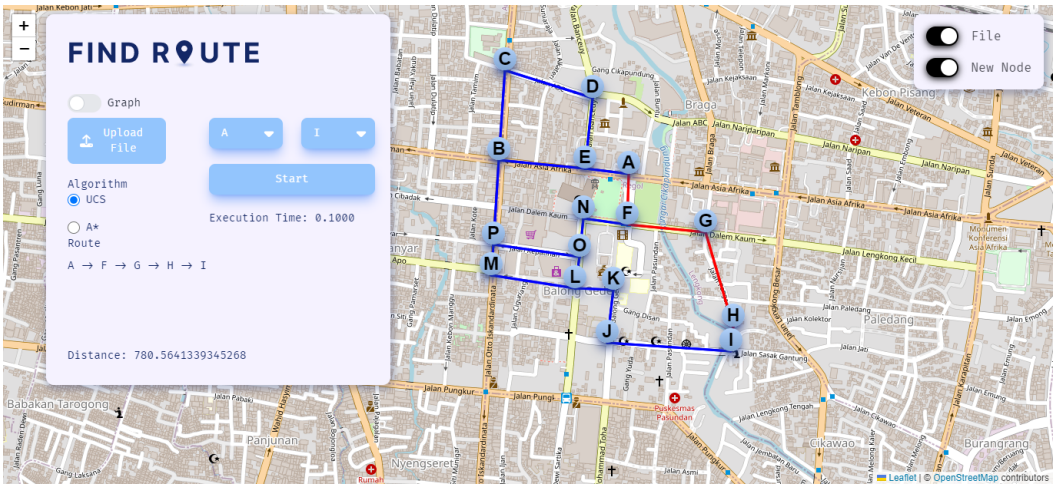
a. Alun.txt
   Start Node: A
   Goal Node: I

```
16
A (-6.921141,107.607668)
B (-6.920768,107.604056)
C (-6.918263,107.604216)
D (-6.919038,107.606670)
E (-6.920995,107.606441)
F (-6.922551,107.607619)
G (-6.922771,107.609810)
H (-6.925376,107.610599)
I (-6.926075,107.610524)
J (-6.925835,107.607071)
K (-6.924356,107.607253)
L (-6.924317,107.606160)
M (-6.923950,107.603842)
N (-6.922388,107.606404)
O (-6.923443,107.606298)
P (-6.923100,107.603892)
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1
0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1
0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0
```

alun.txt



A dari alun.txt*

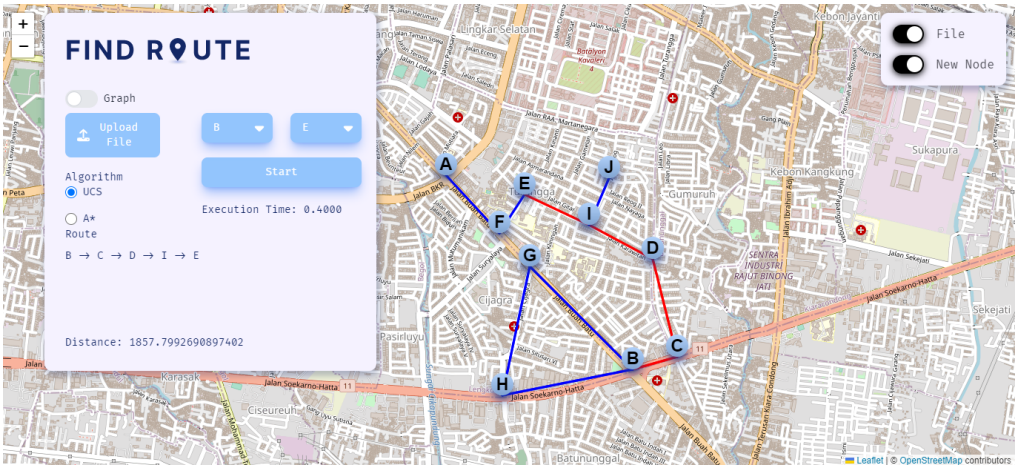


UCS dari alun.txt

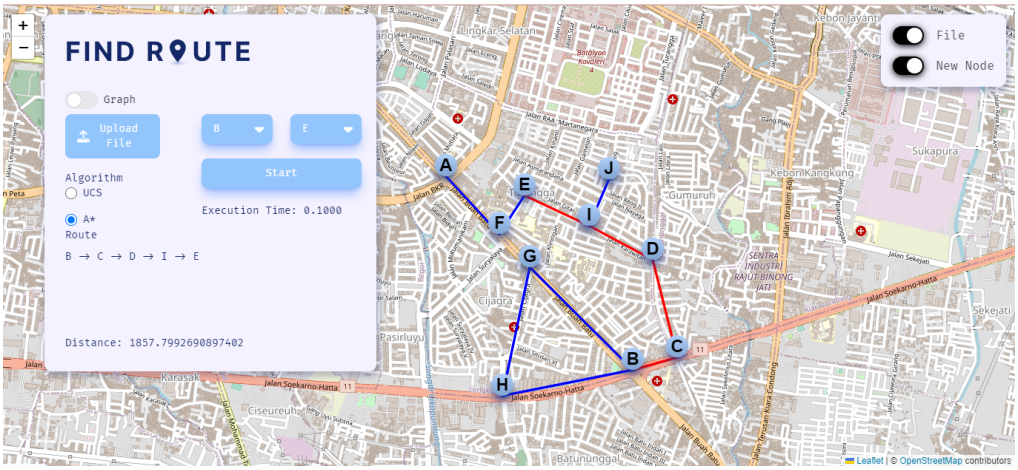
b. Buahbatu.txt
Start Node: N
Goal Node: B

```
10
A (-6.936757,107.622691)
B (-6.947901,107.633486)
C (-6.947165,107.636042)
D (-6.941532,107.634648)
E (-6.93784,107.6272)
F (-6.94008,107.62576)
G (-6.94199,107.62754)
H (-6.94937,107.62593)
I (-6.9396,107.63096)
J (-6.93691,107.63211)
0 0 0 0 0 1 0 0 0 0
0 0 1 0 0 0 1 1 0 0
0 1 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0 1 0
1 0 0 0 1 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0
0 1 0 0 0 0 1 0 0 0
0 0 0 1 1 0 0 0 0 1
0 0 0 0 0 0 0 0 1 0
```

Buahbatu.txt



Hasil A buahbatu.txt*

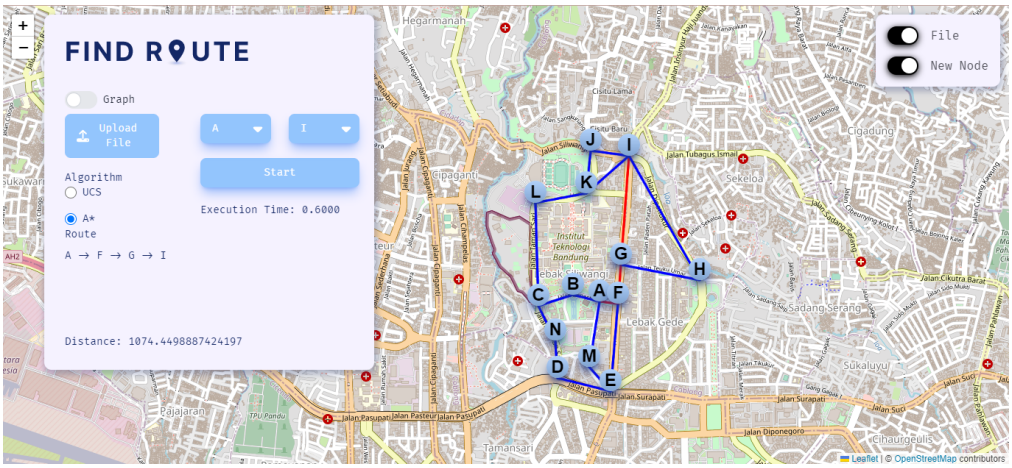


Hasil UCS buahbatu.txt

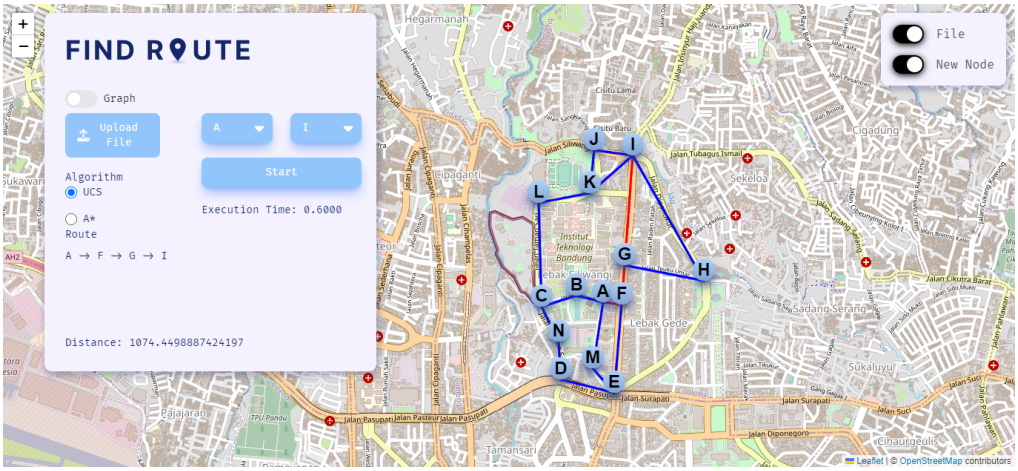
c. Itbdago.txt
Start Node: A
Goal Node: I

```
14
A (-6.893634,107.611970)
B (-6.893252,107.610425)
C (-6.893890,107.608426)
D (-6.898057,107.609559)
E (-6.898827,107.612643)
F (-6.893761,107.613038)
G (-6.891472,107.613236)
H (-6.892391,107.617819)
I (-6.885191,107.613701)
J (-6.884902,107.611468)
K (-6.887359,107.611214)
L (-6.887895,107.608260)
M (-6.897417,107.611396)
N (-6.895880,107.609392)
0 1 0 0 0 1 0 0 0 0 0 0 1 0
1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 1 0 1
0 0 0 0 1 0 0 0 0 0 0 0 0 1
0 0 0 1 0 1 0 0 0 0 0 0 1 0
1 0 0 0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 1 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 0 0
0 0 0 0 0 0 1 1 0 1 1 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0 1 1 0 1 0 0
0 0 1 0 0 0 0 0 0 0 1 0 0 0
1 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0 0 0 0 0
```

itbdago.txt



Hasil A itbdago.txt*

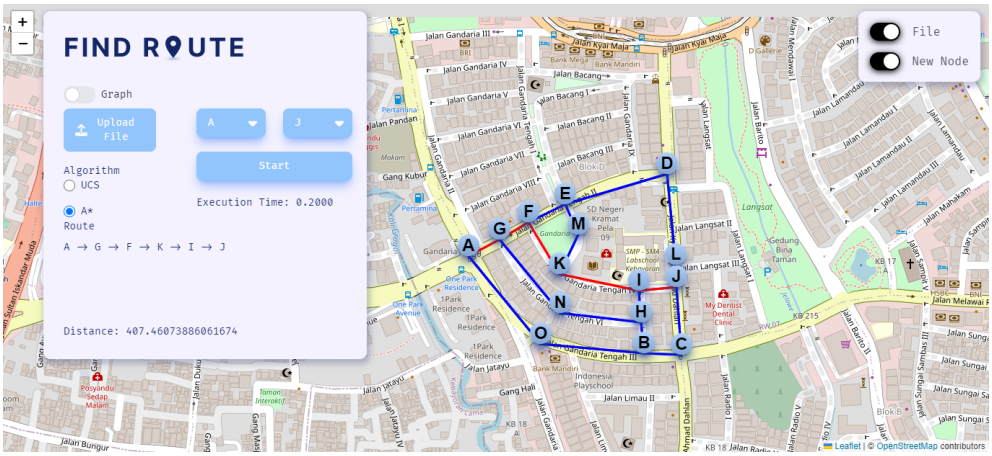


Hasil UCS itbdago.txt

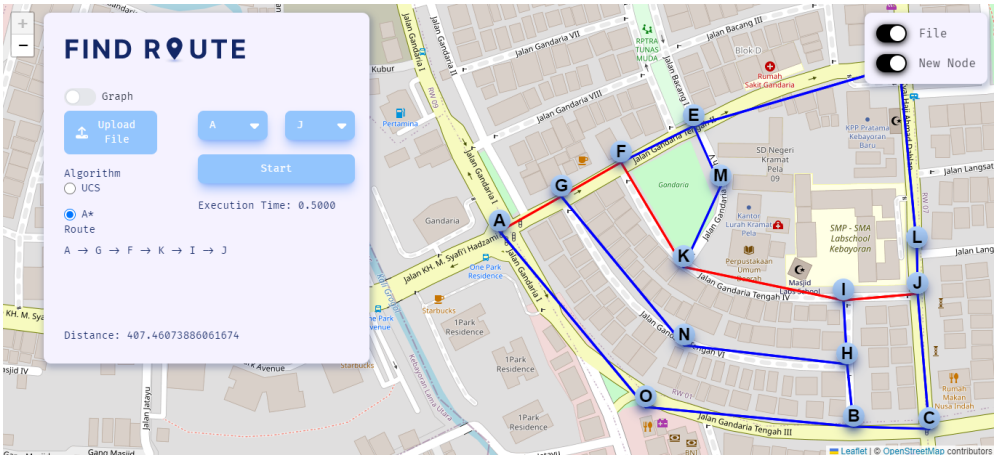
d. Jakarta.txt
Start Node: A
Goal Node: J

```
15
A (-6.245131,106.788753)
B (-6.246568,106.791361)
C (-6.246584,106.791921)
D (-6.243914,106.791703)
E (-6.244357,106.790184)
F (-6.244625,106.789645)
G (-6.244878,106.789209)
H (-6.246105,106.791314)
I (-6.245636,106.791288)
J (-6.245589,106.791838)
K (-6.245394,106.790106)
L (-6.245254,106.791829)
M (-6.244802,106.790382)
N (-6.245968,106.790115)
O (-6.246423,106.789829)
0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
0 0 1 0 0 0 0 1 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 1 0 0 0
0 0 0 1 0 1 0 0 0 0 0 0 1 0 0
0 0 0 0 1 0 1 0 0 0 1 0 0 0 0
1 0 0 0 0 1 0 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0 0 0 0 1 0
0 0 0 0 0 0 0 1 0 1 1 0 0 0 0
0 0 1 0 0 0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 1 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Jakarta.txt



Hasil A jakarta.txt*

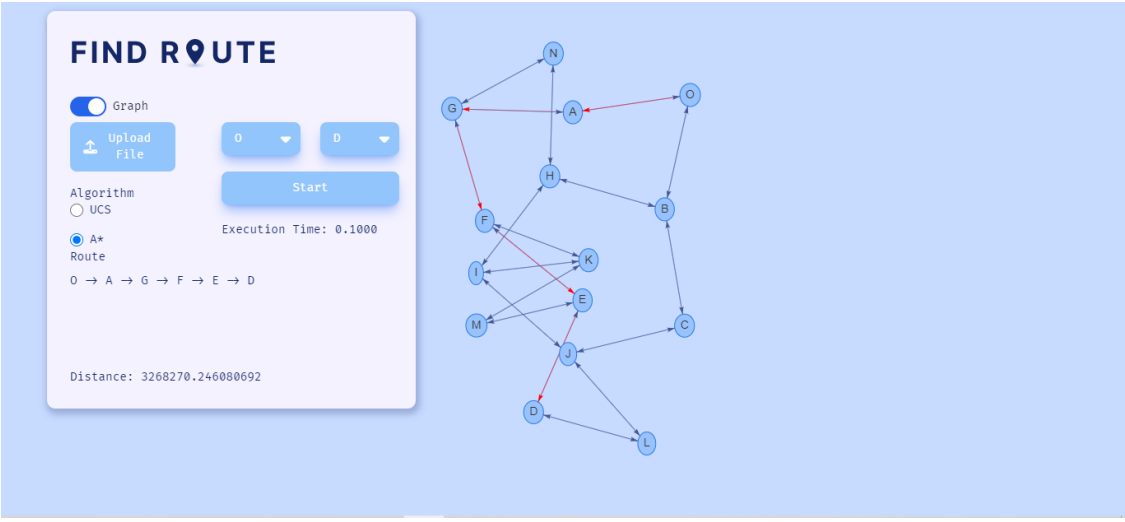


Hasil UCS jakarta.txt

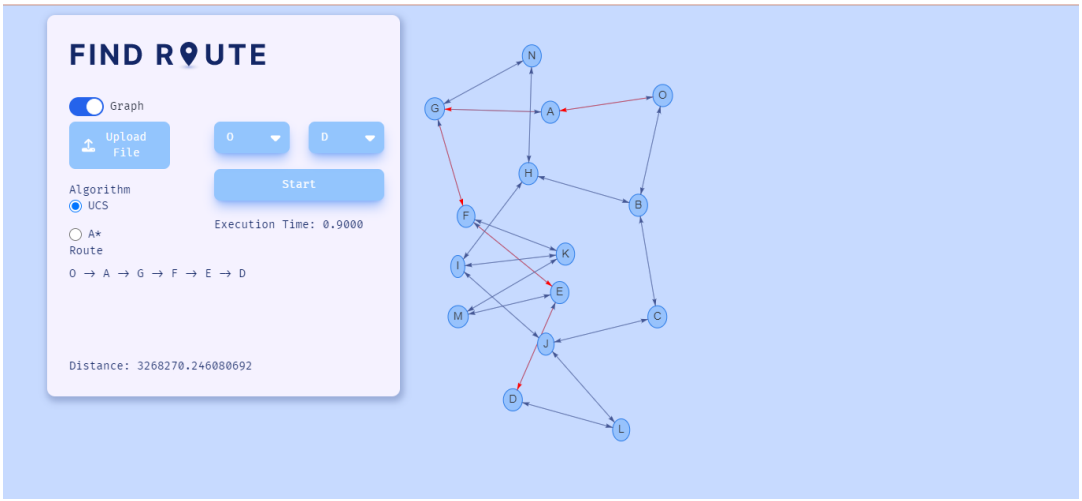
e. Graph.txt
Start Node: O
Goal Node: B

```
15
A (2,3)
B (4,5)
C (3,2)
D (1,3)
E (5,4)
F (5,6)
G (10,8)
H (8,9)
I (11,12)
J (13,4)
K (-1,3)
L (4,9)
M (9,10)
N (11,12)
O (-4,9)
0 0 0 0 0 0 1 0 0 0 0 0 0 1
0 0 1 0 0 0 0 1 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 1 0 0
0 0 0 1 0 1 0 0 0 0 0 0 1 0
0 0 0 0 1 0 1 0 0 0 1 0 0 0
1 0 0 0 0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0 0 0 1 0
0 0 0 0 0 0 0 1 0 1 1 0 0 0
0 0 1 0 0 0 0 0 1 0 0 1 0 0
0 0 0 0 0 1 0 0 1 0 0 0 1 0
0 0 0 1 0 0 0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 1 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0
```

Graph.txt

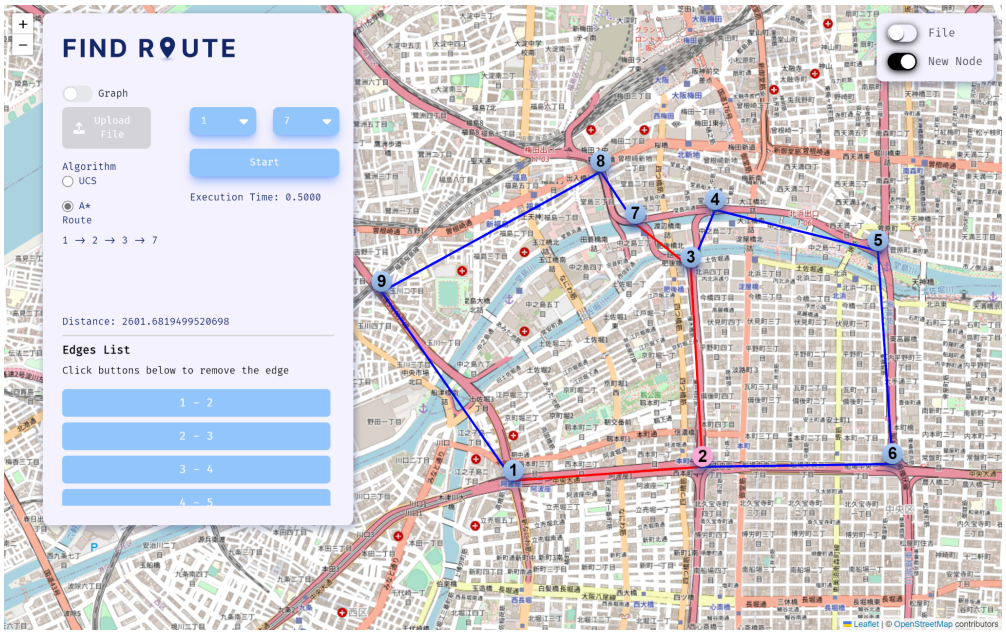


Hasil A graph.txt*

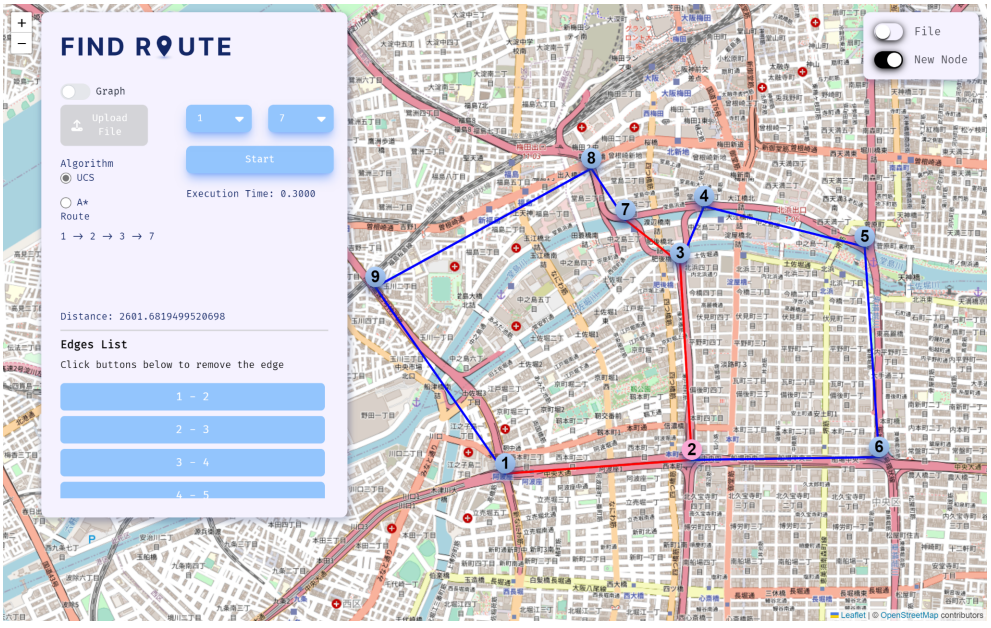


Hasil UCS graph.txt

f. Input menggunakan input peta



*Input langsung dari map dengan menggunakan algoritma A**



Input langsung dari map dengan menggunakan algoritma UCS

4. Alamat Github

https://github.com/sozyGithub/Tucil3_13521132_13521150.git

5. Kesimpulan

Dari kedua algoritma tersebut, dapat ditarik kesimpulan bahwa penggunaan algoritma UCS akan menghasilkan rute yang optimal, namun akan membangkitkan banyak node dan path yang harus ditempuh. Sementara itu, penggunaan algoritma A* akan menghasilkan hasil yang optimal dan jumlah path yang lebih sedikit. Algoritma A* akan memberikan solusi yang lebih optimal jika nilai heuristik yang dihasilkan tidak memberikan estimasi yang terlalu tinggi.

6. Checklist

1	Program dapat menerima input graf	✓
2	Program dapat menghitung lintasan terpendek dengan A* dan UCS	✓
3	Program dapat menghitung lintasan terpendek dengan A* dan UCS	✓
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	✓