Laporan Tugas Kecil 3 IF2122 Strategi Algoritma

Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

Kinantan Arya Bagaspati, 13519044
Jauhar Wibisono, 13519160

Kode Program

FrontEnd
main-page.html

```
<!DOCTYPE html>

<html class='use-all-space'>
  <head>
      {% block head %}
      <title>My Map</title>
      <meta http-equiv='X-UA-Compatible' content='IE=Edge' />
      <meta charset='UTF-8'>
      <meta name='viewport'
          content='width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no'/>
      {% endblock %}
  </head>
  <body>
      <h1>My Map</h1>
      <hr/>
      <h2>Input Node & Edge</h2>
      <h3>Input with file (.txt):</h3>
      <form method='post' enctype="multipart/form-data" border="1px solid black">
        <input type="file" name="upload" accept=".txt" required>
        <input type="submit">
      </form>
  </body>
</html>
```

Results-page.html

```
<!DOCTYPE html>

<html class='use-all-space'>
  <head>
      {% block head %}
      <title>My Map</title>
      <meta http-equiv='X-UA-Compatible' content='IE=Edge' />
      <meta charset='UTF-8'>
      <meta name='viewport'
            content='width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no'/>
      <!-- Replace version in the URL with desired library version -->
      <link rel='stylesheet' type='text/css' href='https://api.tomtom.com/maps-sdk-for-web/cdn/6.x/6.12.0/maps/maps.css'/>
      <!-- styles -->
      <style>
        #map {
          width: 90vw;
          height: 500px;
        }
      </style>
      {% endblock %}
  </head>
```

```
      <body>
          <h1>Results</h1>
          <hr/>
          <form method='post' enctype="multipart/form-data" border="1px solid black">
            <div id='container_start'></div>
            <div id='container_end'></div>
            <input type='submit'>
          </form>
          <br>
          <p>distance = {{ result.distance|int }} meter</p>
          <p>path =
            {% for i in range(result.path|length) %}
              {% if i > 0 %}
                ->
              {% endif %}
              {{ nodes[result.path[i]][0] }}
            {% endfor %}
          </p>
          <div id='map' class='map' ></div>
          <hr/>
          <a href="/">return</a>
          </div>
```

```
<!-- scripts -->
<!-- Replace version in the URL with desired library version -->
<script src='https://api.tomtom.com/maps-sdk-for-web/cdn/6.x/6.12.0/maps/maps-web.min.js'></script>
<script>

  // dropdown handler
  /* When the user clicks on the button,
  toggle between hiding and showing the dropdown content */
  tt.setProductInfo('interactiveMap', '1.0');

  var nodes = {{ nodes|tojson }};
  console.log(nodes);
  var meanx = 0, meany = 0;
  var xmin = 1000, xmax = -1000, ymin = 1000, ymax = -1000;
  for(var i=0; i<nodes.length; i++){
    meanx += nodes[i][2];
    meany += nodes[i][1];
    if(nodes[i][2] > xmax){
      xmax = nodes[i][2];
    }
    if(nodes[i][2] < xmin){
      xmin = nodes[i][2];
    }
    if(nodes[i][1] > ymax){
      ymax = nodes[i][1];
    }
    if(nodes[i][1] < ymin){
      ymin = nodes[i][1]
    }
  }
  meanx /= nodes.length;
  meany /= nodes.length;

  var center_coordinate = [meanx,meany];
  var zoom_value = (2/(xmax-xmin) + 1/(ymax-ymin))
```

```
    var map = tt.map({
        key: 'o1geElO01tAzH56UIRG6010g6cmtDWjz',
        container: 'map',
        center: center_coordinate,
        zoom: 4
    });

    // set marker node
    for (i in nodes){
      var long = nodes[i][1];
      var lat = nodes[i][2];
      var name = nodes[i][0];
      var marker = new tt.Marker().setLngLat([lat, long]).addTo(map);
      var popupOffsets = {
        top: [0, 0],
        bottom: [0, -70],
        'bottom-right': [0, -70],
        'bottom-left': [0, -70],
        left: [25, -35],
        right: [-25, -35]
      };

      var popup = new tt.Popup({offset: popupOffsets}).setHTML(name);
      marker.setPopup(popup).togglePopup();
    }
```

```
// set garis edge
var edges = {{ edge_list|tojson }};
console.log(edges);
map.on('load', function() {
  for (var i in edges){
      map.addLayer({
        'id': 'edge'+i,
        'type': 'line',
        'source': {
            'type': 'geojson',
            'data': {
                'type': 'FeatureCollection',
                'features': [
                    {
                        'type': 'Feature',
                        'geometry': {
                          'type': 'LineString',
                          'properties': {},
                          'coordinates': [
                            [edges[i][1], edges[i][0]],
                            [edges[i][3], edges[i][2]]
                          ]
                        },
                    }
                ]
            }
        },
        'layout': {
          'line-cap': 'round',
          'line-join': 'round'
        },
        'paint': {
          'line-color': '#a1a1a1',
          'line-width': 9
        }
      });
  }
});
```

```
// set garis path
var result = {{ result|tojson }};
console.log(result)
map.on('load', function() {
for (var i=1; i<result.path.length; i++){
    map.addLayer({
      'id': 'path'+i,
      'type': 'line',
      'source': {
          'type': 'geojson',
          'data': {
              'type': 'FeatureCollection',
              'features': [
                  {
                      'type': 'Feature',
                      'geometry': {
                        'type': 'LineString',
                        'properties': {},
                        'coordinates': [
                          [nodes[result.path[i-1]][2], nodes[result.path[i-1]][1]],
                          [nodes[result.path[i]][2], nodes[result.path[i]][1]]
                        ]
                      },
                  }
              ]
          }
      },
      'layout': {
        'line-cap': 'round',
        'line-join': 'round'
      },
      'paint': {
        'line-color': '#ff0000',
        'line-width': 9
      }
    });
  }
});
```

```
// dropdown menus
var values = [];
for (var i in nodes){
  values.push(nodes[i][0]);
}

var select = document.createElement("select");
select.name = 'startnode';
select.id = 'startnode';
select.required = 'required';

for (const val of values) {
  var option = document.createElement("option");
  option.value = val;
  option.text = val.charAt(0).toUpperCase() + val.slice(1);
  select.appendChild(option);
}

var label = document.createElement("label");
label.innerHTML = "Start node: ";
label.htmlFor = "startnode";

document.getElementById("container_start").appendChild(label).appendChild(select);
```

```
            var values = [];
            for (var i in nodes){
              values.push(nodes[i][0])
            }

            var select = document.createElement("select");
            select.name = 'endnode';
            select.id = 'endnode';
            select.required = 'required';

            for (const val of values) {
              var option = document.createElement("option");
              option.value = val;
              option.text = val.charAt(0).toUpperCase() + val.slice(1);
              select.appendChild(option);
            }

            var label = document.createElement("label");
            label.innerHTML = "End node: ";
            label.htmlFor = "endnode";

            document.getElementById("container_end").appendChild(label).appendChild(select);
        </script>
    </body>
</html>
```

BackEnd
interactiveMap.py

```python
from flask import Flask, render_template, request, redirect, url_for
from werkzeug.utils import secure_filename
import os
from Graph import *

app = Flask(__name__)
UPLOAD_FOLDER = os.path.abspath('../test') # directory dokumen-dokumen
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER


def readInput(filename):
    # format input:
    # Baris pertama berisi bilangan bulat N yang menyatakan banyak simpul
    # N baris berikutnya berisi sebuah string dan dua bilangan bulat yang menyatakan nama dan koordinat simpul
    # N baris berikutnya berisi adjacency matrix (sementara boolean)
    inp = open(os.path.join(app.config['UPLOAD_FOLDER'], filename))
    N = int(inp.readline())
    nodes = []
    adjacencyMatrix = [[0 for i in range (N)] for j in range (N)]
    for i in range(N):
        node_name, lng, lat = inp.readline().split(',')
        nodes.append((node_name, float(lng), float(lat)))

    for i in range(N):
        row = inp.readline().split(' ')
        for j in range(N):
            if row[j] == '1':
                adjacencyMatrix[i][j] = 1
    graph = Graph(nodes, adjacencyMatrix)
    return graph
```

```python
# main page
@app.route('/', methods=['GET', 'POST'])
def MainPage():
    if request.method == 'POST':
        if 'upload' in request.files: # website dapet input file
            # dapatkan dan simpan file input
            input_file = request.files['upload']
            input_filename = secure_filename(input_file.filename)
            input_file.save(os.path.join(app.config['UPLOAD_FOLDER'], input_filename))
            # baca input
            graph = readInput(input_filename)
            # jalankan algoritme A*

            # tampilkan hasil
            return redirect(url_for('ResultsPage', filename=input_filename[:-4]))
    return render_template('main-page.html')

# results page
@app.route('/results-<filename>', methods=['GET', 'POST'])
def ResultsPage(filename):
    graph = readInput(filename+'.txt')
    if request.method == 'POST':
        if 'startnode' in request.form and 'endnode' in request.form:
            start_node = request.form['startnode']
            end_node = request.form['endnode']
            return render_template('results-page.html', result=graph.Astar(start_node, end_node, True), nodes=graph.nodes(), edge_list=graph.edgeList())
    return render_template('results-page.html', result={"path":[], "distance":0}, nodes=graph.nodes(), edge_list=graph.edgeList())

if __name__ == '__main__':
    app.run()
```

Graph.py

```python
import math

def euclidianDistance(cor1,cor2):
    radius = 6371000
    kartesian1 = [0,0,0]
    kartesian2 = [0,0,0]
    kartesian1[2] = radius*math.sin(cor1[0]*(math.pi/180))
    kartesian2[2] = radius*math.sin(cor2[0]*(math.pi/180))
    proj1 = radius*math.cos(cor1[0]*(math.pi/180))
    proj2 = radius*math.cos(cor2[0]*(math.pi/180))

    kartesian1[0] = proj1*math.sin(cor1[1]*(math.pi/180))
    kartesian2[0] = proj2*math.sin(cor2[1]*(math.pi/180))
    kartesian1[1] = proj1*math.cos(cor1[1]*(math.pi/180))
    kartesian2[1] = proj2*math.cos(cor2[1]*(math.pi/180))

    sumsquare = 0
    for i in range (3):
        sumsquare += (kartesian2[i]-kartesian1[i])**2
    return (sumsquare**0.5)
```

```python
class Graph:
    #Konstruktor
    def __init__(self, nodes=[], adjacencyMatrix=[]):
        #Beberapa atribut yang dimiliki
        self.__nbNodes = len(nodes)
        self.__nodeNames = []
        self.__idNodes = {}
        self.__coordinates = []
        for i in range (self.__nbNodes):
            self.__nodeNames.append(nodes[i][0])
            self.__idNodes[nodes[i][0]] = i
            self.__coordinates.append((nodes[i][1], nodes[i][2]))
        self.__adjacencyMatrix = [[-1 for i in range (self.__nbNodes)] for j in range (self.__nbNodes)]
        self.__adjacencyList = [[] for i in range (self.__nbNodes)]
        for i in range (self.__nbNodes):
            for j in range (self.__nbNodes):
                if adjacencyMatrix[i][j]>0:
                    #Agar jarak dapat langsung diakses, alih alih menyimpan boolean,
                    #AdjacencyMatrix bernilai -1 jika tidak terhubung, dan jarak 2 node tersebut jika terhubung
                    distance = euclidianDistance(self.__coordinates[i], self.__coordinates[j])
                    self.__adjacencyMatrix[i][j] = distance
                    self.__adjacencyList[i].append((j, distance))
            #print(self.__adjacencyMatrix[i])
```

```python
    #Getters
    def nbNodes(self):
        return self.__nbNodes
    def nodes(self):
        #Mengembalikan array of triple terurut ID berisi nama node dan koordinat node
        toReturn = []
        for i in range (self.__nbNodes):
            toReturn.append([self.__nodeNames[i], self.__coordinates[i][0], self.__coordinates[i][1]])
        return toReturn
    def idNodes(self):
        return self.__idNodes
    def adjacencyMatrix(self):
        toReturn = []
        return self.__adjacencyMatrix
    def adjacencyList(self):
        return self.__adjacencyList
    def edgeList(self):
        edgeCoordinates = []
        for i in range (self.__nbNodes):
            for j in range (i):
                #print(self.__adjacencyMatrix[i][j], end = " ")
                if (self.__adjacencyMatrix[i][j] >= 0):
                    edgeCoordinates.append([self.__coordinates[i][0], self.__coordinates[i][1], self.__coordinates[j][0], self.__coordinates[j][1]])
            #print()
        return edgeCoordinates
```

```
    #Menambah node
    def addNode(self, newNode):
        self.__nodeNames.append(newNode[0])
        self.__idNodes[newNode[0]] = self.__nbNodes
        self.__coordinates.append((newNode[1], newNode[2]))
        self.__adjacencyList.append([])
        for i in range (self.__nbNodes):
            self.__adjacencyMatrix[i].append(-1)
        self.__adjacencyMatrix.append([])
        self.__nbNodes += 1
        for i in range (self.__nbNodes):
            self.__adjacencyMatrix[self.__nbNodes-1].append(-1)

    #Menambah edge yang menghubungkan node berID id1 dan id2
    def addEdgebyID(self, id1, id2):
        distance = euclidianDistance(self.__coordinates[id1], self.__coordinates[id2])
        if(self.__adjacencyMatrix[id1][id2]<0):
            self.__adjacencyMatrix[id1][id2] = distance
            self.__adjacencyMatrix[id2][id1] = distance
            self.__adjacencyList[id1].append((id2, distance))
            self.__adjacencyList[id2].append((id1, distance))

    #Menambah edge yang menghubungkan node bernama node1 dan node2
    def addEdge(self, node1, node2):
        if (node1 in self.__idNodes.keys()) and (node2 in self.__idNodes.keys()):
            id1 = self.__idNodes[node1]
            id2 = self.__idNodes[node2]
            self.addEdgebyID(id1, id2)
```

```
    #Mengembalikan jalur dan jarak hasil Astar berawal dan berakhir pada node bernama nodeto dan nodefrom
    def AstarbyID(self, idfrom, idto, isUsingAM):
        visited = [False for i in range (self.__nbNodes)]
        visitedFrom = [-1 for i in range (self.__nbNodes)]#belum ada yang dikunjungi
        totalDistance = [-1 for i in range (self.__nbNodes)]
        distanceTo = [euclidianDistance(self.__coordinates[i], self.__coordinates[idto]) for i in range (self.__nbNodes)]
        PQPosition = [-1 for i in range (self.__nbNodes)]

        totalDistance[idfrom] = 0
        PQPosition[idfrom] = 0
        PQ = [idfrom] #Nilai awal dari PriorityQueue
        visitedEdges = []
        head = 0
```

```
        while(head < len(PQ) and PQ[head] != idto):
            visited[PQ[head]] = True
            if(isUsingAM):
                for i in range (self.__nbNodes):
                    if(self.__adjacencyMatrix[PQ[head]][i] >= 0) and not(visited[i]):
                        if(totalDistance[i] < 0):
                            visitedFrom[i] = PQ[head]
                            totalDistance[i] = totalDistance[PQ[head]] + self.__adjacencyMatrix[PQ[head]][i]
                            PQPosition[i] = len(PQ)
                            PQ.append(i)
                            toFront = True
                            while(PQPosition[i]-1>head) and (toFront):
                                inFront = PQ[PQPosition[i]-1]
                                if (totalDistance[i] + distanceTo[i] < totalDistance[inFront] + distanceTo[inFront]):
                                    PQPosition[i] -= 1
                                    PQPosition[inFront] += 1
                                    PQ[PQPosition[i]] = i
                                    PQ[PQPosition[inFront]] = inFront
                                else:
                                    toFront = False
                        elif (totalDistance[PQ[head]] + self.__adjacencyMatrix[PQ[head]][i] < totalDistance[i]):
                            visitedFrom[i] = PQ[head]
                            totalDistance[i] = totalDistance[PQ[head]] + self.__adjacencyMatrix[PQ[head]][i]
                            toFront = True
                            while(PQPosition[i]-1>head) and (toFront):
                                inFront = PQ[PQPosition[i]-1]
                                if (totalDistance[i] + distanceTo[i] < totalDistance[inFront] + distanceTo[inFront]):
                                    PQPosition[i] -= 1
                                    PQPosition[inFront] += 1
                                    PQ[PQPosition[i]] = i
                                    PQ[PQPosition[inFront]] = inFront
                                else:
                                    toFront = False
```

```
            else:
                for neighbor in self.__adjacencyList[PQ[head]]:
                    i = neighbor[0]
                    distance = neighbor[1]
                    if(not(visited[i])):
                        if(totalDistance[i] < 0):
                            visitedFrom[i] = PQ[head]
                            totalDistance[i] = totalDistance[PQ[head]] + distance
                            PQPosition[i] = len(PQ)
                            PQ.append(i)
                            toFront = True
                            while(PQPosition[i]-1>head) and (toFront):
                                inFront = PQ[PQPosition[i]-1]
                                if (totalDistance[i] + distanceTo[i] < totalDistance[inFront] + distanceTo[inFront]):
                                    PQPosition[i] -= 1
                                    PQPosition[inFront] += 1
                                    PQ[PQPosition[i]] = i
                                    PQ[PQPosition[inFront]] = inFront
                                else:
                                    toFront = False
                        elif (totalDistance[PQ[head]] + distance < totalDistance[i]):
                            visitedFrom[i] = PQ[head]
                            totalDistance[i] = totalDistance[PQ[head]] + distance
                            toFront = True
                            while(PQPosition[i]-1>head) and (toFront):
                                inFront = PQ[PQPosition[i]-1]
                                if (totalDistance[i] + distanceTo[i] < totalDistance[inFront] + distanceTo[inFront]):
                                    PQPosition[i] -= 1
                                    PQPosition[inFront] += 1
                                    PQ[PQPosition[i]] = i
                                    PQ[PQPosition[inFront]] = inFront
                                else:
                                    toFront = False
            head+=1
```

```
        path = []
        if(head < len(PQ)):
            node = idto
            path.append(node)
            while(node != idfrom):
                node = visitedFrom[node]
                path.append(node)
            path.reverse()
        return {"path": path, "distance": totalDistance[idto]}

    #Mengembalikan jalur dan jarak hasil Astar berawal dan berakhir pada node bernama nodeto dan nodefrom
    def Astar(self, nodeto, nodefrom, isUsingAM):
        idto = self.__idNodes[nodeto]
        idfrom = self.__idNodes[nodefrom]
        return self.AstarbyID(idto, idfrom, isUsingAM)

```

Peta/Graf Input

Baris pertama input berisi bilangan bulat N yang menyatakan banyak simpul. N baris berikutnya berisi sebuah string S dan dua bilangan riil Lat dan Lng yang dipisahkan oleh tanda koma (,). S menyatakan nama simpul, sedangkan Lat dan Lng menyatakan koordinat simpul. Baris berikutnya berisi *adjacency matrix* A berukuran N × N. Baris ke-i kolom ke-j dari A bernilai 1 apabila simpul ke-i dan simpul ke-j dihubungkan oleh sebuah busur.

1. Peta jalan sekitar kampus ITB/Dago

```
16
Surapati - Dago,-6.89894,107.6127
Dipatiukur 1,-6.89466,107.61746
Dipatiukur 2,-6.89243,107.61781
Dago 1,-6.89483,107.61294
Dago 2,-6.89374,107.613
Badaksinga,-6.8973,107.61135
Ganesha,-6.89357,107.61195
Ganesha - Tamansari,-6.89387,107.60843
Dago 4,-6.88527,107.61367
Tamansari 2,-6.88693,107.61144
Siliwangi,-6.88493,107.61157
Sabuga,-6.88768,107.60993
Tamansari 1,-6.88787,107.60822
Dago 3,-6.88739,107.61352
Cikapundung,-6.88336,107.60489
Cihampelas,-6.90002,107.60429
0 1 0 1 0 1 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 0
0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0
1 0 0 0 0 0 1 0 0 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 0 0 1 0 0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0
```

```
0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
```

2. Peta jalan sekitar Alun-Alun Bandung

```
12
Masjid Agung,-6.92128,107.60772
P2,-6.92081,107.60408
P3,-6.92256,107.60758
P4,-6.92236,107.60643
P5,-6.92342,107.6063
P6,-6.9231,107.60394
P7,-6.92152,107.60995
P8,-6.92279,107.60977
P9,-6.92718,107.60361
P10,-6.92733,107.60595
P11,-6.92312,107.6119
P12,-6.93138,107.61246
0 1 1 0 0 0 1 0 0 0 0 0
1 0 0 0 0 1 0 0 0 0 0 0
1 0 0 1 0 0 0 1 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 0 1 0 0
0 1 0 0 1 0 0 0 1 0 0 0
1 0 0 0 0 0 0 1 0 0 0 0
0 0 1 0 0 0 1 0 0 0 1 0
0 0 0 0 0 1 0 0 0 1 0 0
0 0 0 0 1 0 0 0 1 0 0 1
0 0 0 0 0 0 0 1 0 0 0 1
0 0 0 0 0 0 0 0 0 1 1 0
```

3. Peta jalan sekitar Buahbatu

```
15
P1,-6.95409,107.64024
P2,-6.94544,107.64183
P3,-6.94203,107.65268
P4,-6.95217,107.65144
P5,-6.95524,107.6499
P6,-6.95566,107.65462
P7,-6.95898,107.65977
P8,-6.94995,107.66192
P9,-6.93913,107.66381
P10,-6.94795,107.63338
P11,-6.95446,107.63909
P12,-6.96226,107.67256
P13,-6.96763,107.67252
P14,-6.96584,107.63784
P15,-6.93859,107.66971
0 1 0 0 1 0 0 0 0 0 1 0 0 0 0
```

```
1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0
0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 1 0 0 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 0 0 1 0 1 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0 0 0 0 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 0 0 0 1 0 0 0 1 0
0 0 0 0 0 0 1 0 0 0 0 0 1 0 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 0 0 1 0 1 0 0
0 0 0 0 0 0 0 0 1 0 0 1 0 0 0
```

4. Peta jalan sekitar Stadion Kridosono Yogyakarta

```
20
Tugu,-7.78293,110.36704
Gramedia,-7.783,110.37496
Telkom,-7.78685,110.37425
Galeria,-7.78301,110.37926
UKDW,-7.78726,110.37859
Kridosono 1,-7.78828,110.37486
Kridosono 2,-7.78791,110.37316
Mataram,-7.79044,110.3676
Malioboro 1,-7.79009,110.36619
Mangkubumi,-7.78952,110.36619
Jembatan,-7.78969,110.36915
Flyover,-7.78823,110.37843
Pakualaman 1,-7.80169,110.37802
Pakualaman 2,-7.80144,110.36926
Nol Kilometer,-7.80131,110.36475
Lempuyangan,-7.79057,110.37399
Hayam Wuruk,-7.79684,110.37262
Progo,-7.79676,110.36914
Malioboro 2,-7.79638,110.36536
Gayam,-7.79722,110.37763
0 1 0 0 0 0 0 0 0 1 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 0 0 0 0 0
0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0
0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0
1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 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 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 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 0 0 0 0 0 0 1 0 1 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0
```

```
0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0
0 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 1 0 0 0 0 0 1 0 0 1 0 1 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0
```

5. Peta jalan sekitar Alun-Alun Purwokerto

```
17
Masjid Baitussalam,-7.42494,109.22951
Rita Supermall,-7.42516,109.23061
BCA,-7.42629,109.235
Omnia,-7.41977,109.22988
Stasiun,-7.42687,109.23669
BAF,-7.4202,109.23147
Altaf Shop,-7.42326,109.2313
Hotel DBen,-7.4238,109.23505
Kambing ABUYYA,-7.42107,109.23517
RS Margono,-7.41622,109.23063
SMKN2,-7.42244,109.24077
BRI,-7.42414,109.22583
TMP,-7.43413,109.22226
Pasar Manis,-7.41907,109.22694
Mie Cuan,-7.41662,109.24161
Rajawali,-7.43248,109.24393
Andhang Pangrenan,-7.43978,109.24423
0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 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 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
```

6. Peta jalan Romania

```
20
Bucharest,44.457,26.093
Arad,46.181,21.312
Zerind,46.624,21.518
Oradea,47.089,21.907
Sibiu,45.794,24.128
```

```
Fagaras,45.842,24.973
Timisoara,45.756,21.231
Lugoj,45.691,21.903
Mehadia,44.904,22.365
Drobeta,44.639,22.659
Cralova,44.319,23.794
Rimnicu Vilcea,45.099,24.369
Pitesti,44.856,24.869
Giurgiu,43.905,25.969
Urziceni,44.718,26.645
Neamt,47.056,26.506
Iasi,47.158,27.598
Vaslui,46.641,27.728
Hirsova,44.690,27.945
Eforie,44.049,28.653
0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 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
0 1 0 1 0 0 0 0 0 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 0 0
0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
1 0 0 0 1 0 0 0 0 0 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 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 0 0 0 0 0 1 0 1 0 0 0 0 0 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 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
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 0 0 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Beberapa Pasang Simpul dan Lintasan Terpendeknya

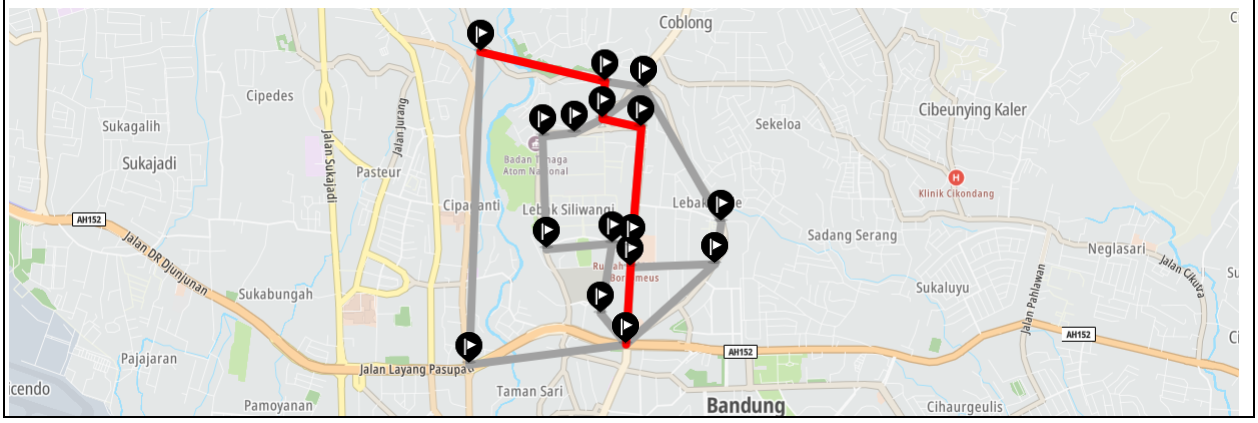
Lintasan terpendek ditunjukkan oleh garis merah pada peta. Apabila tidak terdapat jalur antara dua simpul pilihan, nilai *distance* -1 meter, *path* kosong, dan tidak terdapat garis merah pada peta.

1. Peta jalan sekitar kampus ITB/Dago

Start node: Surapati - Dago
End node: Cikapundung
Submit

distance = 2503 meter

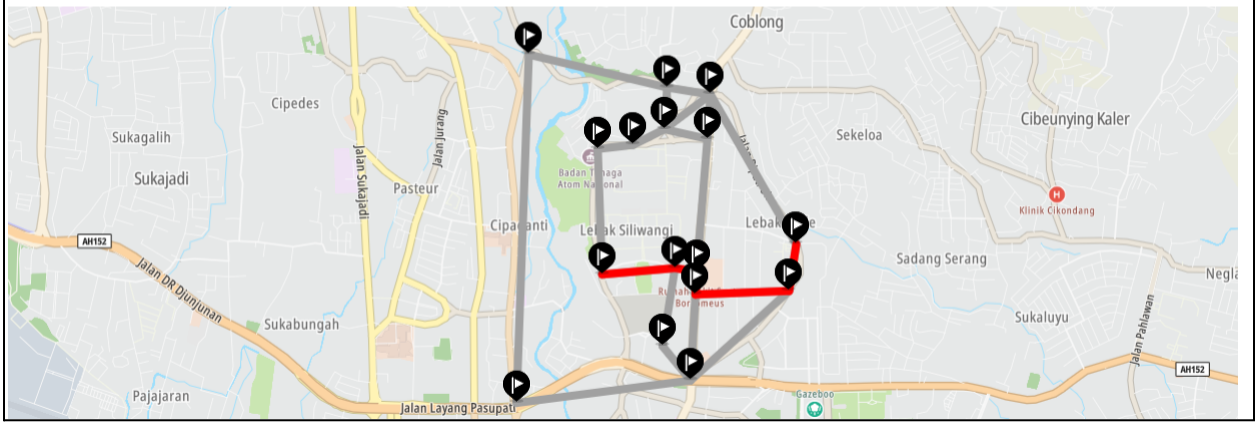
path = Surapati - Dago -> Dago 1 -> Dago 2 -> Dago 3 -> Tamansari 2 -> Siliwangi -> Cikapundung



Start node: Ganesha - Tamansari
End node: Dipatiukur 2
Submit

distance = 1379 meter

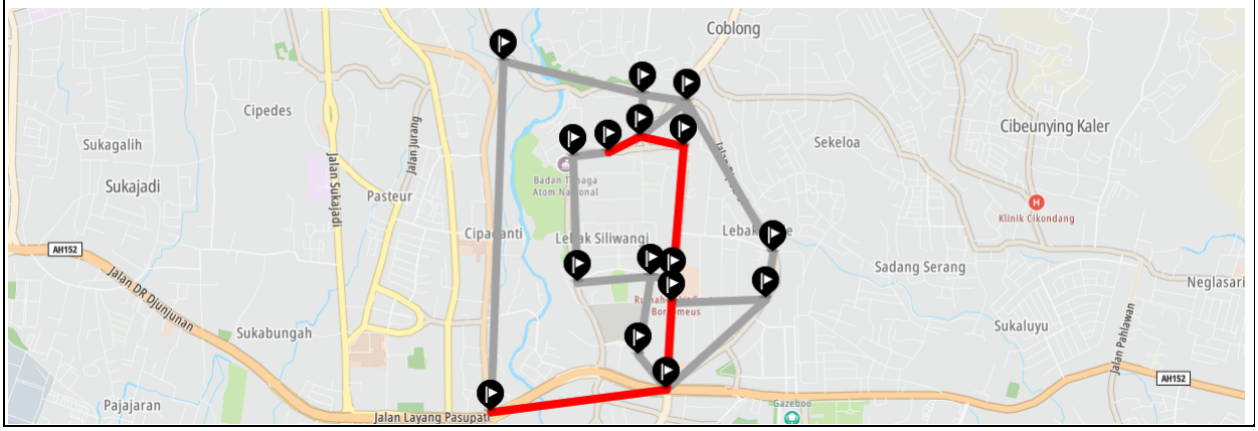
path = Ganesha - Tamansari -> Ganesha -> Dago 2 -> Dago 1 -> Dipatiukur 1 -> Dipatiukur 2



Start node: Cihampelas
End node: Sabuga
Submit

distance = 2645 meter

path = Cihampelas -> Surapati - Dago -> Dago 1 -> Dago 2 -> Dago 3 -> Tamansari 2 -> Sabuga

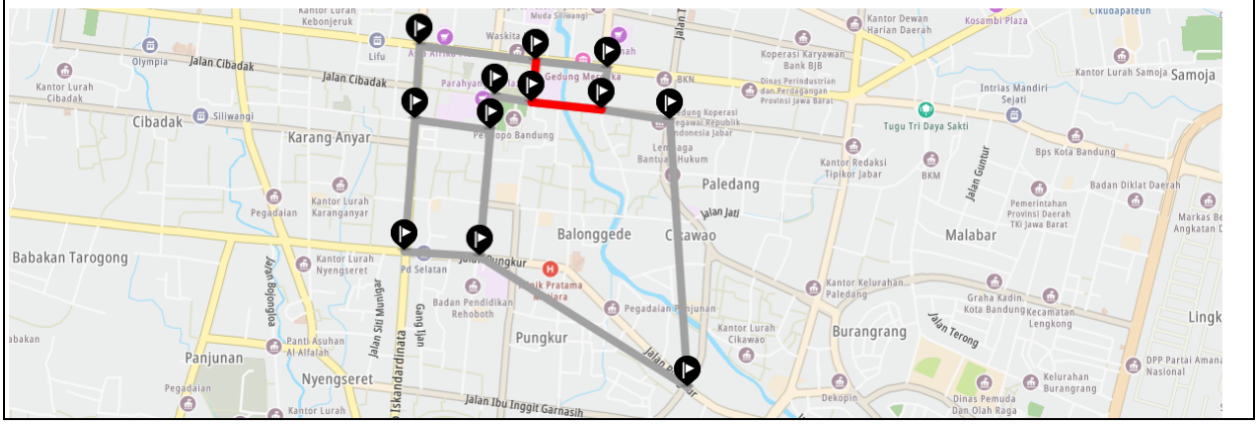


2. Peta jalan sekitar Alun-alun Bandung

Start node: Masjid Agung
End node: P8
Submit

distance = 386 meter

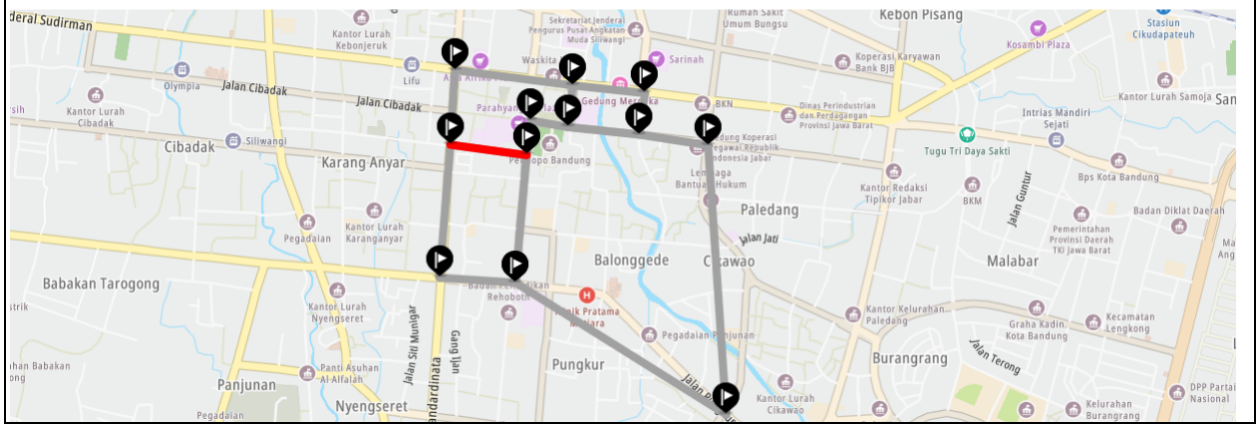
path = Masjid Agung -> P3 -> P8



Start node: P5
End node: P6
Submit

distance = 262 meter

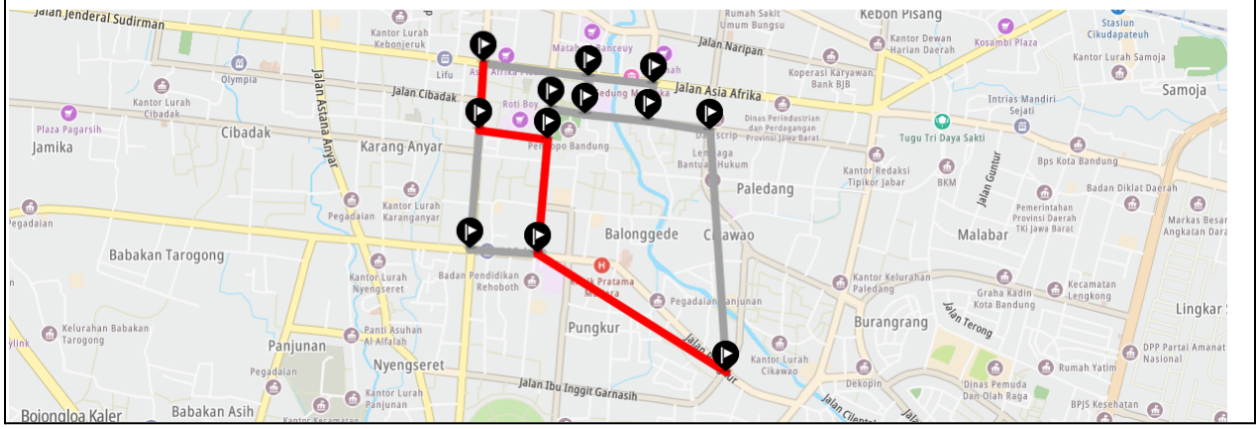
path = P5 -> P6



Start node: P12
End node: P2
Submit

distance = 1802 meter

path = P12 -> P10 -> P5 -> P6 -> P2

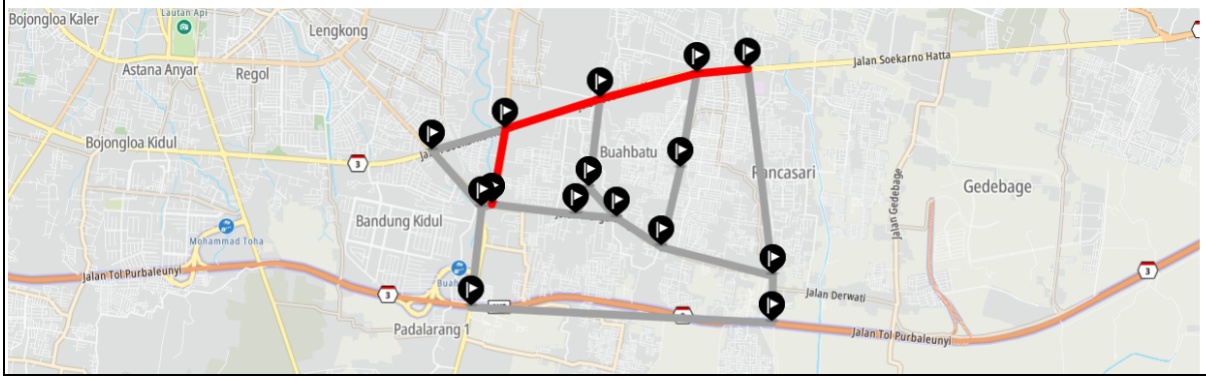


3. Peta jalan sekitar Buahbatu

Start node: P15
End node: P1
Submit

distance = 4158 meter

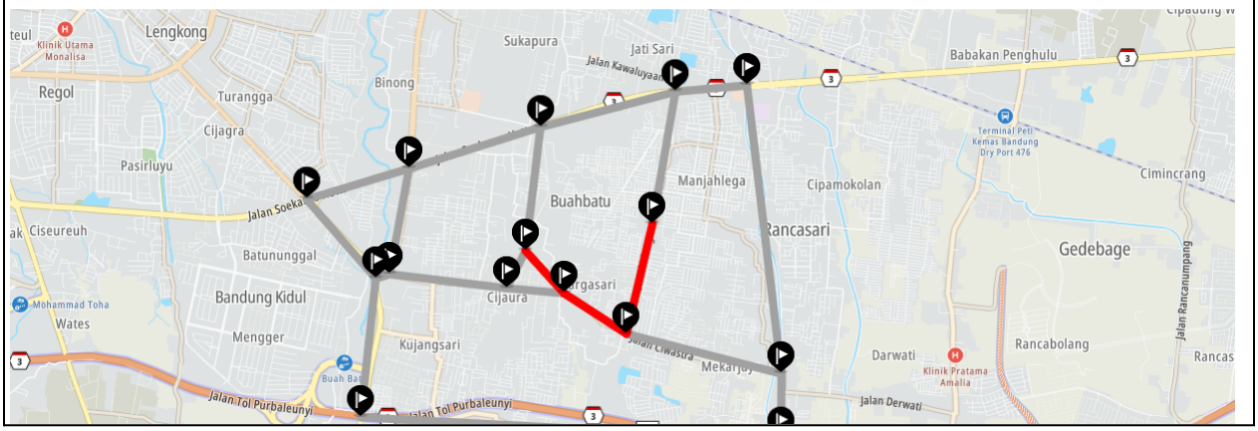
path = P15 -> P9 -> P3 -> P2 -> P1



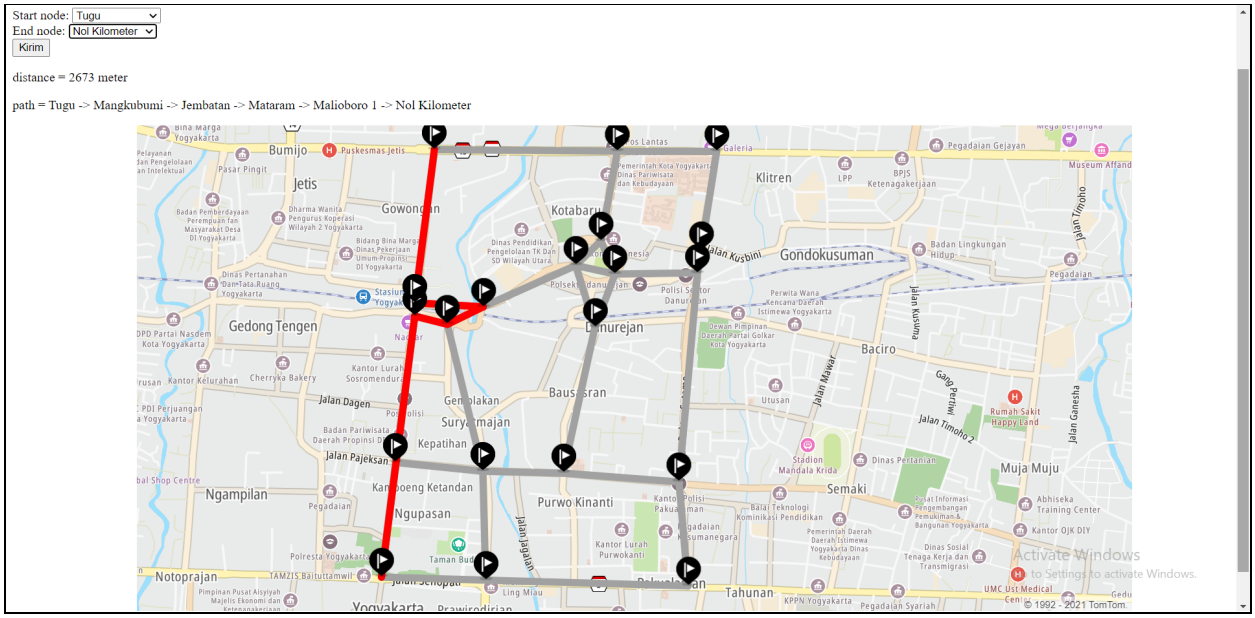
Start node: P8
End node: P4
Submit

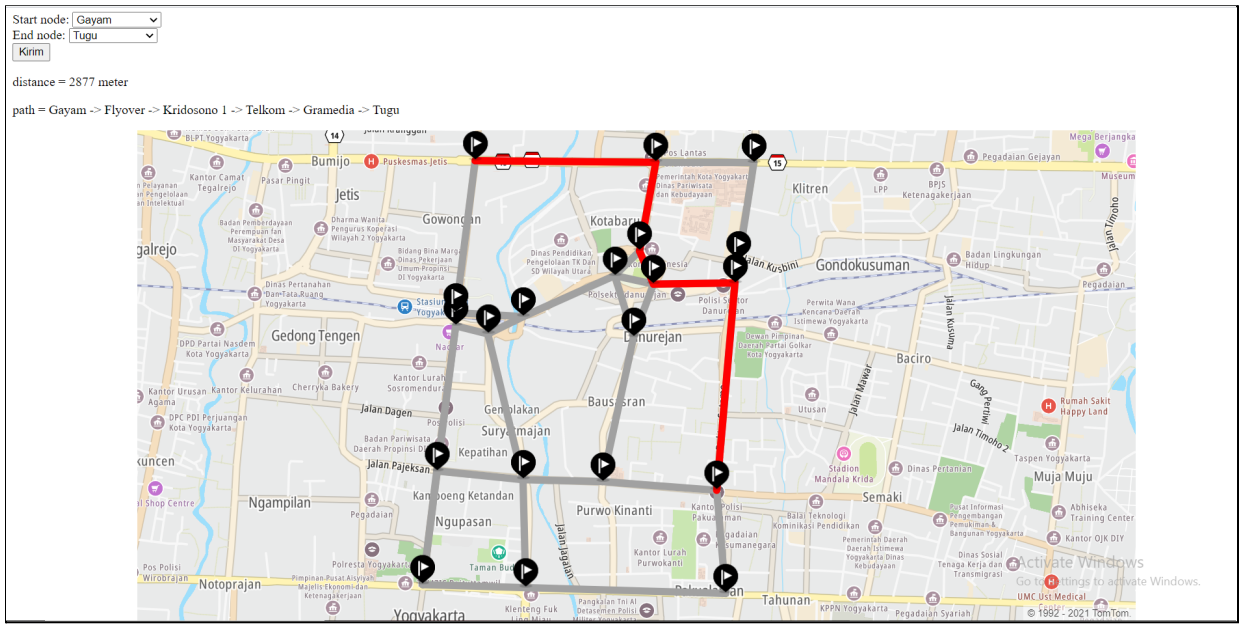
distance = 2232 meter

path = P8 -> P7 -> P6 -> P4

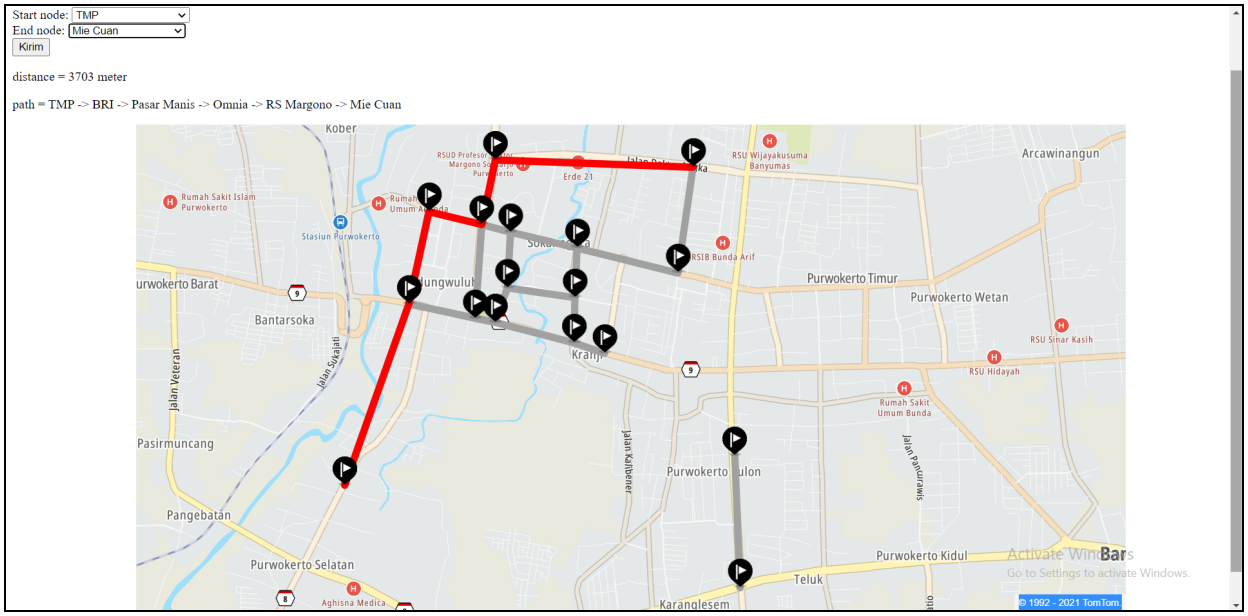


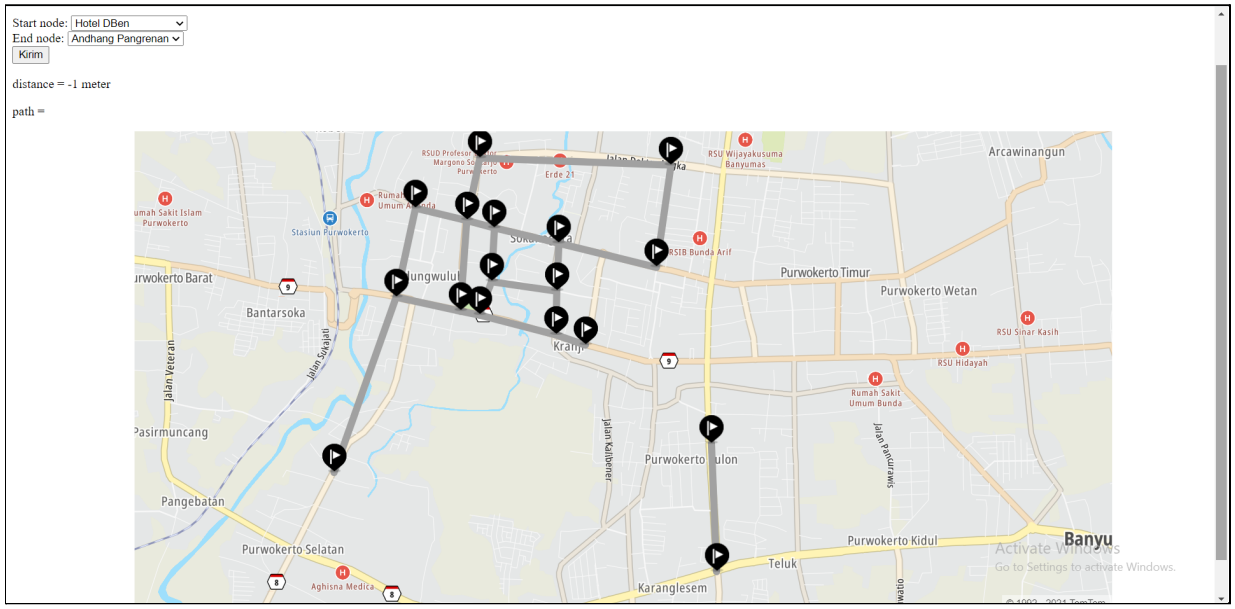
4. Peta jalan sekitar Stadion Kridosono Yogyakarta



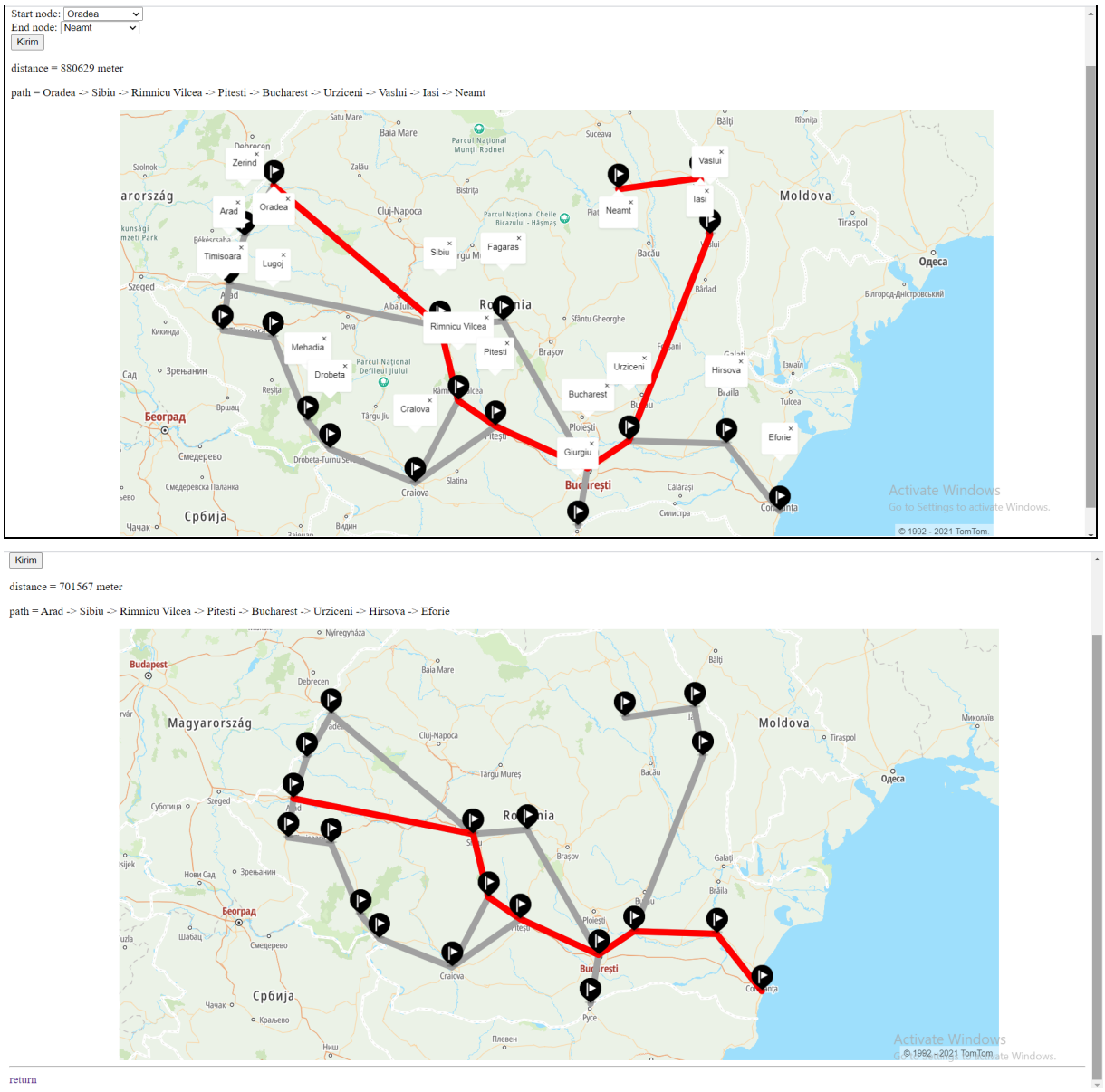


5. Peta jalan sekitar Alun-Alun Purwokerto





6. Peta jalan Romania



Alamat Kode Sumber Program

https://github.com/Jauhar-Wibisono/stima-tucil03.git

Tabel Penilaian

1	Program dapat menerima input graf	Ya
2	Program dapat menghitung lintasan terpendek	Ya

3	Program dapat menampilkan lintasan terpendek serta jaraknya	Ya
4	Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta	Tidak