

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

```
1  <!DOCTYPE html>
2
3  <html class='use-all-space'>
4  |  <head>
5  ||  {% block head %}
6  ||  <title>My Map</title>
7  ||  <meta http-equiv='X-UA-Compatible' content='IE=Edge' />
8  ||  <meta charset='UTF-8'>
9  ||  <meta name='viewport'
10 ||    content='width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no'/>
11 ||  {% endblock %}
12 |  </head>
13 |  <body>
14 |    <h1>My Map</h1>
15 |    <hr/>
16 |    <h2>Input Node & Edge</h2>
17 |    <h3>Input with file (.txt):</h3>
18 |    <form method='post' enctype="multipart/form-data" border="1px solid black">
19 |      <input type="file" name="upload" accept=".txt" required>
20 |      <input type="submit">
21 |    </form>
22 |  </body>
23 |</html>
```

Results-page.html

```
1  <!DOCTYPE html>
2
3  <html class='use-all-space'>
4      <head>
5          {% block head %}
6              <title>My Map</title>
7              <meta http-equiv='X-UA-Compatible' content='IE=Edge' />
8              <meta charset='UTF-8'>
9              <meta name='viewport'
10                 content='width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no' />
11             <!-- Replace version in the URL with desired library version --&gt;
12             &lt;link rel='stylesheet' type='text/css' href='https://api.tomtom.com/maps-sdk-for-web/cdn/6.x/6.12.0/maps/maps.css' /&gt;
13             &lt;!-- styles --&gt;
14         &lt;style&gt;
15             #map {
16                 width: 90vw;
17                 height: 500px;
18             }
19         &lt;/style&gt;
20     {% endblock %}
21 &lt;/head&gt;
22
23 &lt;body&gt;
24     &lt;h1&gt;Results&lt;/h1&gt;
25     &lt;hr/&gt;
26     &lt;form method='post' enctype="multipart/form-data" border="1px solid black"&gt;
27         &lt;div id='container_start'&gt;&lt;/div&gt;
28         &lt;div id='container_end'&gt;&lt;/div&gt;
29         &lt;input type='submit'&gt;
30     &lt;/form&gt;
31     &lt;br&gt;
32     &lt;p&gt;distance = {{ result.distance|int }} meter&lt;/p&gt;
33     &lt;p&gt;path =
34         {% for i in range(result.path|length) %}
35             {% if i &gt; 0 %}
36                 -
37             {% endif %}
38             {{ nodes[result.path[i]][0] }}
39         {% endfor %}
40     &lt;/p&gt;
41     &lt;div id='map' class='map' &gt;&lt;/div&gt;
42     &lt;hr/&gt;
43     &lt;a href="/"&gt;return&lt;/a&gt;
44 &lt;/div&gt;</pre>
```

```

44      <!-- scripts -->
45      <!-- Replace version in the URL with desired Library version -->
46      <script src='https://api.tomtom.com/maps-sdk-for-web/cdn/6.x/6.12.0/maps/maps-web.min.js'></script>
47      <script>
48
49          // dropdown handler
50          /* When the user clicks on the button,
51             toggle between hiding and showing the dropdown content */
52          tt.setProductInfo('interactiveMap', '1.0');
53
54          var nodes = {{ nodes|tojson }};
55          console.log(nodes);
56          var meanx = 0, meany = 0;
57          var xmin = 1000, xmax = -1000, ymin = 1000, ymax = -1000;
58          for(var i=0; i<nodes.length; i++){
59              meanx += nodes[i][2];
60              meany += nodes[i][1];
61              if(nodes[i][2] > xmax){
62                  xmax = nodes[i][2];
63              }
64              if(nodes[i][2] < xmin){
65                  xmin = nodes[i][2];
66              }
67              if(nodes[i][1] > ymax){
68                  ymax = nodes[i][1];
69              }
70              if(nodes[i][1] < ymin){
71                  ymin = nodes[i][1];
72              }
73          }
74          meanx /= nodes.length;
75          meany /= nodes.length;
76
77          var center_coordinate = [meanx,meany];
78          var zoom_value = (2/(xmax-xmin) + 1/(ymax-ymin))

```



```

80          var map = tt.map({
81              key: 'olgeE1001tAzH56UIRG6010g6cmtDWjz',
82              container: 'map',
83              center: center_coordinate,
84              zoom: 4
85          });
86
87          // set marker node
88          for (i in nodes){
89              var long = nodes[i][1];
90              var lat = nodes[i][2];
91              var name = nodes[i][0];
92              var marker = new tt.Marker().setLngLat([lat, long]).addTo(map);
93              var popupOffsets = {
94                  top: [0, 0],
95                  bottom: [0, -70],
96                  'bottom-right': [0, -70],
97                  'bottom-left': [0, -70],
98                  left: [25, -35],
99                  right: [-25, -35]
100             };
101
102             var popup = new tt.Popup({offset: popupOffsets}).setHTML(name);
103             marker.setPopup(popup).togglePopup();
104         }

```

```
106 // set garis edge
107 var edges = {{ edge_list|tojson }};
108 console.log(edges);
109 map.on('load', function() {
110     for (var i in edges){
111         map.addLayer({
112             'id': 'edge'+i,
113             'type': 'line',
114             'source': {
115                 'type': 'geojson',
116                 'data': {
117                     'type': 'FeatureCollection',
118                     'features': [
119                         {
120                             'type': 'Feature',
121                             'geometry': {
122                                 'type': 'LineString',
123                                 'properties': {},
124                                 'coordinates': [
125                                     [edges[i][1], edges[i][0]],
126                                     [edges[i][3], edges[i][2]]
127                                 ]
128                             },
129                         }
130                     ]
131                 }
132             },
133             'layout': {
134                 'line-cap': 'round',
135                 'line-join': 'round'
136             },
137             'paint': {
138                 'line-color': '#a1a1a1',
139                 'line-width': 9
140             }
141         });
142     }
143 });

```

```

145         // set garis path
146         var result = {{ result|tojson }};
147         console.log(result)
148         map.on('load', function() {
149             for (var i=1; i<result.path.length; i++){
150                 map.addLayer({
151                     'id': 'path'+i,
152                     'type': 'line',
153                     'source': {
154                         'type': 'geojson',
155                         'data': {
156                             'type': 'FeatureCollection',
157                             'features': [
158                                 {
159                                     'type': 'Feature',
160                                     'geometry': {
161                                         'type': 'LineString',
162                                         'properties': {},
163                                         'coordinates': [
164                                             [nodes[result.path[i-1]][2], nodes[result.path[i-1]][1]],
165                                             [nodes[result.path[i]][2], nodes[result.path[i]][1]]
166                                         ]
167                                     }
168                                 },
169                             ]
170                         }
171                     },
172                     'layout': {
173                         'line-cap': 'round',
174                         'line-join': 'round'
175                     },
176                     'paint': {
177                         'line-color': '#ff0000',
178                         'line-width': 9
179                     }
180                 });
181             }
182         });

```

```

184         // dropdown menus
185         var values = [];
186         for (var i in nodes){
187             values.push(nodes[i][0]);
188         }
189
190         var select = document.createElement("select");
191         select.name = 'startnode';
192         select.id = 'startnode';
193         select.required = 'required';
194
195         for (const val of values) {
196             var option = document.createElement("option");
197             option.value = val;
198             option.text = val.charAt(0).toUpperCase() + val.slice(1);
199             select.appendChild(option);
200         }
201
202         var label = document.createElement("label");
203         label.innerHTML = "Start node: ";
204         label.htmlFor = "startnode";
205
206         document.getElementById("container_start").appendChild(label).appendChild(select);

```

```

208     var values = [];
209     for (var i in nodes){
210       values.push(nodes[i][0])
211     }
212
213     var select = document.createElement("select");
214     select.name = 'endnode';
215     select.id = 'endnode';
216     select.required = 'required';
217
218     for (const val of values) {
219       var option = document.createElement("option");
220       option.value = val;
221       option.text = val.charAt(0).toUpperCase() + val.slice(1);
222       select.appendChild(option);
223     }
224
225     var label = document.createElement("label");
226     label.innerHTML = "End node: ";
227     label.htmlFor = "endnode";
228
229     document.getElementById("container_end").appendChild(label).appendChild(select);
230   </script>
231 </body>
232 </html>

```

BackEnd

interactiveMap.py

```

11  def readInput(filename):
12      # format input:
13      # Baris pertama berisi bilangan bulat N yang menyatakan banyak simpul
14      # N baris berikutnya berisi sebuah string dan dua bilangan bulat yang menyatakan nama dan koordinat simpul
15      # N baris berikutnya berisi adjacency matrix (sementara boolean)
16      inp = open(os.path.join(app.config['UPLOAD_FOLDER'], filename))
17      N = int(inp.readline())
18      nodes = []
19      adjacencyMatrix = [[0 for i in range (N)] for j in range (N)]
20      for i in range(N):
21          node_name, lng, lat = inp.readline().split(',')
22          nodes.append((node_name, float(lng), float(lat)))
23
24      for i in range(N):
25          row = inp.readline().split(' ')
26          for j in range(N):
27              if row[j][0] == '1':
28                  adjacencyMatrix[i][j] = 1
29
graph = Graph(nodes, adjacencyMatrix)
return graph

```

```

51 # main page
52 @app.route('/', methods=['GET', 'POST'])
53 def MainPage():
54     if request.method == 'POST':
55         if 'upload' in request.files: # website dapet input file
56             # dapatkan dan simpan file input
57             input_file = request.files['upload']
58             input_filename = secure_filename(input_file.filename)
59             input_file.save(os.path.join(app.config['UPLOAD_FOLDER'], input_filename))
60             # baca input
61             graph = readInput(input_filename)
62             # jalankan algoritme A*
63
64             # tampilkan hasil
65             return redirect(url_for('ResultsPage', filename=input_filename[:-4]))
66     return render_template('main-page.html')
67
68 # results page
69 @app.route('/results-<filename>', methods=['GET', 'POST'])
70 def ResultsPage(filename):
71     graph = readInput(filename+'.txt')
72     if request.method == 'POST':
73         if 'startnode' in request.form and 'endnode' in request.form:
74             start_node = request.form['startnode']
75             end_node = request.form['endnode']
76             return render_template('results-page.html', result=graph.Astar(start_node, end_node, True), nodes=graph.nodes(), edge_list=graph.edgeList())
77     return render_template('results-page.html', result={"path":[], "distance":0}, nodes=graph.nodes(), edge_list=graph.edgeList())
78
79 if __name__ == '__main__':
80     app.run()

```

Graph.py

```

1 import math
2
3 def euclidianDistance(corr1,corr2):
4     radius = 6371000
5     kartesian1 = [0,0,0]
6     kartesian2 = [0,0,0]
7     kartesian1[2] = radius*math.sin(corr1[0]*(math.pi/180))
8     kartesian2[2] = radius*math.sin(corr2[0]*(math.pi/180))
9     proj1 = radius*math.cos(corr1[0]*(math.pi/180))
10    proj2 = radius*math.cos(corr2[0]*(math.pi/180))
11
12    kartesian1[0] = proj1*math.sin(corr1[1]*(math.pi/180))
13    kartesian2[0] = proj2*math.sin(corr2[1]*(math.pi/180))
14    kartesian1[1] = proj1*math.cos(corr1[1]*(math.pi/180))
15    kartesian2[1] = proj2*math.cos(corr2[1]*(math.pi/180))
16
17    sumsquare = 0
18    for i in range (3):
19        sumsquare += (kartesian2[i]-kartesian1[i])**2
20    return (sumsquare**0.5)

```

```

22  class Graph:
23      #Konstruktor
24      def __init__(self, nodes=[], adjacencyMatrix=[]):
25          #Beberapa atribut yang dimiliki
26          self.__nbNodes = len(nodes)
27          self.__nodeNames = []
28          self.__idNodes = {}
29          self.__coordinates = []
30          for i in range (self.__nbNodes):
31              self.__nodeNames.append(nodes[i][0])
32              self.__idNodes[nodes[i][0]] = i
33              self.__coordinates.append((nodes[i][1], nodes[i][2]))
34          self.__adjacencyMatrix = [[-1 for i in range (self.__nbNodes)] for j in range (self.__nbNodes)]
35          self.__adjacencyList = [[] for i in range (self.__nbNodes)]
36          for i in range (self.__nbNodes):
37              for j in range (self.__nbNodes):
38                  if adjacencyMatrix[i][j]>0:
39                      #Agar jarak dapat langsung diakses, alih alih menyimpan boolean,
40                      #AdjacencyMatrix bernilai -1 jika tidak terhubung, dan jarak 2 node tersebut jika terhubung
41                      distance = euclidianDistance(self.__coordinates[i], self.__coordinates[j])
42                      self.__adjacencyMatrix[i][j] = distance
43                      self.__adjacencyList[i].append((j, distance))
44          #print(self.__adjacencyMatrix[i])

46      #Getters
47      def nbNodes(self):
48          return self.__nbNodes
49      def nodes(self):
50          #Mengembalikan array of triple terurut ID berisi nama node dan koordinat node
51          toReturn = []
52          for i in range (self.__nbNodes):
53              toReturn.append([self.__nodeNames[i], self.__coordinates[i][0], self.__coordinates[i][1]])
54          return toReturn
55      def idNodes(self):
56          return self.__idNodes
57      def adjacencyMatrix(self):
58          toReturn = []
59          return self.__adjacencyMatrix
60      def adjacencyList(self):
61          return self.__adjacencyList
62      def edgeList(self):
63          edgeCoordinates = []
64          for i in range (self.__nbNodes):
65              for j in range (i):
66                  #print(self.__adjacencyMatrix[i][j], end = " ")
67                  if (self.__adjacencyMatrix[i][j] >= 0):
68                      edgeCoordinates.append([self.__coordinates[i][0], self.__coordinates[i][1], self.__coordinates[j][0], self.__coordinates[j][1]])
69          #print()
70          return edgeCoordinates

```

```

72     #Menambah node
73     def addNode(self, newNode):
74         self.__nodeNames.append(newNode[0])
75         self.__idNodes[newNode[0]] = self.__nbNodes
76         self.__coordinates.append((newNode[1], newNode[2]))
77         self.__adjacencyList.append([])
78         for i in range (self.__nbNodes):
79             self.__adjacencyMatrix[i].append(-1)
80         self.__adjacencyMatrix.append([])
81         self.__nbNodes += 1
82         for i in range (self.__nbNodes):
83             self.__adjacencyMatrix[self.__nbNodes-1].append(-1)
84
85     #Menambah edge yang menghubungkan node berID id1 dan id2
86     def addEdgebyID(self, id1, id2):
87         distance = euclidianDistance(self.__coordinates[id1], self.__coordinates[id2])
88         if(self.__adjacencyMatrix[id1][id2]<0):
89             self.__adjacencyMatrix[id1][id2] = distance
90             self.__adjacencyMatrix[id2][id1] = distance
91             self.__adjacencyList[id1].append((id2, distance))
92             self.__adjacencyList[id2].append((id1, distance))
93
94     #Menambah edge yang menghubungkan node bernama node1 dan node2
95     def addEdge(self, node1, node2):
96         if (node1 in self.__idNodes.keys()) and (node2 in self.__idNodes.keys()):
97             id1 = self.__idNodes[node1]
98             id2 = self.__idNodes[node2]
99             self.addEdgebyID(id1, id2)
100
101     #Mengembalikan jalur dan jarak hasil Astar berawal dan berakhir pada node bernama nodeto dan nodefrom
102     def AstarbyID(self, idfrom, idto, isUsingAM):
103         visited = [False for i in range (self.__nbNodes)]
104         visitedFrom = [-1 for i in range (self.__nbNodes)]#belum ada yang dikunjungi
105         totalDistance = [-1 for i in range (self.__nbNodes)]
106         distanceTo = [euclidianDistance(self.__coordinates[i], self.__coordinates[idto]) for i in range (self.__nbNodes)]
107         PQPosition = [-1 for i in range (self.__nbNodes)]
108
109         totalDistance[idfrom] = 0
110         PPosition[idfrom] = 0
111         PQ = [idfrom] #Nilai awal dari Priority Queue
112         visitedEdges = []
113         head = 0

```

```

114     while(head < len(PQ) and PQ[head] != idto):
115         visited[PQ[head]] = True
116         if(isUsingAM):
117             for i in range (self.__nbNodes):
118                 if(self.__adjacencyMatrix[PQ[head]][i] >= 0) and not(visited[i]):
119                     if(totalDistance[i] < 0):
120                         visitedFrom[i] = PQ[head]
121                         totalDistance[i] = totalDistance[PQ[head]] + self.__adjacencyMatrix[PQ[head]][i]
122                         PQPosition[i] = len(PQ)
123                         PQ.append(i)
124                         toFront = True
125                         while(PQPosition[i]-1>head) and (toFront):
126                             inFront = PQ[PQPosition[i]-1]
127                             if (totalDistance[i] + distanceTo[i] < totalDistance[inFront] + distanceTo[inFront]):
128                                 PQPosition[i] -= 1
129                                 PQPosition[inFront] += 1
130                                 PQ[PQPosition[i]] = i
131                                 PQ[PQPosition[inFront]] = inFront
132                             else:
133                                 toFront = False
134             elif (totalDistance[PQ[head]] + self.__adjacencyMatrix[PQ[head]][i] < totalDistance[i]):
135                 visitedFrom[i] = PQ[head]
136                 totalDistance[i] = totalDistance[PQ[head]] + self.__adjacencyMatrix[PQ[head]][i]
137                 toFront = True
138                 while(PQPosition[i]-1>head) and (toFront):
139                     inFront = PQ[PQPosition[i]-1]
140                     if (totalDistance[i] + distanceTo[i] < totalDistance[inFront] + distanceTo[inFront]):
141                         PQPosition[i] -= 1
142                         PQPosition[inFront] += 1
143                         PQ[PQPosition[i]] = i
144                         PQ[PQPosition[inFront]] = inFront
145                     else:
146                         toFront = False
147
148             else:
149                 for neighbor in self.__adjacencyList[PQ[head]]:
150                     i = neighbor[0]
151                     distance = neighbor[1]
152                     if(not(visited[i])):
153                         if(totalDistance[i] < 0):
154                             visitedFrom[i] = PQ[head]
155                             totalDistance[i] = totalDistance[PQ[head]] + distance
156                             PQPosition[i] = len(PQ)
157                             PQ.append(i)
158                             toFront = True
159                             while(PQPosition[i]-1>head) and (toFront):
160                                 inFront = PQ[PQPosition[i]-1]
161                                 if (totalDistance[i] + distanceTo[i] < totalDistance[inFront] + distanceTo[inFront]):
162                                     PQPosition[i] -= 1
163                                     PQPosition[inFront] += 1
164                                     PQ[PQPosition[i]] = i
165                                     PQ[PQPosition[inFront]] = inFront
166                                 else:
167                                     toFront = False
168             elif (totalDistance[PQ[head]] + distance < totalDistance[i]):
169                 visitedFrom[i] = PQ[head]
170                 totalDistance[i] = totalDistance[PQ[head]] + distance
171                 toFront = True
172                 while(PQPosition[i]-1>head) and (toFront):
173                     inFront = PQ[PQPosition[i]-1]
174                     if (totalDistance[i] + distanceTo[i] < totalDistance[inFront] + distanceTo[inFront]):
175                         PQPosition[i] -= 1
176                         PQPosition[inFront] += 1
177                         PQ[PQPosition[i]] = i
178                         PQ[PQPosition[inFront]] = inFront
179                     else:
180                         toFront = False
180             head+=1

```

```

181     path = []
182     if(head < len(PQ)):
183         node = idto
184         path.append(node)
185         while(node != idfrom):
186             node = visitedFrom[node]
187             path.append(node)
188         path.reverse()
189     return {"path": path, "distance": totalDistance[idto]}
190
191 #Mengembalikan jalur dan jarak hasil Astar berawal dan berakhir pada node bernama nodeto dan nodefrom
192 def Astar(self, nodeto, nodefrom, isUsingAM):
193     idto = self._idNodes[nodeto]
194     idfrom = self._idNodes[nodefrom]
195     return self.AstarbyID(idto, idfrom, isUsingAM)
196

```

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 $N \times N$. 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 0 1
1 0 1 1 0 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 0
1 1 0 0 1 0 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 1 0 0
1 0 0 0 0 0 1 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 0 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 1 0 0 1 0 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	0	1	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

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, 107.61119
P12, -6.93138, 107.61246
0 1 1 0 0 0 1 0 0 0 0 0 0
1 0 0 0 0 1 0 0 0 0 0 0 0
1 0 0 1 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 1 0 1 0 0 0 1 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 1 0 0 0 1 0 0
0 0 0 0 0 1 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 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 1 0 0 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	1	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	0	0	0
1	0	0	1	0	1	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	1	0	1	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	0	0	1
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	1	0	0	0	1
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	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	0	0	1	0
0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

4. Peta jalan sekitar Stadion Kridosono Yogyakarta

0	0	0	0	0	1	1	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	1	0	1	0
0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	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 1 0 0 0 0 0
1 0 1 0 0 0 1 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
1 0 0 0 0 1 0 0 0 1 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 1 0 0 1 0 1 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 1 0 0 0 1 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 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 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 1 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 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

```

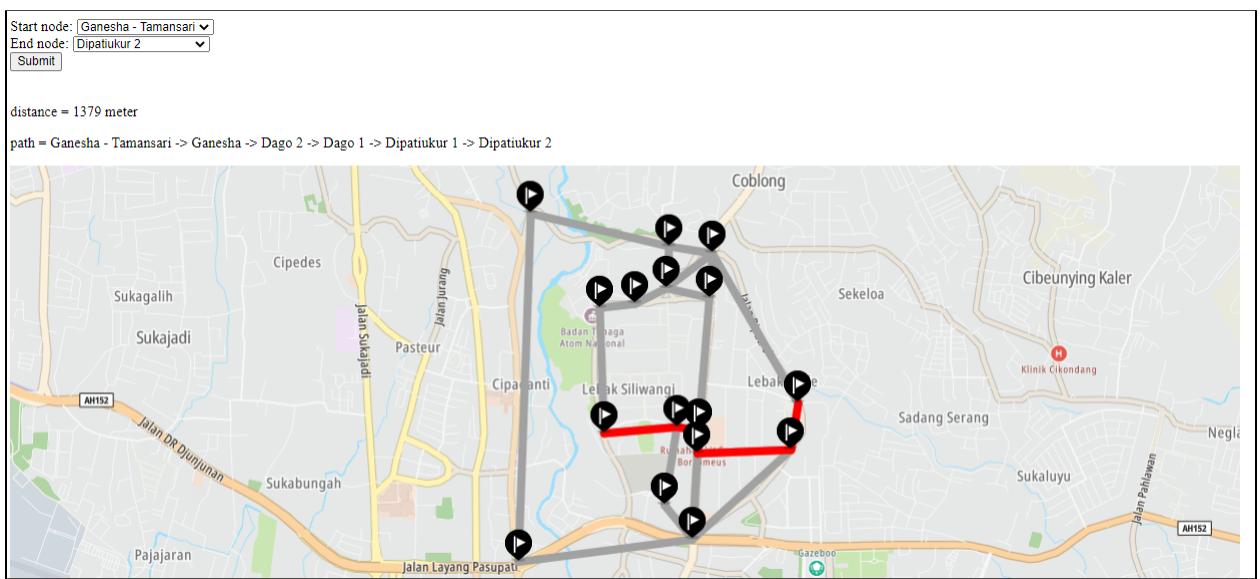
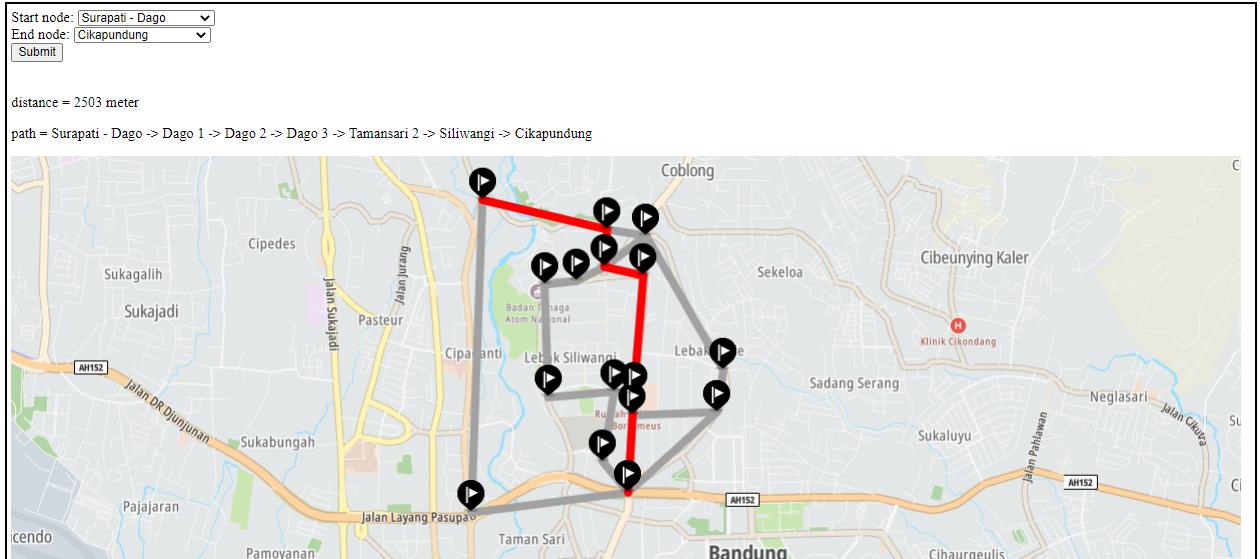
Fagaras,45.842,24.973
Timisoara,45.756,21.231
Lugoj,45.691,21.903
Mehadia,44.904,22.365
Drobeta,44.639,22.659
Cralova,44.319,23.794
Rimnicu Vilcea,45.099,24.369
Pitesti,44.856,24.869
Giurgiu,43.905,25.969
Urziceni,44.718,26.645
Neamt,47.056,26.506
Iasi,47.158,27.598
Vaslui,46.641,27.728
Hirsova,44.690,27.945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

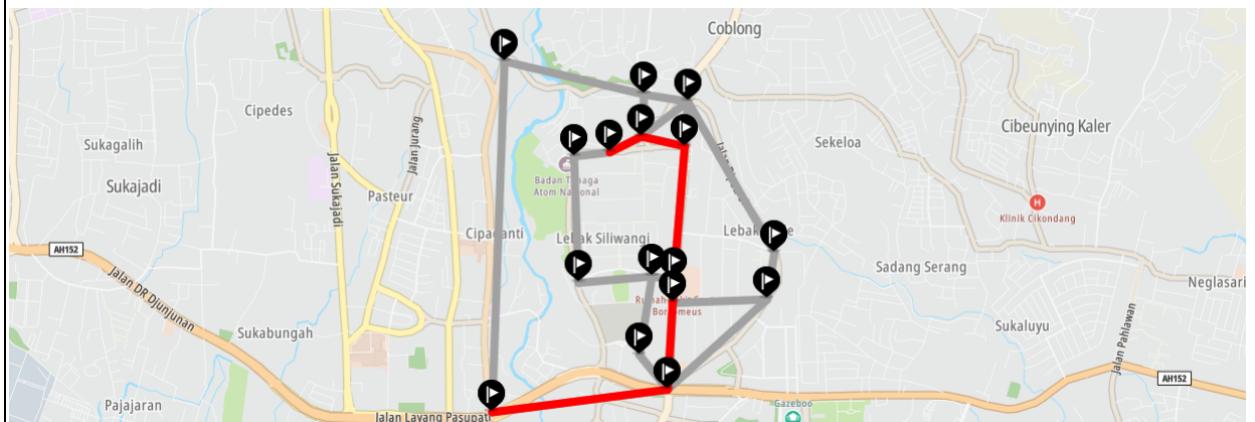
1. Peta jalan sekitar kampus ITB/Dago



Start node: Cihampelas
 End node: Sabuga

distance = 2645 meter

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

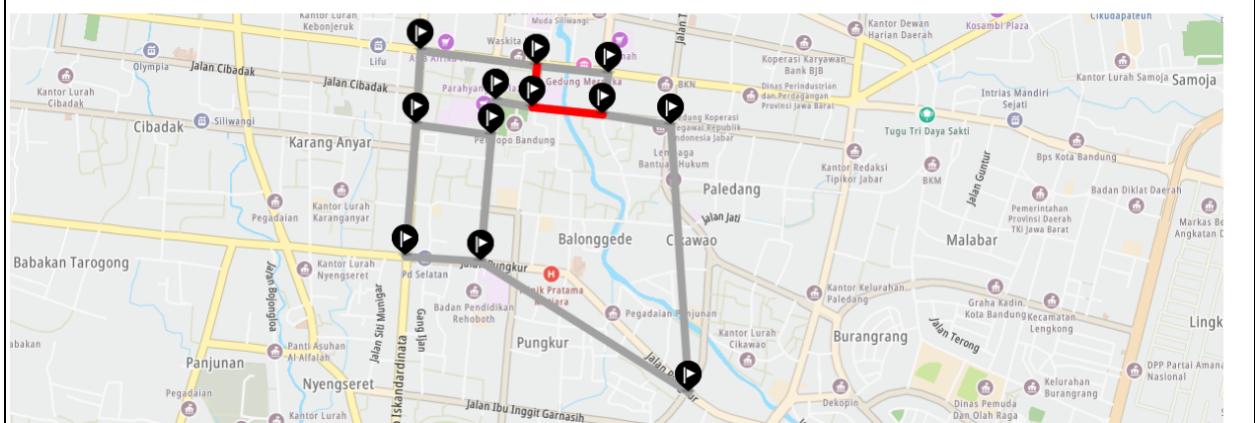


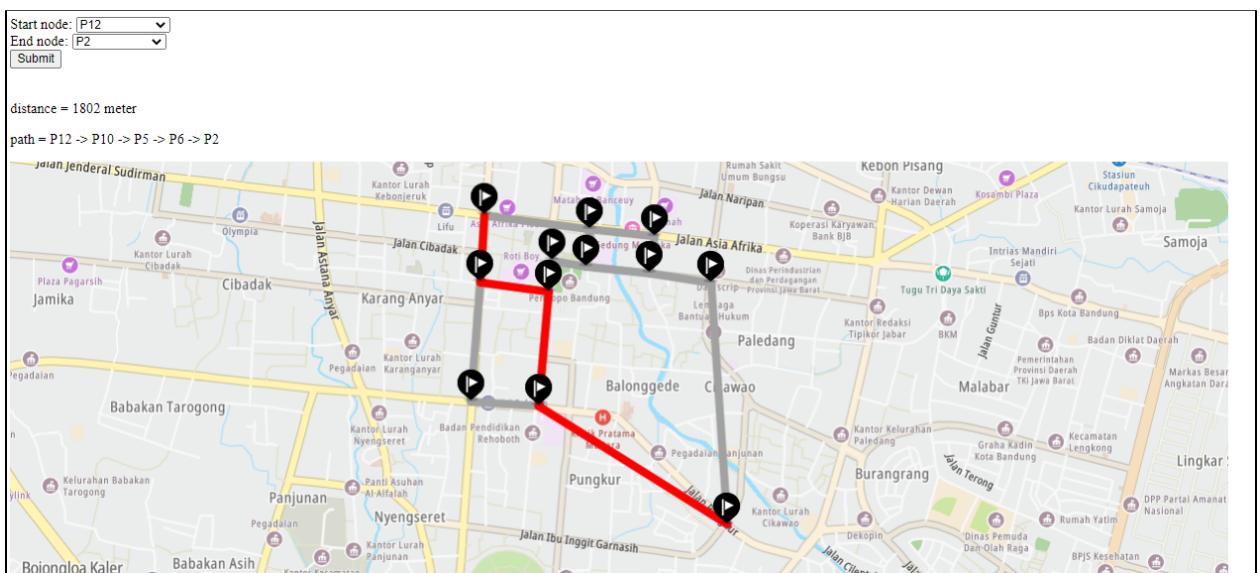
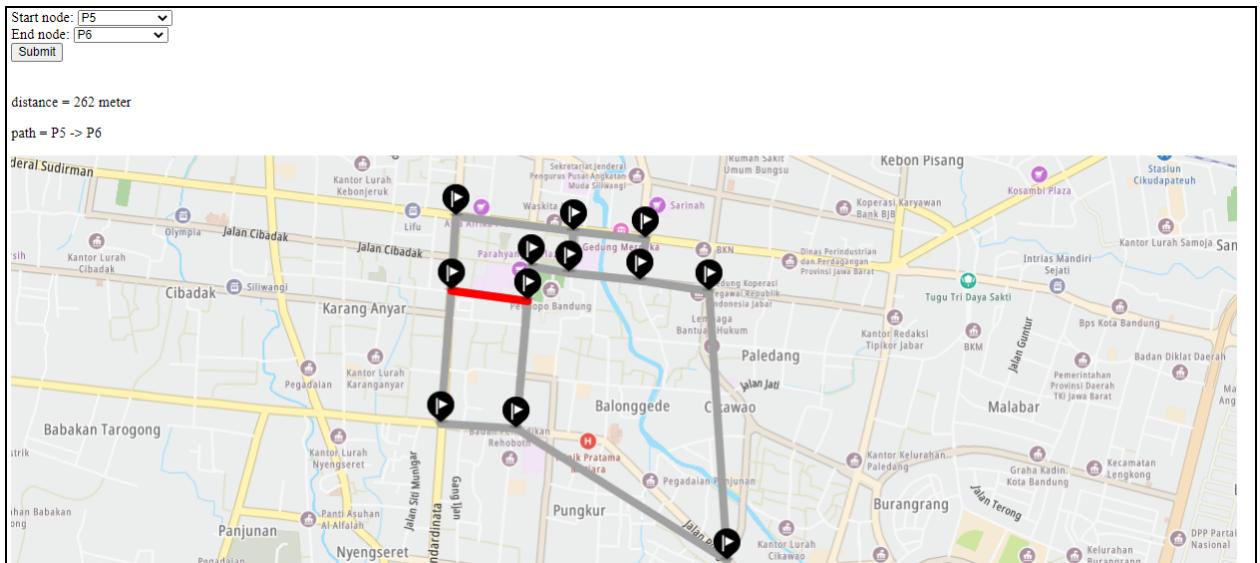
2. Peta jalan sekitar Alun-alun Bandung

Start node: Masjid Agung
 End node: P8

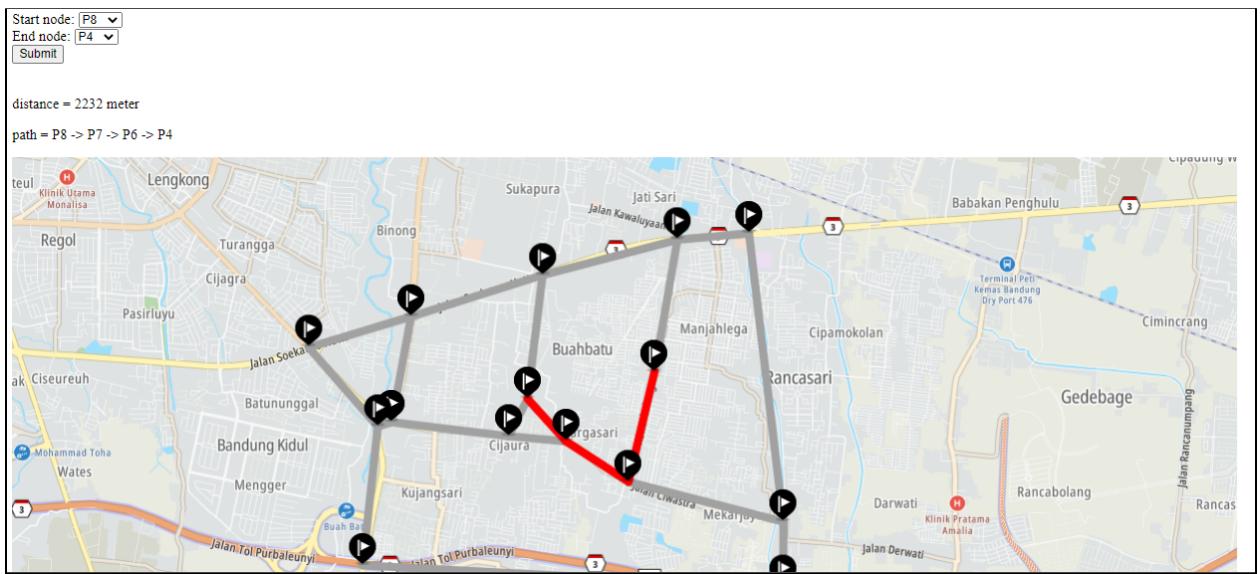
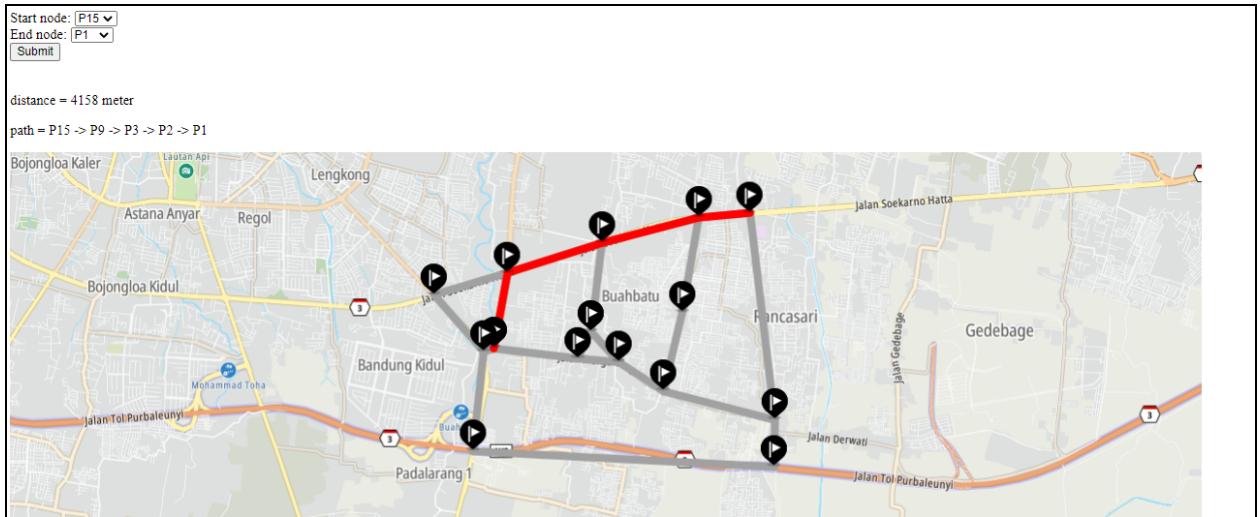
distance = 386 meter

path = Masjid Agung -> P3 -> P8

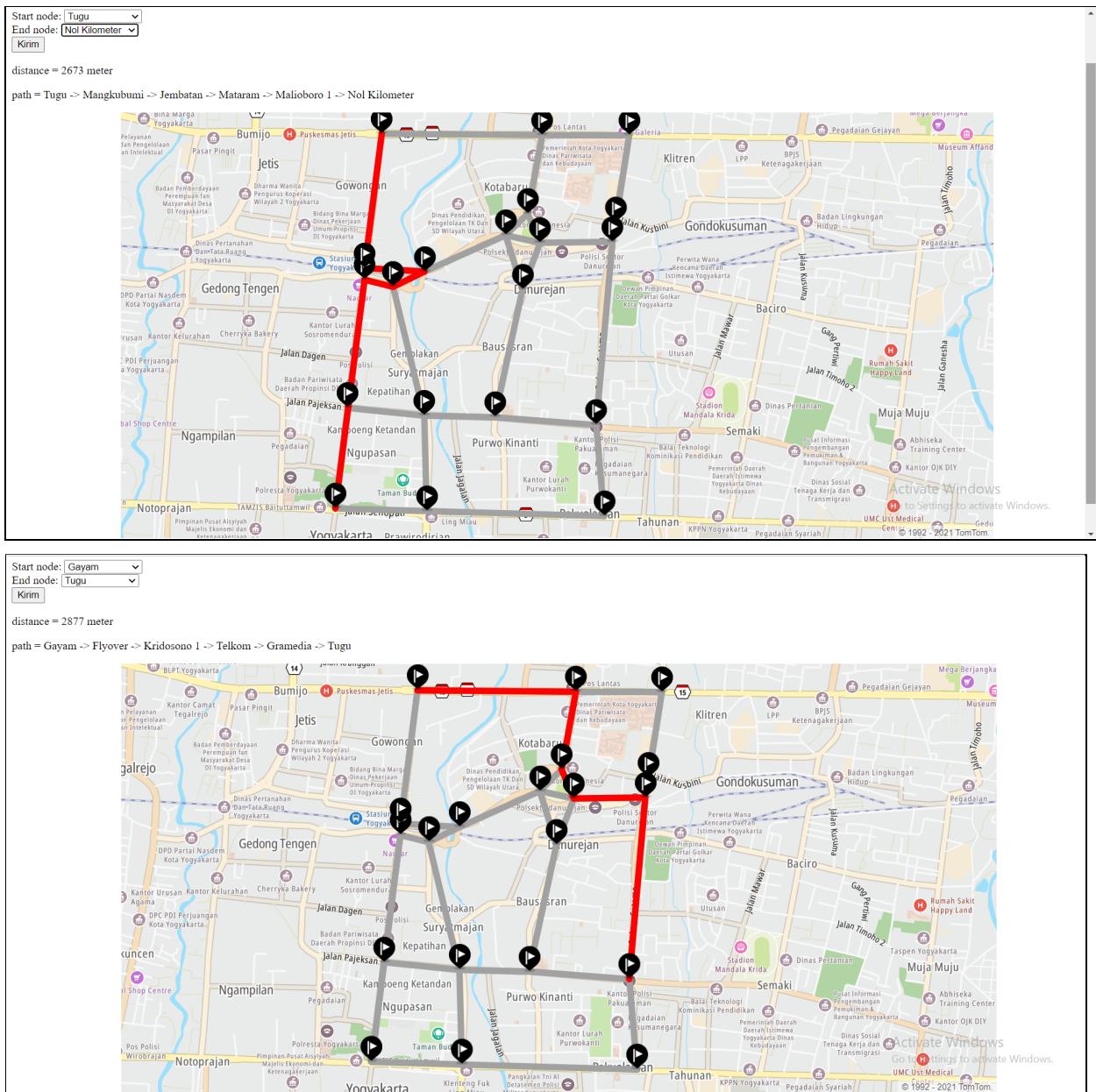




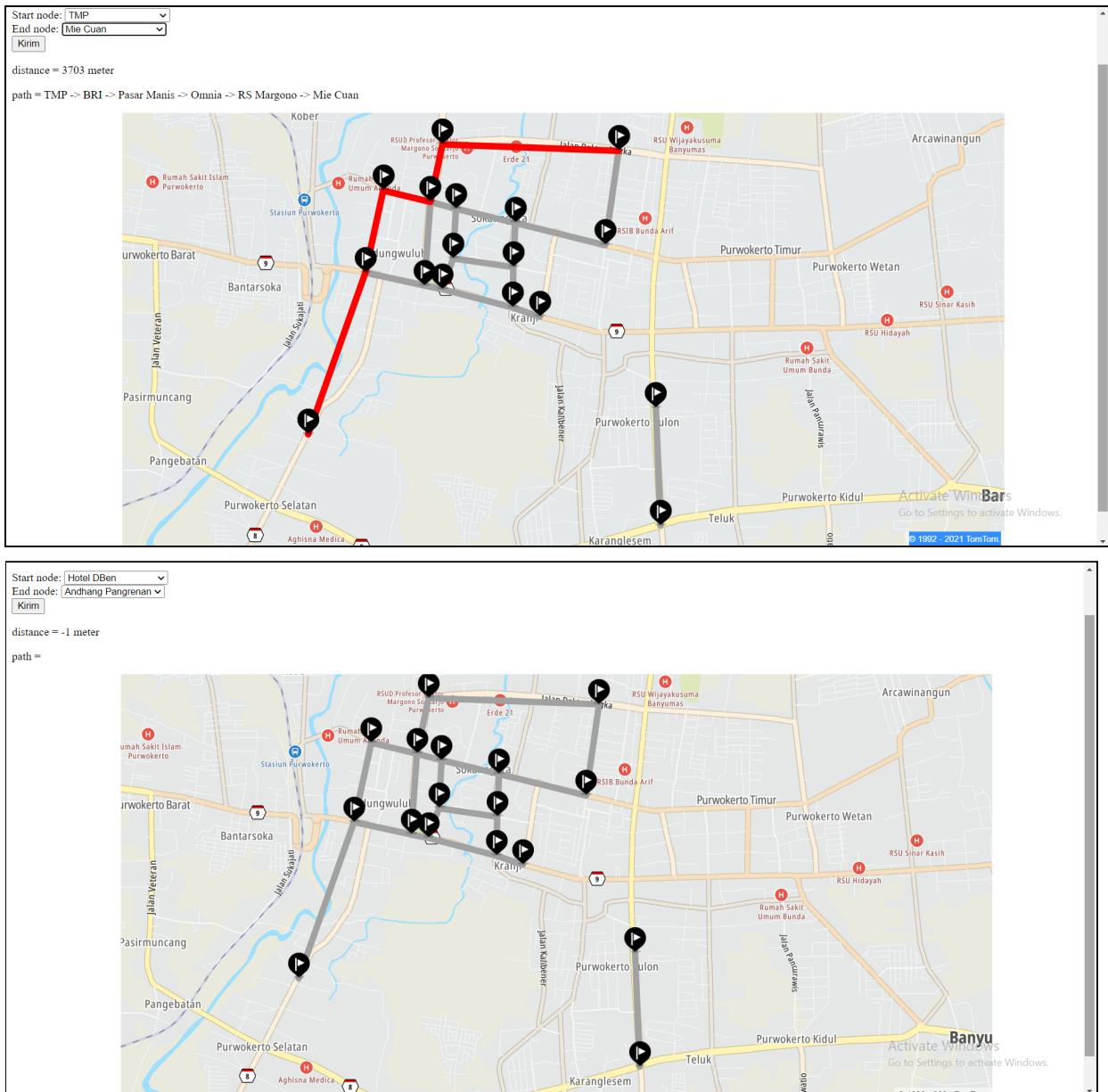
3. Peta jalan sekitar Buahbatu



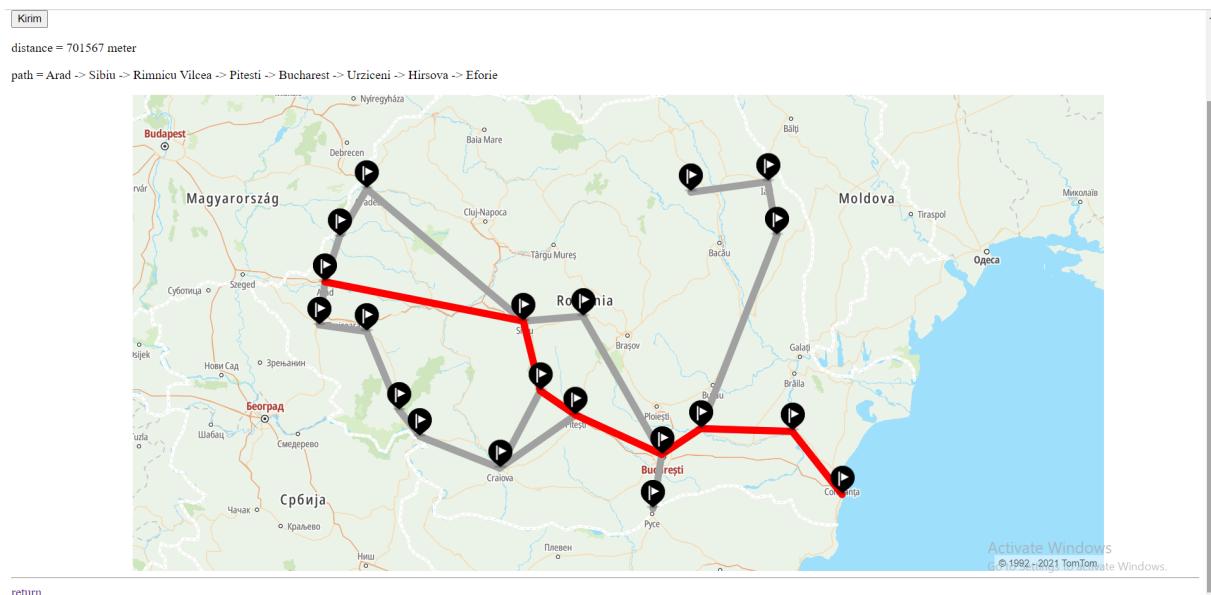
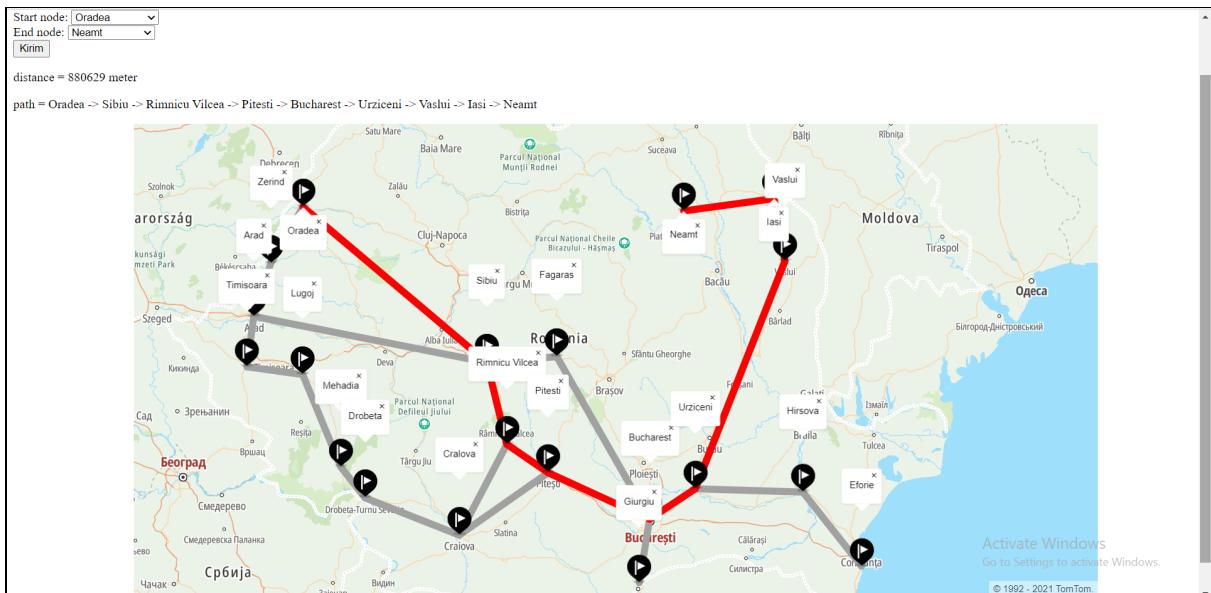
4. Peta jalan sekitar Stadion Kridosono Yogyakarta



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