

Tugas Kecil III IF2211 Strategi Algoritma Semester 2  
Tahun 2020/2021

Implementasi Algoritma A\* untuk Menentukan Lintasan  
Terpendek



Disusun oleh :  
Muhammad Fahmi Alamsyah - 13519077  
Widya Anugrah Putra - 13519105

PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG

## 1. Algoritma A\* yang diimplementasikan

Pertama-tama, website akan membaca file input graf berdasarkan pilihan user. Pada baris pertama berisi banyaknya node yang ada pada graf, misal N, lalu pada N baris selanjutnya berisi koordinat latitude dan longitude dari suatu simpul, beserta nama simpulnya. Selanjutnya file akan berisi matriks ketetanggaan berukuran  $N \times N$ . Program yang kami buat mampu menerima 2 jenis matriks ketetanggaan, matriks yang pertama berupa matriks ketetanggaan yang bertipe matriks boolean (0 untuk tidak ada edge dan 1 untuk ada edge), dan jenis matriks yang kedua bertipe matriks ketetanggaan berbobot (0 untuk tidak ada edge dan  $>0$  untuk menyatakan bobot edge).

Setelah program membaca file input graf, website akan meminta nama node awal dan nama node tujuan dari pengguna. Selanjutnya kedua node akan divalidasi apakah ada pada peta atau tidak dengan menggunakan program bagian backend. Lalu program akan mencari path dari permintaan pengguna menggunakan algoritma A\* sebagai berikut:

- a. Program akan membuat sebuah priority queue yang dibuat khusus untuk menyimpan node mana yang ter-ekspansi, misal namanya OPEN, dan sebuah list yang akan menyimpan node-node mana saja yang sudah mati, misal namanya CLOSED, dan juga list sebesar banyaknya node yang akan berisi predecessor dari node tersebut, misal namanya PARENT. Selanjutnya, masukkan node awal ke dalam queue. Selain itu dibuat flag boolean Found yang akan menunjukkan node tujuan ditemukan atau tidak. Found diinisiasi dengan False.
- b. Lalu lakukan while loop selama queue tidak kosong dan node tujuan belum ditemukan.
- c. Dalam while loop tersebut akan dilakukan:
  - Ambil dan hapus node terdepan dalam queue. Node yang diambil dari queue OPEN tersebut adalah node hidup sekarang. Masukkan node hidup tersebut ke dalam list CLOSED, karena node tersebut langsung dimatikan (tidak akan dikunjungi dalam loop selanjutnya).
  - Sebelum melakukan pengecekan terhadap tetangga node hidup, cek terlebih dahulu apakah node hidup tersebut adalah node tujuan atau tidak. Jika ternyata node tujuan, maka flag Found menjadi True.
  - Jika belum ditemukan, lakukan pengecekan terhadap tetangga-tetangga node hidup. Jika tetangga belum diekspansi (belum ada pada queue OPEN) atau sudah diekspansi tetapi nilai  $f(n)$  dari path sekarang lebih kecil dari nilai  $f(n)$  path sebelumnya, maka set/update nilai  $f(n)$  pada node tersebut, lalu ubah parent dari node tersebut menjadi node hidup. Lalu jika ternyata node tersebut belum diekspansi, tambahkan node tersebut ke dalam queue OPEN (diekspansi).

- d. Setelah keluar dari while loop, program akan mengecek apakah node tujuan dapat dicapai atau tidak dari flag Found. Jika tidak ditemukan, maka program backend akan memberitahu website untuk melakukan peringatan bahwa node tidak ditemukan.
- e. Jika ditemukan, maka program akan mencari jalur dari node awal hingga node tujuan. Program akan membuat list yang akan diisi jalur dari node awal hingga akhir. Misal list yang dibuat bernama Path, maka Path pada awalnya append node tujuan. Pada awalnya dibuat variabel dummy yang diassign dengan node tujuan, lalu dilakukan while loop selama dummy belum sama dengan node awal, maka Path akan append parent dari dummy dan dummy akan menunjuk parent dari dummy. Setelah keluar dari while loop, maka list Path akan di-reverse, sehingga ditemukan jalur yang terpendek dari node awal hingga node tujuan.
- f. Selanjutnya website akan menerima hasil pekerjaan program dan mulai menggambar jalur dan menghitung jarak jalurnya pada peta.

## 2. Source code program yang mengolah data

```

from math import *

class PrioQueueMod(object):
    def __init__(self):
        self.queue = []
    def isEmpty(self):
        return len(self.queue) == 0

    def isMember(self,data): #based on key
        for i in range(len(self.queue)):
            if (self.queue[i][0] == data):
                return True
        return False

    def nthIndex(self, data): #based on key
        for i in range(len(self.queue)):
            if (self.queue[i][0] == data):
                return i
        return -1

    def insert(self,data): #always update the value of f(n), now data consist of gn and hn
        if (self.isMember(data[0])):
            self.queue.pop(self.nthIndex(data[0]))
            self.insert(data)
            return
        self.queue.append(data)
        if (len(self.queue) > 1):
            idxPred = len(self.queue)-2

```

```

    while (idxPred >= 0 and self.queue[idxPred][1]+self.queue[idxPred][2] >
data[1]+data[2]):
        self.queue[idxPred+1] = self.queue[idxPred]
        self.queue[idxPred] = data
        idxPred -= 1

def delete(self):
    return self.queue.pop(0)

def show(self):
    for i in range(len(self.queue)):
        print(str(self.queue[i][0]) + " dengan value g(n): " + str(self.queue[i][1]) + " dan value h(n)
: " +str(self.queue[i][2]))

def getFnKey(self,idx):
    dummy = self.nthIndex(idx)
    if (dummy == -1):
        return 0
    return self.queue[dummy][1] + self.queue[dummy][2]

def distanceInMeter(lat, lng, lat0, lng0):
    x1 = radians(lat)
    x2 = radians(lat0)
    y1 = radians(lng)
    y2 = radians(lng0)
    difx = x1-x2
    dify = y1-y2
    a = sin(difx/2)**2 + cos(x1)*cos(x2)*sin(dify/2)**2
    c = 2 * asin(sqrt(a))
    r = 6378137
    return r*c

def bacaFile(namaFile):
    try:
        f=open('../test/'+ namaFile, 'r')
    except IOError:
        return [],[],False
    N = int(f.readline())
    adjMatrix = [[(-1) for i in range(N)] for i in range(N)]
    listNode = []
    listCoor = []

    for i in range(N):
        line = f.readline()

```

```

idxSpace = line.find(" ")
coorx = float(line[0:idxSpace])
dummy = idxSpace
idxSpace = line.find(" ", dummy+1)
coory = float(line[dummy+1:idxSpace])
listCoor.append((coorx,coory))
listNode.append(line[idxSpace+1:-1])

```

```

idx1 = 0
line = f.readline()
while line != "":
    dummy = 0
    idxSpace = line.find(" ")
    idx2 = 0
    while (idxSpace != -1):
        adjMatrix[idx1][idx2] = int(line[dummy:(idxSpace+1)])
        dummy = idxSpace + 1
        idxSpace = line.find(" ", dummy)
        idx2 += 1
    if (idx1 == N-1) : adjMatrix[idx1][idx2] = int(line[dummy:])
    else : adjMatrix[idx1][idx2] = int(line[dummy:-1])
    idx1 += 1
    line = f.readline()
f.close()
return adjMatrix, listNode, listCoor, True

```

```

def find(array, element):
    try:
        dummy = array.index(element)
    except ValueError:
        return -1
    return array.index(element)

```

```

def isWeightedGraf(adjMatrix):
    for i in range(len(adjMatrix)):
        for j in range(len(adjMatrix[i])):
            if (adjMatrix[i][j] > 1):
                return True
    return False

```

```

def middlePoint(listCoor):
    lat = 0
    long = 0
    for i in range(len(listCoor)):

```

```

lat += listCoor[i][0]
long += listCoor[i][1]
return (lat/len(listCoor),long/len(listCoor))

```

```

def hitungJarakPath(adjMatrix,path,listCoor):
    jarak = 0
    if (isWeightedGraf(adjMatrix)):
        current = 0
        while (current != len(path)-1):
            jarak+= adjMatrix[path[current]][path[current+1]]
            current+=1
        else:
            current = 0
            while (current != len(path)-1):
                jarak+= distanceInMeter(listCoor[path[current]][0],listCoor[path[current]][1],
listCoor[path[current+1]][0],listCoor[path[current+1]][1])
                current += 1
            return jarak

```

```

def main(adjMatrix,listNode,listCoor,node1,node2):
    idx1 = find(listNode,node1)
    idx2 = find(listNode,node2)
    if (idx1 == -1 or idx2 == -1):
        return [],False,False
    closed = []
    queue = PrioQueueMod()
    queue.insert((idx1,0,0))
    parent = [0 for i in range(len(listNode))]
    found = False
    while (not queue.isEmpty() and not found):
        dummy = queue.delete()
        gcostParent = dummy[1]
        current = dummy[0]
        closed.append(current)

        if current == idx2:
            found = True
        if (not found):
            for i in range(len(listNode)):
                if (adjMatrix[current][i] != 0 and find(closed,i) == -1):
                    gcost = gcostParent + distanceInMeter(listCoor[current][0],listCoor[current][1],
listCoor[i][0], listCoor[i][1])
                    hcost = distanceInMeter(listCoor[i][0],listCoor[i][1],listCoor[idx2][0],listCoor[idx2][1])

```

```

if not found:
    return [],True,False

path = []

path.append(idx2)

dummy = idx2

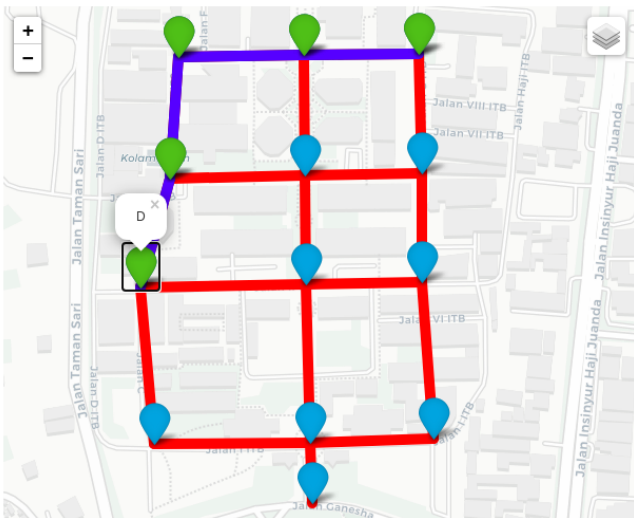
while (dummy != idx1):
    path.append(parent[dummy])
    dummy = parent[dummy]

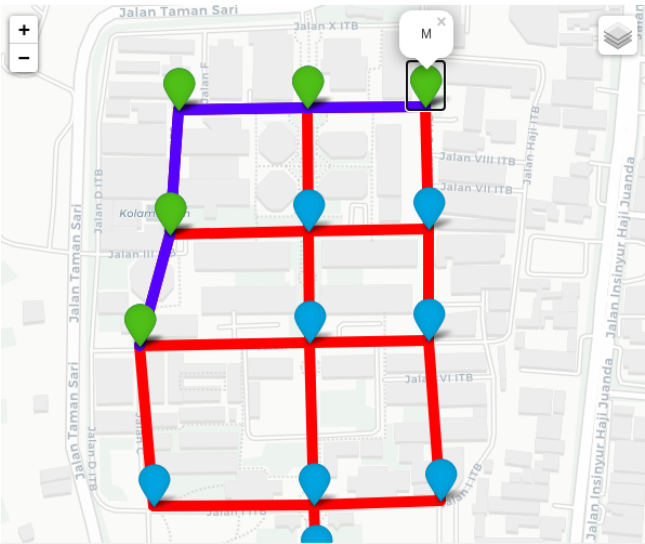
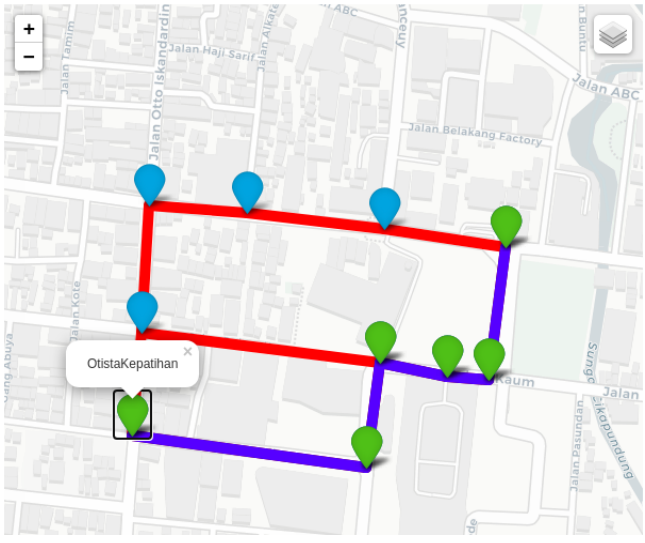
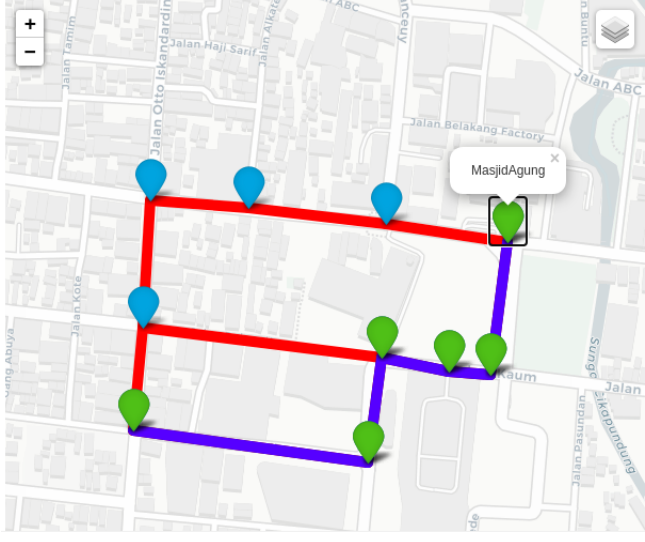
path.reverse()

return path,True,True

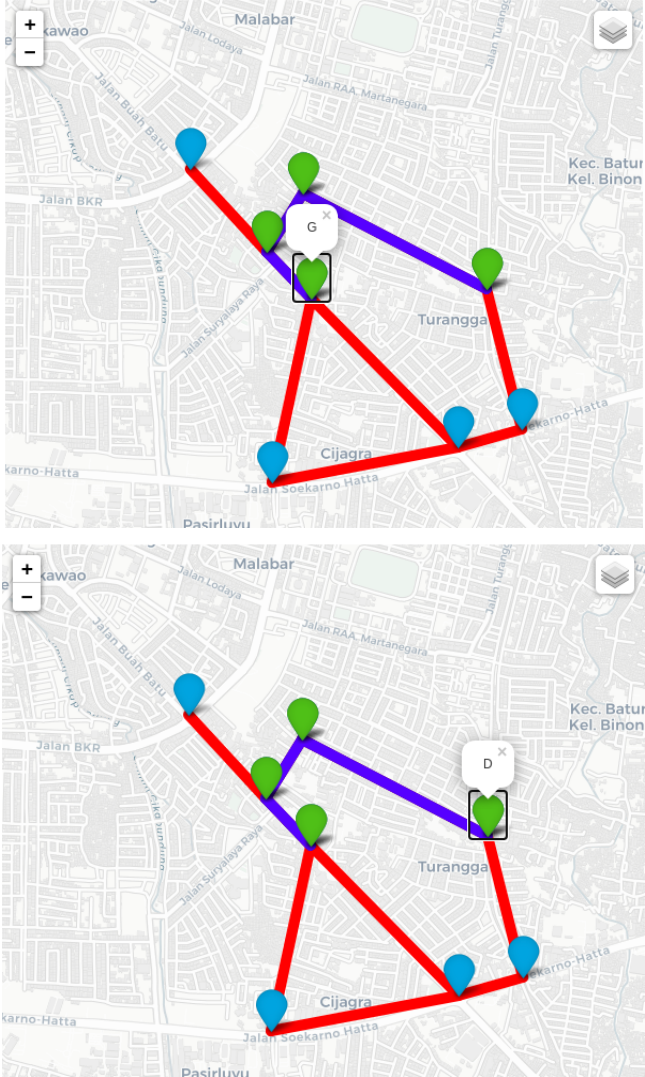
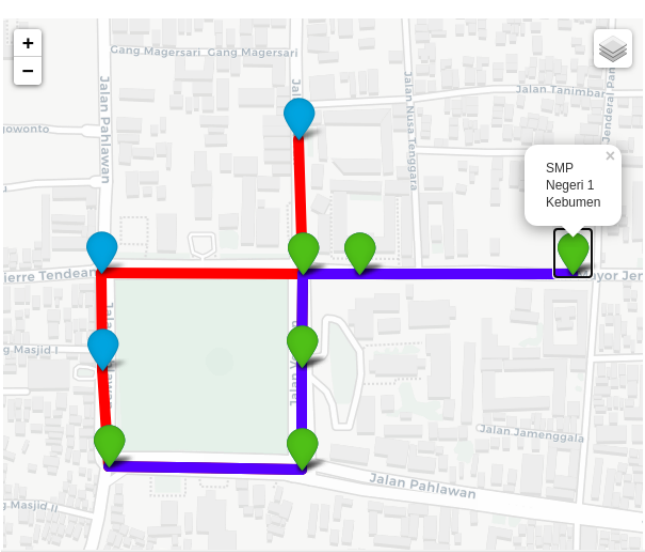
```

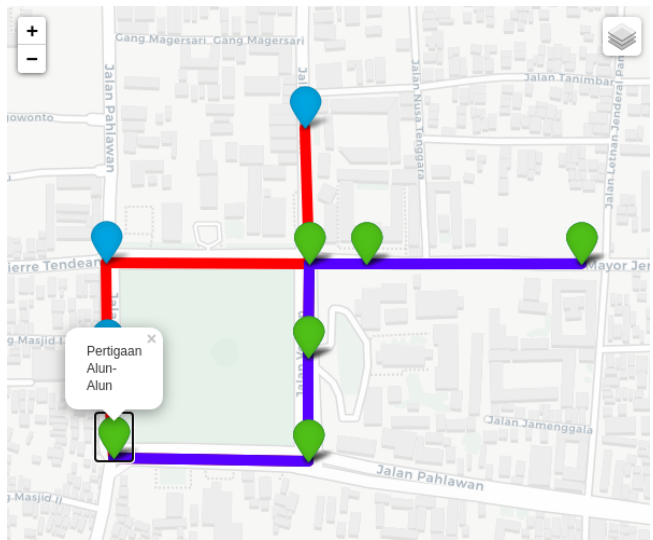
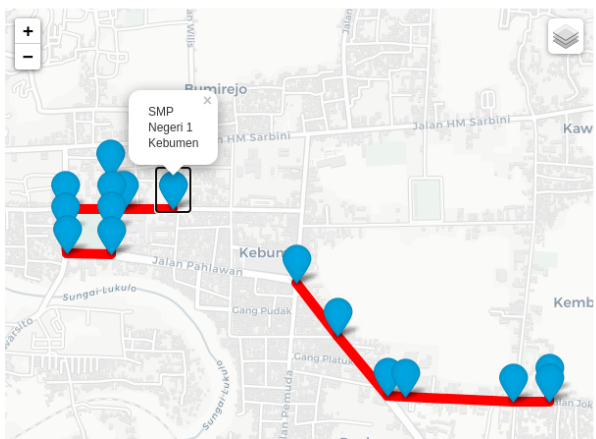
### 3. Contoh input/output :

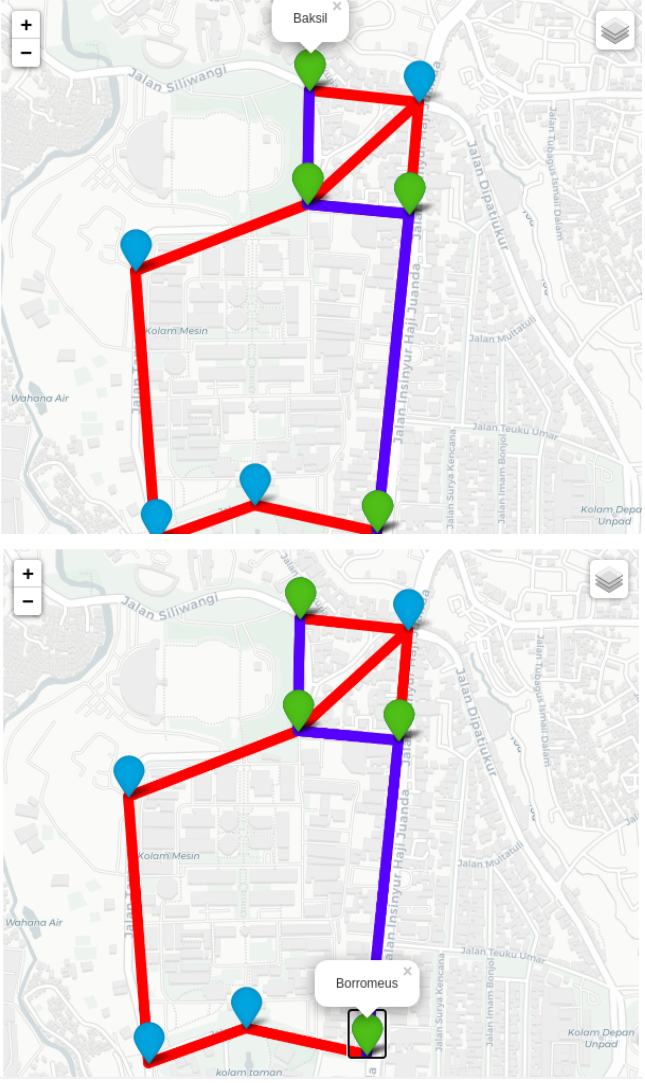
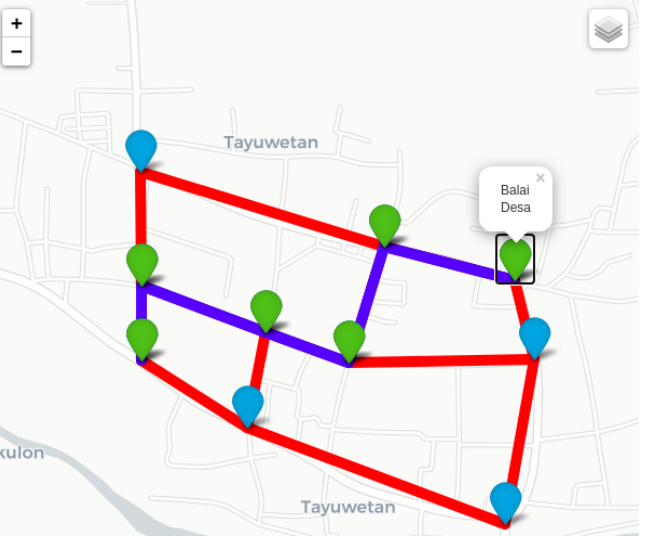
| No | Peta                             | Screenshot   |
|----|----------------------------------|--|
| 1  | ITB<br>Dari simpul M ke simpul D |  |

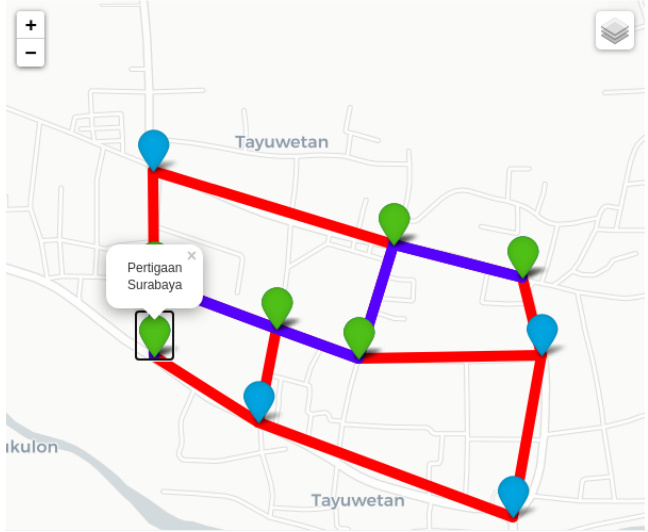
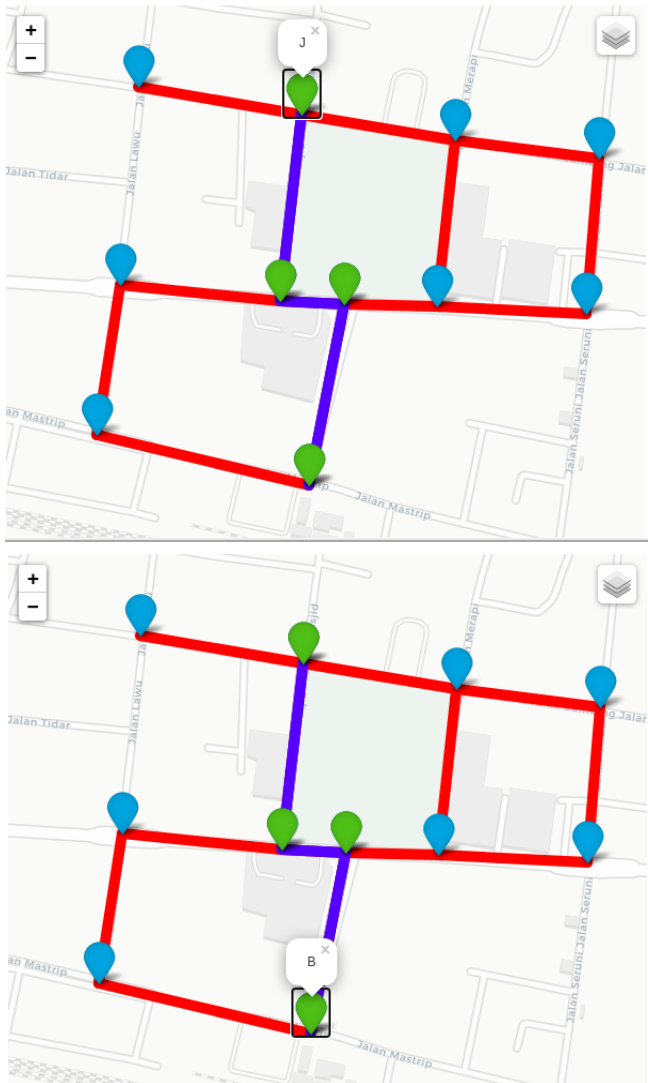
|   |  |   |
|---|--|---|
|   |  |   |
| 2 | Alun-alun Bandung<br>Dari simpul<br>MasjidAgung hingga<br>ke OtistaKepatihan | <br> |



|          |   |   |
|----------|---|---|
| <p>3</p> | <p>Buahbatu<br/>Dari simpul G hingga ke simpul D</p>                              |  <p>The top map shows a path from node G (green heart) to node D (green heart) via Cijagra (blue dot) and Turangga (blue dot). The path is highlighted in red. The bottom map shows the same path from G to D via Cijagra and Turangga, but the path is highlighted in blue.</p> |
| <p>4</p> | <p>Kebumen<br/>Dari simpul SMP Negeri 1 Kebumen ke simpul Pertigaan Alun-Alun</p> |  <p>The map shows a path from SMP Negeri 1 Kebumen (green heart) to Pertigaan Alun-Alun (blue dot). The path is highlighted in red. The path starts at SMP Negeri 1 Kebumen, goes south to a junction, then west to Pertigaan Alun-Alun.</p>                                    |

|   |  |  |
|---|--|--|
|   |  |    |
| 5 | <p>Kebumen<br/>Dari simpul Rumah<br/>Alam ke simpul SMP<br/>Negeri 1 Kebumen</p> <p>Kasus tidak ditemukan<br/>path</p> | <div>Tidak ditemukan jalan menuju lokasi tujuan</div>  <div>Tidak ditemukan jalan menuju lokasi tujuan</div>  |

|          |   |  |
|----------|---|--|
| <p>6</p> | <p>Sekitar ITB<br/>Dari simpul Baksil ke<br/>simpul Borromeus</p>           |  <p>The top map shows a network of red and blue lines connecting nodes in the ITB area. The nodes are marked with blue and green hearts. The path starts at Baksil (green heart) and ends at Borromeus (blue heart). The bottom map shows the same network, but the path starts at Borromeus (green heart) and ends at Baksil (blue heart).</p> |
| <p>7</p> | <p>Tayu<br/>Dari simpul Balai Desa<br/>ke simpul Pertigaan<br/>Surabaya</p> |  <p>The map shows a network of red and blue lines connecting nodes in the Tayu area. The nodes are marked with blue and green hearts. The path starts at Balai Desa (green heart) and ends at Pertigaan Surabaya (blue heart).</p>   |

|   |                                     |   |
|---|-------------------------------------|---|
|   |                                     |   |
| 8 | Blitar<br>Dari simpul J ke simpul B |  |

|   |                                       |   |
|---|---------------------------------------|---|
| 9 | Monas<br>Dari simpul Q ke<br>simpul B |  |
|---|---------------------------------------|---|

#### 4. Alamat repository program

Project ini dapat dilihat di <https://github.com/widyaput/Tucil3STIMA> yang akan dipublic ketika *deadline* selesai.

|   |   |   |
|---|---|---|
| 1 | Program dapat menerima input graf   | ✓ |
| 2 | Program dapat menghitung lintasan terpendek   | ✓ |
| 3 | Program dapat menampilkan lintasan terpendek serta jaraknya                         | ✓ |
| 4 | Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta | ✓ |

