LAPORAN TUGAS KECIL III
IMPLEMENTASI ALGORITMA A* UNTUK MENENTUKAN LINTASAN TERPENDEK
IF2211 Strategi Algoritma

Oleh :

Haikal Lazuardi Fadil - 13519027
Harith Fakhiri Setiawan - 13519161

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2021

BAB I
Kode Program

astar.py:

src > astar.py

```python
from math import radians, cos, sin, asin, sqrt
# Membaca text dari file, mengembalikan multi dimentional array persis seperti file.txt
def bacaTextFile(x):
    f = open(x,'r')
    file = f.readlines()
    operand = []
    #melakukan replace beberapa character seperti koma, titik, dan newline
    for lines in file:
        lines = lines.replace("\n"," ")
        operand.append(lines)
    opfinal = []
    for lines in operand:
        lines = lines.split()
        opfinal.append(lines)
    return opfinal

### Membaca multidimensional array dari file yang dibaca, mengembalikan listofnode ###
def listOfNode(x):
    starting = int(x[0][0])
    node = []
    for i in range(1,starting+1):
        node.append(x[i][0]) #append list node dari array input
    return node

# Membuat dan mengembalikan array berbentuk list of tupple,
# yang berisi tupple ketetanggaan atau edges dari graph
def tetangga(x,listnode):
    matrix = []
    starting = int(x[0][0])
    for i in range(starting + 1, len(x)):
        for j in range(len(x[starting + 1])):
```

src > astar.py

```python
    for i in range(starting + 1, len(x)):
        for j in range(len(x[starting + 1])):
            if(int(x[i][j]) > 0):
                #append edge node pada matrix yang bertetangga apabila bernilai 1
                matrix.append((listnode[i-(starting+1)],listnode[j]))
    return matrix

# Membuat list of list of location dari masing2 node #
def coordinates(x):
    matrix =[]
    temp = []
    starting = int(x[0][0])
    for i in range(1, starting+1):
        temp.append(x[i][1])
    for loc in temp:
        loc = loc.replace('(',' ')
        loc = loc.replace(')',' ')
        loc = loc.replace(',',' ')
        matrix.append(loc)
    loc = []
    for lines in matrix :
        lines = lines.split()
        loc.append(lines)
    return loc

```

src > astar.py

```python
# Membaca elemen dari list of list of location, mengembalikan jarak dengan metode haversine #
def jarak(latitudeA, longitudeA, latitudeB, longitudeB):
    longitudeA, latitudeA, longitudeB, latitudeB = map(radians, [longitudeA, latitudeA, longitudeB, latitudeB])
    dlon = longitudeB - longitudeA
    dlat = latitudeB - latitudeA
    a = sin(dlat/2)**2 + cos(latitudeA) * cos(latitudeB) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371 * (1000)
    dist = round(c*r, 3)
    return dist

# Membaca array pada bagian matrix ketetanggan, kemudian membuat matrix bobot yang berisikan
# jarak yang dihitung dengan metode haversine formula antar simpul.
def matrixJarak(arr, loc):
    matrix = []
    starting = int(arr[0][0])
    for i in range(starting+1, len(arr)):
        temp = []
        for j in range(len(loc)):
            temp.append(jarak(float(loc[i-(starting+1)][0]), float(loc[i-(starting+1)][1]), float(loc[j][0]), float(l
        matrix.append(temp)
    return matrix

```

src > astar.py

```python

# Membaca matrix ketetanggan, mengembalikan tetangga dari suatu node
def getNeighbour(arrNeigh, node):
    matrix = []
    for elmt in arrNeigh:
        if elmt[0] == node:
            matrix.append(elmt[1])
            #mengappend tetangga dari node yang ada di input
            #berdasarkan array ketetanggaan
    return matrix

# Menerima sebuah array dari listnode, kemudian mencari dan mengembalikan
# indeks dari node yang dicari
def getIdx(array, node):
    i = 0
    found = False
    while(found == False and i < len(array)):
        if (array[i] == node):
            found = True
        else:
            i += 1
    if (found == True):
        return i
    else:
        return -1 #mengembalikan -1 apabila node tidak ditemukan

# Mengurutkan list tuple yang berisi node asal, node tujuan, berdasarkan jarak antara keduanya
def sortingFn(fnCandidate):
    # algoritma selection short
    for i in range(len(fnCandidate)):
```

src > astar.py

```python
        min_idx = i
        for j in range(i+1, len(fnCandidate)):
            if float(fnCandidate[min_idx][1]) > float(fnCandidate[j][1]):
                min_idx = j
        fnCandidate[i], fnCandidate[min_idx] = fnCandidate[min_idx], fnCandidate[i]
    return fnCandidate

```

src > astar.py

```python
# Mencari jalur terpendek dari node asal ke node tujuan #
def closestPath(srcNode, destNode, arrNeigh, Distance, listNode):
    candidate = []
    candidateNode = [srcNode]
    visited = []
    destIdx = getIdx(listNode, destNode) #index node yang dituju
    if (srcNode == destNode):
        return [srcNode, 0]
    else :
        currentNode = srcNode
        predSrc = 0
        first = True
        while (len(candidate) > 0 or first == True):
            first = False
            currNeighbour = getNeighbour(arrNeigh, currentNode)
            for node in currNeighbour:
                if (node not in candidateNode):
                    pred = currentNode #menyimpan prenode
                    currentCheck = node
                    predIdx = getIdx(listNode,pred) #indeks prenode
                    idx = getIdx(listNode, currentCheck) #indeks node yang sedang di cek
                    srcToN = predSrc + float(Distance[idx][predIdx]) #gn
                    nToDest = float(Distance[idx][destIdx]) #hn
                    temp = srcToN + nToDest #fn
                    candidate.append((currentCheck, temp, srcToN, currentNode))
            candidate = sortingFn(candidate) #sorting list candidate berdasarkan fn
            #memastikan untuk tidak berenti iterasi ketika terdapat kasus dimana len(candidate) = 1 dan perlu dipop,
            #namun masih terdapat tetangga yang harus diiterasi dari current sehingga masih memenuhi syarat while
            if  (len(getNeighbour(arrNeigh, currentNode)) > 0 and len(candidate) == 1):
                candidate.append(('',9999999999,0,''))
```

src > astar.py

```python
                candidate.append(('',9999999999,0,''))
            a = candidate.pop(0) #pop elemen pertama hasil sorting
            visited.append((a[3],a[0],a[1],a[2])) #append prenode,currnode,fn,gn
            candidateNode.append(a[0])
            currentNode = a[0]
            predSrc = float(a[2])
        finalMove = getMinFn(destNode, visited)
        if (str(finalMove[1]) == str(destNode)):
            path = derivate(destNode,srcNode,visited)
            return path
        else:
            print("jalur tidak ditemukan")
            return []

# Mencari node tujuan dari list tuple visited(yang sudah dikunjungi) berdasarkan
# jarak terpendek mengembalikan 1 elemen dari visited
def getMinFn(destNode,visited):
    min = 999999999
    result = visited[0]
    for i in range (len(visited)):
        if (min > float(visited[i][2]) and visited[i][1] == destNode):
            min = float(visited[i][2])
            result = visited[i]
    return result

```

src > astar.py

```
168  # menerima input destination dan sourcenode, serta visited, kemudian menelusuri
169  # path dari destination node ke source node, dan mengembalikan list of tupple
170  # yang berisi node, dan jarak yang ditempuh dari src node
171  def derivate(destNode,srcNode,visited):
172      path = []
173      nodeBacktrack = destNode
174      destTuple = getMinFn(destNode, visited) #append terakhir
175      found = False
176      while (found == False):
177          for i in range(len(visited)):
178              if (nodeBacktrack == visited[i][1]):
179                  path.append((destTuple[1],destTuple[3]))
180                  nodeBacktrack = destTuple[0] #backtrack mundur
181                  destNode = destTuple[0]
182                  if(destTuple[0] == srcNode):
183                      if(srcNode not in path):
184                          path.append((srcNode, 0))
185                      found = True #berhenti loop
186                  destTuple = getMinFn(destNode, visited)
187      path.reverse() #reverse list hasil append
188      return path
```

main.py:

src > main.py

```
1   from astar import bacaTextFile, listOfNode, tetangga, coordinates, jarak, matrixJarak, getNeighbour, getIdx, sortingF
2   from web import server
3   import sys
4
5   #apabila hanya menjalankan 'python main.py' pada terminal, maka akan dijalankan main python standar
6   if len(sys.argv) == 1:
7       filename = str(input("Masukkan nama file : "))
8       fileDir = "../test/" + filename
9       arr = bacaTextFile(fileDir)
10      # ['A', 'B', 'C', 'D', ...] A,B,C,D,... adalah node
11      listNode = listOfNode(arr)
12      # [('A', 'B'), ('A', 'C'), ...] B tetangga A, C tetangga A
13      arrNeigh = tetangga(arr, listNode)
14      # [['1', '2'], ['2', '1'], ['1','1'], ...] elmt pertama adalah latitude, dan kedua adalah longitude
15      loc = coordinates(arr)
16      matrixDistance = matrixJarak(arr, loc)
17      # 0 2 1
18      # 2 0 2
19      # 1 2 0
20      src = str(input("Masukkan node awal : "))  # A
21      dest = str(input("Masukkan node tujuan : "))  # G
22      # [('A', 0), ('B', 20.62), ('G', 159.84)], elmt kedua dari tuple dalam satuan meter
23      jalan = closestPath(src, dest, arrNeigh, matrixDistance, listNode)
24      print(jalan)
25
```

src > main.py

```
26  # apabila pada terminal 'dijalankan python main.py gui' maka akan menjalankan GUI berupa
27  # web lokal yang telah dibuat
28  else:
29      if sys.argv[1] == "gui":
30          server.run(port=5000, debug=True)
31
32      else:
33          print("Invalid command, either run \"python3 main.py web\" or \"python3 main.py\"")
34
```

web.py:

src > web.py

```python
from flask_cors import CORS
from flask import Flask, request, send_from_directory
from werkzeug.utils import secure_filename
from astar import bacaTextFile, listOfNode, coordinates, matrixJarak, tetangga, closestPath
import os

server = Flask(__name__)
CORS(server)


@server.route("/")
def index():
    return send_from_directory(os.path.join("static", "html"), "index.html")


@server.route("/read-file", methods=["POST"])
def read_file():
    f = request.files["file"]
    f.save(os.path.join("uploads", secure_filename(f.filename)))

    arr = bacaTextFile(secure_filename(f.filename))

    nodeArr = listOfNode(arr)
    neighArr = tetangga(arr, nodeArr)
    coordArr = coordinates(arr)
    distMatr = matrixJarak(arr, coordArr)

    nodeArrDict = []
    for index, node in enumerate(nodeArr):
```

src > web.py

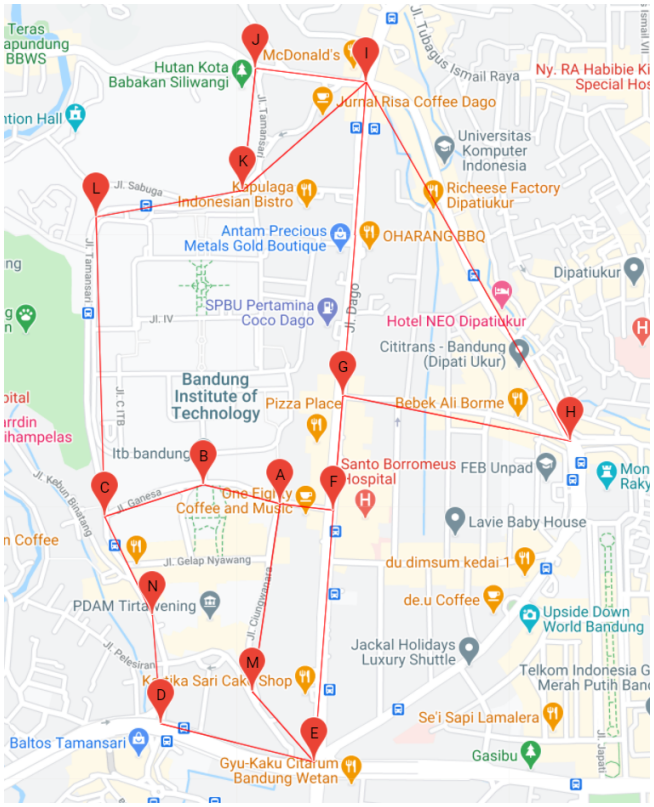
```python
        for index, node in enumerate(nodeArr):
            nodeName = node[0]
            nodeDict = {
                "name": nodeName,
                "title": nodeName,
                "position": {
                    "lng": float(coordArr[index][1]),
                    "lat": float(coordArr[index][0]),
                }
            }
            nodeArrDict.append(nodeDict)

    neighArrDict = []
    for index, neighbor in enumerate(neighArr):
        neighDict = {
            "rel": list(neighbor),
            "distance": float(distMatr[nodeArr.index(neighbor[0])][nodeArr.index(neighbor[1])])
        }
        neighArrDict.append(neighDict)

    return {"nodes": nodeArrDict, "neighbors": neighArrDict}


```

```python
@server.route("/get-path", methods=["POST"])
def get_path():
    req = request.json
    arr = bacaTextFile(os.path.join("uploads", req["filename"]))
    nodeArr = listOfNode(arr)
    neighArr = tetangga(arr, nodeArr)
    coordArr = coordinates(arr)
    distMatr = matrixJarak(arr, coordArr)
    path = closestPath(req["source"], req["destination"],       Find related code in Tucil3-Stima
                       neighArr, distMatr, nodeArr)

    retval = []
    for p in path:
        retval.append({
            "node": p[0],
            "cost": p[1]
        })

    return {"paths": retval}

```

BAB II
Peta/Graf Input

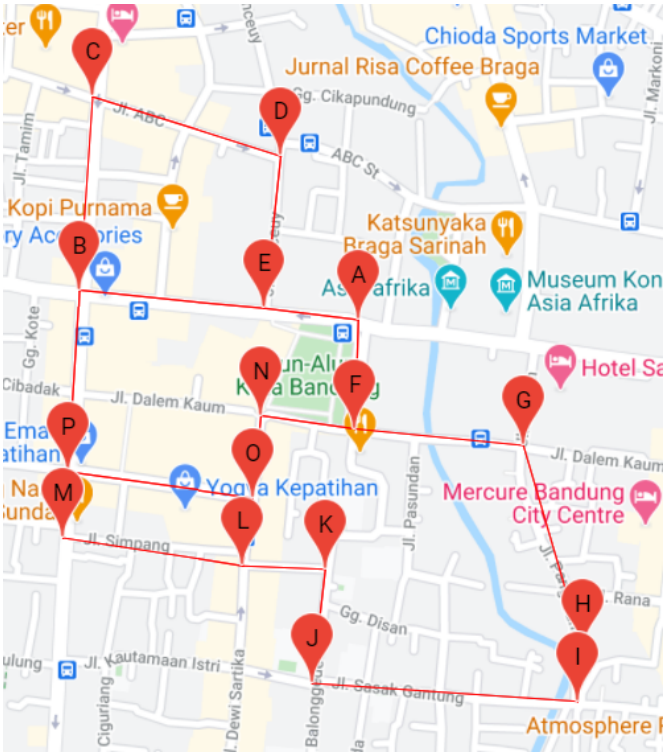
Tucil3-Stima > test > itbdago.txt

```
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
```



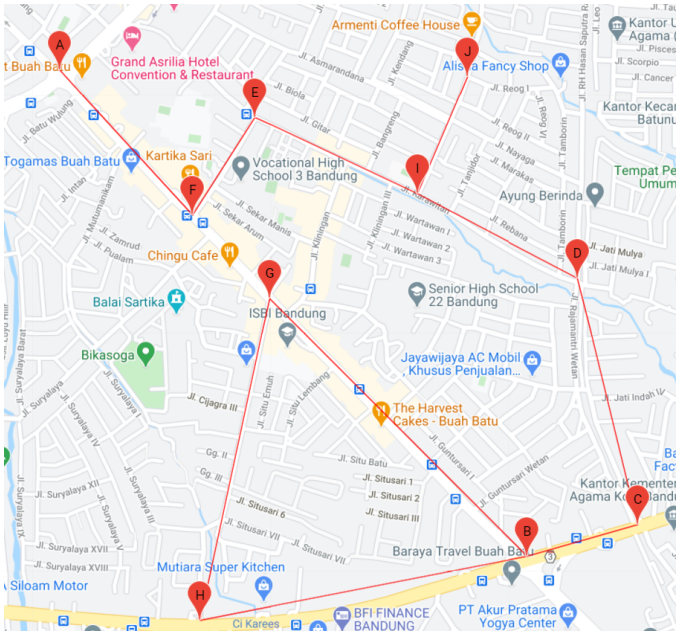
Gambar 2.1 File Input ITB/Dago

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



Gambar 2.2 File Input Alun-Alun Bandung

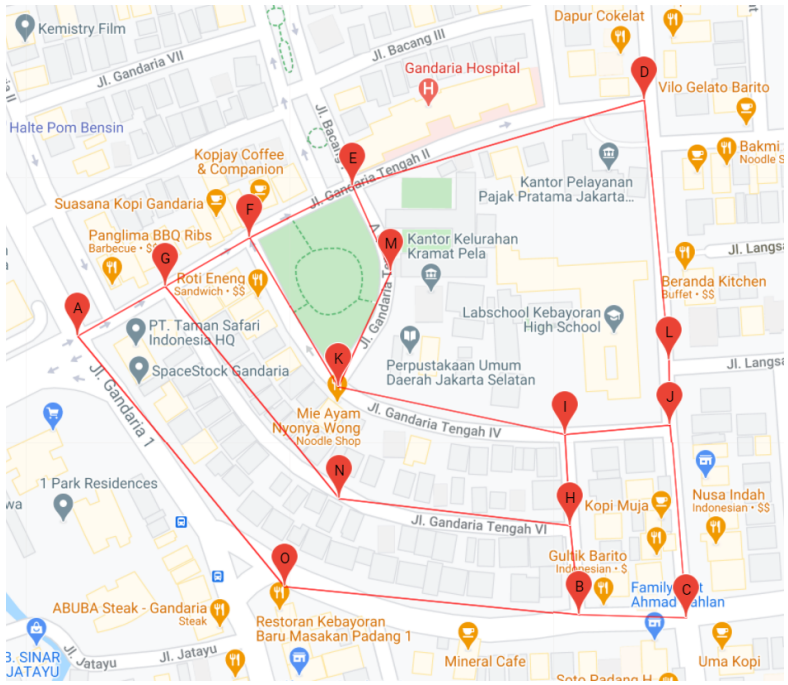
```
C: > Users > Haikal > Downloads > Tucil3-S
10
A (-6.936757,107.622691)
B (-6.947901,107.633486)
C (-6.947165,107.636042)
D (-6.941532,107.634648)
E (-6.93784,107.6272)
F (-6.94008,107.62576)
G (-6.94199,107.62754)
H (-6.94937,107.62593)
I (-6.9396,107.63096)
J (-6.93691,107.63211)
0 0 0 0 0 1 0 0 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
```



Gambar 2.3 File Input Buahbatu

C: > Users > Haikal > Downloads > Tucil3-Stima >

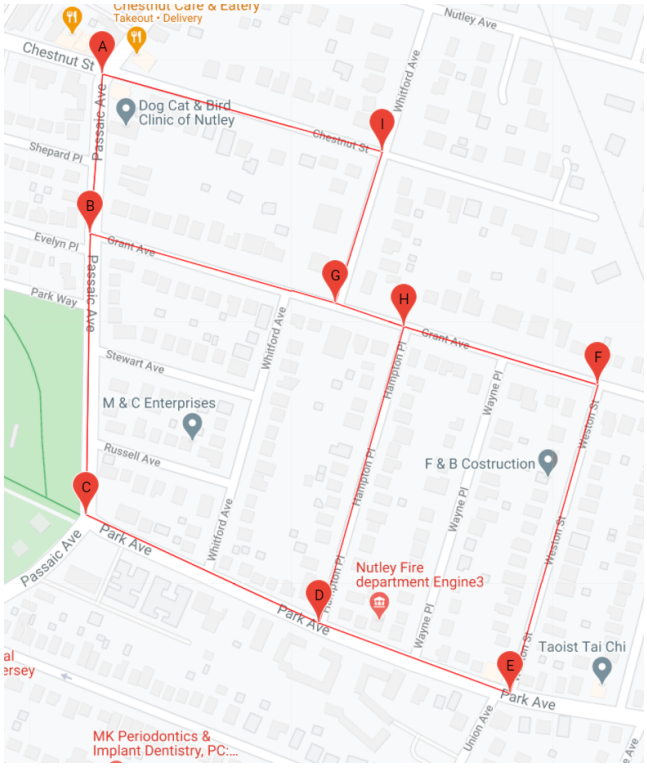
```
15
A (-6.245131,106.788753)
B (-6.246568,106.791361)
C (-6.246584,106.791921)
D (-6.243914,106.791703)
E (-6.244357,106.790184)
F (-6.244625,106.789645)
G (-6.244878,106.789209)
H (-6.246105,106.791314)
I (-6.245636,106.791288)
J (-6.245589,106.791838)
K (-6.245394,106.790106)
L (-6.245254,106.791829)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



Gambar 2.4 File Input Jakarta

test > newyork.txt

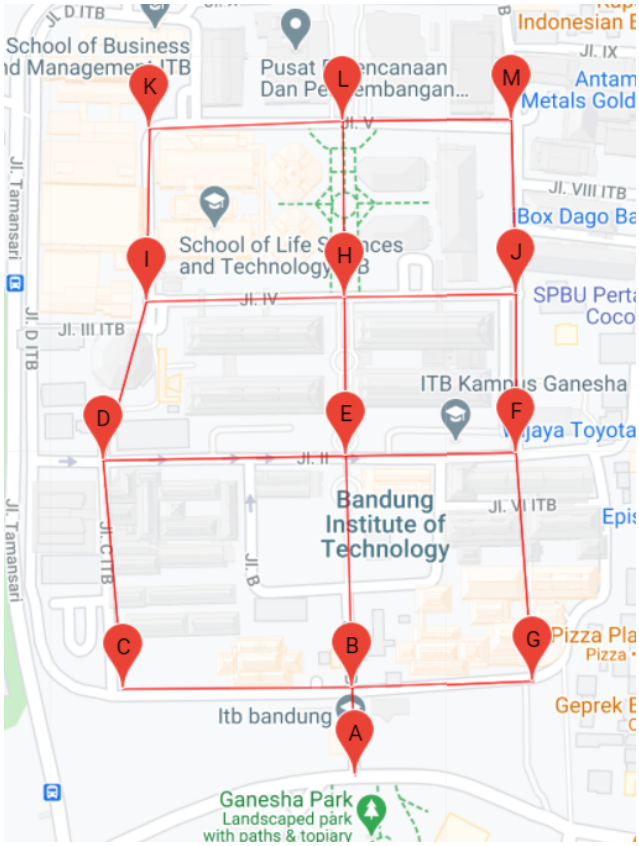
```
9
A (40.817856,-74.154211)
B (40.816586,-74.154335)
C (40.814353,-74.154383)
D (40.813489,-74.151930)
E (40.812932,-74.149927)
F (40.815382,-74.148998)
G (40.816032,-74.151758)
H (40.815851,-74.151037)
I (40.817233,-74.151270)
0 1 0 0 0 0 0 0 1
1 0 1 0 0 0 1 0 0
0 1 0 1 0 0 0 0 0
0 0 1 0 1 0 0 1 0
0 0 0 1 0 1 0 0 0
0 0 0 0 1 0 0 1 0
0 1 0 0 0 0 0 1 1
0 0 0 1 0 1 1 0 0
1 0 0 0 0 0 1 0 0
```



Gambar 2.5 File Input New York

test > itb.txt

```
13
A (-6.893220,107.610454)
B (-6.892613,107.610428)
C (-6.892625,107.608842)
D (-6.891049,107.608705)
E (-6.891012,107.610386)
F (-6.890995,107.611570)
G (-6.892568,107.611680)
H (-6.889913,107.610374)
I (-6.889943,107.609006)
J (-6.889898,107.611563)
K (-6.888757,107.609019)
L (-6.888697,107.610370)
M (-6.888690,107.611537)
0 1 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 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
0 1 0 1 0 1 0 1 0 0 0 0 0
0 0 0 0 1 0 1 0 0 1 0 0 0
0 1 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 1 1 0 1 0
0 0 0 1 0 0 0 1 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 1 0 0 1 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 1 0
```

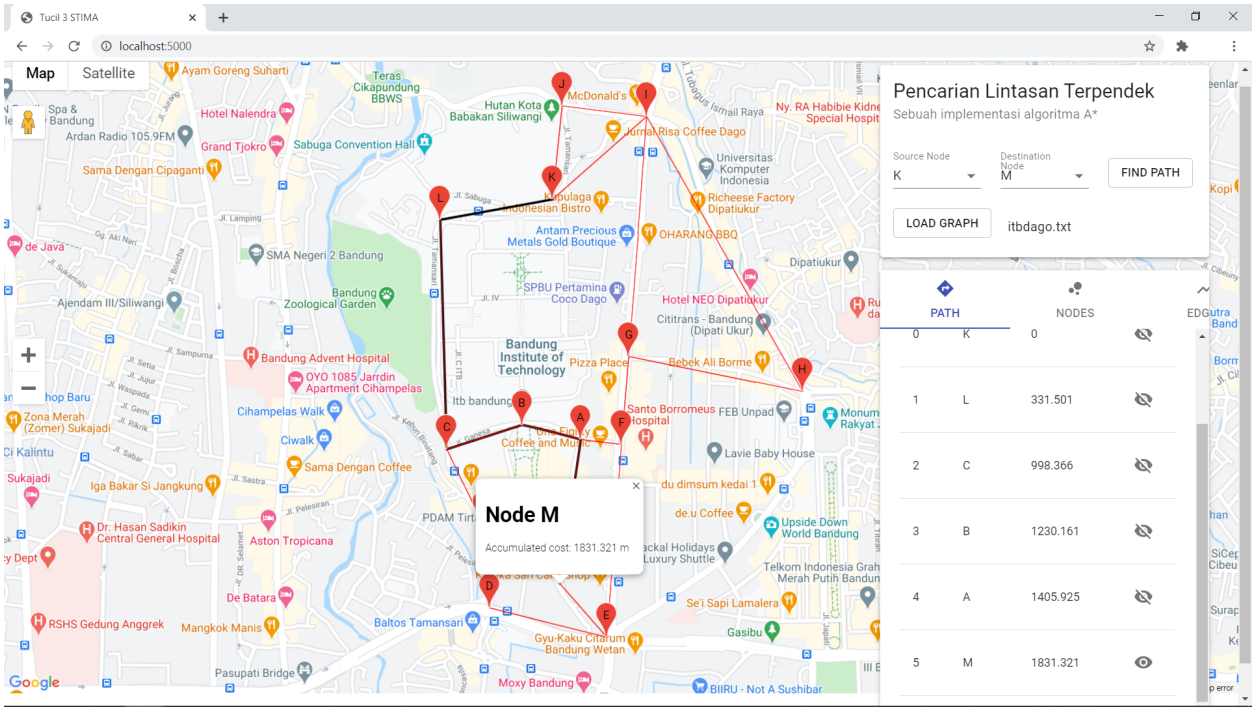


Gambar 2.6 File Input ITB

BAB III
Screenshot Program

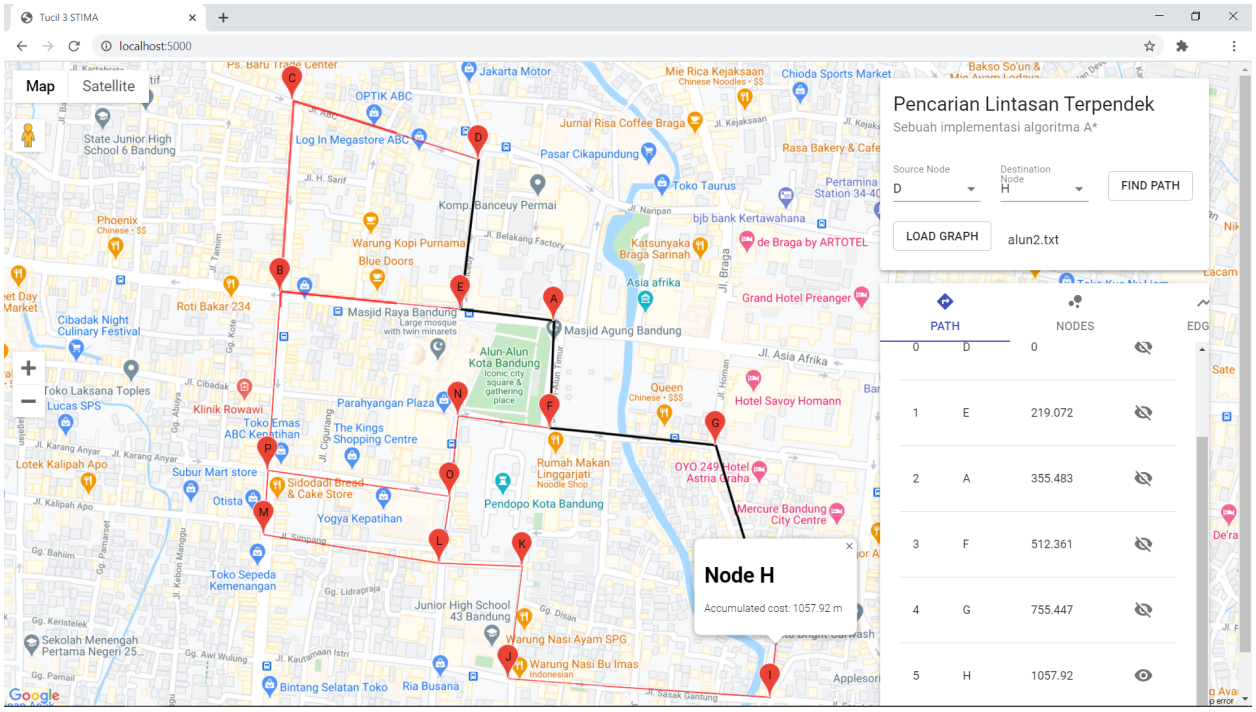
itbdago.txt

Source Node : K, Destination Node : M



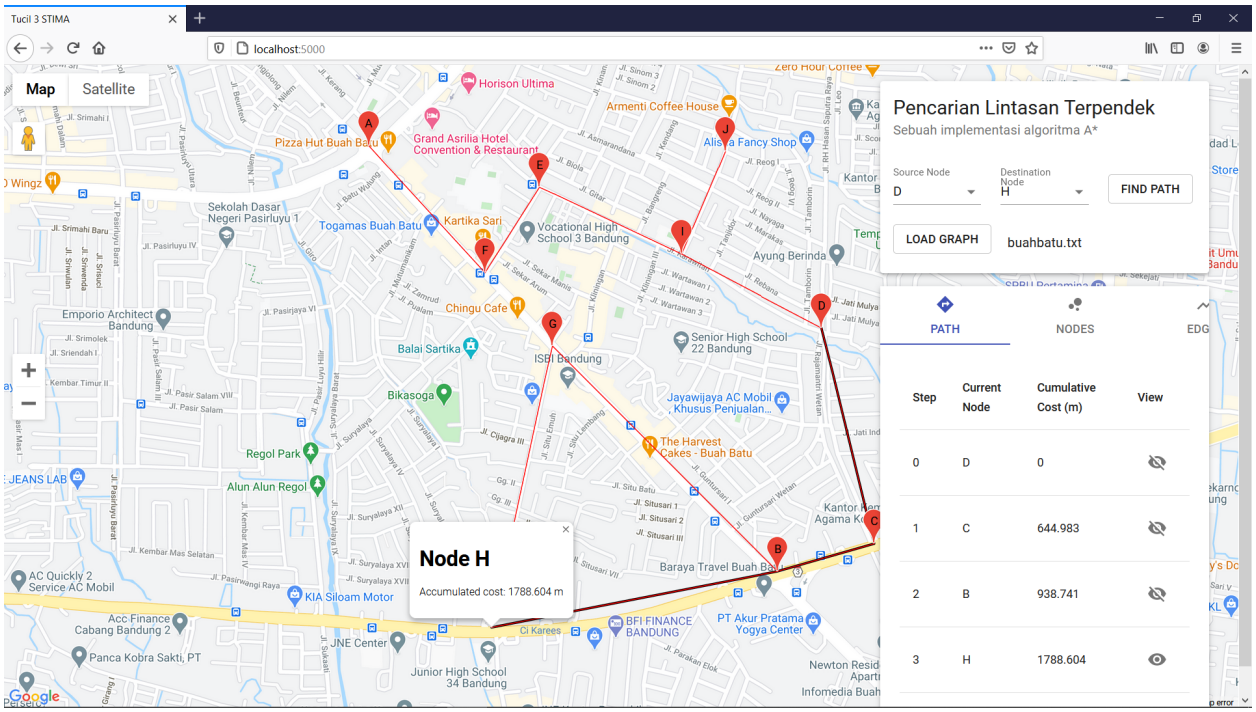
alun2.txt

Source Node : D, Destination Node : H



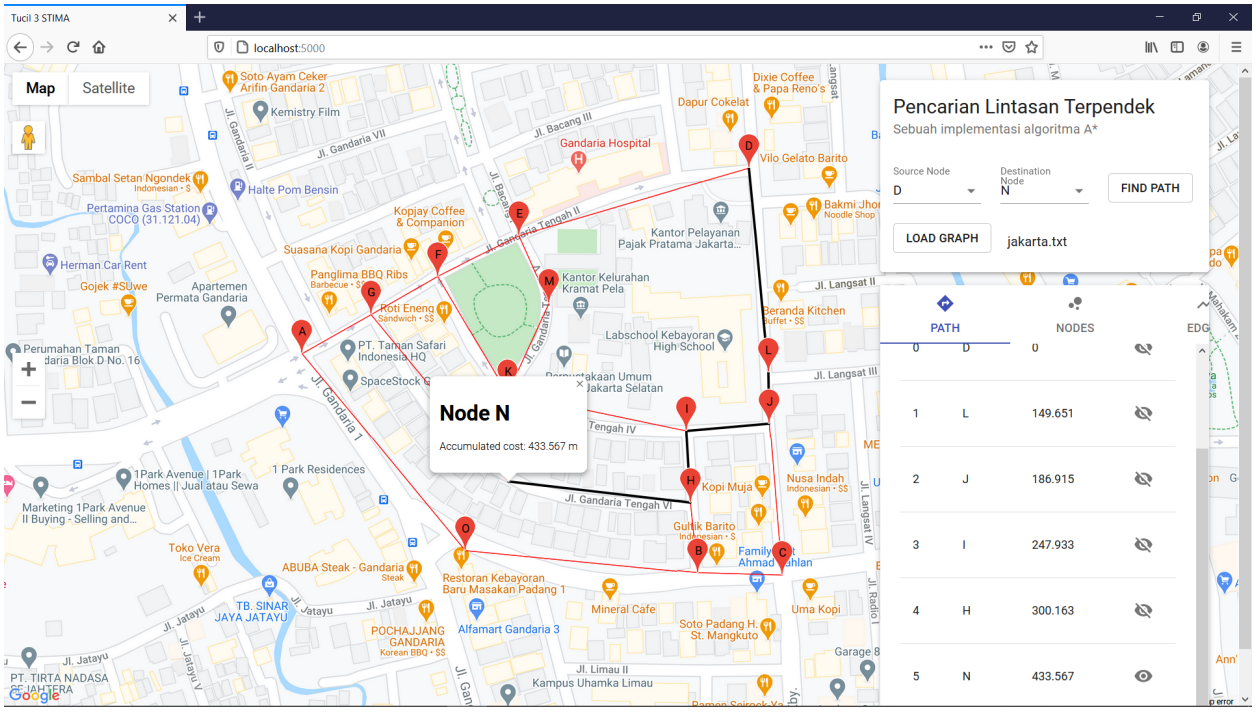
buahbatu.txt

Source Node : D, Destination Node : H



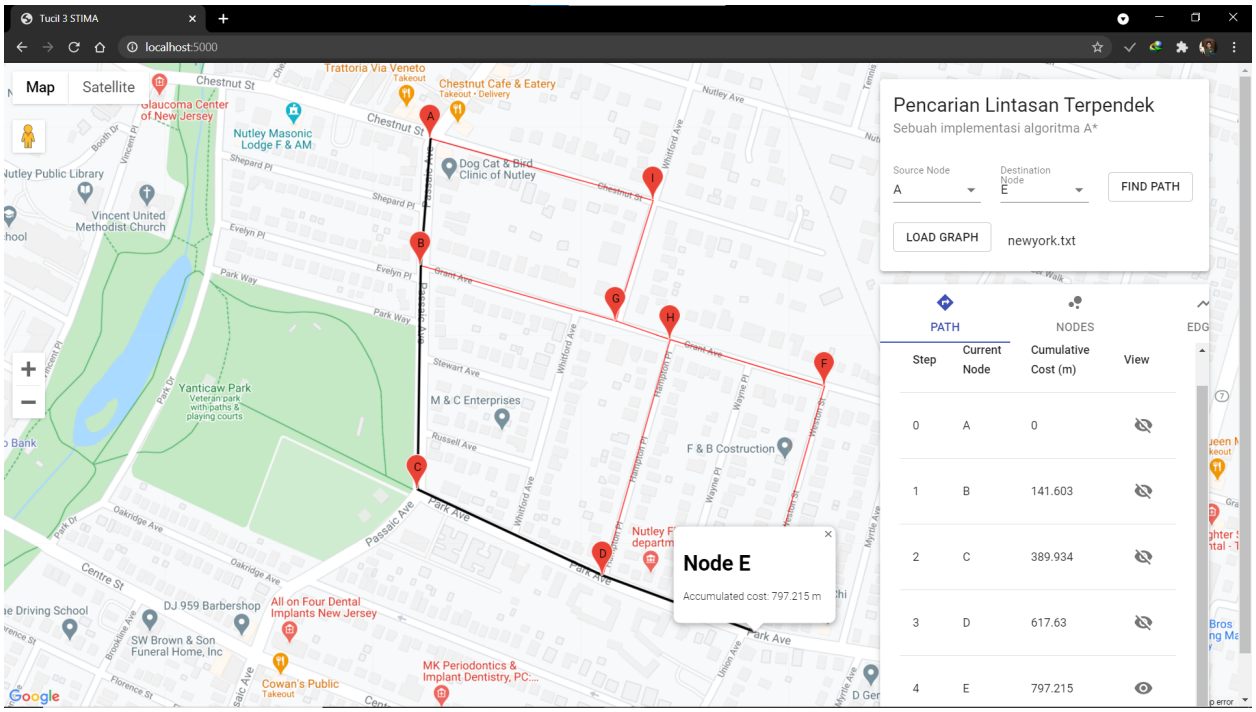
jakarta.txt

Source Node : D, Destination Node : N



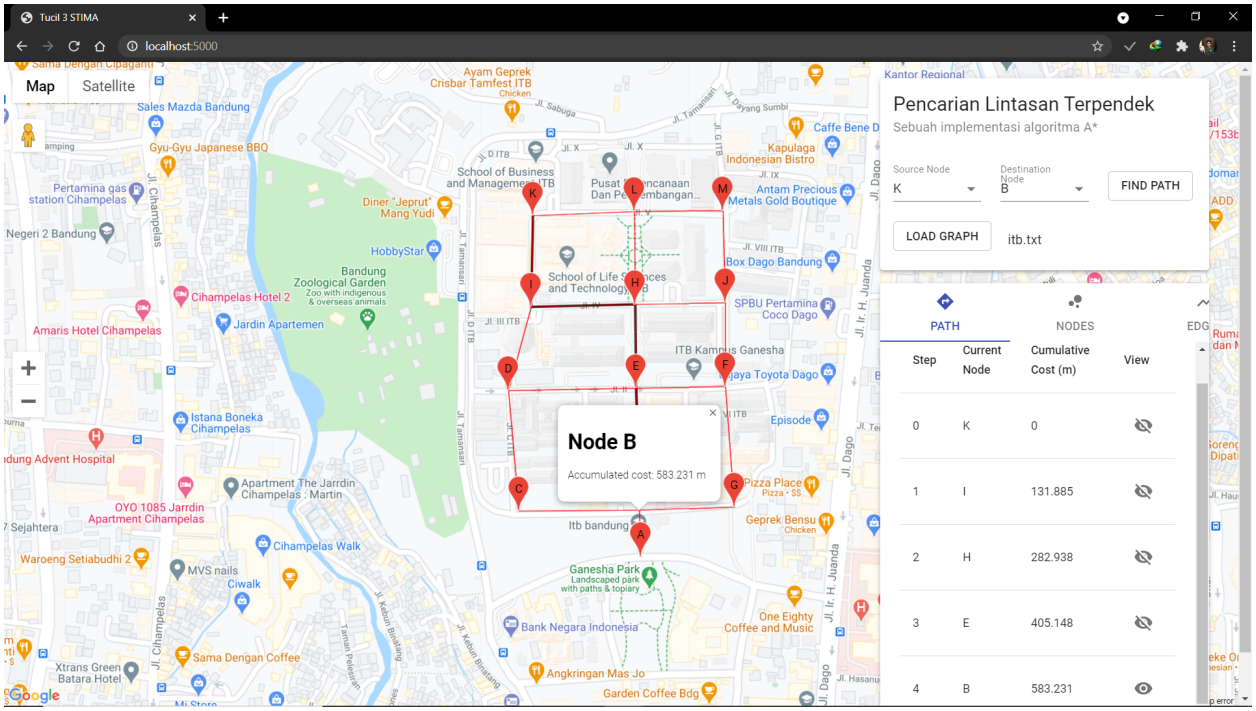
newyork.txt

Source Node : A, Destination Node : E

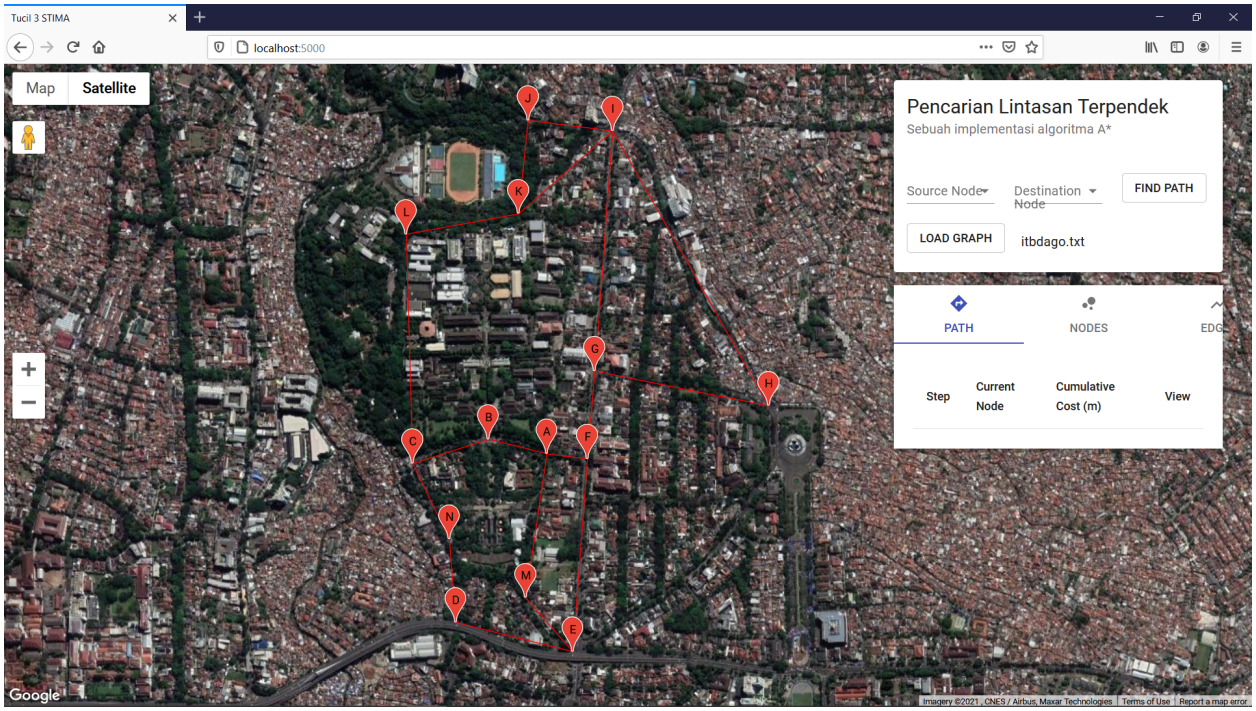


itb.txt

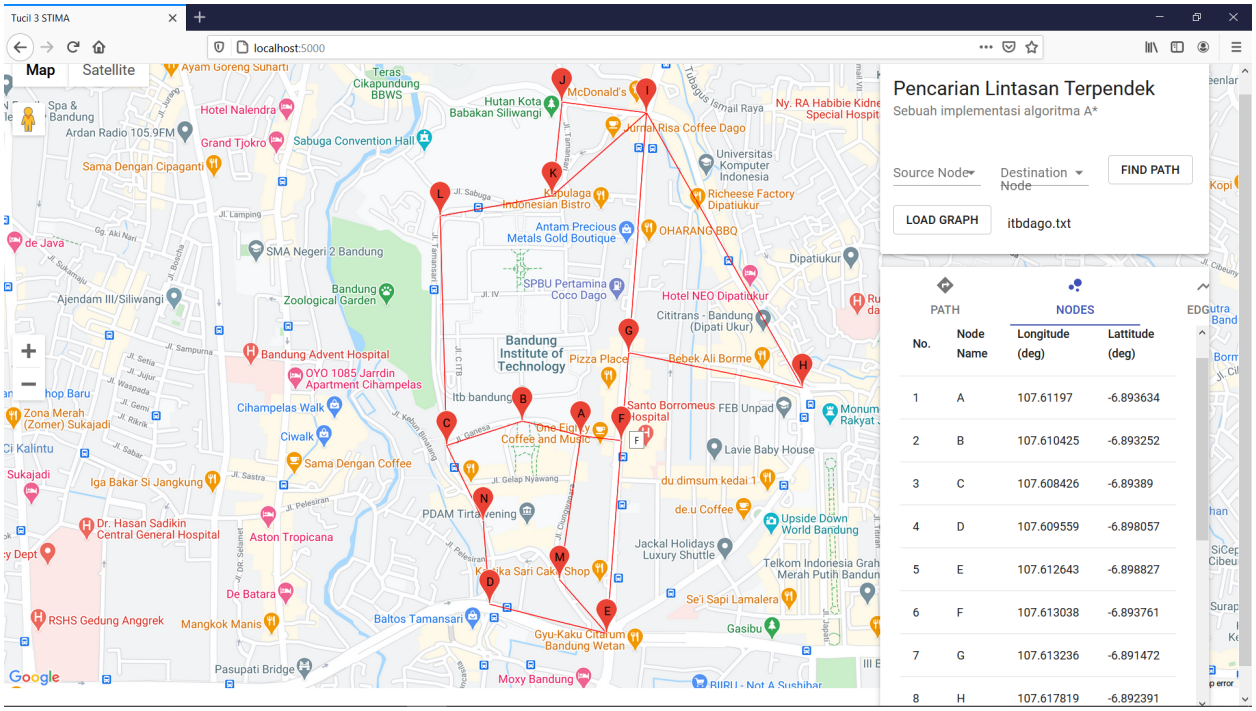
Source Node : K, Destination Node : B



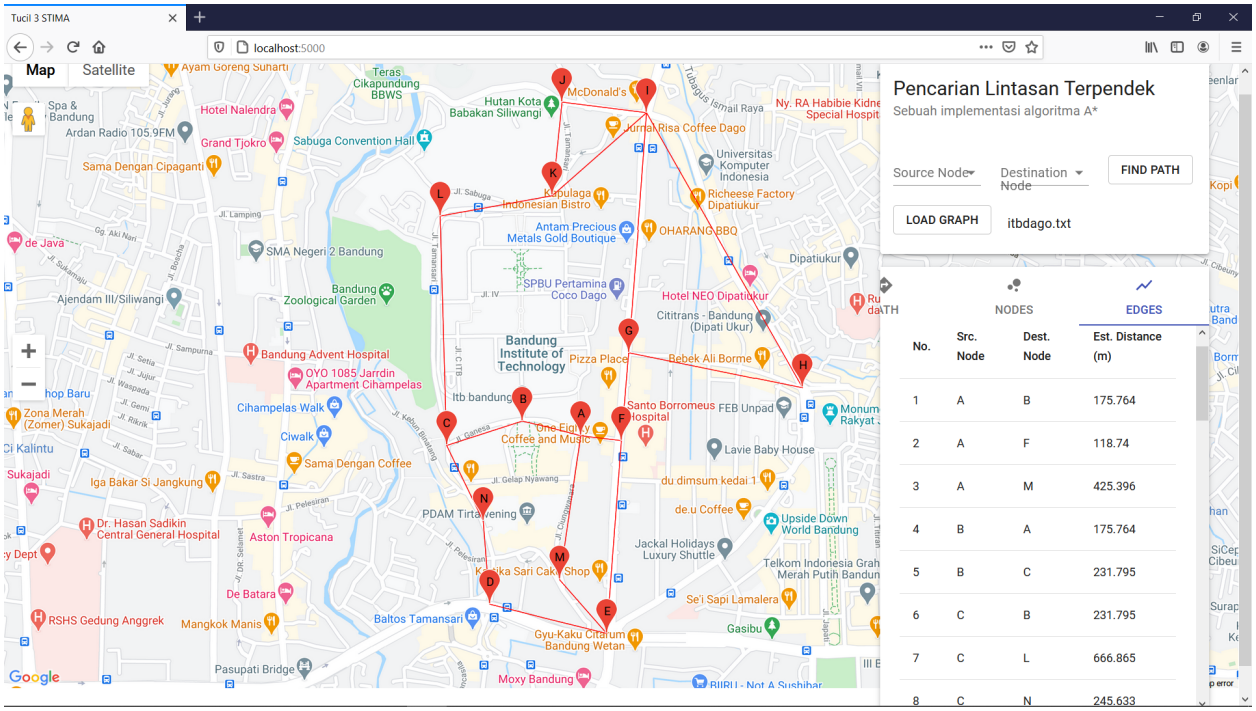
mode Satellite



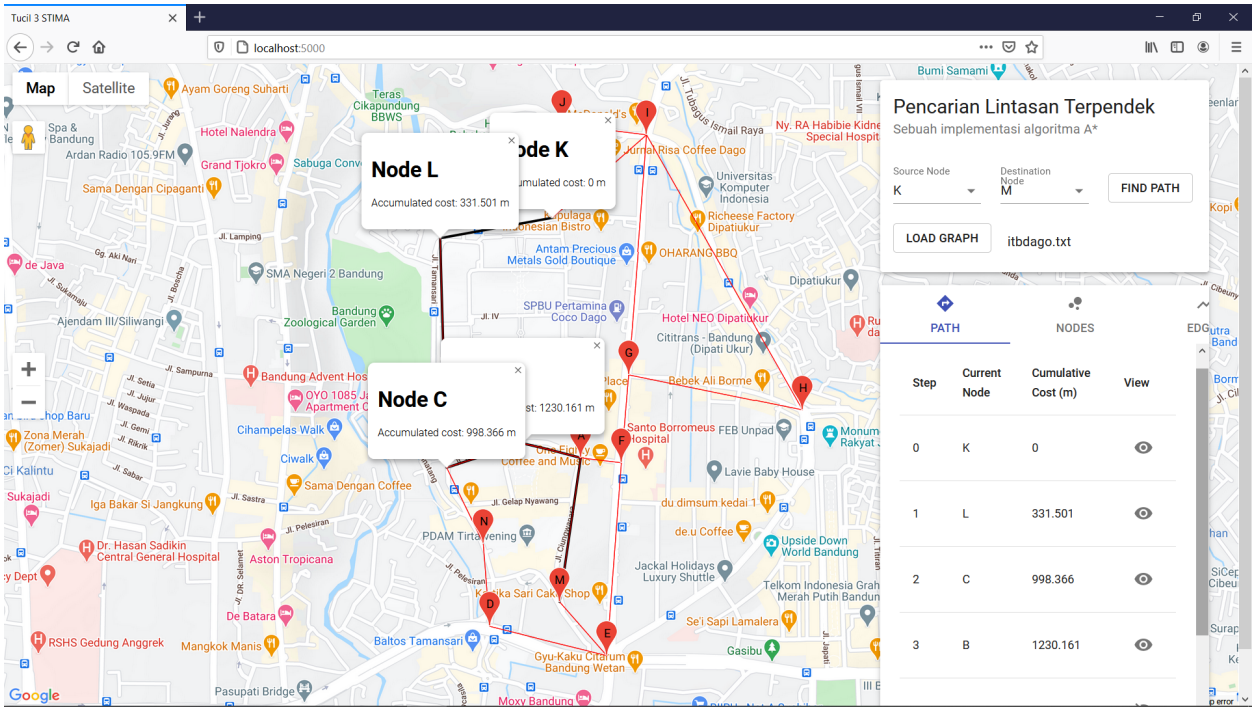
Fitur Nodes



Fitur Edges



Fitur view pada path, apabila view diubah menjadi seen akan memberikan informasi accumulated distance pada setiap node yang view nya telah diubah menjadi seen.



BAB IV
Link

Github : https://github.com/haikallf/Tucil3Stima

BAB V
Tabel Penilaian

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	√