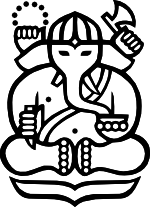
Laporan Tugas Kecil 3 IF2211 Strategi Algoritma

Penyelesaian Persoalan *Travelling Salesperson Problem* dengan Algoritma *Branch and Bound*



Oleh:

Kanisius Kenneth Halim

13515008

K-02

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2017

# DESKRIPSI PERSOALAN

Buatlah program untuk menyelesaikan persoalan *Travelling Salesperson Problem* (TSP) dengan menggunakan Algoritma *Branch and Bound.* Nilai *bound* dihitung dengan *reduced cost matrix* dan dengan Bobot Tur Lengkap. Untuk penyelesaian dengan memanfaatkan *reduced cost matrix*, masukan berupa graf berarah. Sedangkan untuk penyelesaian dengan memanfaatkan Bobot Tur Lengkap, masukan berupa graf tidak berarah.

# KODE PROGRAM

Program dibuat menggunakan bahasa Python 2.7 dengan lingkungan sistem operasi UNIX-like. Program dibuat dengan cukup modular, dan kedua algoritma yang digunakan untuk menyelesaikan persoalan TSP diatas dapat di akses dari Main program.

Program mengambil input graf dari file teks yang berada pada subfolder “asset” yang harus berada pada folder yang sama dengan lokasi program dijalankan. File text yang digunakan sebagai input harus memiliki nama file “rcm\_XX.txt” untuk input graf pada algoritma *reduced cost matrix* dan “blt\_XX.txt” untuk input graf pada algoritma bobot tur lengkap.

Program akan mengeluarkan graf dalam berupa gambar pada subfolder “out” sesuai dengan nama file input dengan ekstensi file diganti dengan file PNG (Portable Network Graphic).

Library yang digunakan untuk melakukan rendering graf menjadi file PNG adalah library graph-tool *(*[*https://graph-tool.skewed.de)*](https://graph-tool.skewed.de))untuk bahasa Python, library harus terinstall dan dapat digunakan untuk Python 2.7 agar program dapat digunakan, selain itu juga digunakan library PIL(Python Imaging Library) untuk mengakses gambar dari dalam lingkungan interpreter Python dan ditampilkan ketika rendering graf telah berhasil.

## File main.py

File main.py adalah file yang berinteraksi dengan user, file main menerima input dari user untuk memilih persoalan mana yang akan diselesaikan oleh program, berikut adalah *source code* dari file main.py:

**from** **\_\_future\_\_** **import** print\_function

**try**:

**from** **PIL** **import** Image

**except**:

**print**("PIL not found, Please install with 'pip install pillowcase'")

quit()

**import** **rcm**

**import** **copy**

**import** **util**

**import** **btl**

**import** **time**

**import** **draw**

INF = float("inf")

**def** printPath(l):

**print**(l[0],end="")

**for** i **in** range (1,len(l)):

**print**("->",l[i],end="",sep="")

**print**("->1")

**print**()

*#Fungsi Main*

**def** Main():

*#Input*

**print**("1. Reduced Cost Matrix")

**print**("2. Bobot Tur Lengkap")

choice = input(">")

**if**(choice == 1):

pref = "rcm"

di = True

**elif**(choice == 2):

pref = "blt"

di = False

**else**:

exit()

tc = raw\_input("Testcase: ")

fname = pref +"\_" +tc

start = time.time()

adj = []

*#Load File*

util.readFile(fname,adj)

**print**("Input Graph:")

util.PrintMatrix(adj)

*#Proses*

**if**(choice == 1):

ret = rcm.startRCM(adj)

cost = ret[0]

count = ret[2]

**else**:

ret = btl.startBNB(adj)

cost = ret[0]

count = ret[2]

*#Output*

**print**("Cost: ", cost)

**print**("Path: ",end ="")

printPath(ret[1])

**print**("Simpul Hidup",count);

img = draw.draw\_graph(adj,ret[1],di,fname)

ima = Image.open(img)

ima.show()

stop = time.time()

**print**("Execution Time: ",stop-start,"s",sep="")

*#Program Utama untuk memanggil fungsi main*

Main()

## File rcm.py

File rcm.py adalah file yang berisi tentang segala fungsi yang dibutuhkan untuk melakukan penyelesaian permasalahan dengan algoritma *reduced cost matrix*, berikut adalah *source code* untuk file ini:

**from** **\_\_future\_\_** **import** print\_function

**import** **copy**

**try**:

**import** **Queue**

**except**:

**print**("Please use Python2")

quit()

**import** **util**

INF = float("inf")

*#Melakukan Reduksi Matrix terhadap baris*

**def** reduce\_by\_row(adj):

n = len(adj)

lb = 0

**for** i **in** range(0,n):

min = INF

**for** j **in** range(0,n):

**if** (adj[i][j] < min):

min = adj[i][j];

**for** j **in** range(0,n):

**if**(adj[i][j] != INF):

adj[i][j] = adj[i][j]-min

**if**(min != INF):

lb+=min

**return** lb

*#Melakukan reduksi Matrix terhadap kolom*

**def** reduce\_by\_col(adj):

n = len(adj)

lb =0

**for** i **in** range(0,n):

min = INF

**for** j **in** range(0,n):

**if**(adj[j][i] < min):

min = adj[j][i]

**for** j **in** range(0,n):

**if**(adj[j][i] != INF):

adj[j][i] = adj[j][i]-min

**if**(min != INF):

lb+=min

**return** lb

*#Melakukan Reduksi Matrix total*

**def** reduce\_all(adj):

lb =0;

lb += reduce\_by\_row(adj)

lb += reduce\_by\_col(adj)

**return** lb

**def** select(x,i,j):

adj = copy.deepcopy(x)

n = len(adj)

**for** a **in** range(0,n):

adj[i][a] = INF

adj[a][j] = INF

adj[j][0] = INF

adj[j][i] = INF

**return** adj

**def** genNode(root):

lb = root[0]

bname = root[1]

i = bname[len(bname)-1]

adj = root[2]

ret = []

**for** j **in** range(0,len(adj)):

**if**(adj[i-1][j] != INF):

c = (adj[i-1][j])

temp = select(adj,i-1,j)

s = reduce\_all(temp)

ret.append((lb+s+c,bname + [j+1],temp))

**return** ret

**def** startRCM(root):

n = len(root)

q = Queue.PriorityQueue()

count = 1

adj = copy.deepcopy(root)

lb = reduce\_all(adj)

bname = []

bname.append(1)

q.put( (lb,bname,adj))

f = True

ret = 0

cost = 0

**while**((**not** q.empty()) **and** f):

temp = q.get()

genNode(temp)

x = genNode(temp)

**if**(**not** x):

f=False

ret = temp[1]

cost=temp[0]

**else**:

**for** y **in** x:

q.put(y)

count+=1

**return** (cost,ret,count)

## File btl.py

File btl.py adalah file yang berisi tentang segala fungsi yang dibutuhkan untuk melakukan penyelesaian permasalahan dengan algoritma bobot tur lengkap, berikut adalah *source code* untuk file ini:

**from** **\_\_future\_\_** **import** print\_function

**import** **copy**

**try**:

**import** **Queue**

**except**:

**print**("Please use Python2")

INF = float("inf")

**def** isInPath(name,path):

ret = -1

**for** i **in** range(0,len(path)):

**if**(name==path[i]):

ret = i

**break**

**return** ret

**def** genCost(adj,path):

cx = 0

**for** i **in** range(0,len(path)):

cx += adj[path[i]-1][path[(i+1)%len(path)]-1]

**return** cx

**def** getCost(adj,bname):

sumx = 0

**for** i **in** range(0,len(adj)):

cx = 0

idx = isInPath(i+1,bname)

x = adj[i][:]

**if**(idx == -1):

x.sort()

cx += x[0] + x[1]

**elif**(idx == 0):

y = bname[1]

min = INF

**for** j **in** range(0,len(adj)):

**if**(adj[i][j] < min **and** j!=y-1):

min = adj[i][j]

cx+= min + adj[i][y-1]

**elif**(idx == len(bname)-1):

y = bname[len(bname)-2]

curr = INF

**for** j **in** range(0,len(adj)):

**if**(adj[i][j] < min **and** j!=y-1):

min = adj[i][j]

cx+= min + adj[i][y-1]

**else**:

y = bname[idx-1]

z = bname[idx+1]

cx+=(adj[i][y-1])+(adj[i][z-1])

sumx+=cx

**return** sumx/2

**def** genNode(root):

bname = root[1]

adj = root[2]

i = bname[len(bname)-1]

ret = []

**for** j **in** range(0,len(adj)):

**if**(adj[i-1][j] != INF **and** isInPath(j+1,bname) == -1 ):

ret.append((getCost(adj,bname+[j+1]),bname + [j+1],adj))

**return** ret

**def** findSol(cost,path,adj,count):

ret = genNode((cost,path,adj))

**if**(len(adj)-len(path)==1):

mini = (INF,[],0)

**for** i **in** ret:

**if**(i[0] < mini[0]):

mini = i

**return** (mini[0],mini[1],count)

**else**:

q = Queue.PriorityQueue()

**for** x **in** ret:

count[0]+=1

q.put((x[0],count[0],x[1],x[2]))

ctemp = (INF,[],0)

costnow = INF

**while**(**not** q.empty()):

temp = q.get()

**if**(temp[0] <= costnow):

ret = findSol(temp[0],temp[2],temp[3],count)

costnow = genCost(adj,ret[1])

**if**(costnow <= ctemp[0]):

ctemp = ret

**return** ctemp

**def** startBNB (root):

count = [1]

ret = findSol(0,[1],root,count)

cost = genCost(root,ret[1])

path = ret[1]

count = count[0]

**return** (cost,path,count)

## File util.py

File util.py berisikan fungsi yang digunakan dalam program seperti untuk membaca file dan menulis matrix, berikut adalah source code untuk file ini:

**from** **\_\_future\_\_** **import** print\_function

INF = float("inf")

*#Mencetak Matrix ke layar*

**def** PrintMatrix(l):

**for** i **in** l:

**for** j **in** i:

**print** (j, end = '**\t**')

**print**()

**def** readFile(fname, adj):

**try**:

infile = open("asset/"+fname+".txt","r")

strin = infile.read()

strin = strin.split('**\n**')

**for** stln **in** strin:

stln = stln.split()

l = []

**for** c **in** stln:

**if**(int(c) == -999):

l.append(INF)

**else**:

l.append(int(c))

adj.append(l)

**except**:

**print**("File Not Found")

exit()

## File draw.py

File ini berisikan fungsi untuk melakukan rendering graf dari matrix ketetanggaan dan solusi dari program, berikut adalah source code untuk file ini

**try**:

**from** **graph\_tool.all** **import** \*

**except**:

**print**("Please Install graph-tool for Python from https://graph-tool.skewed.de")

INF = float("inf")

**def** draw\_graph(adj,path,di,fname):

g = Graph(directed=di)

vprop = g.new\_vertex\_property("string")

eprop = g.new\_edge\_property("string")

vprop\_color = g.new\_vertex\_property("string")

eprop\_color = g.new\_edge\_property("string")

n = len(adj)

l = []

*#Add Vertex*

**for** i **in** range(0,n):

temp = g.add\_vertex()

vprop[temp] = i+1

vprop\_color[temp] = "black"

*#Dummy Vertex untuk Mendapatkan Circular Layout*

root = g.add\_vertex()

vprop[root] = "x"

vprop\_color[root]="none"

*#Add Edge*

**for** i **in** range(0,n):

**if**(**not** di):

**for** j **in** range(i,n):

**if**(adj[i][j] != INF):

temp = g.add\_edge(g.vertex(i),g.vertex(j))

eprop[temp] = adj[i][j]

eprop\_color[temp]= "black"

**else**:

**for** j **in** range(0,n):

**if**(adj[i][j] != INF):

temp = g.add\_edge(g.vertex(i),g.vertex(j))

eprop[temp] = adj[i][j]

eprop\_color[temp]= "black"

*#Dummy Edge ke dummy Vertex agar terbentuk bentuk circular*

**for** i **in** range(0,n):

temp = g.add\_edge(root,g.vertex(i))

eprop\_color[temp] = "none"

*#Mengubah warna path menjadi merah*

**for** i **in** range(0,len(path)):

src = g.vertex(path[i]-1)

i = (i+1)%len(path)

dest = g.vertex(path[i]-1)

eprop\_color[g.edge(src,dest)] = "red"

**try**:

os.mkdir("out")

**except**:

dummy=0

*#Render Graf*

g.vertex\_properties["name"]=vprop

g.edge\_properties["weight"]=eprop

g.vertex\_properties["color"]=vprop\_color

g.edge\_properties["color"]=eprop\_color

x= g.vertex\_properties["color"]

oname="out/"+fname+".png"

graph\_draw(

g,

pos=radial\_tree\_layout(g,n),

vertex\_text=g.vertex\_properties["name"],

vertex\_font\_size=40,

vertex\_color="none",

vertex\_fill\_color=x,

edge\_pen\_width=5.0,

edge\_text=g.edge\_properties["weight"],

edge\_font\_size=12,

edge\_color=g.edge\_properties["color"],

output\_size=(1000,1000),

output = oname,

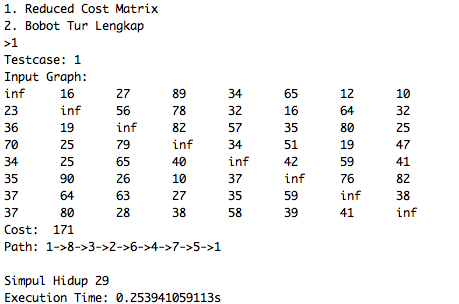
bg\_color=[1,1,1,1]

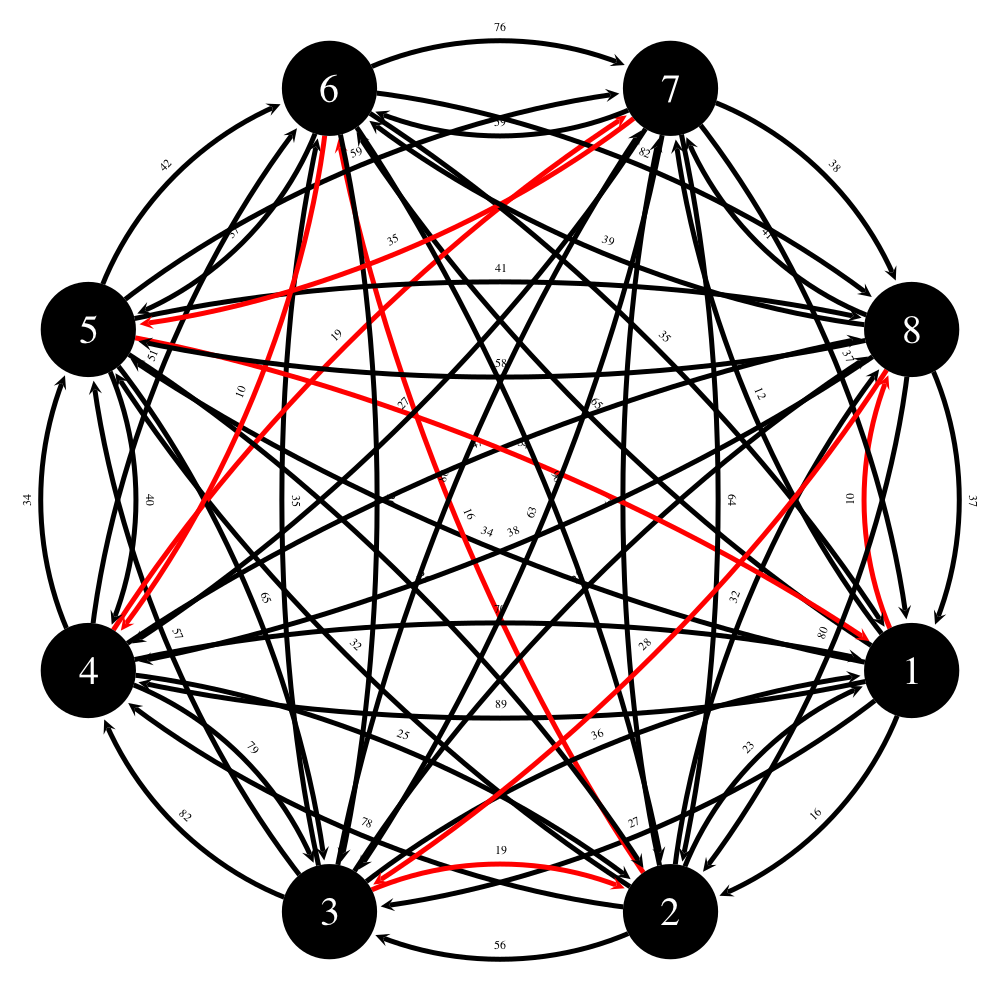
)

**return** oname

# SCREENSHOT PROGRAM

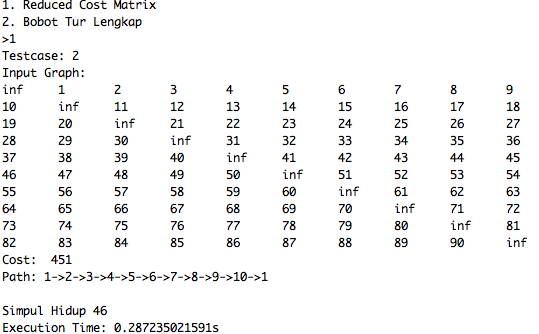
## Screenshot Graf Uji 1 Reduced Cost Matrix

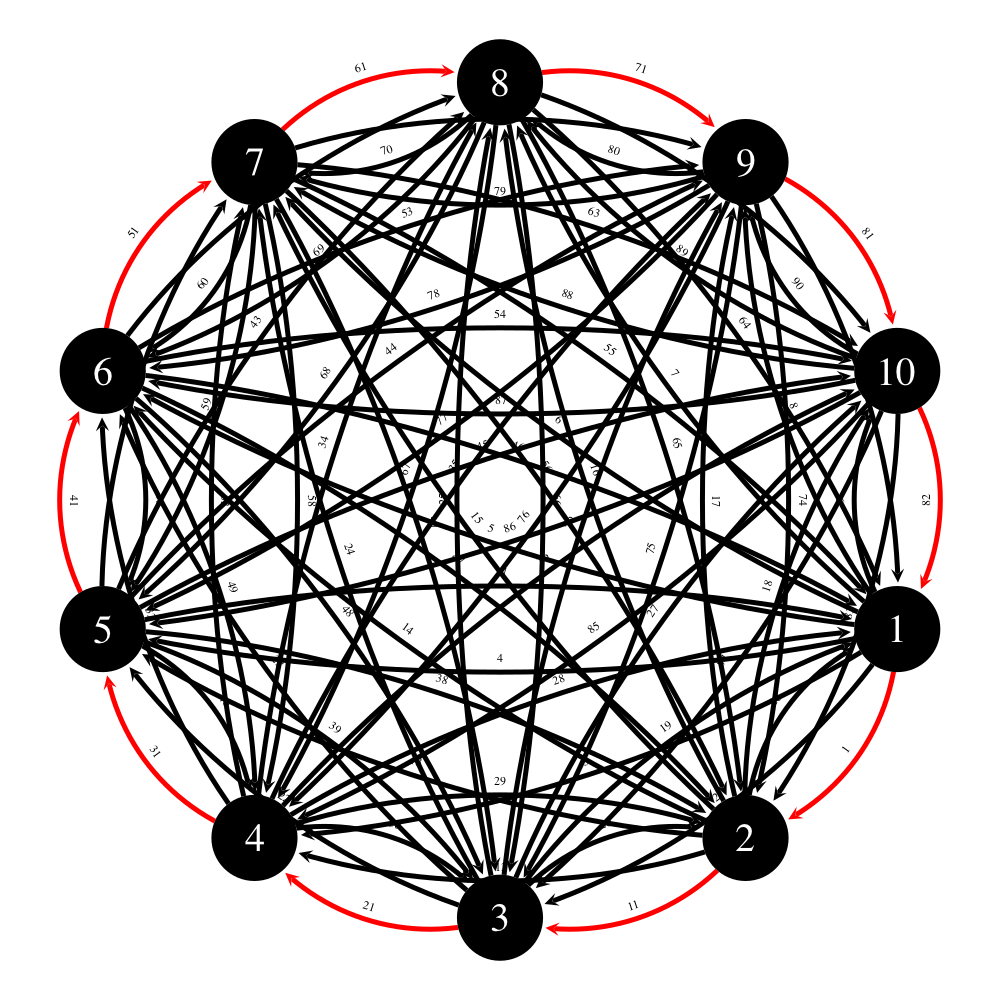




Gambar . out/rcm\_1.png

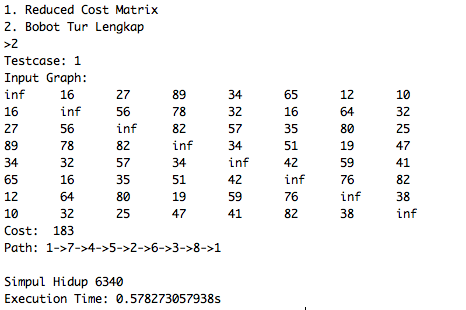
## Screenshot Graf Uji 2 Reduced Cost Matrix

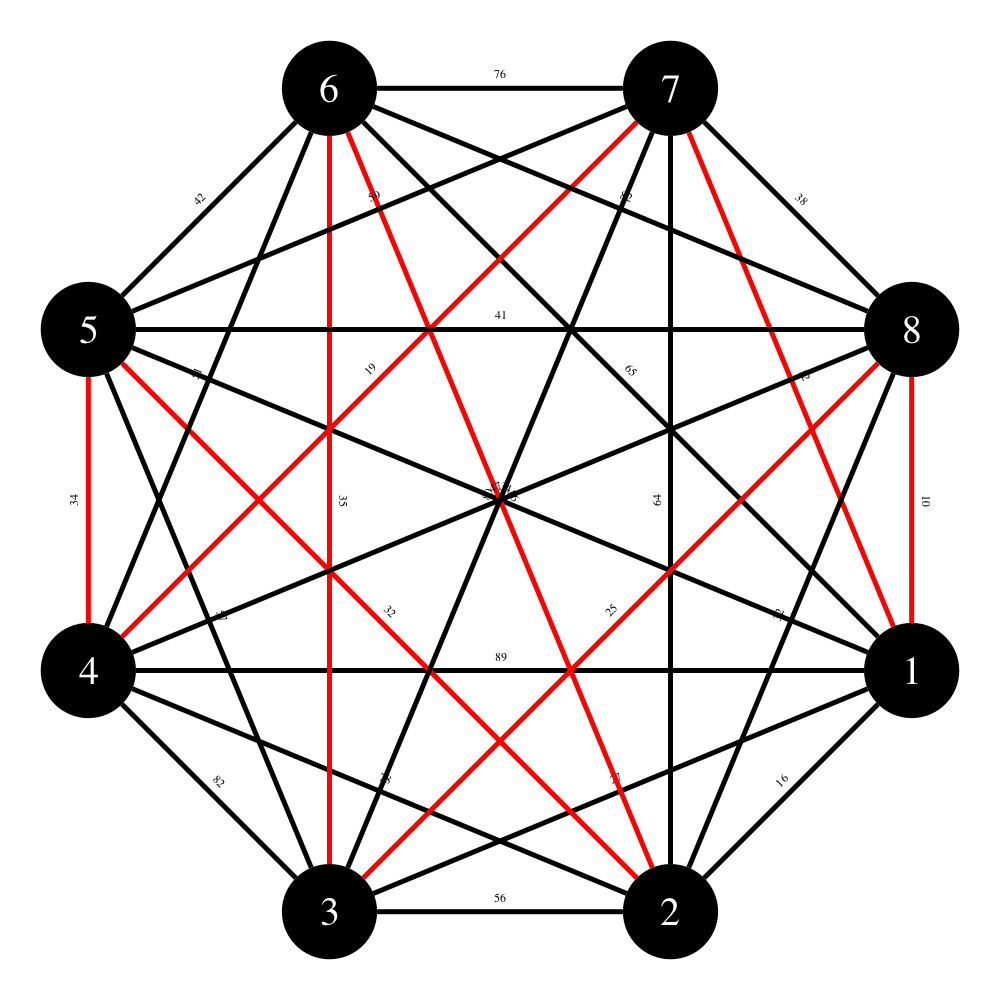




Gambar . out/rcm\_2.png

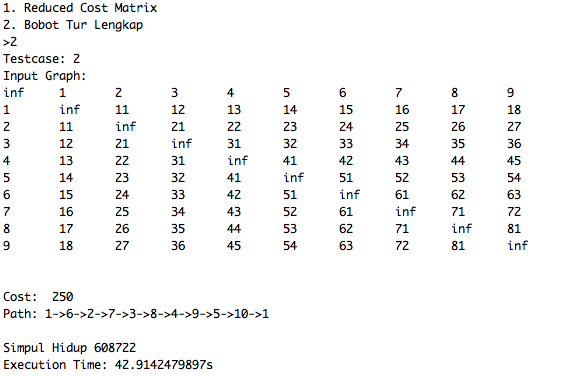
## Screenshot Graf Uji 1 Bobot Tur Lengkap

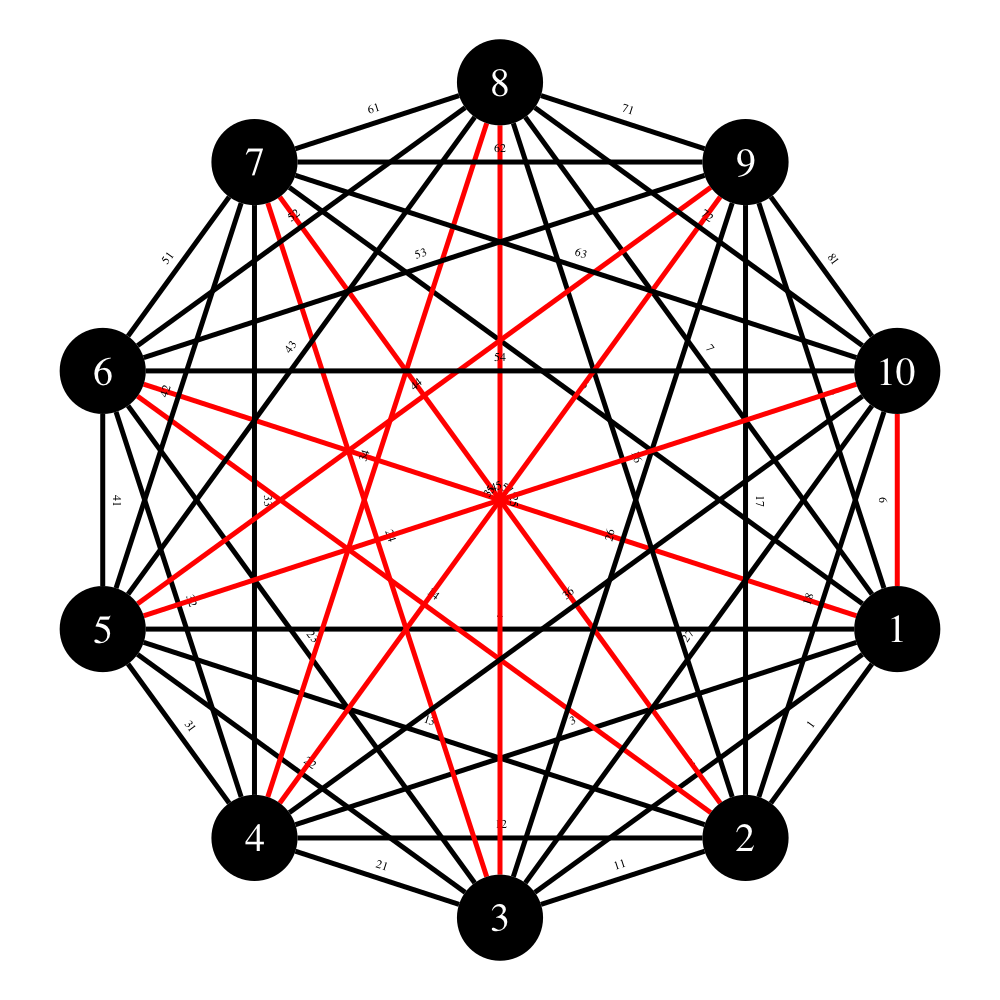




Gambar . out/blt\_1.png

## Screenshot Graf Uji 2 Bobot Tur Lengkap





Gambar . blt\_2.png

# REFERENSI

[*https://graph-tool.skewed.de/static/doc/index.htm*l](https://graph-tool.skewed.de/static/doc/index.html), diakses pada 5 April 2017

[*https://docs.python.org/2/*](https://docs.python.org/2/), diakses pada 5 April 2017

[*http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2016-2017/stima16-17.htm*](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2016-2017/stima16-17.htm), diakses pada 5 April 2017

[*http://www.bogotobogo.com/python/python\_PriorityQueue\_heapq\_Data\_Structure.php*](http://www.bogotobogo.com/python/python_PriorityQueue_heapq_Data_Structure.php), diakses pada 4 April 2017