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

1. 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.

2. 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)* 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.

2.1 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()
```

2.2 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)
```

2.3 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)
```

2.4 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()
```

2.5 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"],
```

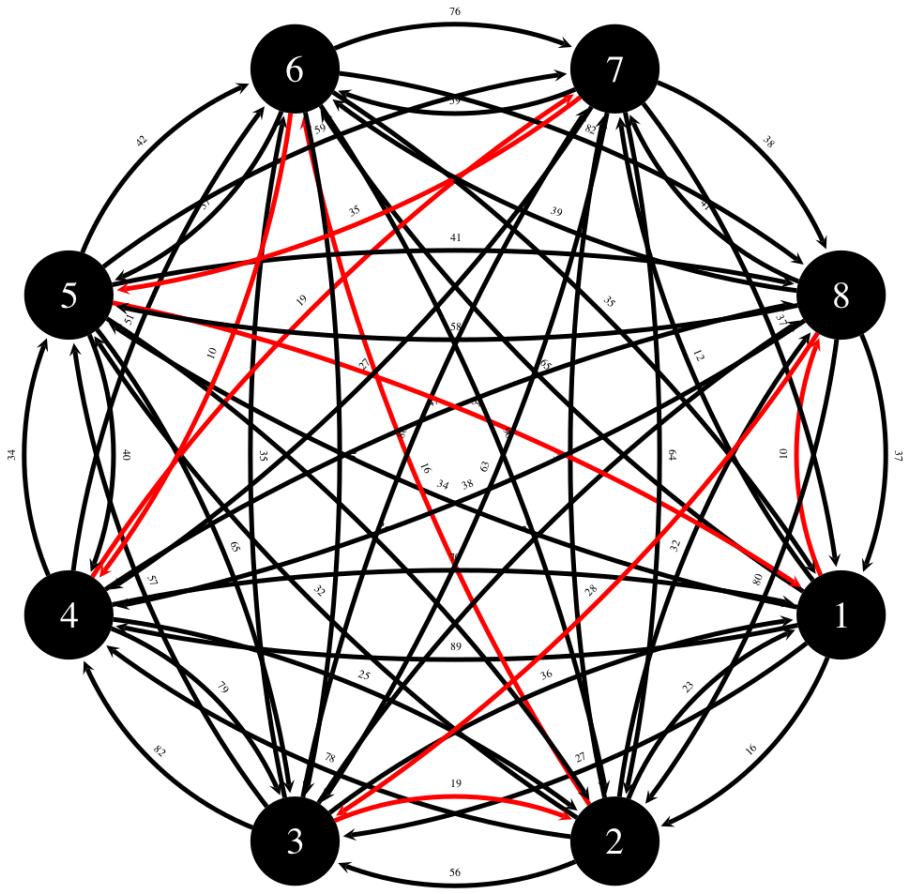
```
    output_size=(1000,1000),
        output = oname,
        bg_color=[1,1,1,1]
        )
        return oname
```

3. SCREENSHOT PROGRAM

3.1 Screenshot Graf Uji 1 Reduced Cost Matrix

```
1. Reduced Cost Matrix
2. Bobot Tur Lengkap
>1
Testcase: 1
Input Graph:
inf     16      27      89      34      65      12      10
23      inf     56      78      32      16      64      32
36      19      inf     82      57      35      80      25
70      25      79      inf     34      51      19      47
34      25      65      40      inf     42      59      41
35      90      26      10      37      inf     76      82
37      64      63      27      35      59      inf     38
37      80      28      38      58      39      41      inf
Cost:  171
Path: 1->8->3->2->6->4->7->5->1

Simpul Hidup 29
Execution Time: 0.253941059113s
```

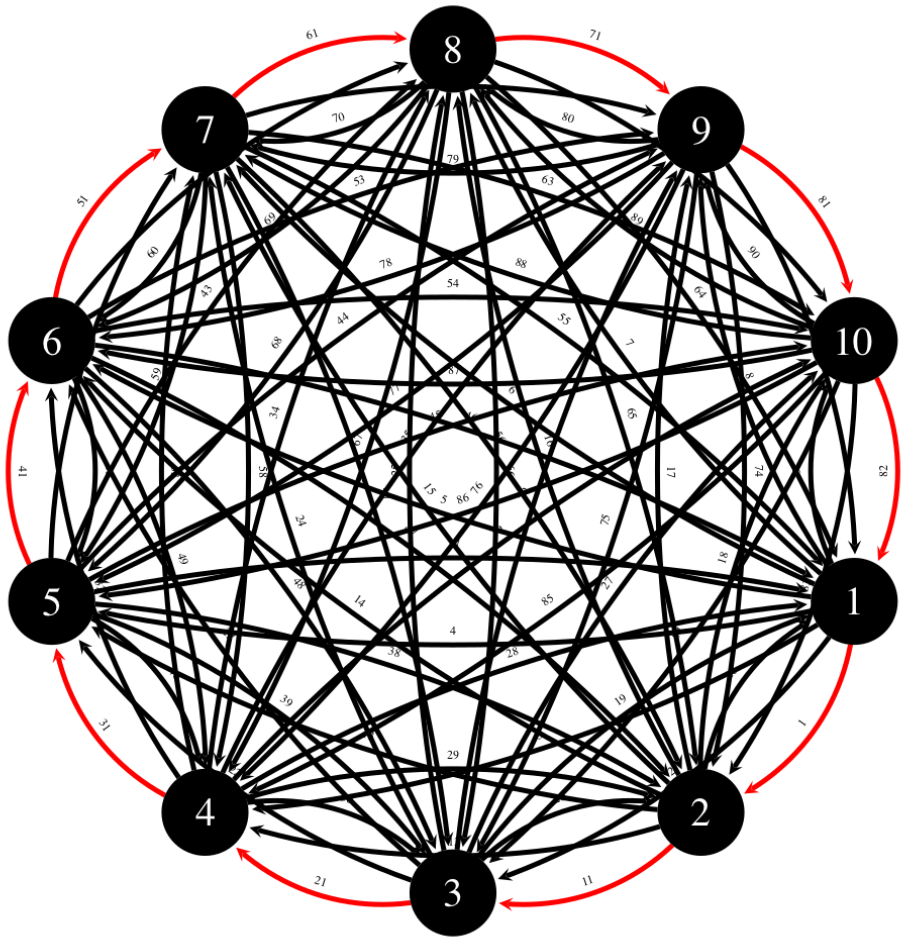


Gambar 1. out/rcm_1.png

3.2 Screenshot Graf Uji 2 Reduced Cost Matrix

```
1. Reduced Cost Matrix
2. Bobot Tur Lengkap
>1
Testcase: 2
Input Graph:
inf     1       2       3       4       5       6       7       8       9
10      inf     11      12      13      14      15      16      17      18
19      20      inf     21      22      23      24      25      26      27
28      29      30      inf     31      32      33      34      35      36
37      38      39      40      inf     41      42      43      44      45
46      47      48      49      50      inf     51      52      53      54
55      56      57      58      59      60      inf     61      62      63
64      65      66      67      68      69      70      inf     71      72
73      74      75      76      77      78      79      80      inf     81
82      83      84      85      86      87      88      89      90      inf
Cost:  451
Path: 1->2->3->4->5->6->7->8->9->10->1

Simpul Hidup 46
Execution Time: 0.287235021591s
```

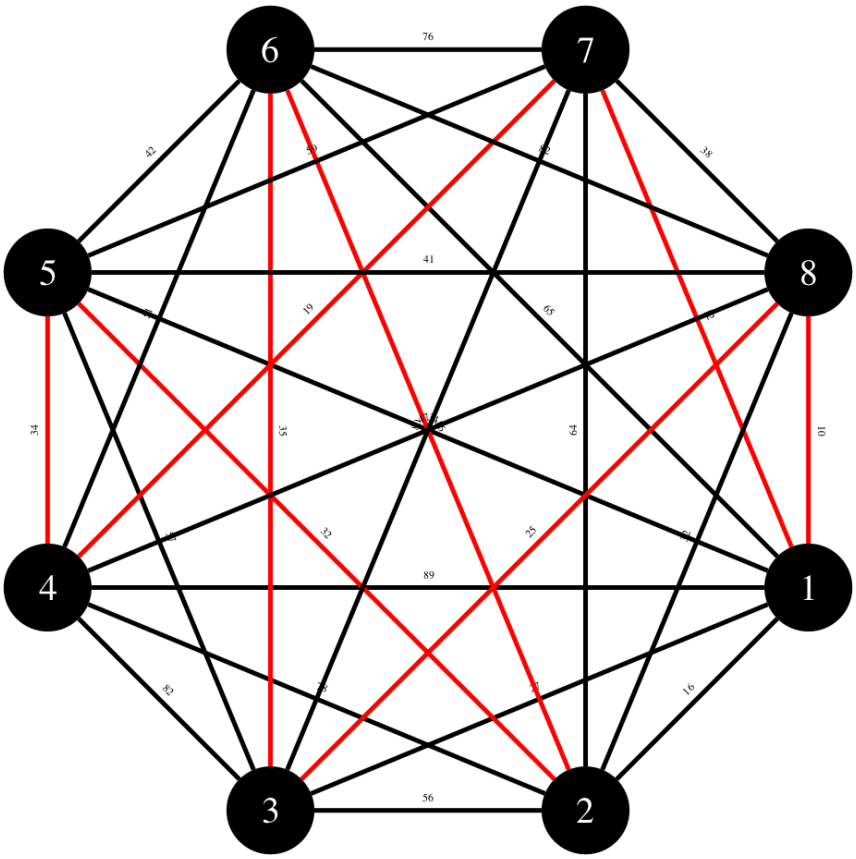


Gambar 2. out/rcm_2.png

3.3 Screenshot Graf Uji 1 Bobot Tur Lengkap

```
1. Reduced Cost Matrix
2. Bobot Tur Lengkap
>2
Testcase: 1
Input Graph:
inf     16      27      89      34      65      12      10
16      inf     56      78      32      16      64      32
27      56      inf     82      57      35      80      25
89      78      82      inf     34      51      19      47
34      32      57      34      inf     42      59      41
65      16      35      51      42      inf     76      82
12      64      80      19      59      76      inf     38
10      32      25      47      41      82      38      inf
Cost:  183
Path: 1->7->4->5->2->6->3->8->1

Simpul Hidup 6340
Execution Time: 0.578273057938s
```

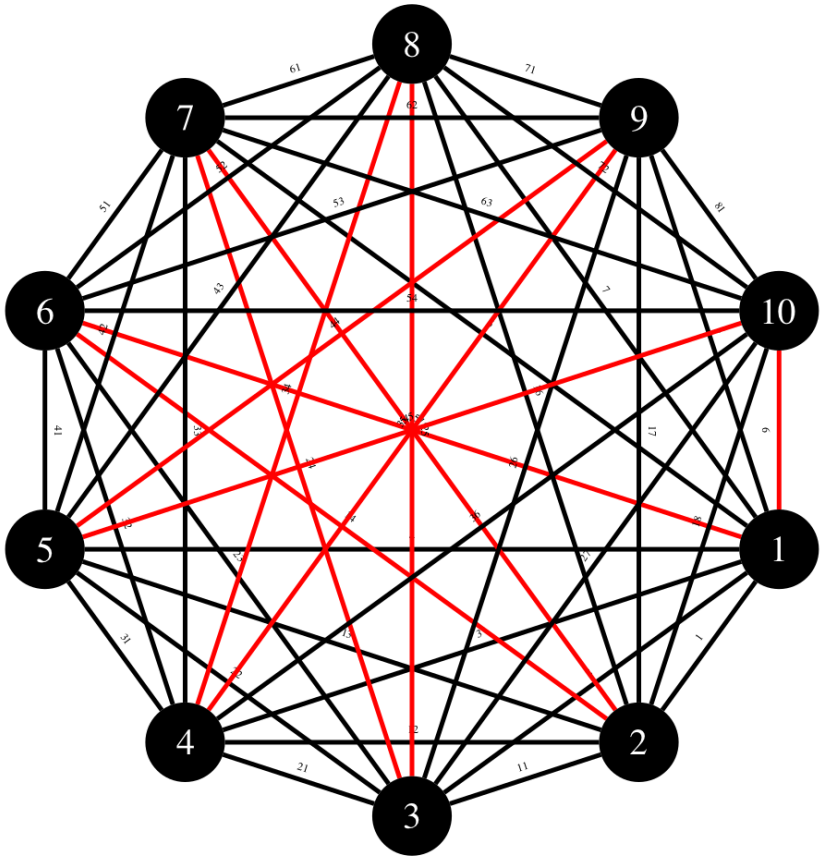


Gambar 3. out/blt_1.png

3.4 Screenshot Graf Uji 2 Bobot Tur Lengkap

```
1. Reduced Cost Matrix
2. Bobot Tur Lengkap
>2
Testcase: 2
Input Graph:
inf     1       2       3       4       5       6       7       8       9
1       inf     11      12      13      14      15      16      17      18
2       11      inf     21      22      23      24      25      26      27
3       12      21      inf     31      32      33      34      35      36
4       13      22      31      inf     41      42      43      44      45
5       14      23      32      41      inf     51      52      53      54
6       15      24      33      42      51      inf     61      62      63
7       16      25      34      43      52      61      inf     71      72
8       17      26      35      44      53      62      71      inf     81
9       18      27      36      45      54      63      72      81      inf
Cost:  250
Path: 1->6->2->7->3->8->4->9->5->10->1

Simpul Hidup 608722
Execution Time: 45.4492228031s
```



Gambar 4. blt_2.png

4. REFERENSI

*https://graph-tool.skewed.de/static/doc/index.htm*l, diakses pada 5 April 2017
https://docs.python.org/2/, diakses pada 5 April 2017
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, diakses pada 4 April 2017