REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN

CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00202034732, 21 September 2020
Pencipta		
Nama	:	**Wamiliana, Akmal Junaidi dkk**
Alamat	:	Jl. Ebony E3 No. 2 RT. 028, Lk. I, Kel. Beringin Raya, Kec. Kemiling, Bandar Lampung 35158, Bandar Lampung, Lampung, 35158
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**Wamiliana, Akmal Junaidi dkk**
Alamat	:	Jl. Ebony E3 No. 2 RT. 028, Lk. I, Kel. Beringin Raya, Kec. Kemiling, Bandar Lampung 35158, Bandar Lampung, Lampung, 35158
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Program Komputer**
Judul Ciptaan	:	**Penentuan Degree Constrained Minimum Spanning Tree Dengan Algoritma Sollin**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	14 Januari 2020, di Bandar Lampung
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor pencatatan	:	000204441

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n. MENTERI HUKUM DAN HAK ASASI MANUSIA
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL

Dr. Freddy Harris, S.H., LL.M., ACCS.
NIP. 196611181994031001

LAMPIRAN PENCIPTA

No	Nama	Alamat
1	Wamiliana	Jl. Ebony E3 No. 2 RT. 028, Lk. I, Kel. Beringin Raya, Kec. Kemiling, Bandar Lampung 35158
2	Akmal Junaidi	Jl. U. Suropati Gg. Mataram No. 50 RT. 005, Lk. II, Kel. Labuhan Ratu Raya, Kec. Labuhan Ratu, Bandar Lampung 35142
3	Amanto	Jl. Hi. Komarudin, No.35, Kel. Rajabasa Raya, Kec. Rajabasa, Bandar Lampung 35144
4	Febi Mudiyanto	Desa Gedung Ratu, RT/RW. 008/003, Kec. Anak Ratu Aji, Kab. Lampung Tengah 35513

LAMPIRAN PEMEGANG

No	Nama	Alamat
1	Wamiliana	Jl. Ebony E3 No. 2 RT. 028, Lk. I, Kel. Beringin Raya, Kec. Kemiling, Bandar Lampung 35158
2	Akmal Junaidi	Jl. U. Suropati Gg. Mataram No. 50 RT. 005, Lk. II, Kel. Labuhan Ratu Raya, Kec. Labuhan Ratu, Bandar Lampung 35142
3	Amanto	Jl. Hi. Komarudin, No.35, Kel. Rajabasa Raya, Kec. Rajabasa, Bandar Lampung 35144
4	Febi Mudiyanto	Desa Gedung Ratu, RT/RW. 008/003, Kec. Anak Ratu Aji, Kab. Lampung Tengah 35513

Formulir Permohonan Pencatatan Ciptaan

Data Permohonan		
Nomor Permohonan	:	EC00202034732
Tanggal Pengajuan	:	21-09-2020
Jenis Ciptaan	:	Program Komputer
Judul Ciptaan	:	Penentuan Degree Constrained Minimum Spanning Tree dengan Algoritma Sollin
Uraian Ciptaan	:	Program komputer ini adalah aplikasi yang dapat digunakan untuk menyelesaikan permasalahan yang berkaitan dengan persoalan koneksi dan instalasi jaringan. Input aplikasi berupa sistem jaringan misalnya jaringan pipa air bersih, dimana pada setiap titik percabangan memiliki keterbatasan sambungan. Hal ini diterapkan untuk menjamin keterhandalan jaringan pipa air tersebut.
Tanggal dan tempat diumumkan pertama kali	:	Bandar Lampung,14-01-2020

Pencipta		
Nama	Alamat	Kebangsaan
Wamiliana	Jl. Ebony E3 No. 2 RT. 028, Lk. I, Kel. Beringin Raya, Kec. Kemiling, Bandar Lampung 35158	Indonesia
Akmal Junaidi	Jl. U. Suropati Gg. Mataram No. 50 RT. 005, Lk. II, Kel. Labuhan Ratu Raya, Kec. Labuhan Ratu, Bandar Lampung 35142	Indonesia
Amanto	Jl. Hi. Komarudin, No.35, Kel. Rajabasa Raya, Kec. Rajabasa, Bandar Lampung 35144	Indonesia
Febi Mudiyanto	Desa Gedung Ratu, RT/RW. 008/003, Kec. Anak Ratu Aji, Kab. Lampung Tengah 35513	Indonesia

Pemegang		
Nama	Alamat	Kebangsaan
Wamiliana	Jl. Ebony E3 No. 2 RT. 028, Lk. I, Kel. Beringin Raya, Kec. Kemiling, Bandar Lampung 35158	Indonesia
Akmal Junaidi	Jl. U. Suropati Gg. Mataram No. 50 RT. 005, Lk. II, Kel. Labuhan Ratu Raya, Kec. Labuhan Ratu, Bandar Lampung 35142	Indonesia
Amanto	Jl. Hi. Komarudin, No.35, Kel. Rajabasa Raya, Kec. Rajabasa, Bandar Lampung 35144	Indonesia
Febi Mudiyanto	Desa Gedung Ratu, RT/RW. 008/003, Kec. Anak Ratu Aji, Kab. Lampung Tengah 35513	Indonesia

Lampiran
KTP Peringatan Detail

Jakarta, 21-09-2020
Pemohon/Kuasa

t.t.d.

Tanda Tangan

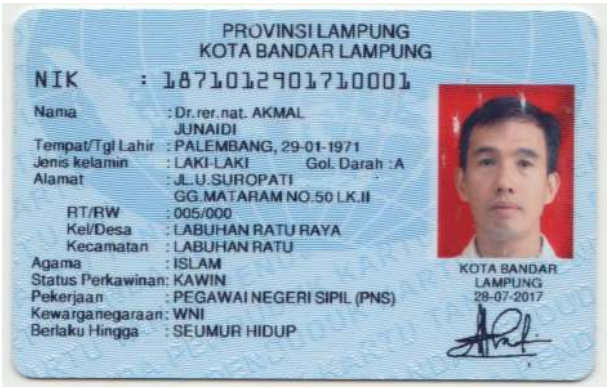
Nama Lengkap Akmal Junaidi

Catatan: Jika dalam jangka waktu 5(lima) hari kerja belum mendapatkan surat pencatatan ciptaan, agar menghubungi email: **permohonan.ciptadesain@dgip.go.id**

Scan KTP Pemohon dan Pencipta: Penentuan *Degree Constrained Minimum Spanning Tree* dengan Algoritma Sollin

1. Wamiliana

2. Akmal Junaidi



3. Amanto

4. Febi Mudiyanto

SURAT PERNYATAAN

Yang bertanda tangan di bawah ini, pemegang hak cipta:

N a m a	:	Wamiliana
Kewarganegaraan	:	Indonesia
Alamat	:	Jl. Ebony E3 No. 2 RT. 028, Lk. I, Kel. Beringin Raya, Kec. Kemiling Bandar Lampung 35158

N a m a	:	Akmal Junaidi
Kewarganegaraan	:	Indonesia
Alamat	:	Jl. U. Suropati Gg. Mataram No. 50 RT. 005, Lk. II, Kel. Labuhan Ratu Raya, Kec. Labuhan Ratu, Bandar Lampung 35142

N a m a	:	Amanto
Kewarganegaraan	:	Indonesia
Alamat	:	Jl. Hi. Komarudin, No.35, Kel. Rajabasa Raya, Kec. Rajabasa, Bandar Lampung 35144

N a m a	:	Febi Mudiyanto
Kewarganegaraan	:	Indonesia
Alamat	:	Desa Gedung Ratu, RT/RW. 008/003, Kec. Anak Ratu Aji, Kab. Lampung Tengah 35513

Dengan ini menyatakan bahwa:

1. Karya Cipta yang saya mohonkan:
   Berupa : Program Komputer
   Berjudul : Penentuan *Degree Constrained Minimum Spanning Tree* dengan Algoritma Sollin.

   - Tidak meniru dan tidak sama secara esensial dengan Karya Cipta milik pihak lain atau obyek kekayaan intelektual lainnya sebagaimana dimaksud dalam Pasal 68 ayat (2);
   - Bukan merupakan Ekspresi Budaya Tradisional sebagaimana dimaksud dalam Pasal 38;
   - Bukan merupakan Ciptaan yang tidak diketahui penciptanya sebagaimana dimaksud dalam Pasal 39;
   - Bukan merupakan hasil karya yang tidak dilindungi Hak Cipta sebagaimana dimaksud dalam Pasal 41 dan 42;
   - Bukan merupakan Ciptaan seni lukis yang berupa logo atau tanda pembeda yang digunakan sebagai merek dalam perdagangan barang/jasa atau digunakan sebagai lambang organisasi, badan usaha, atau badan hukum sebagaimana dimaksud dalam Pasal 65 dan;
   - Bukan merupakan Ciptaan yang melanggar norma agama, norma susila, ketertiban umum, pertahanan dan keamanan negara atau melanggar peraturan perundang-undangan sebagaimana dimaksud dalam Pasal 74 ayat (1) huruf d Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

2. Sebagai pemohon mempunyai kewajiban untuk menyimpan asli contoh ciptaan yang dimohonkan dan harus memberikan apabila dibutuhkan untuk kepentingan penyelesaian sengketa perdata maupun pidana sesuai dengan ketentuan perundang-undangan.

3. Karya Cipta yang saya mohonkan pada Angka 1 tersebut di atas tidak pernah dan tidak sedang dalam sengketa pidana dan/atau perdata di Pengadilan.

4. Dalam hal ketentuan sebagaimana dimaksud dalam Angka 1 dan Angka 3 tersebut di atas saya / kami langgar, maka saya / kami bersedia secara sukarela bahwa:
   a. Permohonan karya cipta yang saya ajukan dianggap ditarik kembali; atau
   b. Karya Cipta yang telah terdaftar dalam Daftar Umum Ciptaan Direktorat Hak Cipta, Direktorat Jenderal Hak Kekayaan Intelektual, Kementerian Hukum Dan Hak Asasi Manusia R.I dihapuskan sesuai dengan ketentuan perundang-undangan yang berlaku.

c. Dalam hal kepemilikan Hak Cipta yang dimohonkan secara elektronik sedang dalam berperkara dan/atau sedang dalam gugatan di Pengadilan maka status kepemilikan surat pencatatan elektronik tersebut ditangguhkan menunggu putusan Pengadilan yang berkekuatan hukum tetap.

Demikian Surat pernyataan ini saya/kami buat dengan sebenarnya dan untuk dipergunakan sebagimana mestinya.

Bandar Lampung, 21 September 2020

1. (Wamiliana)
Pemegang Hak Cipta*

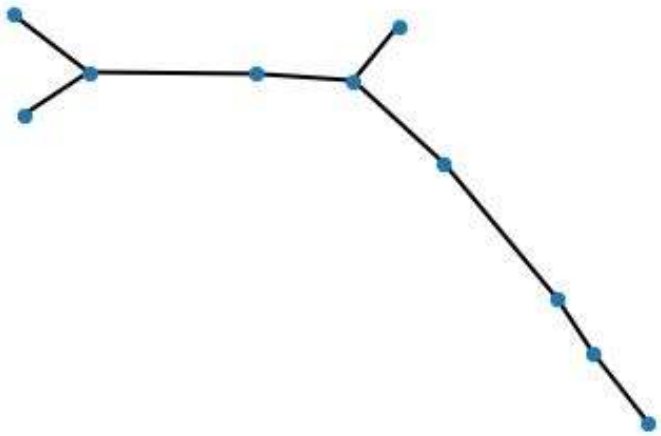
2. (Akmal Junaidi)
Pemegang Hak Cipta*

3. (Amanto)
Pemegang Hak Cipta*

4. (Febi Mudiyanto)
Pemegang Hak Cipta*

Tim DCMST Universitas Lampung

Panduan Penggunaan Program Komputer Penentuan Degree Constrained Minimum Spanning Tree dengan Algoritma Sollin



Panduan ini berisi penjelasan dan contoh penggunaan program Penentuan Degree Constrained Minimum Spanning Tree dengan Algoritma Sollin. Script program komputer ini terdiri dari beberapa bagian, berikut adalah penjelasannya :

1. *Import* Library

```
import numpy as np
import math
import collections
```

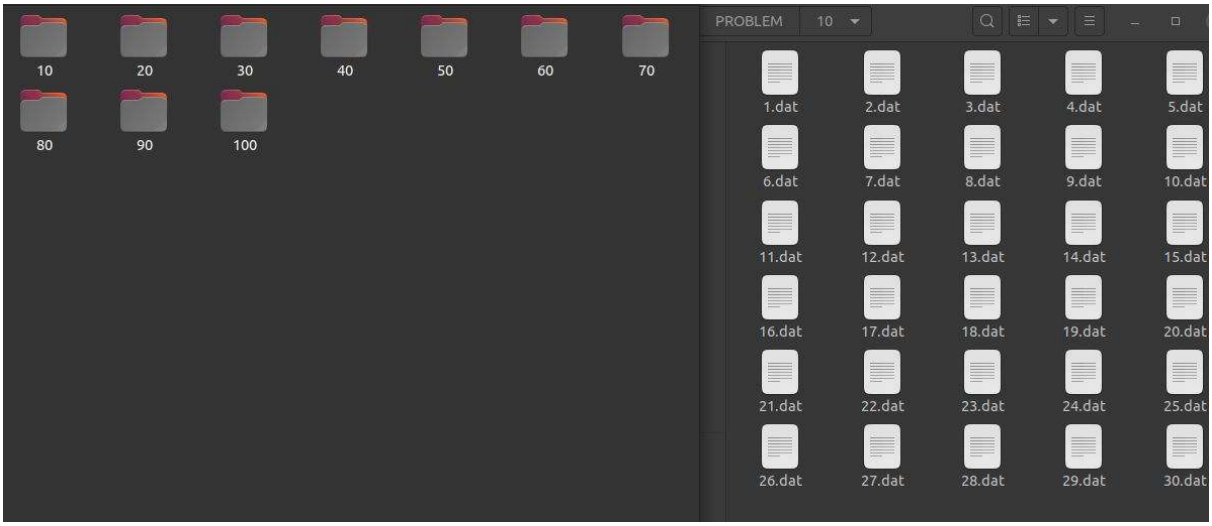
Library yang digunakan dalam program ini adalah numpy, math, dan collections. Numpy digunakan untuk operasi matriks/tabel. Math digunakan untuk proses perhitungan matematika. Collections digunakan untuk menghitung banyaknya nilai dalam suatu tabel.

2. *Load* Data

```
#load data
data = np.loadtxt('F:\\PROBLEM\\10\\1.dat') #directory file data
data = data.astype(int)
data = data.reshape(len(data),1)
```

Data disimpan dalam file ".dat" didalamnya terdapat list bobot garis, kemudian disiapkan dengan *reshape* menjadi bentuk vektor.

Berikut ini merupakan tampilan folder PROBLEM dan 30 file dat per folder, file ini berisi bobot dengan sebanyak jumlah garis pada graf lengkap titik tersebut.



3. *Function fase1*

```
#function untuk menghitung jumlah titik graf
def jumlahtitik(data):
  ...

#function untuk membuat matriks sumber dan tujuan
def matriks(n):
  ...

#function untuk membuat matriks tetangga
def matriks_adjacent(tab,n):
  ...
```

Function ini digunakan untuk menghitung banyak titik (*vertex*) dari data, kemudian membuat matriks sumber dan tujuan sebagai inisialisasi titik graf, selanjutnya membentuk matriks tetangga untuk representasi graf dalam bentuk matriks.

Dibawah ini proses pemanggilan *function* tersebut.

```
n = jumlahtitik(data) #menghitung jumlah titik
vertex_list = list(range(1,n+1))
X = matriks(n) #membuat tabel sumber, tujuan
tab = np.concatenate((X,data),axis=1) #membuat tabel sumber,tujuan,bobot
m = matriks_adjacent(tab,n) #membuat matriks tetangga
```

Berikut adalah gambar contoh matriks tetangga dari file 10 titik dengan nama file “1.dat”.

m - NumPy array

	0	1	2	3	4	5	6	7	8	9
0	0	115	74	42	955	712	660	697	450	806
1	115	0	391	504	597	537	452	709	452	784
2	74	391	0	939	472	372	272	250	644	354
3	42	504	939	0	212	515	132	248	603	642
4	955	597	472	212	0	935	70	715	328	24
5	712	537	372	515	935	0	426	987	72	227
6	660	452	272	132	70	426	0	144	814	82
7	697	709	250	248	715	987	144	0	740	703
8	450	452	644	603	328	72	814	740	0	886
9	806	784	354	642	24	227	82	703	886	0

4. Script DCMST

```
#---------------Source Code Algoritma Modified Sollin - (DCMST)-------------

#function untuk mencari garis terpendek dari setiap titik
def g_terpendek_setiaptitik(n):
  ...

#function untuk mengecek derajat (degree)
def  degree(tabel):
  ...

#function untuk mencari mst fase awal
def carimst_fase1(tabel):
  ...

#function untuk mereduksi element list yang duplicate
def unique(list1):
  ...

#function untuk mengkombinasikan setM
def combine(setM):
  ...
```

Script DCMST mengandung beberapa *function* dengan berlandaskan pada algoritma sollin yang telah dimodifikasi sehingga dapat menangani masalah *Degree Restricted*.

Berikut ini bagian penggunaan function diatas,

```
derajat = 3 #inisialisasi batas derajat
b_list = [] #titik tabu untuk menyimpan titik dengan degree >= 3
tabel = g_terpendek_setiaptitik(n) #fase 1 pencarian garis terpendek setiap titik
b_list = degree(tabel) #menyimpan titik yang mencapai batas derajat ke black list

#mengecek degree
degree(tabel)
setM,mst = carimst_fase1(tabel)
```

Script pemanggilan diatas mengandung inisialisasi derajat, variabel tabu (b_list), pencarian garis terpendek setiap titik, pengecekan *degree,* dan output awal komponen dan mst.

Kemudian menghitung ulang derajat dengan *degree counter* agar dapat diproses dalam langkah selanjutnya.

```
xdegree = collections.Counter(tabel[:,0])
ydegree = collections.Counter(tabel[:,1])
```

5. Mereduksi garis yang terhubung dalam titik dengan degree > 3

```
#function untuk mereduksi degree > 3
def reduksidegree():
  ...

b_list = degree(tabel) #mengecek degree dan memasukkan dalam black list

combine(setM) #menggabungkan komponen di setM
```

setM, mst, serta tabel yang telah didapat sebelumnya diolah lagi dengan cara mereduksi titik yang derajatnya lebih dari 3 kemudian melakukan combine sehingga terbentuk lagi setM baru.

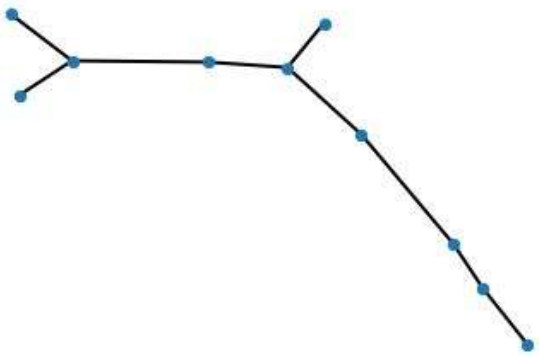
Selanjutnya *script* berikut ini akan dilakukan proses iterasi sampai didapatkan banyak komponen adalah satu.

```
#function menghubungkan komponen
def iterasiMenghubungkanKomponen(setM):
  ...

print(mst) #nilai DCMST dengan d <= 3
```

6. Hasil

	0	1	2
0	1	2	115
1	1	3	74
2	1	4	42
3	4	7	132
4	5	7	70
5	5	10	24
6	6	9	72
7	6	10	227
8	7	8	144



```
In [9]: mst
Out[9]: 900
```

Hasil disimpan dalam variabel “tabel” yang mengandung titik sumber (kolom 0), titik tujuan (kolom 1) dan bobot (kolom 2). Kemudian ketika direpresentasikan dalam bentuk graf didapatlah MST seperti pada gambar dan nilai MST disimpan dalam variabel “mst” pada contoh kali ini mst adalah 900.

Tim DCMST Universitas Lampung

Source Code Program Komputer Penentuan Degree Constrained Minimum Spanning Tree dengan Algoritma Sollin

(Bahasa Python)

```
import numpy as np
import math
import collections

#Load data
data=np.loadtxt('F:\\PROBLEM\\10\\1.dat') #directory file data
data=data.astype(int)
data=data.reshape(len(data),1)
#-------------------------------------------------
#function untuk menghitung jumlah titik graf
def jumlahtitik(data):
    jumlah_garis=len(data)
    n=(1+math.sqrt(1+(8*jumlah_garis)))/2
    return int(n)
#-----------------------------------------
#function untuk membuat matriks sumber dan tujuan
def matriks(n):
    c=list(range(n,0,-1))
    d=list(range(1,n+1))
    X=[[],[]]
    for t in d:
        i=1
        a=t+1
        while (i < c[t-1]):
            X[0].append(t)
            X[1].append(a)
            i += 1
            a += 1

    return np.asarray(X).T
#-----------------------------------------
#function untuk membuat matriks tetangga
def matriks_adjacent(tab,n):
    m=np.zeros((n,n), dtype=int)
```

```
    for i in range(len(tab)):
        a=int(tab[i][0])
        b=int(tab[i][1])
        m[a-1,b-1]=tab[i][2]
    m += m.T
    return m
#-----------------------------------------------------
n=jumlahtitik(data) #menghitung jumlah titik
vertex_list=list(range(1,n+1))
X=matriks(n) #membuat tabel sumber, tujuan
tab=np.concatenate((X,data),axis=1)  #membuat tabel sumber,tujuan, bobot
m=matriks_adjacent(tab,n) #membuat matriks tetangga

#-----------------Source Code Algoritma Modified Sollin - (DCMST) -------------------
#function untuk mencari garis terpendek setiap titik
def g_terpendek_setiaptitik(n):
    c=0
    d=0
    tabel=np.zeros((n,3))
    k=n-1
    while k>0:
        for i in range(n):

            min=max(m[i])
            for j in range(n):
                if m[i][j]>0 and m[i][j]<min:
                    min=m[i][j]
                    c=i+1
                    d=j+1
            if c>d:
                c,d=d,c

            tabel[k][0],tabel[k][1],tabel[k][2]=c,d,min
            k -= 1
    tabel=np.unique(tabel,axis=0)

    return tabel
#-----------------------------------------------------
#function untuk mengecek degree
def degree(tabel):
    xdegree=collections.Counter(tabel[:,0])
    ydegree=collections.Counter(tabel[:,1])
    for t in range(n,0,-1):
        tot_degree=(xdegree[t]+ydegree[t])
        if tot_degree>=derajat:
            if t not in b_list:
                b_list.append(t)
            if t in vertex_list:
```

```
                vertex_list.remove(t)
    return b_list
#-------------------------------------------------

def cekdegree(tabel):
    #cek degree
    xdegree=collections.Counter(tabel[:,0])
    ydegree=collections.Counter(tabel[:,1])

    #remove vertex list yang degree nya >= 3
    for t in vertex_list:
            tot_degree=(xdegree[t]+ydegree[t])
            if tot_degree>=derajat:
                if t not in b_list:
                    b_list.append(t)
                if t in vertex_list:
                    vertex_list.remove(t)

#-------------------------------------------------
def carimst_fase1(tabel):
    setM=[]
    titik=[]
    m=[]
    mst=0
    for k in range(len(tabel)):
        cekdegree(tabel)
        a,b=tabel[k][0],tabel[k][1]
        m=[[a,b]]
        bobot=tabel[k][2]

        #print(a==m[0][0] and b==m[0][1])

        if setM:
            l=0
            while l<len(setM):
                if (m[0][0] in setM[l]):
                    for ll in range(l+1,len(setM)):
                        if (m[0][1] in setM[ll]):
                            #print(m[0][0],m[0][1])
                            setM[ll].append(m[0][0])

                            break
                    setM[l]+=m[0]
                    mst=mst+bobot
                    setM[l]=list(np.unique(setM[l]))
                    l=len(setM)
                elif (m[0][1] in setM[l]):
                    for ll in range(l+1, len(setM)):
```

```
                        if (m[0][0] in setM[ll]):
                            #print(m[0][0],m[0][1])
                            setM[ll].append(m[0][1])
                            #print(b)
                            break

                        #print(a,b)
                    setM[l]+=m[0]
                    mst=mst+bobot
                    setM[l]=list(np.unique(setM[l]))
                    l=len(setM)


                elif (m[0][0] not in setM[l]) and (m[0][1] not in setM[l]):
                    if l==len(setM)-1:
                        setM.append(m[0])
                        mst=mst+bobot
                        l=len(setM)
                    else:
                        l=l+1
            titik.append(a)
            titik.append(b)
        else:
            setM.append(m[0])
            mst=mst+bobot
            titik.append(a)
            titik.append(b)
        titik=list(np.unique(titik))

    combine(setM)
    return setM,mst

def unique(list1):
    x = np.array(list1)
    x = np.unique(x)

    return list(x)


def combine(setM):
    p=0
    while p<len(setM):
        for n in setM[p]:
            k=0
            while k<len(setM):
                if p!=k:
                    if n in setM[k]:
                        setM[p]+=setM[k]
```

```
                        setM.remove(setM[k])

                k+=1
        setM[p]=unique(setM[p])
        p+=1


derajat=3 #inisialisasi batas derajat
b_list=[] #titik tabu untuk menyimpan titik dengan degree >= 3
tabel=g_terpendek_setiaptitik(n) #fase 1 pencarian garis terpendek setiap titik
b_list=degree(tabel) #memasukan ke black_list

#cek degree
cekdegree(tabel)
setM,mst=carimst_fase1(tabel)

xdegree=collections.Counter(tabel[:,0])
ydegree=collections.Counter(tabel[:,1])

#mereduksi degree
#lanjutan degree untuk >3 reduksi
for t in range(n,0,-1):
    cekdegree(tabel)
    b_list=degree(tabel)
    tot_degree=(xdegree[t]+ydegree[t])
    b_list=degree(tabel)
    if tot_degree>derajat:
        temptabel=np.delete(tabel,2,1)
        i,j=np.where(temptabel==t)#cek posisi titik t
        temp={}
        nn=len(i)
        ix=i[nn-1]
        for ii in i:
            temp[tabel[ii][0],tabel[ii][1]]=tabel[ii][2]
        temp=(sorted(temp.items(), key =
             lambda kv:(kv[1], kv[0])))

        while (len(temp))>derajat:


aa,bb,cc=temp[len(temp)-1][0][0],temp[len(temp)-1][0][1],temp[len(temp)-1][1]
            #print(aa,bb,cc)
            for ix in range(len(tabel)):
                if tabel[ix][0]==aa and tabel[ix][1]==bb and tabel[ix][2]==cc:
                    xx=ix

            if aa==t:
                T=bb
```

```
            else:
                T=aa

            cari_tl={}
            oo=0
            for xix in range(len(setM)):
                if T in setM[xix]:
                    komponen=setM[xix]

            for o in vertex_list:
                if o != T and o not in komponen:
                        if o != T:
                            cari_tl[o,T]=m[int(o-1)][int(T-1)]
                        oo+=1
            cari_tl=(sorted(cari_tl.items(), key =
                 lambda kv:(kv[1], kv[0])))

            o,T,cc=cari_tl[0][0][0],cari_tl[0][0][1],cari_tl[0][1]
            if o>T:
                o,T=T,o

            tabel[xx][0],tabel[xx][1],tabel[xx][2]=o,T,cc
            tabel=np.unique(tabel, axis=0)
            del temp[len(temp)-1]


cekdegree(tabel)
b_list=degree(tabel)

xdegree=collections.Counter(tabel[:,0])
ydegree=collections.Counter(tabel[:,1])
setM,mst=carimst_fase1(tabel)
combine(setM)#menghubungkan beberapa komponen

#iterasiMenghubungkanKomponen(setM)
iterasi=1
while len(setM)>1:

    b_list=degree(tabel)
    cekdegree(tabel)
    hub={}
    iter=1;
    for x in range(len(setM)):
        for z in setM[x]:
            for y in range(len(setM)):
                if x!=y and (z in vertex_list) and (z not in b_list):
                    min=max(m[int(z-1),:])
                    c=z
```

```
                for aa in setM[y]:
                    if (z!=aa) and (aa in vertex_list) and (aa not in b_list):
                        d=aa
                        if m[int(z-1),int(aa-1)] < min:
                            min=m[int(z-1),int(aa-1)]
                            e=min
                            c=z
                            d=aa

                    hub[c,d]=min
                    iter+=1

hub=(sorted(hub.items(), key =
             lambda kv:(kv[1], kv[0])))
c=hub[0][0][0]
d=hub[0][0][1]
min=hub[0][1]

if (c<d) and (c in vertex_list) and (d in vertex_list):
    tabel=np.append(tabel,[[c,d,min]],axis=0)
    setM.append([c,d])
    t=[c,d]
    mst=mst+min

elif (c in vertex_list) and (d in vertex_list):
    tabel=np.append(tabel,[[d,c,min]],axis=0)
    setM.append([c,d])
    t=[d,c]
    mst=mst+min

b_list=degree(tabel)
cekdegree(tabel)
combine(setM)
tabel=np.unique(tabel,axis=0)
iterasi+=1
```