LAPORAN KUIS 2:
PROGRAM KOMPUTER DAN ANALISIS KASUS MODEL PENUGASAN DAN ALGORITMA DIJKSTRA

Ditulis oleh:

Nuzha Musyafira 051116 4000 0014

Dosen:

M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

Riset Operasi – IF84923

Departemen Informatika

Fakultas Teknologi Informasi dan Komunikasi (FTIK)

Institut Teknologi Sepuluh Nopember (ITS)

2018

SUMPAH DAN PERNYATAAN

Demi Allah (Tuhan) Yang Maha Esa, maka dengan ini, saya bersumpah dan menyatakan dengan sebenar-benarnya bahwa saya mengerjakan jawaban soal Kuis 2 ini secara sendiri dan mandiri, tidak melakukan kecurangan dalam bentuk apa pun, tidak menyalin/menjiplak/melakukan plagiat pekerjaan/karya orang lain, serta tidak menerima bantuan pengerjaan dalam bentuk apa pun dari orang lain. Saya bersedia menerima semua konsekuensi dalam bentuk apa pun, apabila saya ternyata terbukti melakukan kecurangan dan/atau penyalinan/penjiplakan/plagiat pekerjaan/karya orang lain.

Surabaya, 17 November 2018

Nuzha Musyafira
051116 4000 0014

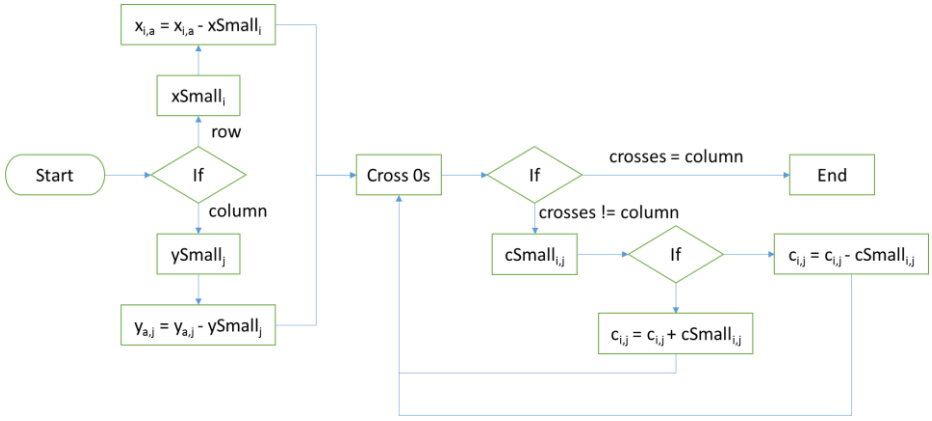
DAFTAR ISI

MODEL PENUGASAN

I. Algoritma dan Penjelasan

Model penugasan merupakan kasus khusus *transformation problem* dimana tujuannya adalah untuk menetapkan pekerjaan atau pekerja j ke mesin atau pekerja i sehingga biaya yang dikeluarkan dapat diminimalkan.

II. Diagram Alur



III. Pseudocode

```
def isEqual(lenRow,lenCol):
    global table
    if lenRow<lenCol:
        temp=[]
        for x in range(lenCol):
            temp.append(0)
        for x in range(lenCol - lenRow):
            table.append(temp)
        lenRow=lenCol
    elif lenRow>lenCol:
        for x in range(lenRow):
            for y in range(lenRow - lenCol):
                table[x].append(0)
        lenCol=lenRow
```

Fungsi di atas digunakan sebagai validasi pada saat inisialisasi program, yakni apabila jumlah baris tidak sama dengan jumlah kolom, makan akan dibentuk suatu *dummy* untuk memenuhi. Paramater lenRow adalah jumlah baris dan lenCol adalah jumlah kolom.

```
def minValue(x,y,table2):
    if y==1:
        table2=list(zip(*table2))
    temp=[]
```

```
4  for i in range(x):
5      temp.append(min(table2[i]))
6  return temp
```

Fungsi di atas adalah fungsi untuk mencari nilai elemen terkecil dari setiap baris maupun kolom. Parameter x adalah jumlah baris atau kolom, y adalah *condition flag* untuk menentukan apakah yang dicari merupakan baris atau kolom, dan table2 adalah tabel penugasan.

```
def toZero(x,y,min):
1  for i in range(x):
2      for j in range(x):
3          if y==1:
4              table[j][i]=table[j][i]-min[i]
5          else:
6              table[i][j]=table[i][j]-min[i]
```

Fungsi di atas adalah fungsi untuk mengurangi semua elemen dengan nilai elemen terkecil dari setiap baris maupun kolom. Parameter x adalah jumlah baris atau kolom, y adalah *condition flag* untuk menentukan apakah yang diproses merupakan baris atau kolom, dan min adalah *array* yang berisi nilai elemen terkecil.

```
def countZero(x,y,table2):
1  if y==1:
2      table2=list(zip(*table2))
3  temp=[]
4  for i in range(x):
5      temp.append(table2[i].count(0))
6  return temp
```

Fungsi di atas adalah fungsi untuk menghitung jumlah elemen yang bernilai 0 dari setiap baris maupun kolom. Parameter x adalah jumlah baris atau kolom, y adalah *condition flag* untuk menentukan apakah yang dicari baris atau kolom, dan table2 adalah tabel penugasan.

```
def sortNumZero(arr):
1  val=len(arr)-1
2  temp=[]
3  for a in arr:
4      for i in range(len(a)):
5          temp2=[]
6          temp2.append(a[i])
7          temp2.append(val)
8          temp2.append(i)
9          temp.append(temp2)
10     val-=1
11 temp.sort(key=lambda z: (-z[0], -z[1]))
12 return temp
```

Fungsi di atas adalah fungsi untuk mengurutkan jumlah elemen yang bernilai 0 dari setiap baris maupun kolom berdasarkan jumlah terbanyak dan pemrioritasan pada baris. Parameter arr adalah array yang menyimpan indeks baris atau kolom dan jumlah elemen 0 nya.

```
def crossZero(table2,sortZ,counter):
    row=[]
    col=[]
    for i in range(len(table2)):
        row.append(0)
        col.append(0)
    for i in sortZ:
        if i[1]==1:
            if table2[i[2]].count(0)>0:
                for j in range(len(table2[i[2]])):
                    if table2[i[2]][j]==0:
                        table2[i[2]][j]='#'
                counter+=1
                row[i[2]]=1
        else:
            temp=list(zip(*table2))
            if temp[i[2]].count(0)>0:
                for j in range(len(temp[i[2]])):
                    if table2[j][i[2]]==0:
                        table2[j][i[2]]='#'
                counter+=1
                col[i[2]]=1
    if counter<len(table2):
        temp2=[]
        for s in range(len(table2)):
            for x in range(len(table2)):
                if row[s]==0 and col[x]==0:
                    temp2.append(table2[s][x])
        newMin=min(temp2)
        for s in range(len(table2)):
            for x in range(len(table2)):
                if table2[s][x]=='#':
                    table2[s][x]=0
                if row[s]==0 and col[x]==0:
                    table2[s][x]-=newMin
                elif row[s]==1 and col[x]==1:
                    table2[s][x]+=newMin
        numZ=[]
        numZ.append(countZero(4,0,table2))
        numZ.append(countZero(4,1,table2))
```

```
40      sortZ=sortNumZero(numZ)
41      crossZero(table2,sortZ,0)
42  else:
43      for s in table2:
44          for x in range(len(s)):
45              if s[x]=='#':
46                  s[x]=0
```

Fungsi di atas adalah fungsi untuk melakukan *crossing* nilai 0. Parameter table2 adalah tabel penugasan, sortZ adalah skala prioritas manakah baris atau kolom yang akan ditarik garis terlebih dahulu, dan counter adalah jumlah garis yang telah ditarik.

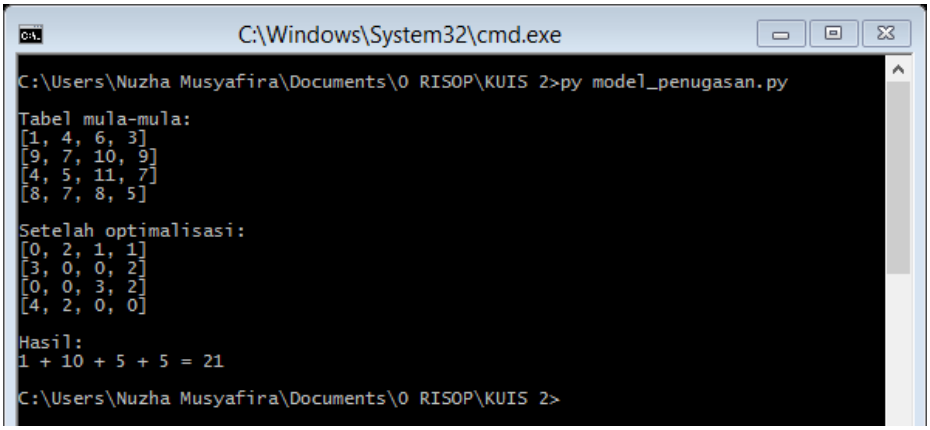
```
def costs(x,table2):
1   flag=[]
2   for i in range(x):
3       flag.append(0)
4   temp2=[]
5   while sum(flag)<=x:
6       c=[]
7       for a in table2:
8           c.append(a.count(0))
9       if c.count(1)==0:
10          Break
11      temp=[]
12      idx=c.index(1)
13      i=idx
14      j=table2[idx].index(0)
15      flag[j]=1
16      for f in range(x):
17          if table2[f][j]==0:
18              table2[f][j]='*'
19      c[idx]=0
20      table2[i][j]='*'
21      temp.append(i)
22      temp.append(j)
23      temp2.append(temp)
24  for s in table2:
25      for x in range(len(s)):
26          if s[x]=='*':
27              s[x]=0
28  temp2=sorted(temp2)
29  return temp2
```

Fungsi di atas adalah fungsi untuk menyeleksi dan mengalokasikan berdasarkan elemen yang bernilai 0 yang telah dioptimalisasi. Parameter x adalah jumlah baris atau kolom dan table2 adalah tabel penugasan.

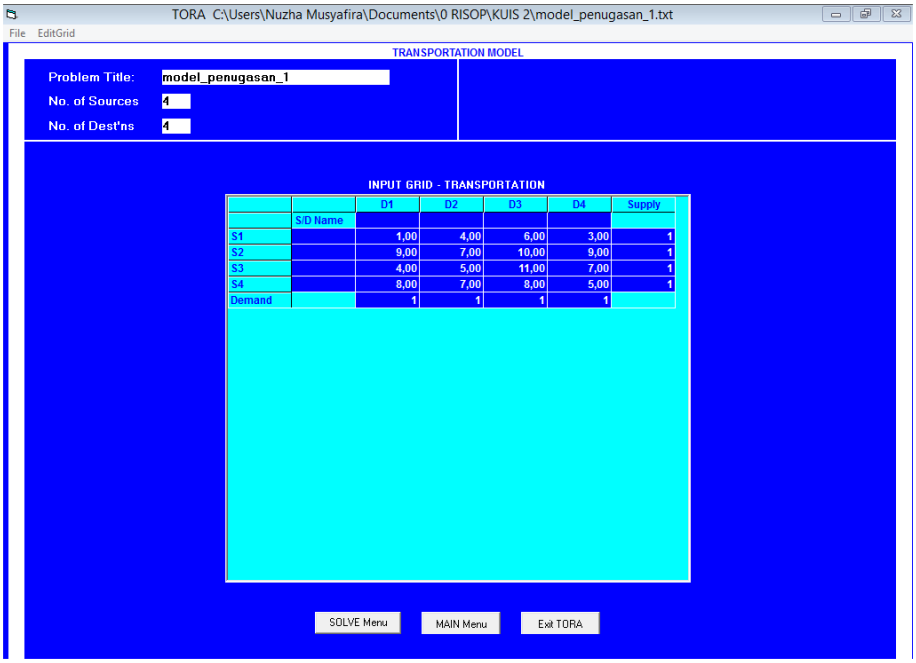
IV. Studi Kasus I

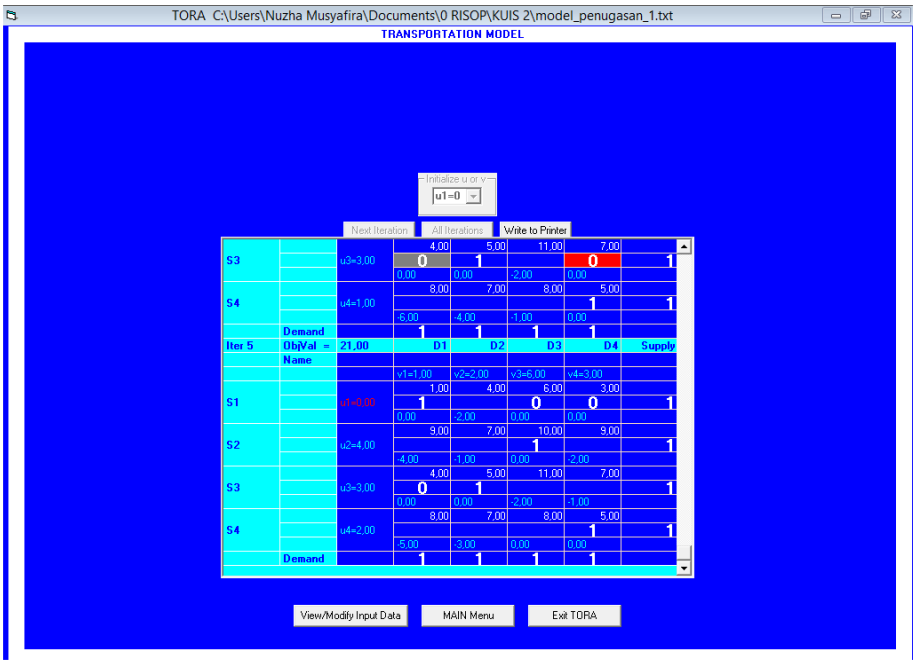
Pekerja \ Mesin	1	2	3	4
1	1	4	6	3
2	9	7	10	9
3	4	5	11	7
4	8	7	8	5

I. Hasil Output



II. TORA





III. Analisis Algoritma

Permasalahan ini mula-mula dipetakan dalam bentuk tabel penugasan dengan sejumlah i baris dan j kolom. Kemudian, memastikan semua baris dan kolom sudah memiliki nilai 0 dengan cara mengurangi setiap elemen pada baris atau kolom dengan nilai elemen terkecil pada setiap baris atau kolom yang bersangkutan. Setelah itu, ditarik garis yang melalui nilai 0 dengan jumlah lebih banyak, dengan catatan apabila ada kesamaan antara baris dan kolom, maka diutamakan baris. Apabila jumlah garis yang ditarik sama dengan jumlah kolom, maka alokasikan ongkos minimum. Jika tidak, maka ulangi iterasi dengan mencari nilai terkecil dari elemen yang tidak dilewati garis, kemudian

dikurangkan terhadapnya, dan ditambah terhadap elemen yang dilewati persimpangan garis.

IV. Source Code

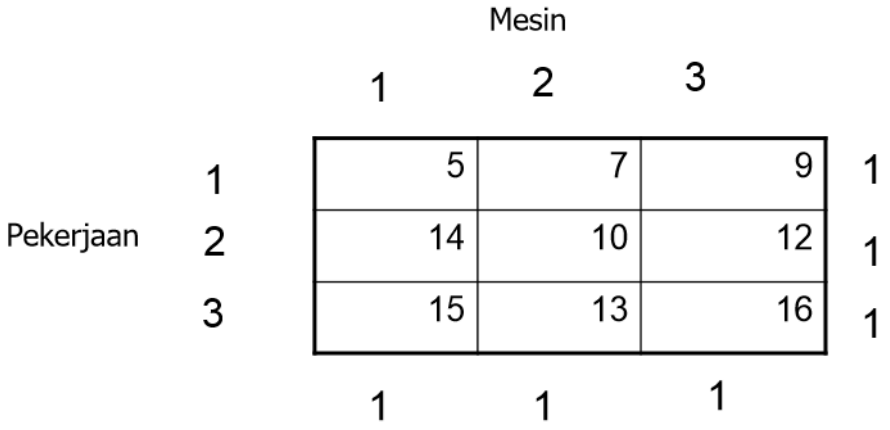
```
    def isEqual(lenRow,lenCol):
        global table
        if lenRow<lenCol:
            temp=[]
            for x in range(lenCol):
                temp.append(0)
            for x in range(lenCol - lenRow):
                table.append(temp)
            lenRow=lenCol
        elif lenRow>lenCol:
            for x in range(lenRow):
                for y in range(lenRow - lenCol):
                    table[x].append(0)
            lenCol=lenRow

    def sortNumZero(arr):
        val=len(arr)-1
        temp=[]
        for a in arr:
            for i in range(len(a)):
                temp2=[]
                temp2.append(a[i])
                temp2.append(val)
                temp2.append(i)
                temp.append(temp2)
            val-=1
        temp.sort(key=lambda z: (-z[0], -z[1]))
        return temp

    def countZero(x,y,table2):
        if y==1:
            table2=list(zip(*table2))
        temp=[]
        for i in range(x):
            temp.append(table2[i].count(0))
        return temp

    def toZero(x,y,min):
        for i in range(x):
            for j in range(x):
                if y==1:
                    table[j][i]=table[j][i]-min[i]
                else:
                    table[i][j]=table[i][j]-min[i]
```

```
def minValue(x,y,table2):
    if y==1:
        table2=list(zip(*table2))
    temp=[]
    for i in range(x):
        temp.append(min(table2[i]))
    return temp

def crossZero(table2,sortZ,counter):
    row=[]
    col=[]
    for i in range(len(table2)):
        row.append(0)
        col.append(0)
    for i in sortZ:
        if i[1]==1:
            if table2[i[2]].count(0)>0:
                for j in range(len(table2[i[2]])):
                    if table2[i[2]][j]==0:
                        table2[i[2]][j]='#'
                counter+=1
                row[i[2]]=1
        else:
            temp=list(zip(*table2))
            if temp[i[2]].count(0)>0:
                for j in range(len(temp[i[2]])):
                    if table2[j][i[2]]==0:
                        table2[j][i[2]]='#'
                counter+=1
                col[i[2]]=1
    if counter<len(table2):
        temp2=[]
        for s in range(len(table2)):
            for x in range(len(table2)):
                if row[s]==0 and col[x]==0:
                    temp2.append(table2[s][x])
        newMin=min(temp2)
        for s in range(len(table2)):
            for x in range(len(table2)):
                if table2[s][x]=='#':
                    table2[s][x]=0
                if row[s]==0 and col[x]==0:
                    table2[s][x]-=newMin
                elif row[s]==1 and col[x]==1:
                    table2[s][x]+=newMin
        numZ=[]
        numZ.append(countZero(4,0,table2))
        numZ.append(countZero(4,1,table2))
```

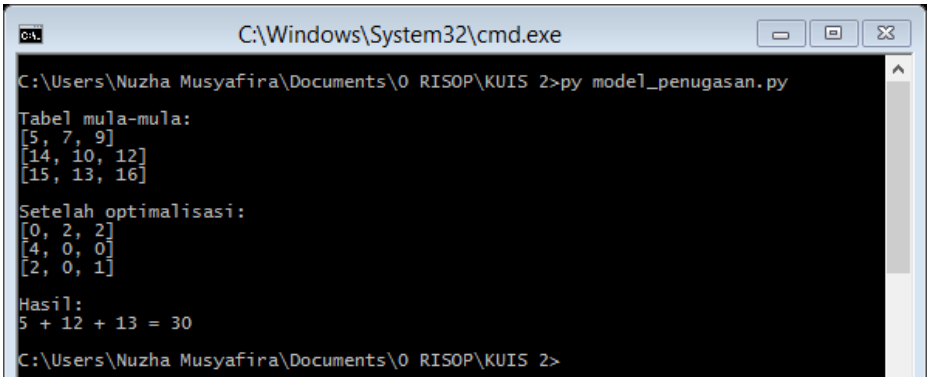
```
            sortZ=sortNumZero(numZ)
            crossZero(table2,sortZ,0)
        else:
            for s in table2:
                for x in range(len(s)):
                    if s[x]=='#':
                        s[x]=0

    def costs(x,table2):
        flag=[]
        for i in range(x):
            flag.append(0)
        temp2=[]
        while sum(flag)<=x:
            c=[]
            for a in table2:
                c.append(a.count(0))
            if c.count(1)==0:
                break
            temp=[]
            idx=c.index(1)
            i=idx
            j=table2[idx].index(0)
            flag[j]=1
            for f in range(x):
                if table2[f][j]==0:
                    table2[f][j]='*'
            c[idx]=0
            table2[i][j]='*'
            temp.append(i)
            temp.append(j)
            temp2.append(temp)
        for s in table2:
            for x in range(len(s)):
                if s[x]=='*':
                    s[x]=0
        temp2=sorted(temp2)
        return temp2

    table=[
           [1,4,6,3],
           [9,7,10,9],
           [4,5,11,7],
           [8,7,8,5],
          ]
    lenRow=len(table)
    lenCol=len(table[0])
    isEqual(lenRow,lenCol)
    import copy
```

```
        copyTable=copy.deepcopy(table)
        minVal=minValue(lenRow,0,table)
        toZero(lenRow,0,minVal)
        minVal=minValue(lenCol,1,table)
        toZero(lenCol,1,minVal)
        numZero=[]
        numZero.append(countZero(lenRow,0,table))
        numZero.append(countZero(lenCol,1,table))
        sortedZero=sortNumZero(numZero)
        crossZero(table,sortedZero,0)
        result=costs(lenRow,table)
        print("\nTabel mula-mula:")
        for i in copyTable:
            print(i)
        print("\nSetelah optimalisasi:")
        for i in table:
            print(i)
        print("\nHasil:")
        res=0
        for i in range(len(result)):
            res+=copyTable[result[i][0]][result[i][1]]
            if i==len(result)-1:

        print(copyTable[result[i][0]][result[i][1]],"=",re
        s)
            else:
           print(copyTable[result[i][0]][result[i][1]],"+
",end='')
```

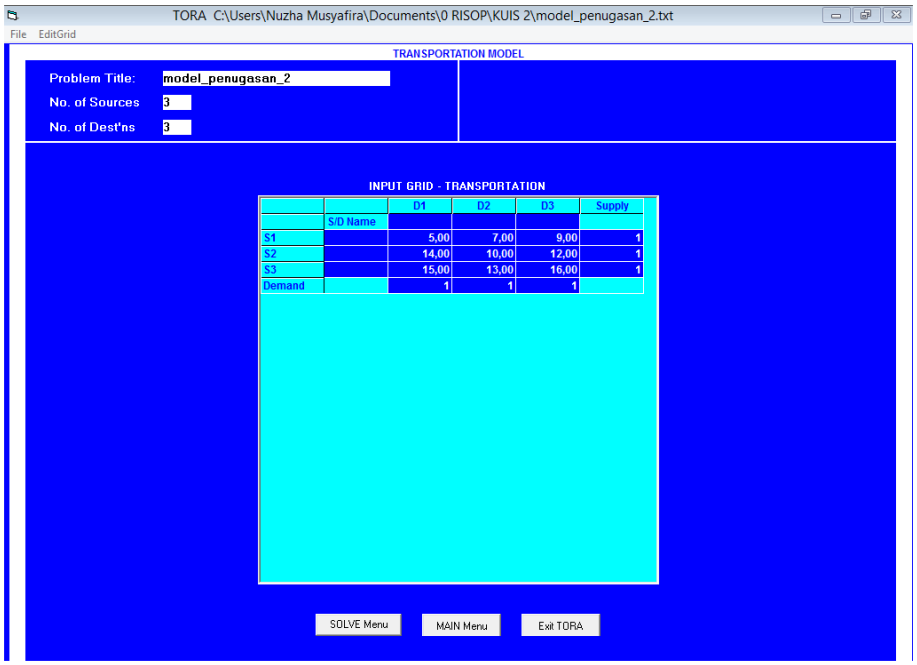
V. Studi Kasus II

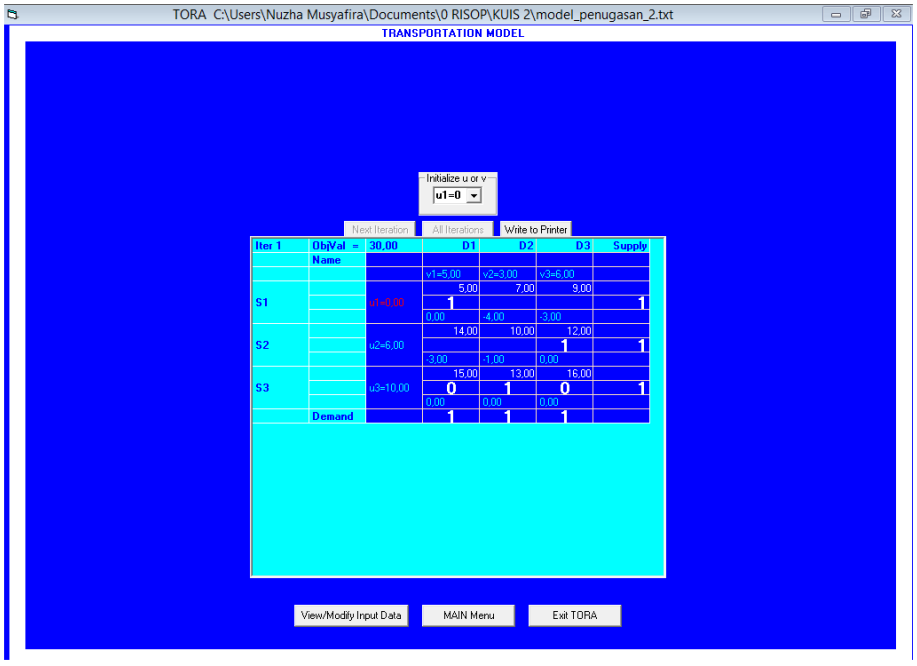


I. Hasil Output


C:\Windows\System32\cmd.exe
C:\Users\Nuzha Musyafira\Documents\0 RISOP\KUIS 2>py model_penugasan.py
Tabel mula-mula:
[5, 7, 9]
[14, 10, 12]
[15, 13, 16]
Setelah optimalisasi:
[0, 2, 2]
[4, 0, 0]
[2, 0, 1]
Hasil:
5 + 12 + 13 = 30
C:\Users\Nuzha Musyafira\Documents\0 RISOP\KUIS 2>

II. TORA


TORA C:\Users\Nuzha Musyafira\Documents\0 RISOP\KUIS 2\model_penugasan_2.txt
File EditGrid
TRANSPORTATION MODEL
Problem Title: model_penugasan_2
No. of Sources 3
No. of Dest'ns 3
INPUT GRID - TRANSPORTATION
S/D Name D1 D2 D3 Supply
S1 5,00 7,00 9,00 1
S2 14,00 10,00 12,00 1
S3 15,00 13,00 16,00 1
Demand 1 1 1
SOLVE Menu
MAIN Menu
Exit TORA


TORA C:\Users\Nuzha Musyafira\Documents\0 RISOP\KUIS 2\model_penugasan_2.txt
TRANSPORTATION MODEL
Initialize u or v
u1=0
Next Iteration
All Iterations
Write to Printer
Iter 1 ObjVal = 30,00 D1 D2 D3 Supply
Name
v1=5,00 v2=3,00 v3=6,00
S1 u1=0,00 5,00 7,00 9,00 1 0,00 -4,00 -3,00
S2 u2=6,00 14,00 10,00 12,00 1 1 -3,00 -1,00 0,00
S3 u3=10,00 15,00 13,00 16,00 0 1 0 1 0,00 0,00 0,00
Demand 1 1 1
View/Modify Input Data
MAIN Menu
Exit TORA

III. Analisis Algoritma

Permasalahan ini mula-mula dipetakan dalam bentuk tabel penugasan dengan sejumlah i baris dan j kolom. Kemudian, memastikan semua baris dan kolom sudah memiliki nilai 0 dengan cara mengurangi setiap elemen pada baris atau kolom dengan nilai elemen terkecil pada setiap baris atau kolom yang bersangkutan. Setelah itu, ditarik garis yang melalui nilai 0 dengan jumlah lebih banyak, dengan catatan apabila ada kesamaan antara baris dan kolom, maka diutamakan baris. Apabila jumlah garis yang ditarik sama dengan jumlah kolom, maka alokasikan ongkos minimum. Jika tidak, maka ulangi iterasi dengan mencari nilai terkecil dari elemen yang tidak dilewati garis, kemudian dikurangkan terhadapnya, dan ditambah terhadap elemen yang dilewati persimpangan garis.

IV. Source Code

```
    def isEqual(lenRow,lenCol):
        global table
        if lenRow<lenCol:
            temp=[]
            for x in range(lenCol):
                temp.append(0)
            for x in range(lenCol - lenRow):
                table.append(temp)
            lenRow=lenCol
        elif lenRow>lenCol:
            for x in range(lenRow):
                for y in range(lenRow - lenCol):
                    table[x].append(0)
            lenCol=lenRow

    def sortNumZero(arr):
        val=len(arr)-1
        temp=[]
        for a in arr:
            for i in range(len(a)):
                temp2=[]
                temp2.append(a[i])
                temp2.append(val)
                temp2.append(i)
                temp.append(temp2)
            val-=1
        temp.sort(key=lambda z: (-z[0], -z[1]))
        return temp

    def countZero(x,y,table2):
        if y==1:
            table2=list(zip(*table2))
        temp=[]
```

```
        for i in range(x):
            temp.append(table2[i].count(0))
        return temp

    def toZero(x,y,min):
        for i in range(x):
            for j in range(x):
                if y==1:
                    table[j][i]=table[j][i]-min[i]
                else:
                    table[i][j]=table[i][j]-min[i]

    def minValue(x,y,table2):
        if y==1:
            table2=list(zip(*table2))
        temp=[]
        for i in range(x):
            temp.append(min(table2[i]))
        return temp

    def crossZero(table2,sortZ,counter):
        row=[]
        col=[]
        for i in range(len(table2)):
            row.append(0)
            col.append(0)
        for i in sortZ:
            if i[1]==1:
                if table2[i[2]].count(0)>0:
                    for j in range(len(table2[i[2]])):
                        if table2[i[2]][j]==0:
                            table2[i[2]][j]='#'
                    counter+=1
                    row[i[2]]=1
            else:
                temp=list(zip(*table2))
                if temp[i[2]].count(0)>0:
                    for j in range(len(temp[i[2]])):
                        if table2[j][i[2]]==0:
                            table2[j][i[2]]='#'
                    counter+=1
                    col[i[2]]=1
        if counter<len(table2):
            temp2=[]
            for s in range(len(table2)):
                for x in range(len(table2)):
                    if row[s]==0 and col[x]==0:
                        temp2.append(table2[s][x])
            newMin=min(temp2)
```

```
            for s in range(len(table2)):
                for x in range(len(table2)):
                    if table2[s][x]=='#':
                        table2[s][x]=0
                    if row[s]==0 and col[x]==0:
                        table2[s][x]-=newMin
                    elif row[s]==1 and col[x]==1:
                        table2[s][x]+=newMin
            numZ=[]
            numZ.append(countZero(4,0,table2))
            numZ.append(countZero(4,1,table2))
            sortZ=sortNumZero(numZ)
            crossZero(table2,sortZ,0)
        else:
            for s in table2:
                for x in range(len(s)):
                    if s[x]=='#':
                        s[x]=0

    def costs(x,table2):
        flag=[]
        for i in range(x):
            flag.append(0)
        temp2=[]
        while sum(flag)<=x:
            c=[]
            for a in table2:
                c.append(a.count(0))
            if c.count(1)==0:
                break
            temp=[]
            idx=c.index(1)
            i=idx
            j=table2[idx].index(0)
            flag[j]=1
            for f in range(x):
                if table2[f][j]==0:
                    table2[f][j]='*'
            c[idx]=0
            table2[i][j]='*'
            temp.append(i)
            temp.append(j)
            temp2.append(temp)
        for s in table2:
            for x in range(len(s)):
                if s[x]=='*':
                    s[x]=0
        temp2=sorted(temp2)
        return temp2
```

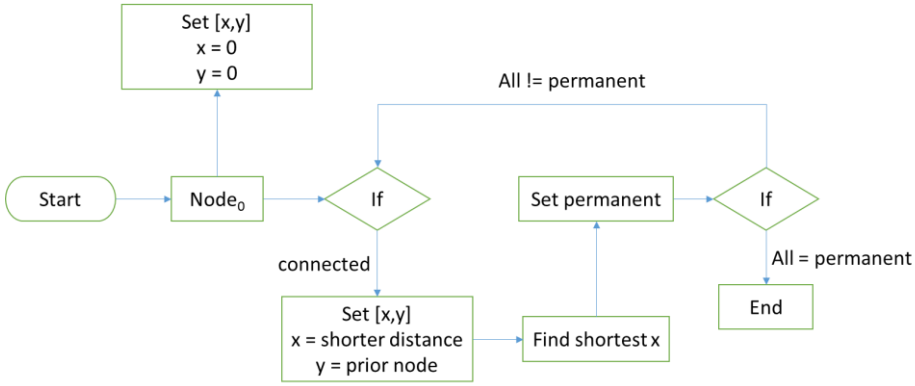
```
      table=[
             [5,7,9],
             [14,10,12],
             [15,13,16],
            ]
      lenRow=len(table)
      lenCol=len(table[0])
      isEqual(lenRow,lenCol)
      import copy
      copyTable=copy.deepcopy(table)
      minVal=minValue(lenRow,0,table)
      toZero(lenRow,0,minVal)
      minVal=minValue(lenCol,1,table)
      toZero(lenCol,1,minVal)
      numZero=[]
      numZero.append(countZero(lenRow,0,table))
      numZero.append(countZero(lenCol,1,table))
      sortedZero=sortNumZero(numZero)
      crossZero(table,sortedZero,0)
      result=costs(lenRow,table)
      print("\nTabel mula-mula:")
      for i in copyTable:
          print(i)
      print("\nSetelah optimalisasi:")
      for i in table:
          print(i)
      print("\nHasil:")
      res=0
      for i in range(len(result)):
          res+=copyTable[result[i][0]][result[i][1]]
          if i==len(result)-1:

      print(copyTable[result[i][0]][result[i][1]],"=",re
      s)
          else:
            print(copyTable[result[i][0]][result[i][1]],"+
",end='')
```

ALGORITMA DIJKSTRA

I. Algoritma dan Penjelasan

Algoritma Dijkstra merupakan algoritma *greedy* untuk menentukan jarak terpendek dari titik awal ke setiap titik lain dalam grafik yang memiliki bobot (*weighted graph*).

II. Diagram Alur



III. Pseudocode

def dijkstra(matrix,s,d,n):	
1	r=[]
2	for i in range(n):
3	temp=[]
4	temp.append(0)
5	temp.append(0)
6	r.append(temp)
7	s-=1
8	d-=1
9	dist=[]
10	setPermanent=[]
11	for i in range(n):
12	dist.append(10000)
13	setPermanent.append(0)
14	dist[s]=0
15	b=0
16	for i in range(n-1):
17	a=minDistance(dist,setPermanent,n)
18	setPermanent[a]=1
19	r[a][1]=b
20	b=a
21	for j in range(n):
22	if setPermanent[j]==0 and matrix[a][j]>0 and dist[a]!=10000 and dist[a]+matrix[a][j]<dist[j]:

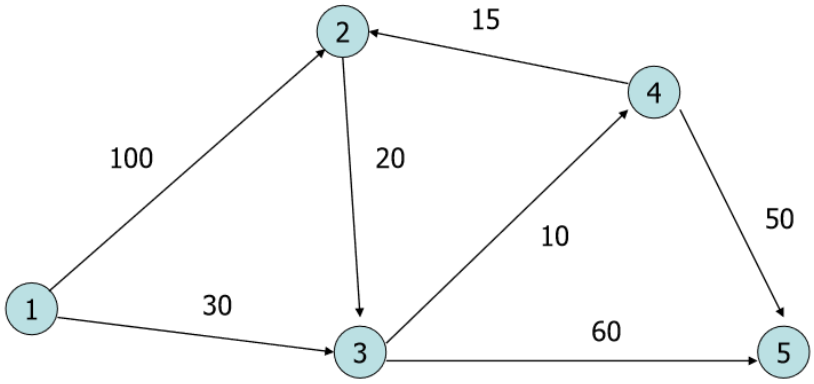
```
23          dist[j]=dist[a]+matrix[a][j]
24      r[a][0]=dist[a]
25  return r
```

Fungsi di atas adalah fungsi untuk memproses algoritma Dijkstra. Parameter matrix adalah array yang berisi *adjacency matrix* yang melambangkan hubungan dan jarak antar *node*, s adalah *source node*, d adalah *destination node*, dan n adalah jumlah seluruh *node*.

```
def minDistance(dist2,set,x):
1   minimum=10000
2   idx=0
3   for i in range(x):
4       if set[i]==0 and dist2[i]<=minimum:
5           minimum=dist2[i]
6           idx=i
7   return idx
```

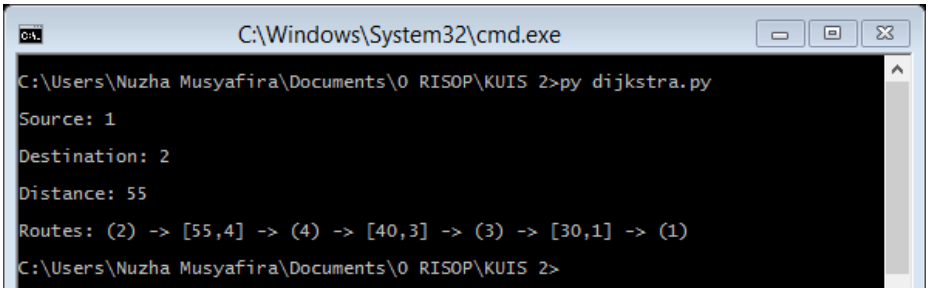
Fungsi di atas adalah fungsi untuk mencari jarak terdekat antar *node*. Parameter dist2 adalah *array* yang menyimpan jarak, set adalah *array* yang menyimpan apakah permanen atau tidak, dan x adalah jumlah seluruh *node*.

IV. Studi Kasus

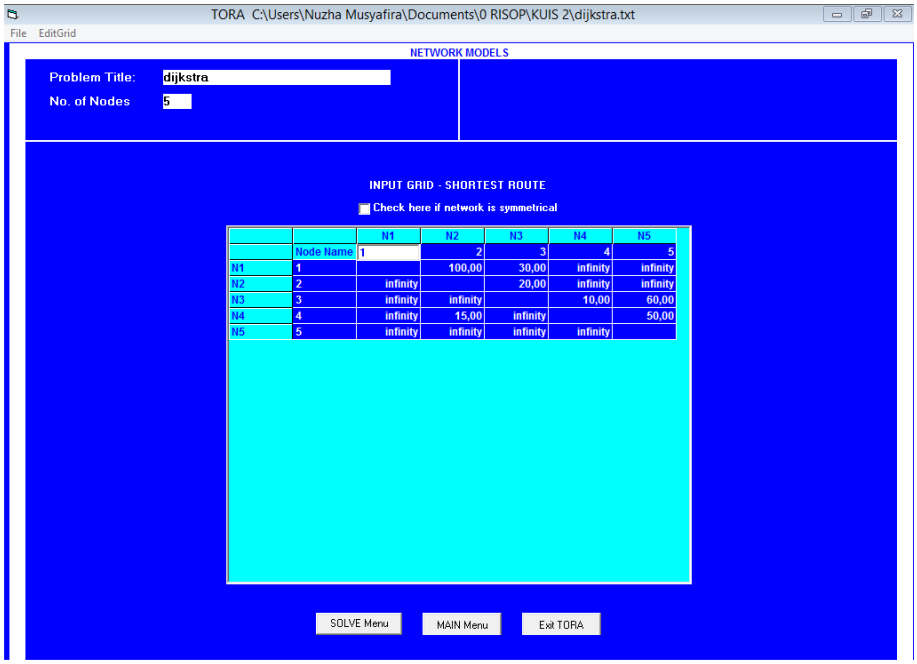


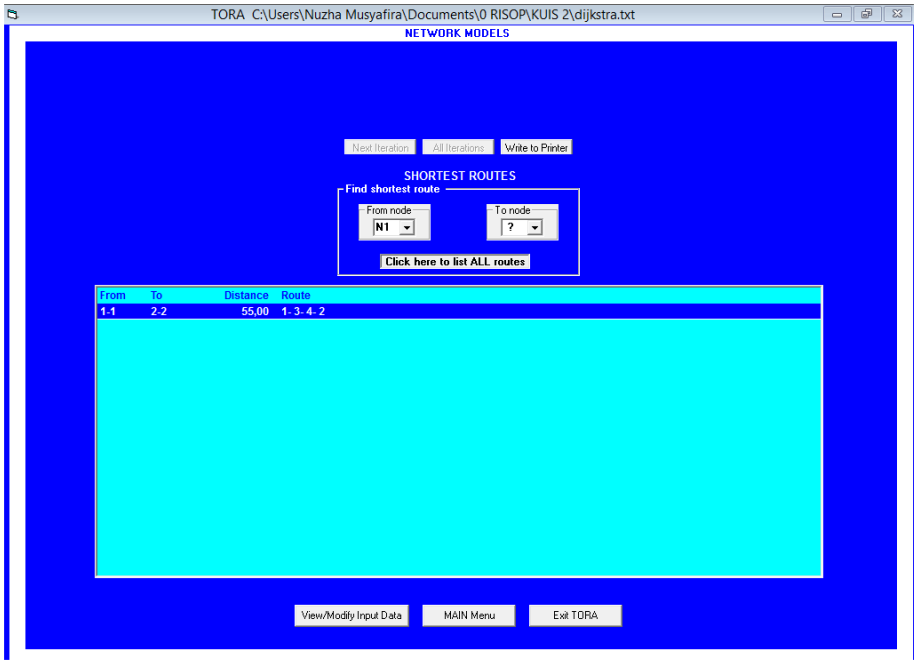
Misal untuk mengetahui rute terpendek dari node 1 ke node 2 dari contoh gambar jaringan di atas.

I. Hasil Output



II. TORA





III. Analisis Algoritma

Algoritma merupakan algoritma pencari jarak yang *greedy*. Mula-mula, node awal ditandai dengan [0,-] dan diberi status permanen karena ke dirinya sendiri. Kemudian, lakukan iterasi untuk setiap *node* yang dapat berhubungan, kemudian set jarak dan *node* sebelumnya, dan beri status sementara. Lakukan pencarian jarak hingga jarak terkecil kemudian baru nyatakan status sebagai permanen. Apabila semua status *node* sudah permanen, maka pencarian telah selesai. Jika tidak, maka ulangi iterasi hingga semua menjadi permanen.

IV. Source Code

```
      def minDistance(dist2,set,x):
            minimum=10000
            idx=0
            for i in range(x):
                  if set[i]==0 and dist2[i]<=minimum:
                        minimum=dist2[i]
                        idx=i
            return idx

      def dijkstra(matrix,s,d,n):
            r=[]
            for i in range(n):
                  temp=[]
                  temp.append(0)
                  temp.append(0)
                  r.append(temp)
            s-=1
            d-=1
            dist=[]
            setPermanent=[]
            for i in range(n):
                  dist.append(10000)
                  setPermanent.append(0)
            dist[s]=0
            b=0
            for i in range(n-1):
                  a=minDistance(dist,setPermanent,n)
                  setPermanent[a]=1
                  r[a][1]=b
                  b=a
                  for j in range(n):
                        if     setPermanent[j]==0     and
      matrix[a][j]>0     and     dist[a]!=10000     and
      dist[a]+matrix[a][j]<dist[j]:
                              dist[j]=dist[a]+matrix[a][j]
                  r[a][0]=dist[a]
            return r

      adjMatrix = [
                     [0,100,30,0,0],
                     [0,0,20,0,0],
                     [0,0,0,10,60],
                     [0,15,0,0,50],
                     [0,0,0,0,0]
                    ]
      nodes=len(adjMatrix)
      src=1
      dst=2
```

```
      route=dijkstra(adjMatrix,src,dst,nodes)
      print("\nSource:", src)
      print("\nDestination:", dst)
      print("\nDistance:", route[dst-1][0])
      print("\nRoutes: ", end='')
      while(dst>src):
            print("(%d)" %dst, "-> ", end='')
            x=route[dst-1][0]
            y=route[dst-1][1]+1
            print("[%d" %x, end='')
            print(",%d" %y, end='')
            print("] -> ", end='')
            dst=y
print("(%d)" %src)
```