LAPORAN
UJIAN AKHIR SEMESTER:
PROGRAM KOMPUTER DAN ANALISIS KASUS ALGORITMA FLOYD

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 Ujian Akhir Semester (UAS) 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, 04 Desember 2018

Nuzha Musyafira
051116 4000 0014

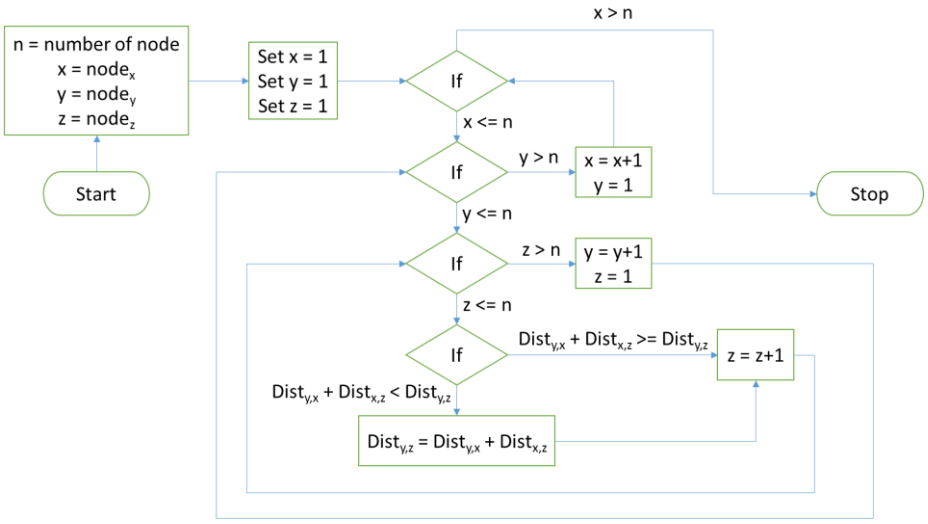
DAFTAR ISI

ALGORITMA FLOYD

I. Algoritma dan Penjelasan

Algoritma Floyd merupakan algortima yang digunakan untuk mencari rute terpendek antara 2 node dalam suatu network. Algoritma Floyd lebih general dibanding Dijkstra karena menghitung rute terpendek di antara semua node yang ada.

II. Diagram Alur



III. Pseudocode

```
def floyd(m2,n2):
1    d=[]
2    p=[]
3    inf=1000
4    for x in m2:
5        temp=[]
6        temp2=[]
7        for y in x:
8            temp.append(y)
9            temp2.append(-1)
10       d.append(temp)
11       p.append(temp2)
12   for x in range(n2):
13       for y in range(n2):
14           if x==y:
15               p[x][y]=0
16           elif d[x][y]!=inf:
17               p[x][y]=x
18           else:
19               p[x][y]=-1
```

```
20  for x in range(n2):
21      for y in range(n2):
22          for z in range(n2):
23              if d[y][x]+d[x][z]<d[y][z]:
24                  d[y][z]=d[y][x]+d[x][z]
25                  p[y][z]=p[x][z]
26  return d,p
```

Fungsi di atas adalah fungsi untuk menemukan rute dengan jarak terpendek menggunakan algoritma Floyd. Parameter m2 merupakan matriks yang menyimpan tabel jarak mula-mula dengan ukuran n2 x n2 yang juga didapat dari parameter yang lainnya. Fungsi ini akan mengembalikan (*return*) matriks d yang berisi jarak (*distance*) terpendek dan p yang berisi *initial path* antar node.

```
def printAll(p2,n2):
1   print("")
2   for x in range(n2):
3       for y in range(n2):
4           if x!=y and p2[x][y]!=-1:
5               print("Path dari",x+1,"ke",y+1,"=",x+1,"->",end='')
6               printPath(p2,x,y)
7               print("",y+1)
```

Fungsi di atas adalah fungsi untuk mencetak seluruh rute dari node satu ke node lain, yaitu node selain node diri sendiri dan node yang tidak bisa dicapai. Parameter p2 merupakan matriks yang berisi *initial path* antar node dan n2 adalah ukuran matriks atau dengan kata lain jumlah node. Fungsi ini tidak mengembalikan apa-apa (*void*), namun mencetak rute dengan memanggil fungsi lain untuk mendapatkan *complete path* dari *initial path* p2. Karena indeks tiap node dimulai dari 0, maka variabel x dan y ditambah dengan 1 ketika dicetak.

```
def printPath(p3,x2,y2):
1   if p3[x2][y2]!=x2:
2       printPath(p3,x2,p3[x2][y2])
3       print("",p3[x2][y2]+1,"->",end='')
```

Fungsi di atas adalah fungsi rekursif yang digunakan untuk mencetak *complete path* antara node asal dan node tujuan. Parameter yang digunakan yaitu p3 yang merupakan matriks berisi *initial path*, x2 sebagai indeks node asal, dan y2 sebagai indeks node tujuan. Karena indeks tiap node dimulai dari 0, maka variabel p3[x2][y2] ditambah dengan 1 ketika dicetak.

IV. Studi Kasus I

For the network in figure 6.21 find the shortest route between every two nodes. The distances (in miles) are given on the arcs. Arc (3,5) is directional so that no trafic is alowed from node 5 to node 3. All the other arcs allow trafic in both directions.

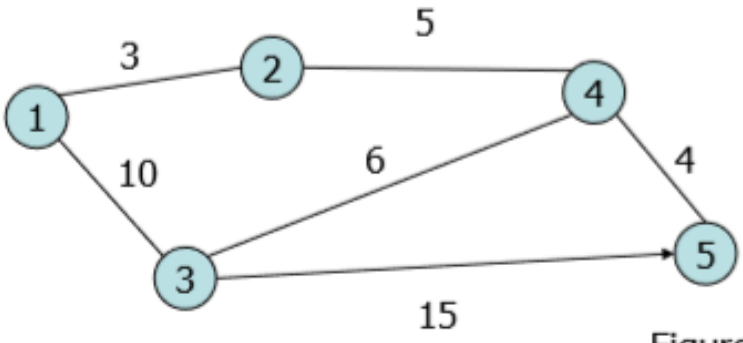
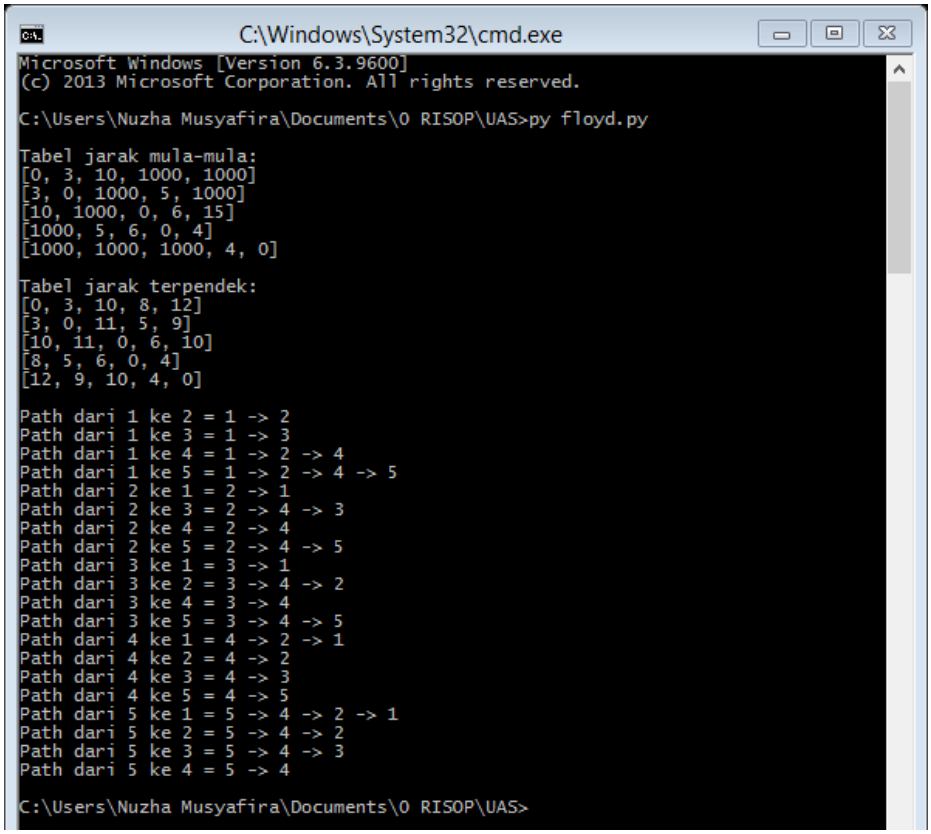
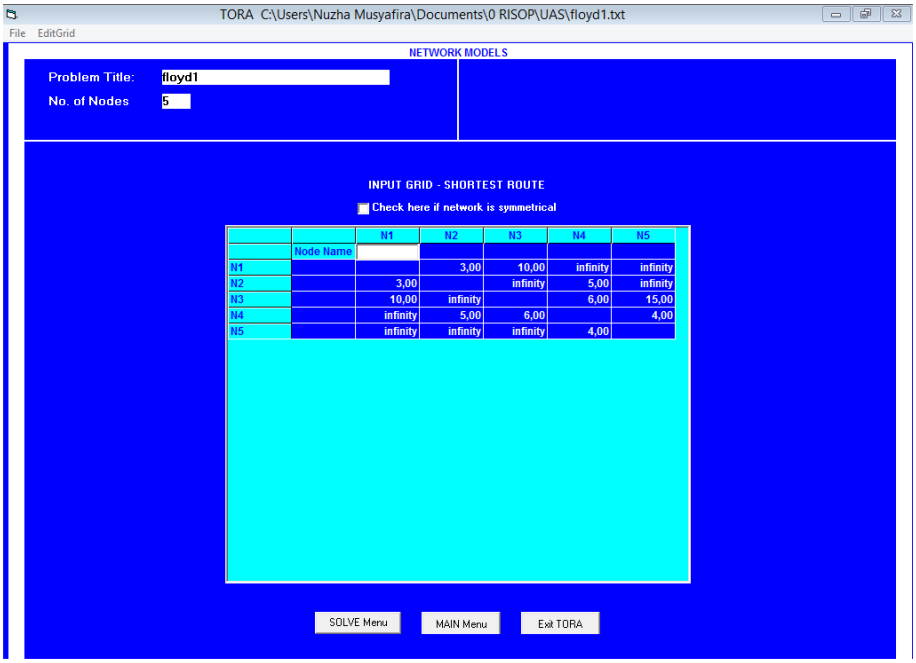


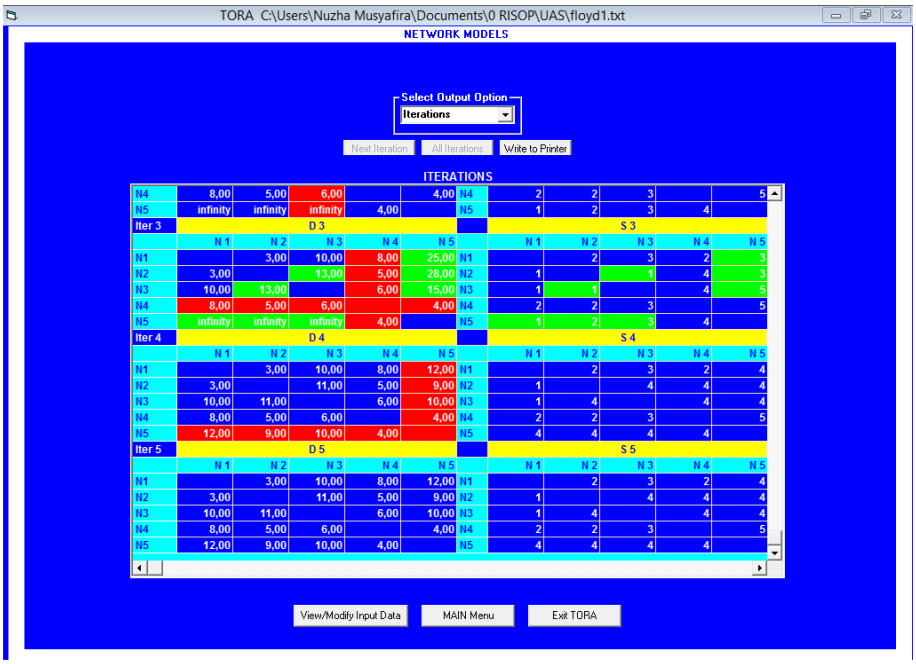
Figure 6.21

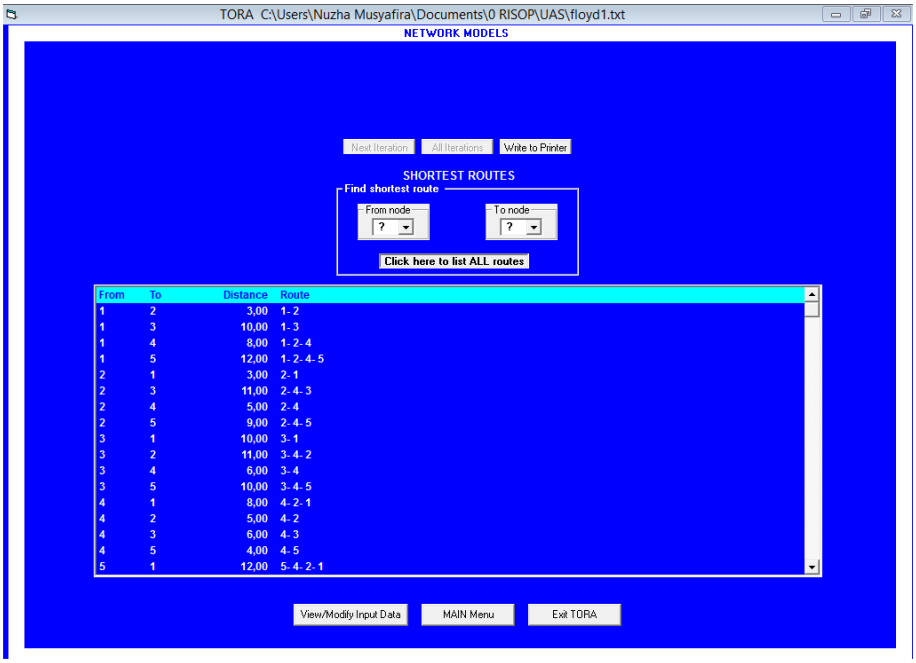
I. Hasil Output

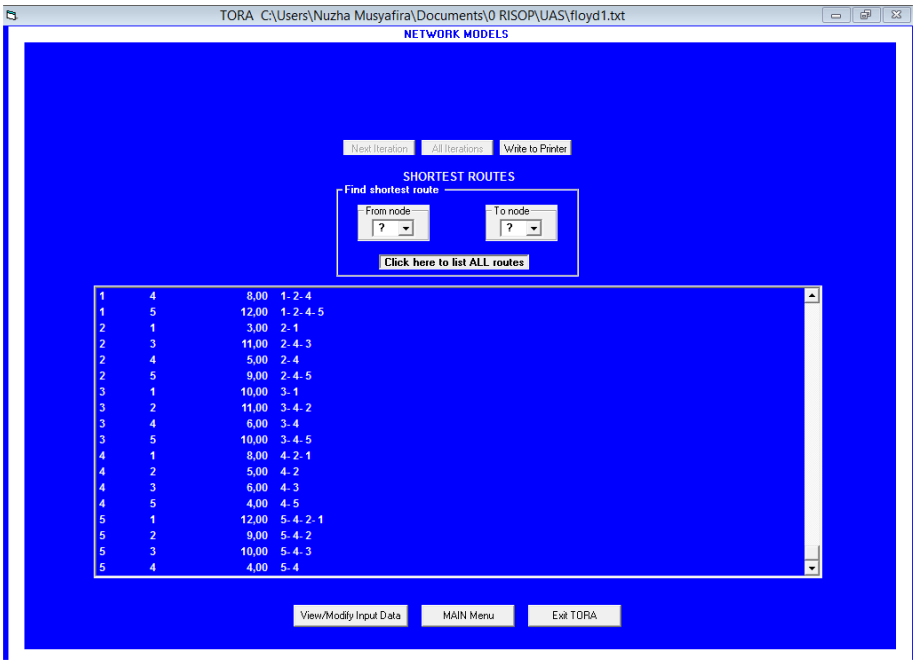


II. TORA


TORA C:\Users\Nuzha Musyafira\Documents\0 RISOP\UAS\floyd1.txt
File EditGrid
NETWORK MODELS
Problem Title: floyd1
No. of Nodes 5
INPUT GRID - SHORTEST ROUTE
Check here if network is symmetrical
Node Name
N1 N2 N3 N4 N5
N1 3,00 10,00 infinity infinity
N2 3,00 infinity 5,00 infinity
N3 10,00 infinity 6,00 15,00
N4 infinity 5,00 6,00 4,00
N5 infinity infinity infinity 4,00
SOLVE Menu
MAIN Menu
Exit TORA


TORA C:\Users\Nuzha Musyafira\Documents\0 RISOP\UAS\floyd1.txt
NETWORK MODELS
Select Output Option
Iterations
Next Iteration
All Iterations
Write to Printer
ITERATIONS
Iter 3
D 3
S 3
Iter 4
D 4
S 4
Iter 5
D 5
S 5
View/Modify Input Data
MAIN Menu
Exit TORA





III. Analisis Algoritma

Algoritma Floyd merupakan algoritma pencari jarak yang mengdaptasi dari *Floyd's Triangle*. Mula-mula, dibentuk *adjacency matrix* dengan ukuran n x n (n adalah jumlah node) yang menggambarkan *direct distance* antar node. Indeks pada baris mengindikasikan node asal dan indeks pada kolom mengindikasikan node tujuan. Node dengan tujuan adalah dirinya sendiri diset 0, node yang tidak dapat terhubung secara langsung ke node lain diset INF. Setelah itu, lakukan iterasi sebanyak n x n x n, dengan asumsi tiap iterasi mewakili $node_x$, $node_y$, dan $node_z$. Apabila jarak $node_y$ ke $node_x$ ditambah dengan jarak $node_x$ ke $node_z$ kurang dari jarak $node_y$ ke $node_z$, maka jarak $node_y$ ke $node_z$ diset menjadi jarak $node_y$ ke $node_x$ ditambah dengan jarak

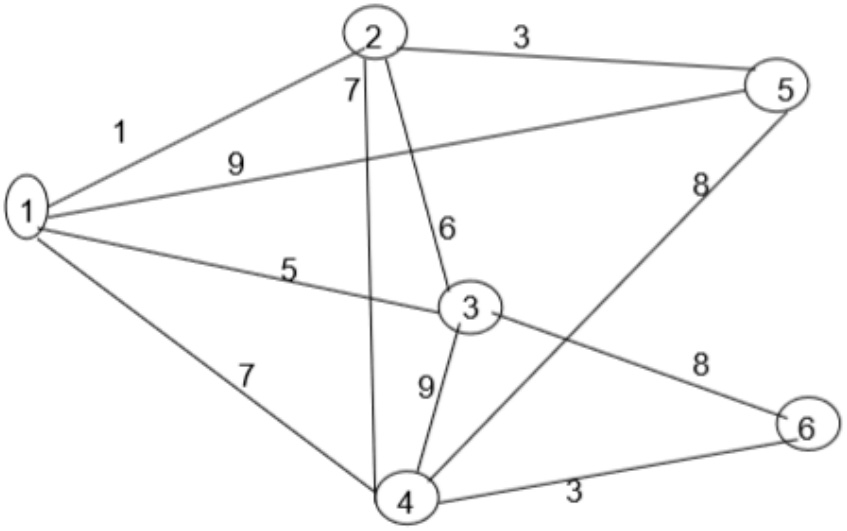
$node_x$ ke $node_z$. Ulangi hingga akhir iterasi untuk mendapatkan rute dengan jarak terpendek.

IV. Source Code

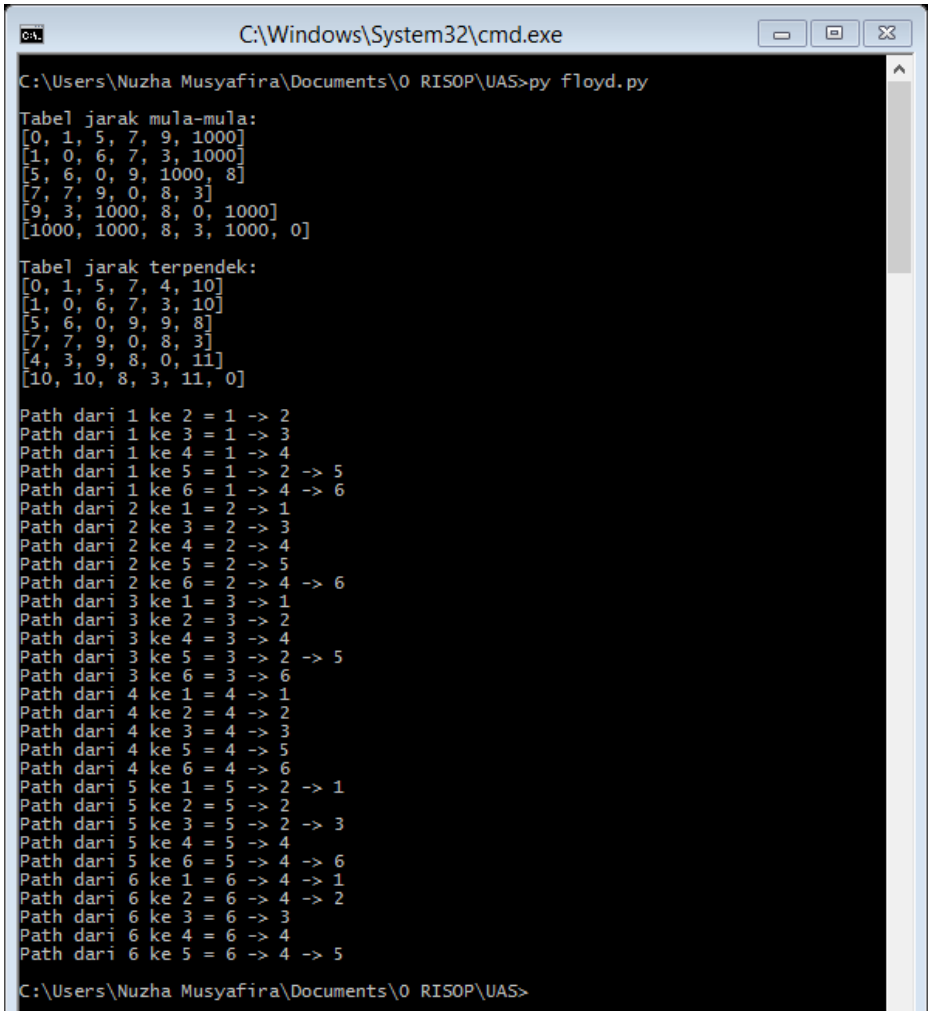
```
def floyd(m2,n2):
    d=[]
    p=[]
    inf=1000
    for x in m2:
        temp=[]
        temp2=[]
        for y in x:
            temp.append(y)
            temp2.append(-1)
        d.append(temp)
        p.append(temp2)
    for x in range(n2):
        for y in range(n2):
            if x==y:
                p[x][y]=0
            elif d[x][y]!=inf:
                p[x][y]=x
            else:
                p[x][y]=-1
    for x in range(n2):
        for y in range(n2):
            for z in range(n2):
                if d[y][x]+d[x][z]<d[y][z]:
                    d[y][z]=d[y][x]+d[x][z]
                    p[y][z]=p[x][z]
    return d,p

def printAll(p2,n2):
    print("")
    for x in range(n2):
        for y in range(n2):
            if x!=y and p2[x][y]!=-1:
                print("Path
dari",x+1,"ke",y+1,"=",x+1,"->",end='')
                printPath(p2,x,y)
                print("",y+1)

def printPath(p3,x2,y2):
    if p3[x2][y2]!=x2:
        printPath(p3,x2,p3[x2][y2])
        print("",p3[x2][y2]+1,"->",end='')

inf=1000
```

```
m=[
   [0,3,10,inf,inf],
   [3,0,inf,5,inf],
   [10,inf,0,6,15],
   [inf,5,6,0,4],
   [inf,inf,inf,4,0]
  ]
n=len(m)
print("\nTabel jarak mula-mula:")
for x in m:
     print(x)
result,path=floyd(m,n)
print("\nTabel jarak terpendek:")
for x in result:
     print(x)
printAll(path,n)
```

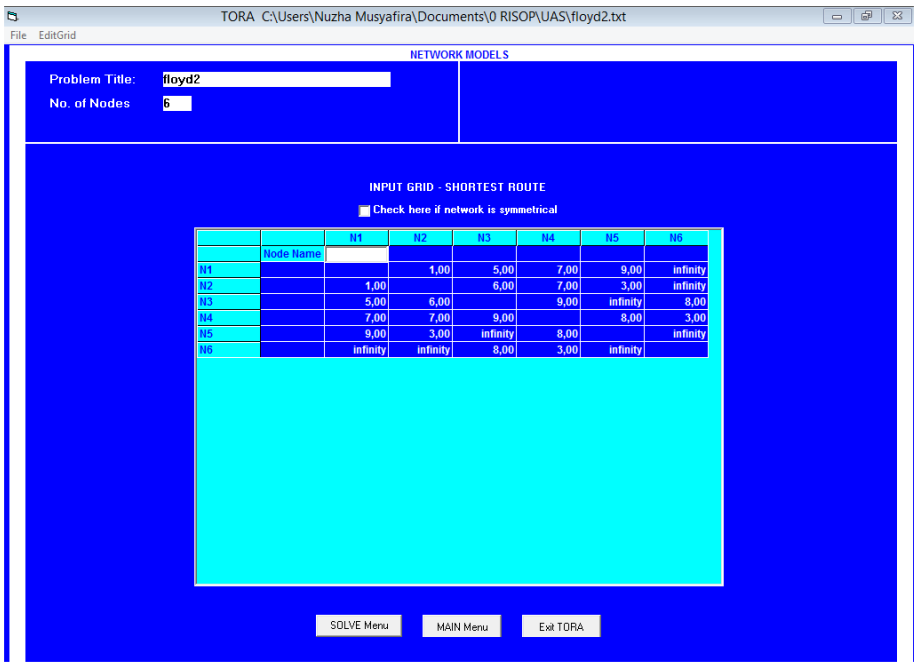
V. Studi Kasus II



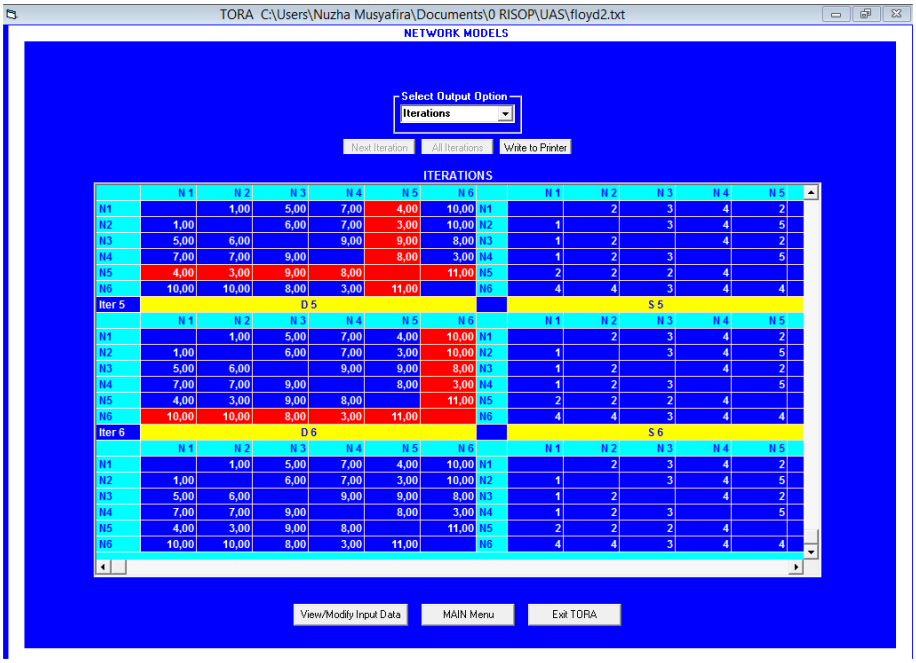
I. Hasil Output

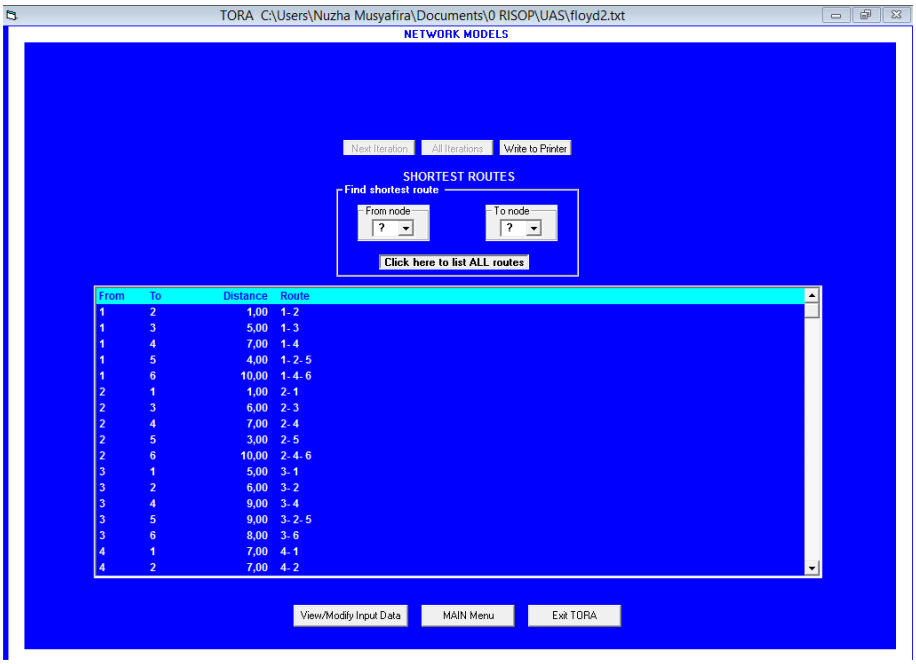


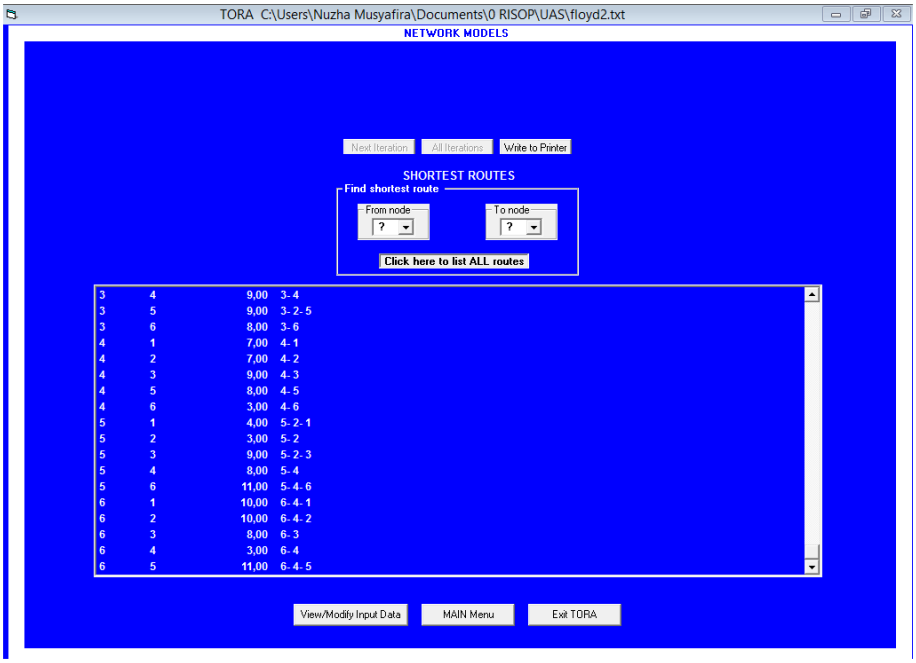
II. TORA



		N1	N2	N3	N4	N5	N6
	Node Name						
N1			1,00	5,00	7,00	9,00	infinity
N2		1,00		6,00	7,00	3,00	infinity
N3		5,00	6,00		9,00	infinity	8,00
N4		7,00	7,00	9,00		8,00	3,00
N5		9,00	3,00	infinity	8,00		infinity
N6		infinity	infinity	8,00	3,00	infinity	


TORA C:\Users\Nuzha Musyafira\Documents\0 RISOP\UAS\floyd2.txt
NETWORK MODELS
Select Output Option
Iterations
Next Iteration
All Iterations
Write to Printer
ITERATIONS
View/Modify Input Data
MAIN Menu
Exit TORA


TORA C:\Users\Nuzha Musyafira\Documents\0 RISOP\UAS\floyd2.txt
NETWORK MODELS
Next Iteration
All Iterations
Write to Printer
SHORTEST ROUTES
Find shortest route
From node
To node
Click here to list ALL routes
View/Modify Input Data
MAIN Menu
Exit TORA



III. Analisis Algoritma

Algoritma Floyd merupakan algoritma pencari jarak yang mengdaptasi dari *Floyd's Triangle*. Mula-mula, dibentuk *adjacency matrix* dengan ukuran n x n (n adalah jumlah node) yang menggambarkan *direct distance* antar node. Indeks pada baris mengindikasikan node asal dan indeks pada kolom mengindikasikan node tujuan. Node dengan tujuan adalah dirinya sendiri diset 0, node yang tidak dapat terhubung secara langsung ke node lain diset INF. Setelah itu, lakukan iterasi sebanyak n x n x n, dengan asumsi tiap iterasi mewakili $node_x$, $node_y$, dan $node_z$. Apabila jarak $node_y$ ke $node_x$ ditambah dengan jarak $node_x$ ke $node_z$ kurang dari jarak $node_y$ ke $node_z$, maka jarak $node_y$ ke $node_z$ diset menjadi jarak $node_y$ ke $node_x$ ditambah dengan jarak $node_x$ ke $node_z$. Ulangi hingga akhir iterasi untuk mendapatkan rute dengan jarak terpendek.

IV. Source Code

```
def floyd(m2,n2):
    d=[]
    p=[]
    inf=1000
    for x in m2:
        temp=[]
        temp2=[]
        for y in x:
            temp.append(y)
            temp2.append(-1)
        d.append(temp)
        p.append(temp2)
    for x in range(n2):
```

```
                for y in range(n2):
                        if x==y:
                                p[x][y]=0
                        elif d[x][y]!=inf:
                                p[x][y]=x
                        else:
                                p[x][y]=-1
        for x in range(n2):
                for y in range(n2):
                        for z in range(n2):
                                if d[y][x]+d[x][z]<d[y][z]:
                                        d[y][z]=d[y][x]+d[x][z]
                                        p[y][z]=p[x][z]
        return d,p

def printAll(p2,n2):
        print("")
        for x in range(n2):
                for y in range(n2):
                        if x!=y and p2[x][y]!=-1:
                                print("Path
dari",x+1,"ke",y+1,"=",x+1,"->",end='')
                                printPath(p2,x,y)
                                print("",y+1)

def printPath(p3,x2,y2):
        if p3[x2][y2]!=x2:
                printPath(p3,x2,p3[x2][y2])
                print("",p3[x2][y2]+1,"->",end='')

inf=1000
m=[
   [0,1,5,7,9,inf],
   [1,0,6,7,3,inf],
   [5,6,0,9,inf,8],
   [7,7,9,0,8,3],
   [9,3,inf,8,0,inf],
   [inf,inf,8,3,inf,0]
  ]
n=len(m)
print("\nTabel jarak mula-mula:")
for x in m:
        print(x)
result,path=floyd(m,n)
print("\nTabel jarak terpendek:")
for x in result:
        print(x)
printAll(path,n)
```