

**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

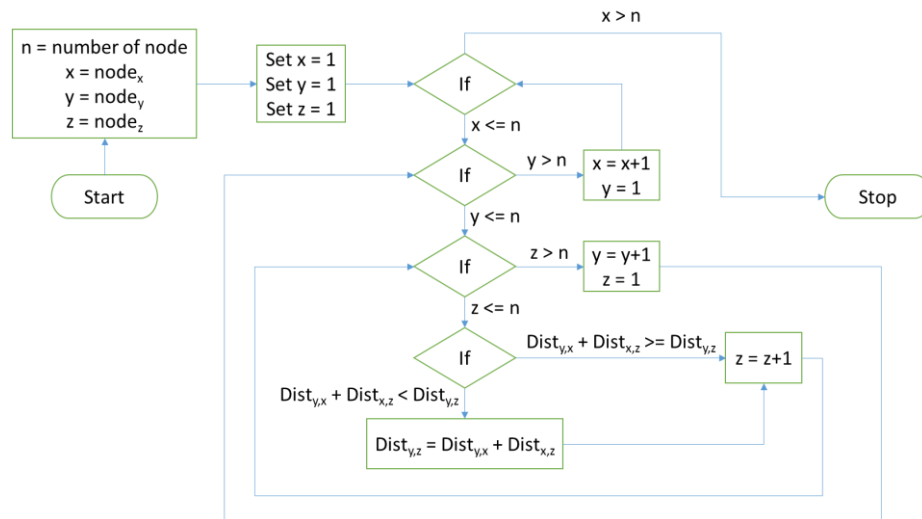
Daftar Isi .....	3
Algoritma Floyd .....	4
I. Algoritma dan Penjelasan.....	4
II. Diagram Alur .....	4
III. Pseudocode .....	4
IV. Studi Kasus I .....	5
I. Hasil Output .....	6
II. TORA.....	6
III. Analisis Algoritma.....	8
IV. Source Code .....	9
V. Studi Kasus II .....	10
I. Hasil Output .....	10
II. TORA.....	11
III. Analisis Algoritma.....	13
IV. Source Code .....	13

# ALGORITMA FLOYD

## I. Algoritma dan Penjelasan

Algoritma Floyd merupakan algoritma 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 traffic is allowed from node 5 to node 3. All the other arcs allow traffic in both directions.

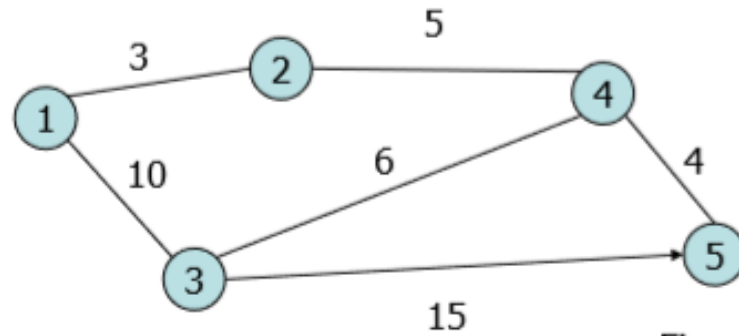


Figure 6.21

## I. Hasil Output

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Nuzha Musyafira\Documents\0 RISOP\UAS>py floyd.py

Tabel jarak mula-mula:
[0, 3, 10, 1000, 1000]
[3, 0, 1000, 5, 1000]
[10, 1000, 0, 6, 15]
[1000, 5, 6, 0, 4]
[1000, 1000, 1000, 4, 0]

Tabel jarak terpendek:
[0, 3, 10, 8, 12]
[3, 0, 11, 5, 9]
[10, 11, 0, 6, 10]
[8, 5, 6, 0, 4]
[12, 9, 10, 4, 0]

Path dari 1 ke 2 = 1 -> 2
Path dari 1 ke 3 = 1 -> 3
Path dari 1 ke 4 = 1 -> 2 -> 4
Path dari 1 ke 5 = 1 -> 2 -> 4 -> 5
Path dari 2 ke 1 = 2 -> 1
Path dari 2 ke 3 = 2 -> 4 -> 3
Path dari 2 ke 4 = 2 -> 4
Path dari 2 ke 5 = 2 -> 4 -> 5
Path dari 3 ke 1 = 3 -> 1
Path dari 3 ke 2 = 3 -> 4 -> 2
Path dari 3 ke 4 = 3 -> 4
Path dari 3 ke 5 = 3 -> 4 -> 5
Path dari 4 ke 1 = 4 -> 2 -> 1
Path dari 4 ke 2 = 4 -> 2
Path dari 4 ke 3 = 4 -> 3
Path dari 4 ke 5 = 4 -> 5
Path dari 5 ke 1 = 5 -> 4 -> 2 -> 1
Path dari 5 ke 2 = 5 -> 4 -> 2
Path dari 5 ke 3 = 5 -> 4 -> 3
Path dari 5 ke 4 = 5 -> 4

C:\Users\Nuzha Musyafira\Documents\0 RISOP\UAS>
  
```

## 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

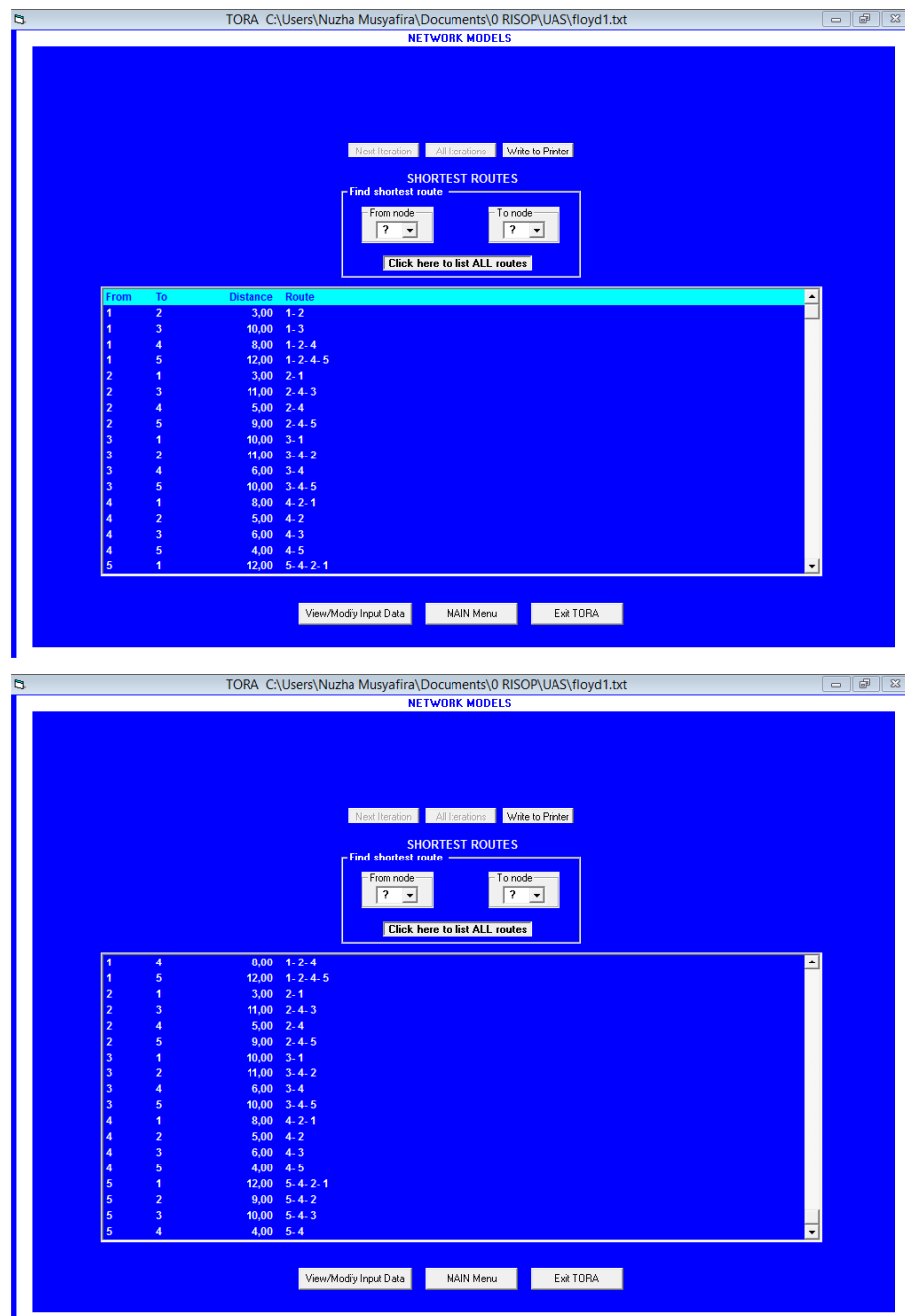
Select Output Option  
Iterations

Next Iterations All Iterations Write to Printer

ITERATIONS

	N1	N2	N3	N4	N5		N1	N2	N3	N4	N5
N4	8,00	5,00	6,00				2	2	3		5
N5	infinity	infinity	infinity	4,00			1	2	3	4	
Iter 3											
	N1	N2	N3	N4	N5		N1	N2	N3	N4	N5
N1		3,00	10,00	8,00	25,00			2	3	2	3
N2	3,00		13,00	5,00	28,00		1		4	4	3
N3	10,00	13,00		6,00	15,00		1	1		4	6
N4	8,00	5,00	6,00		4,00		2	2	3		5
N5	infinity	infinity	infinity	4,00			1	2	3	4	
Iter 4											
	N1	N2	N3	N4	N5		N1	N2	N3	N4	N5
N1		3,00	10,00	8,00	12,00			2	3	2	4
N2	3,00		11,00	5,00	9,00				4	4	4
N3	10,00	11,00		6,00	10,00		1	4		4	4
N4	8,00	5,00	6,00		4,00		2	2	3		5
N5	12,00	9,00	10,00	4,00			4	4	4	4	
Iter 5											
	N1	N2	N3	N4	N5		N1	N2	N3	N4	N5
N1		3,00	10,00	8,00	12,00			2	3	2	4
N2	3,00		11,00	5,00	9,00				4	4	4
N3	10,00	11,00		6,00	10,00		1	4		4	4
N4	8,00	5,00	6,00		4,00		2	2	3		5
N5	12,00	9,00	10,00	4,00			4	4	4	4	

View/Modify Input Data MAIN Menu Exit TORA



### III. Analisis Algoritma

Algoritma Floyd merupakan algoritma pencari jarak yang mengadaptasi dari *Floyd's Triangle*. Mula-mula, dibentuk *adjacency matrix* dengan ukuran  $n \times n$  ( $n$  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  $n \times n \times n$ , 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<sub>x</sub> ke node<sub>z</sub>. 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='')

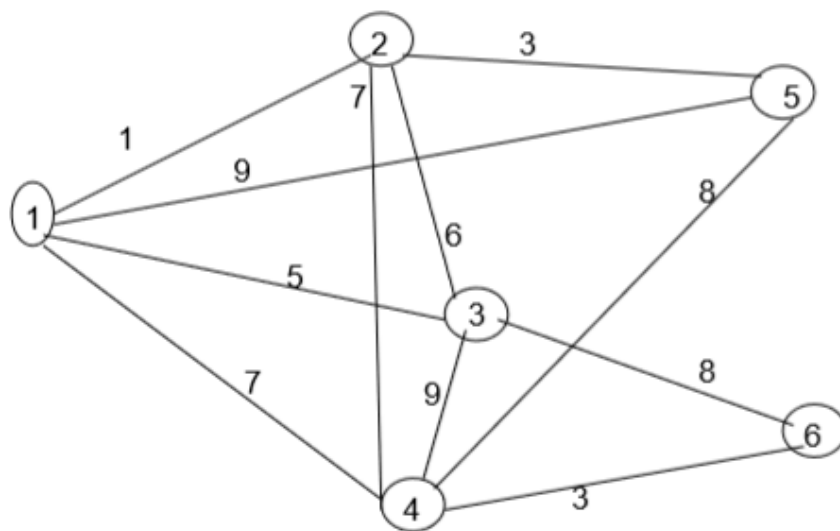
inf=1000
```

```

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

```

C:\Windows\System32\cmd.exe

C:\Users\Nuzha Musyafira\Documents\0 RISOP\UAS>py floyd.py

Tabel jarak mula-mula:
[0, 1, 5, 7, 9, 1000]
[1, 0, 6, 7, 3, 1000]
[5, 6, 0, 9, 1000, 8]
[7, 7, 9, 0, 8, 3]
[9, 3, 1000, 8, 0, 1000]
[1000, 1000, 8, 3, 1000, 0]

Tabel jarak terpendek:
[0, 1, 5, 7, 4, 10]
[1, 0, 6, 7, 3, 10]
[5, 6, 0, 9, 9, 8]
[7, 7, 9, 0, 8, 3]
[4, 3, 9, 8, 0, 11]
[10, 10, 8, 3, 11, 0]

Path dari 1 ke 2 = 1 -> 2
Path dari 1 ke 3 = 1 -> 3
Path dari 1 ke 4 = 1 -> 4
Path dari 1 ke 5 = 1 -> 2 -> 5
Path dari 1 ke 6 = 1 -> 4 -> 6
Path dari 2 ke 1 = 2 -> 1
Path dari 2 ke 3 = 2 -> 3
Path dari 2 ke 4 = 2 -> 4
Path dari 2 ke 5 = 2 -> 5
Path dari 2 ke 6 = 2 -> 4 -> 6
Path dari 3 ke 1 = 3 -> 1
Path dari 3 ke 2 = 3 -> 2
Path dari 3 ke 4 = 3 -> 4
Path dari 3 ke 5 = 3 -> 2 -> 5
Path dari 3 ke 6 = 3 -> 6
Path dari 4 ke 1 = 4 -> 1
Path dari 4 ke 2 = 4 -> 2
Path dari 4 ke 3 = 4 -> 3
Path dari 4 ke 5 = 4 -> 5
Path dari 4 ke 6 = 4 -> 6
Path dari 5 ke 1 = 5 -> 2 -> 1
Path dari 5 ke 2 = 5 -> 2
Path dari 5 ke 3 = 5 -> 2 -> 3
Path dari 5 ke 4 = 5 -> 4
Path dari 5 ke 6 = 5 -> 4 -> 6
Path dari 6 ke 1 = 6 -> 4 -> 1
Path dari 6 ke 2 = 6 -> 4 -> 2
Path dari 6 ke 3 = 6 -> 3
Path dari 6 ke 4 = 6 -> 4
Path dari 6 ke 5 = 6 -> 4 -> 5

C:\Users\Nuzha Musyafira\Documents\0 RISOP\UAS>

```

## II. TORA

TORA C:\Users\Nuzha Musyafira\Documents\0 RISOP\UAS\floyd2.txt

File EditGrid

NETWORK MODELS

Problem Title:

No. of Nodes:

INPUT GRID - SHORTEST ROUTE

☐ Check here if network is symmetrical

	Node Name	N1	N2	N3	N4	N5	N6
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	

SOLVE Menu MAIN Menu Exit TORA

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

	N 1	N 2	N 3	N 4	N 5	N 6		N 1	N 2	N 3	N 4	N 5	
N1		1,00		5,00	7,00	4,00	10,00	N1					
N2	1,00		5,00	7,00	3,00	10,00	N2		1		3	4	
N3	5,00		6,00	9,00	9,00	8,00	N3		1	2		4	
N4	7,00	7,00		9,00	8,00	3,00	N4		1	2	3		
N5	4,00	3,00		9,00	8,00	11,00	N5		2	2	2	4	
N6	10,00	10,00		8,00	3,00	11,00	N6		4	4	3	4	
Iter 5	D 5							S 5					
	N 1	N 2	N 3	N 4	N 5	N 6		N 1	N 2	N 3	N 4	N 5	
N1		1,00		5,00	7,00	4,00	10,00	N1					
N2	1,00		5,00	7,00	3,00	10,00	N2		1		3	4	
N3	5,00		6,00	9,00	9,00	8,00	N3		1	2		4	
N4	7,00	7,00		9,00	8,00	3,00	N4		1	2	3		
N5	4,00	3,00		9,00	8,00	11,00	N5		2	2	2	4	
N6	10,00	10,00		8,00	3,00	11,00	N6		4	4	3	4	
Iter 6	D 6							S 6					
	N 1	N 2	N 3	N 4	N 5	N 6		N 1	N 2	N 3	N 4	N 5	
N1		1,00		5,00	7,00	4,00	10,00	N1					
N2	1,00		5,00	7,00	3,00	10,00	N2		1		3	4	
N3	5,00		6,00	9,00	9,00	8,00	N3		1	2		4	
N4	7,00	7,00		9,00	8,00	3,00	N4		1	2	3		
N5	4,00	3,00		9,00	8,00	11,00	N5		2	2	2	4	
N6	10,00	10,00		8,00	3,00	11,00	N6		4	4	3	4	

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

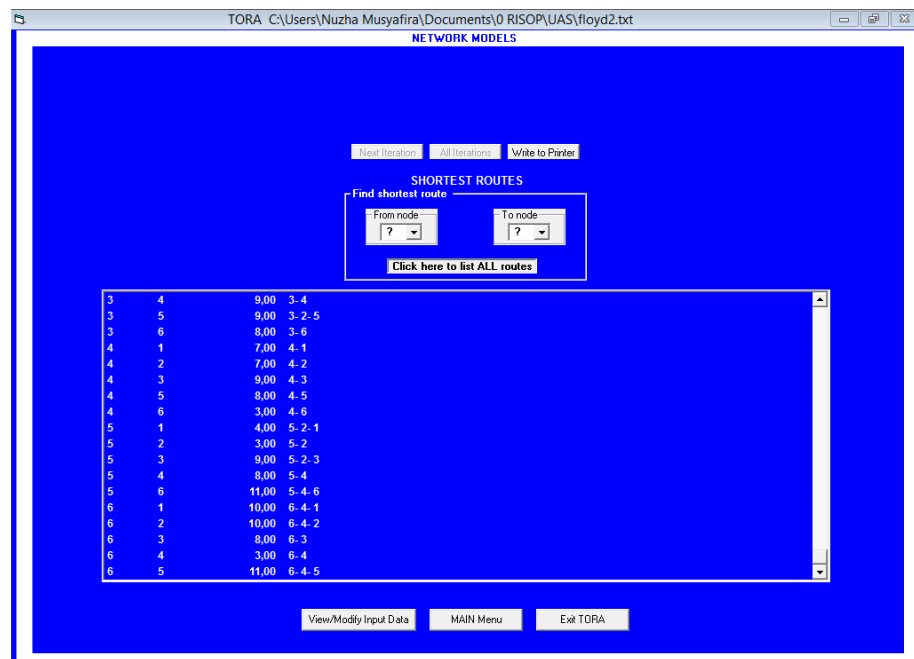
Find shortest route

From node ? To node ?

Click here to list ALL routes

From	To	Distance	Route
1	2	1,00	1-2
1	3	5,00	1-3
1	4	7,00	1-4
1	5	4,00	1-2-5
1	6	10,00	1-4-6
2	1	1,00	2-1
2	3	6,00	2-3
2	4	7,00	2-4
2	5	3,00	2-5
2	6	10,00	2-4-6
3	1	5,00	3-1
3	2	6,00	3-2
3	4	9,00	3-4
3	5	9,00	3-2-5
3	6	8,00	3-6
4	1	7,00	4-1
4	2	7,00	4-2

View/Modify Input Data MAIN Menu Exit TORA



### III. Analisis Algoritma

Algoritma Floyd merupakan algoritma pencari jarak yang mengadaptasi dari *Floyd's Triangle*. Mula-mula, dibentuk *adjacency matrix* dengan ukuran  $n \times n$  ( $n$  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  $n \times n \times n$ , 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)

```