TUGAS VII

CARA ROUTER MENENTUKAN RUTE TERBAIK DENGAN ALGORITMA

DISUSUN OLEH:

NAMA : ARUM CANTIKA PUTRI

NIM : 09011181419022

KELAS : SK 5 A

FAKULTAS ILMU KOMPUTER

JURUSAN SISTEM KOMPUTER

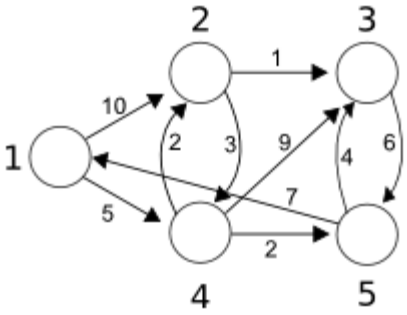
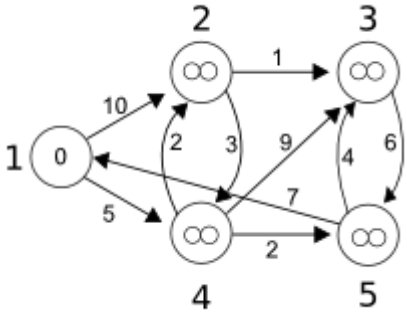
UNIVERSITAS SRIWIJAYA

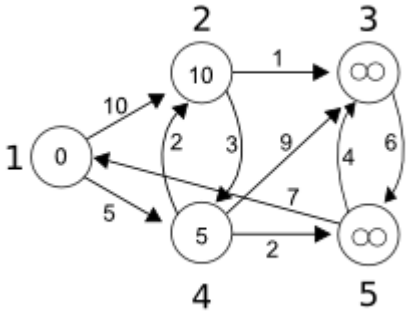
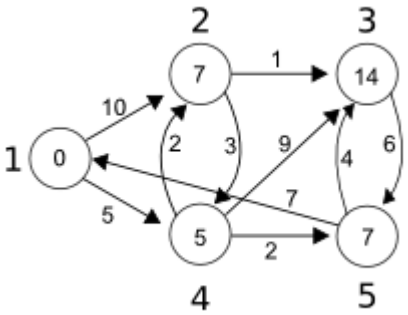
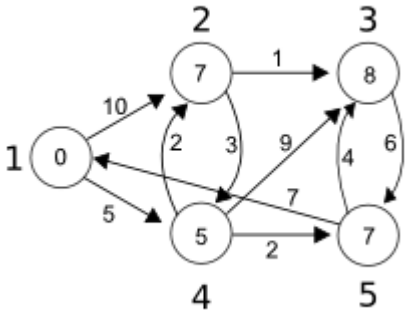
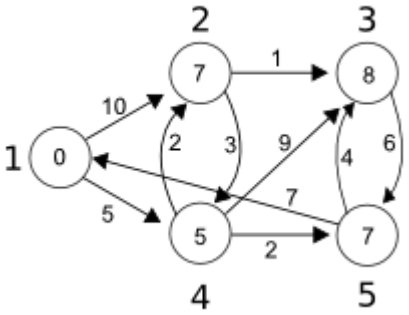
2016

CARA ROUTER MENENTUKAN JALAN TERBAIK DENGAN ALGORITMA

1. Algoritma Dijkstra

Ruter mencari jalan terbaik dengan algoritma Dijkstra atau algoritma shortest path first (SPF). Algoritma ini memperbaiki informasi database dari informasi topologi. Algoritma distance vector memiliki informasi yang tidak spesifik tentang distance network dan tidak mengetahui jarak router.

	1. Mengambil node 1 dan 3
	2. Mengambil jalur terpendek ke semua node menjadi tak terhingga. Tandai panjang jalur terpendek ke sumber sebagai 0

	3. Mengoptimalkan jalur ke node 2 dan 4
	4. Mengulang langkah ini 4 kali untuk memastikan semua node optimal
	5. Tidak ada optimasi untuk node 5 dan 3
	6. Jalur terpendek dari Dijkstra

```
Foreach node set distance[node] = HIGH
SettledNodes = empty
UnSettledNodes = empty

Add sourceNode to UnSettledNodes
distance[sourceNode]= 0

while (UnSettledNodes is not empty) {
  evaluationNode = getNodeWithLowestDistance(UnSettledNodes)
  remove evaluationNode from UnSettledNodes
    add evaluationNode to SettledNodes
    evaluatedNeighbors(evaluationNode)
}

getNodeWithLowestDistance(UnSettledNodes){
  find the node with the lowest distance in UnSettledNodes and return it
}
```

```
evaluatedNeighbors(evaluationNode){
  Foreach destinationNode which can be reached via an edge from
evaluationNode AND which is not in SettledNodes {
    edgeDistance = getDistance(edge(evaluationNode, destinationNode))
    newDistance = distance[evaluationNode] + edgeDistance
    if (distance[destinationNode]  > newDistance) {
      distance[destinationNode]  = newDistance
      add destinationNode to UnSettledNodes
    }
  }
}
```

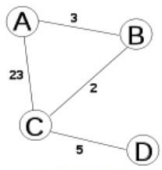
2. Algoritma Belman-Ford

Algoritma Bellman-Ford merupakan algoritma untuk mencari shortest path, dengan menghitung jarak terpendek pada sebuah graf berbobot, atau menghitung semua jarak terpendek yang berawal dari satu titik node.

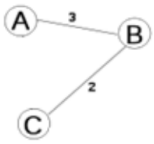
Pseudocode Algoritma Bellman-Ford

```
procedure BellmanFord(list vertices, list edges, vertex source)
   // This implementation takes in a graph, represented as lists of vertices
   // and edges, and modifies the vertices so that their distance and
   // predecessor attributes store the shortest paths.

   // Step 1: Initialize graph
   for each vertex v in vertices:
       if v is source then v.distance := 0
       else v.distance := infinity
       v.predecessor := null

   // Step 2: relax edges repeatedly
   for i from 1 to size(vertices)-1:
       for each edge uv in edges: // uv is the edge from u to v
           u := uv.source
           v := uv.destination
           if u.distance + uv.weight < v.distance:
               v.distance := u.distance + uv.weight
               v.predecessor := u

   // Step 3: check for negative-weight cycles
   for each edge uv in edges:
       u := uv.source
       v := uv.destination
       if u.distance + uv.weight < v.distance:
           error "Graph contains a negative-weight cycle"
```

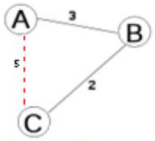
Contoh Routing dengan Bellman-Ford



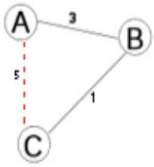
Gambar 2. Router A, B, C, D dan Bobot Sisinya



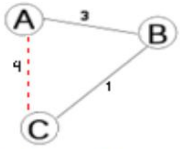
3. Kasus pada Router A, B, C



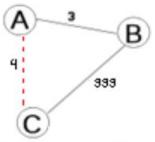
4. Seakan-akan Ada Path pada Router A-C, C-A



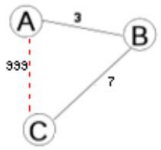
5. Update Cost pada Router B-C menjadi 1



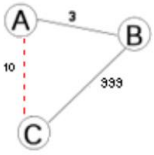
Gambar 6. Distance Cost yang Baru



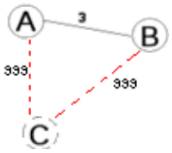
7. Link Antara B-C Diset Menjadi 999



Gambar 8. Perubahan Link Hasil Update



Gambar 9. Perubahan Selanjutnya



Gambar 10. Kondisi Akhir

Algoritma ini terdiri dari langkah-langkah berikut:

1. Setiap simpul menghitung jarak antara dirinya dan semua node lain dalam AS dan menyimpan informasi ini dalam sebuah tabel.

2. Setiap node mengirimkan tabel untuk semua node tetangganya.

3. Ketika sebuah node menerima tabel jarak dari tetangga-tetangganya, ia menghitung rute terpendek ke semua node lainnya dan melakukan updating tabelnya sendiri untuk mencerminkan perubahan apapun.

dari A	via A	via B	via C	via D
A				
B		3		
C			23	
D				

dari B	via A	via B	via C	via D
A	3			
B				
C			2	
D				

dari C	via A	via B	via C	via D
A	23			
B		2		
C				
D				5

dari D	via A	via B	via C	via D
A				
B				
C			5	
D				

Tabel 1 Routing Table saat T=0

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D			28	

from B	via A	via B	via C	via D
to A	3		25	
to B				
to C	26		2	
to D			7	

from C	via A	via B	via C	via D
to A	23	5		
to B	26	2		
to C				
to D				5

from D	via A	via B	via C	via D
to A			28	
to B			7	
to C			5	
to D				

Tabel 2 Routing Table saat T=1

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		7	
to B				
to C	8		2	
to D	31		7	

from C	via A	via B	via C	via D
to A	23	5		33
to B	26	2		12
to C				
to D	51	9		5

from D	via A	via B	via C	via D
to A			10	
to B			7	
to C			5	
to D				

Tabel 3 Routing Table saat T=2

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		7	
to B				
to C	8		2	
to D	13		7	

from C	via A	via B	via C	via D
to A	23	5		15
to B	26	2		12
to C				
to D	33	9		5

from D	via A	via B	via C	via D
to A			10	
to B			7	
to C			5	
to D				

Tabel 4 Routing Table saat T=3