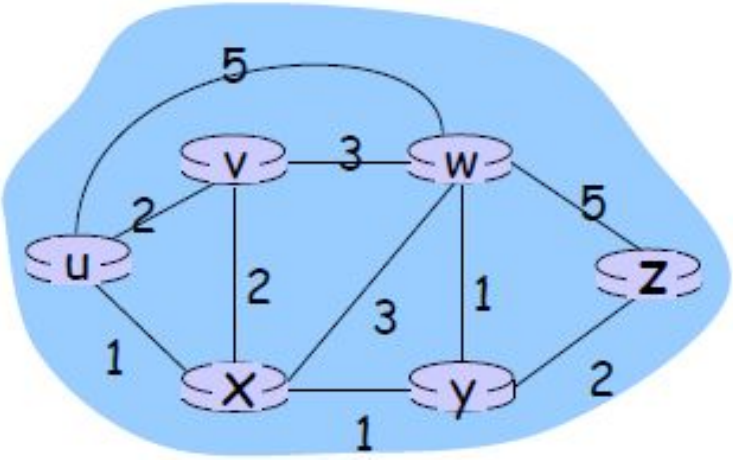
Computer Networks & IOT (18B11CS311)

Even Semester_2023

Network Layer

Routing Algorithms

Graph abstraction

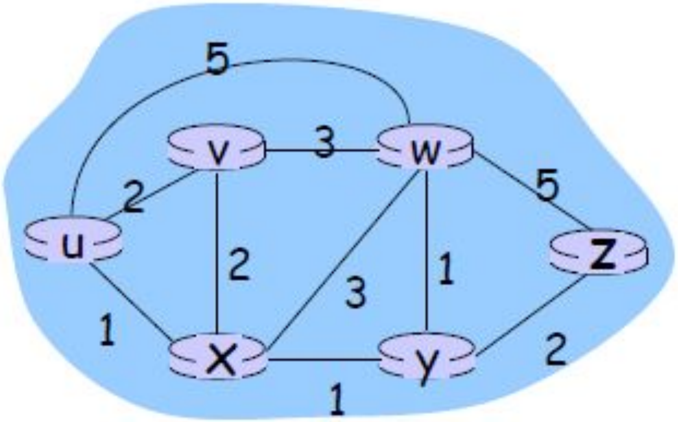


Graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Graph abstraction: costs



- $c(x,x')$ = cost of link (x,x')
  - e.g., $c(w,z) = 5$
- cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

Global:

- all routers have complete topology, link cost info
- "link state" algorithms

Decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Static or dynamic?

Static:

- routes change slowly over time

Dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

Routing Protocol

- Intra domain
  - Distance Vector (RIP-routing information protocol)
  - Link State (OSPF-open shortest path first)
- Inter domain
  - Path Vector (BGP)

Distance Vector Routing Algorithm

Routers using distance-vector protocol do not have knowledge of the entire path to a destination. Instead they use two methods:

1. Direction in which router or exit interface a packet should be forwarded.
2. Distance from its destination

"Direction" usually means the next hop address and the exit interface.

"Distance" is a measure of the cost to reach a certain node. The least cost route between any two nodes is the route with minimum distance.

Distance Vector

Local Signpost

- Direction
- Distance

Routing Table

For each destination list:

- Next Node
- Distance

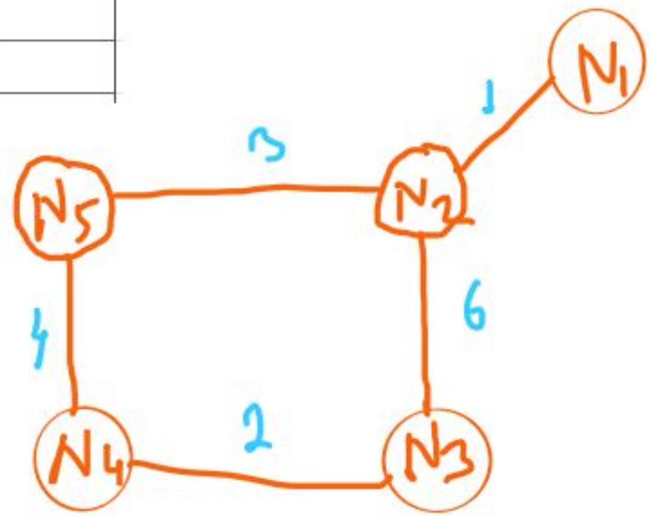
dest	next	dist

Table Synthesis

- Neighbors exchange table entries
- Determine current best next hop
- Inform neighbors
  - Periodically
  - After changes

Destination	Distance	Next
N1		
N2		
N3		
N4		
N5		

Destination	Distance	Next
N1		
N2		
N3		
N4		
N5		



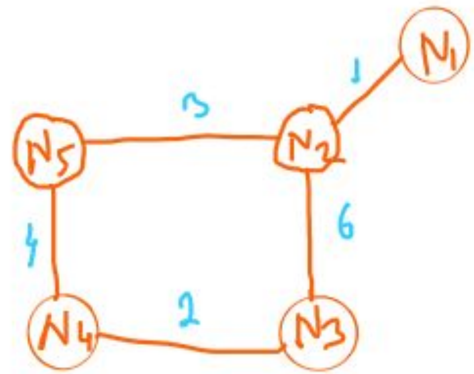
Destination	Distance	Next
N1		
N2		
N3		
N4		
N5		

Destination	Distance	Next
N1		
N2		
N3		
N4		
N5		

Destination	Distance	Next
N1		
N2		
N3		
N4		
N5		

Destination	Distance	Next
N1	∞	—
N2	3	N2
N3	∞	—
N4	4	N4
N5	0	N5

Destination	Distance	Next
N1	0	N1
N2	1	N2
N3	∞	—
N4	∞	—
N5	∞	—



Destination	Distance	Next
N1	1	N1
N2	0	N2
N3	6	n3
N4	∞	—
N5	3	N3

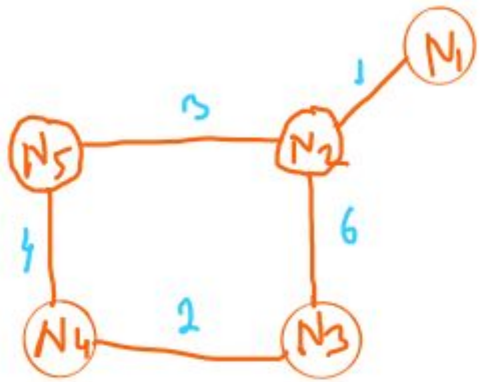
Destination	Distance	Next
N1	0	N1
N2	1	N2
N3	∞	—
N4	∞	—
N5	∞	—

Destination	Distance	Next
N1	∞	—
N2	6	N2
N3	0	N3
N4	2	N4
N5	∞	—

Router	From
N1	N2
N2	N1,N3,N5
N3	N2, N4
N4	N3,N5
N5	N2, N4

Destination	Distance	Next
N1	∞	
N2	3	N2
N3	∞	
N4	4	N4
N5	0	N5

Destination	Distance	Next
N1	0	N1
N2	1	N2
N3	∞	
N4	∞	
N5	∞	



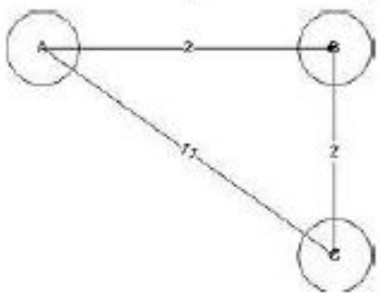
Destination	Distance	Next
N1	1	N1
N2	0	N2
N3	6	n3
N4	∞	
N5	3	N3

Destination	Distance	Next
N1	0	N1
N2	1	N2
N3	∞	
N4	∞	
N5	∞	

Destination	Distance	Next
N1	∞	
N2	6	N2
N3	0	N3
N4	2	N4
N5	∞	

Routing Loop?

- In the simplest version, a routing loop of size two, node A thinks that the path to some destination (call it C) is through its neighboring node, node B. At the same time, node B thinks that the path to C starts at node A.
- Thus, whenever traffic for C arrives at either A or B, it will loop endlessly between A and B, unless some mechanism exists to prevent that behavior.
- For example, in the network given below, **node A** is transmitting data to node **C** via node **B**. If the link between nodes **B** and **C** goes down and **B** has not yet informed node**A** about the breakage, node **A** transmits the data to node **B** assuming that the link **A-B-C** is operational and of lowest cost. Node **B** knows of the broken link and tries to reach node **C** via node **A**, thus sending the original data back to node **A**. Furthermore, node **A**receives the data that it originated back from node **B** and consults its routing table. Node **A**'s routing table will say that it can reach node **C** via node **B** (because it still has not been informed of the break) thus sending its data back to node **B** creating an infinite loop.



******This hop limit was introduced to avoid routing loops, but also means that networks that are connected through more than 15 routers are unreachable**

The Count-to-Infinity Problem

The algorithm does not prevent routing loops from happening and suffers from the **count-to-infinity problem**.

The Count-to-Infinity Problem

A's Routing Table

to	via (next hop)	cost
C	B	2

B's Routing Table

to	via (next hop)	cost
C	C	1

now link B-C goes down

C	B	2					C	-	oo
			C	2 →	← C	oo			
C	-	oo					C	A	3
			C	oo →	← C	3			
C	B	4					C	-	oo
			C	4 →	← C	oo			

34

Count-to-Infinity

- The reason for the count-to-infinity problem is that each node only has a "next-hop-view"
- For example, in the first step, A did not realize that its route (with cost 2) to C went through node B

- How can the Count-to-Infinity problem be solved?

Count-to-Infinity

Solution 1: Always advertise the entire path in an update message (**Path vectors**)

- If routing tables are large, the routing messages require substantial bandwidth
- BGP uses this solution

Solution 2: Never advertise the cost to a neighbor if this neighbor is the next hop on the current path **(Split Horizon)**

- Example: A would not send the first routing update to B, since B is the next hop on A's current route to C
- Split Horizon does not solve count-to-infinity in all cases!

Solution 3: **(Split Horizon with poison reverse)**

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

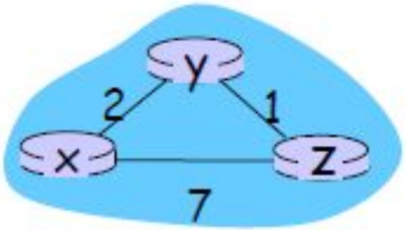
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$

node x table

cost to	x	y	z
from x	0	2	7
from y	∞	∞	∞
from z	∞	∞	∞

cost to	x	y	z
from x	0	2	3
from y	2	0	1
from z	7	1	0

cost to	x	y	z
from x	0	2	3
from y	2	0	1
from z	3	1	0

node y table

cost to	x	y	z
from x	∞	∞	∞
from y	2	0	1
from z	∞	∞	∞

cost to	x	y	z
from x	0	2	7
from y	2	0	1
from z	7	1	0

cost to	x	y	z
from x	0	2	3
from y	2	0	1
from z	3	1	0

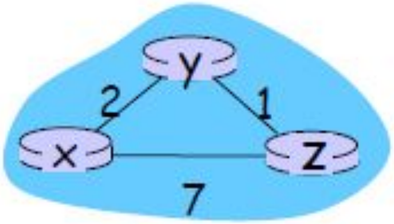
node z table

cost to	x	y	z
from x	∞	∞	∞
from y	∞	∞	∞
from z	7	1	0

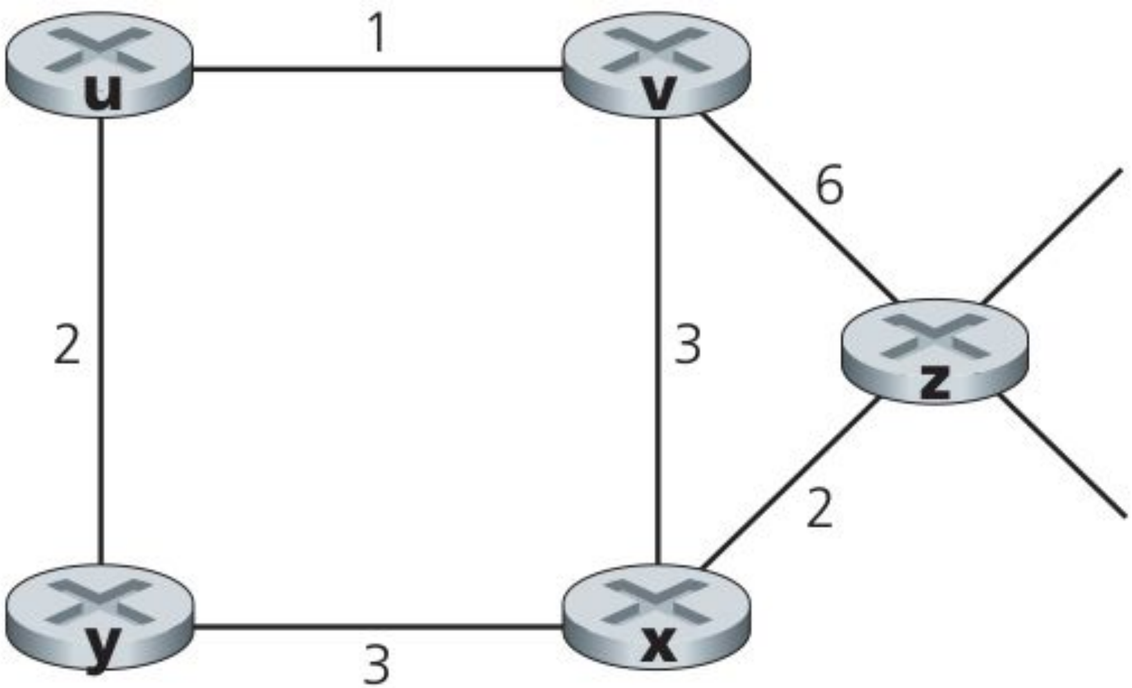
cost to	x	y	z
from x	0	2	7
from y	2	0	1
from z	3	1	0

cost to	x	y	z
from x	0	2	3
from y	2	0	1
from z	3	1	0

time



Consider the network shown below, and assume that each node initially knows the costs to each of its neighbors.
Consider the distance-vector algorithm and show the distance table entries at node z.



Distance vector routing algorithm exchanges the information with the neighbors and works asynchronously.

According to the distance vector algorithm, any node m computes the distance vector using the following formulas:

$$D_m(m) = 0$$

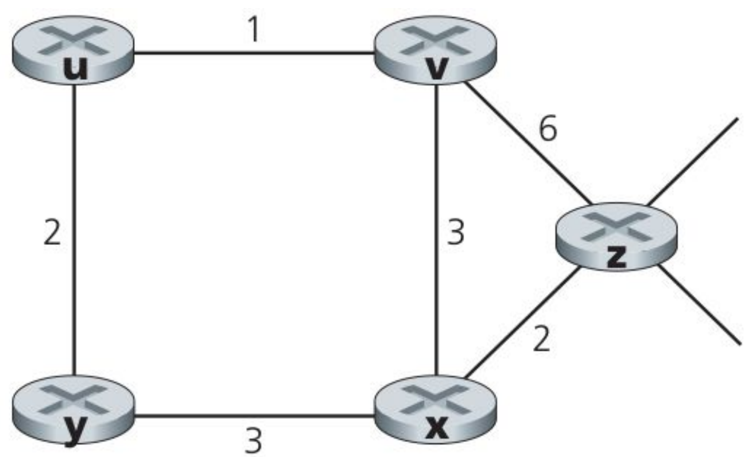
$$D_m(n) = \min\{c(m,n) + D_n(n), c(m,n) + D_o(n)\}$$

$$D_m(o) = \min\{c(m,n) + D_n(o), c(m,o) + D_o(o)\}$$

Note: NA is used when there is no distance value.

Construct the distance vector table for node z from the network diagram:

	u	*v*	*x*	*y*	*z*
v	*NA*	*NA*	*NA*	*NA*	*NA*
x	*NA*	*NA*	*NA*	*NA*	*NA*
z	*NA*	6	2	*NA*	0



Now update the table with costs of all the neighboring nodes.

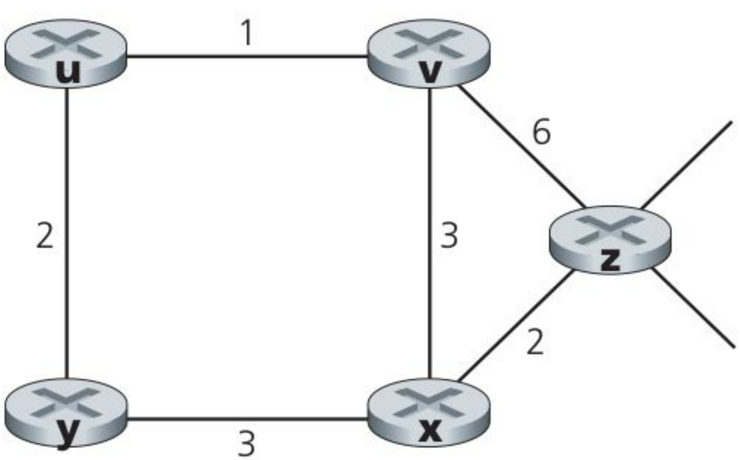
	u	*v*	*x*	*y*	*z*
v	1	0	3	*NA*	6
x	*NA*	3	0	3	2
z	*NA*	6	2	*NA*	0

Update the table with minimum costs using the distance vector routing algorithm:

Example: *v* to *y*, two paths are available. *v*-*u*-*y* and *v*-*x*-*y* with costs 3 and 6 respectively. So, *v*-*u*-*y* is the path with minimum cost. Hence update the table with this value.

	u	*v*	*x*	*y*	*z*
v	1	0	3	3	5
x	4	3	0	3	2
z	6	5	2	5	0

Therefore, at node *z*, the above table will be computed by the distance vector routing algorithm.

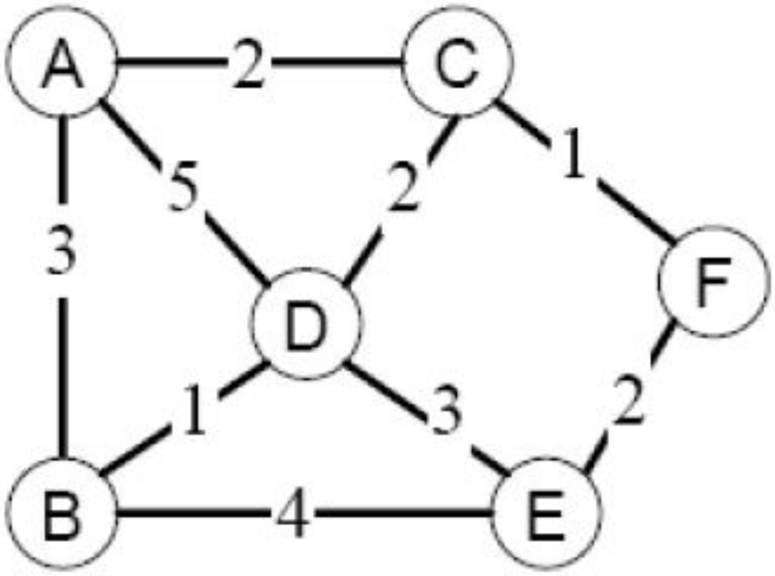


Questions..

1. Consider the network shown below.

(a)Use distance vector routing (Bellman-Ford algorithm) to find the set of shortest paths from all nodes to destination node B.

(b) Suppose link B–D fails; show the first two iterations of what happens next.



Link-State Routing Algorithm

A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.'s

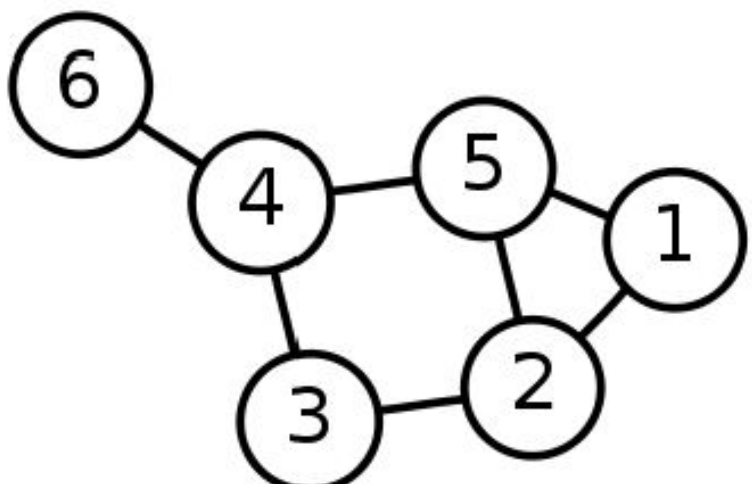
Notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijkstra's algorithm

Single-Source Shortest Path Problem

Single-Source Shortest Path Problem - The problem of finding shortest paths from a source vertex *v* to all other vertices in the graph.



Dijkstra's algorithm

Dijkstra's algorithm - is a solution to the single-source shortest path problem in graph theory.

Works on both directed and undirected graphs. However, all edges must have nonnegative weights.

Approach: Greedy

Input: Weighted graph $G=\{E,V\}$ and source vertex $v \in V$, such that all edge weights are nonnegative

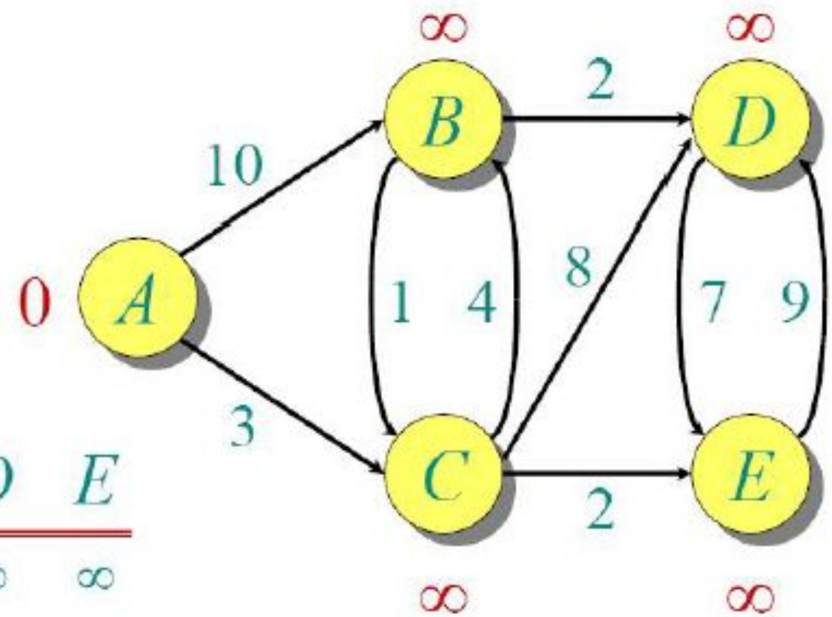
Output: Lengths of shortest paths (or the shortest paths themselves) from a given source vertex $v \in V$ to all other vertices

Dijkstra's algorithm - Pseudocode

```
dist[s] ←o                                    (distance to source vertex is zero)
for  all v ∈ V–{s}
      do  dist[v] ←∞                          (set all other distances to infinity)
S←∅                                           (S, the set of visited vertices is initially empty)
Q←V                                           (Q, the queue initially contains all
vertices)
while Q ≠∅                                    (while the queue is not empty)
do   u ← mindistance(Q,dist)                  (select the element of Q with the min.
distance)
    S←S∪{u}                                   (add u to list of visited vertices)
     for all v ∈ neighbors[u]
          do  if   dist[v] > dist[u] + w(u, v)              (if new shortest path found)
                   then      d[v] ←d[u] + w(u, v)     (set new value of shortest path)
                         (if desired, add traceback code)
return dist
```

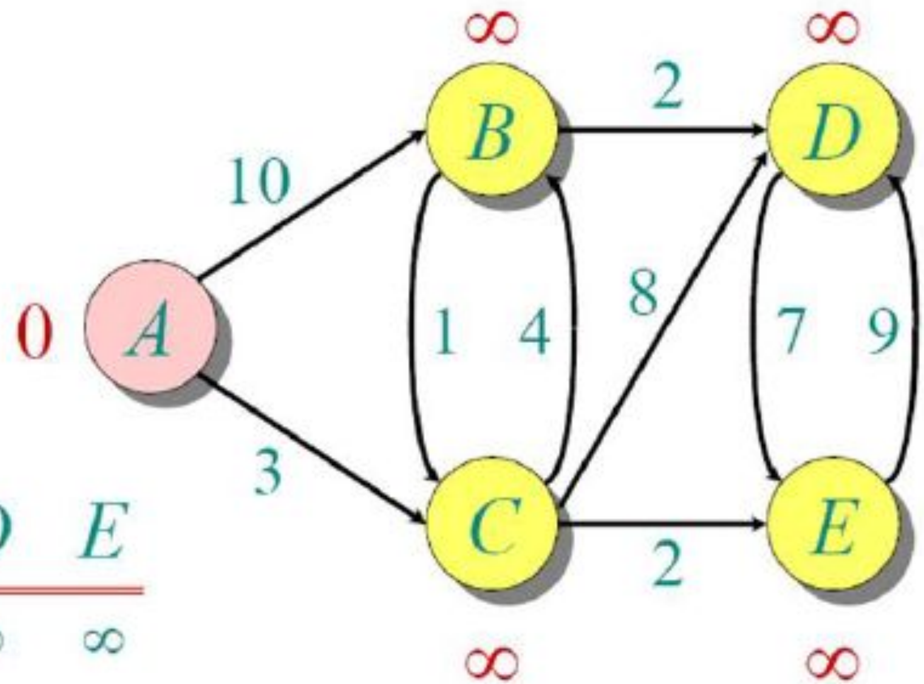
Dijkstra Example(1)

Initialize:

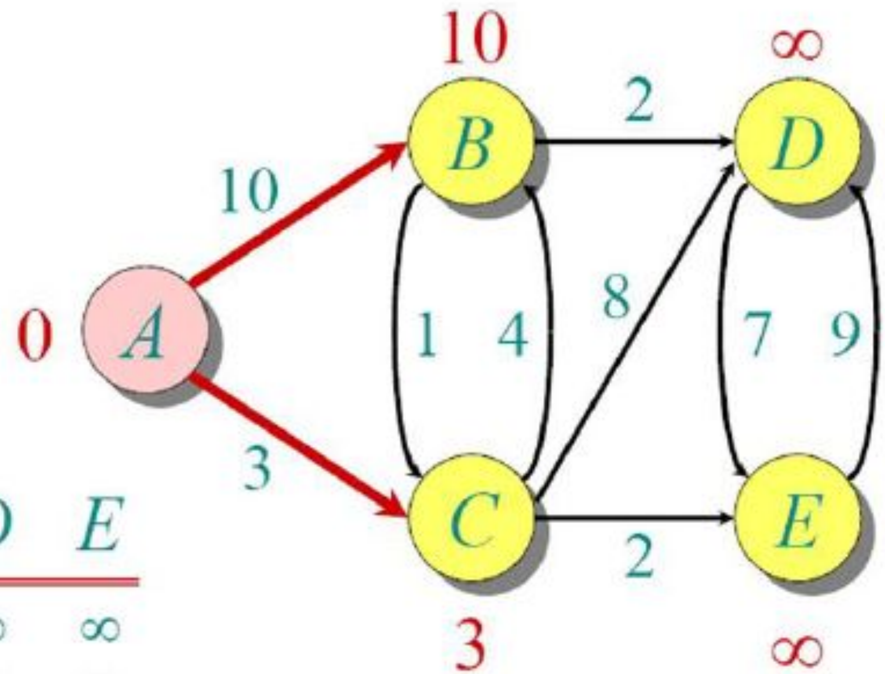


Q:	A	B	C	D	E
	0	∞	∞	∞	∞

S: {}

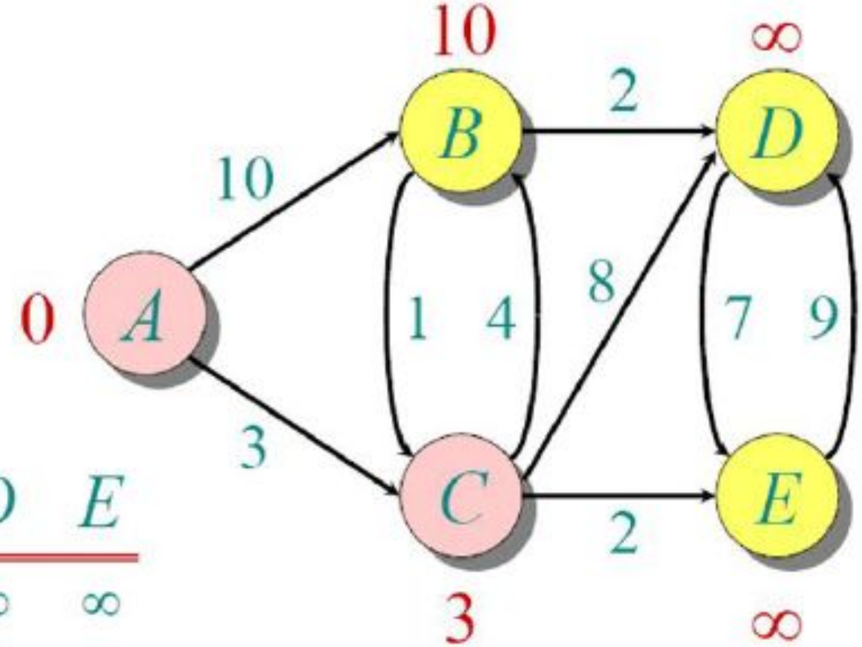


Q:	A	B	C	D	E
	0	∞	∞	∞	∞



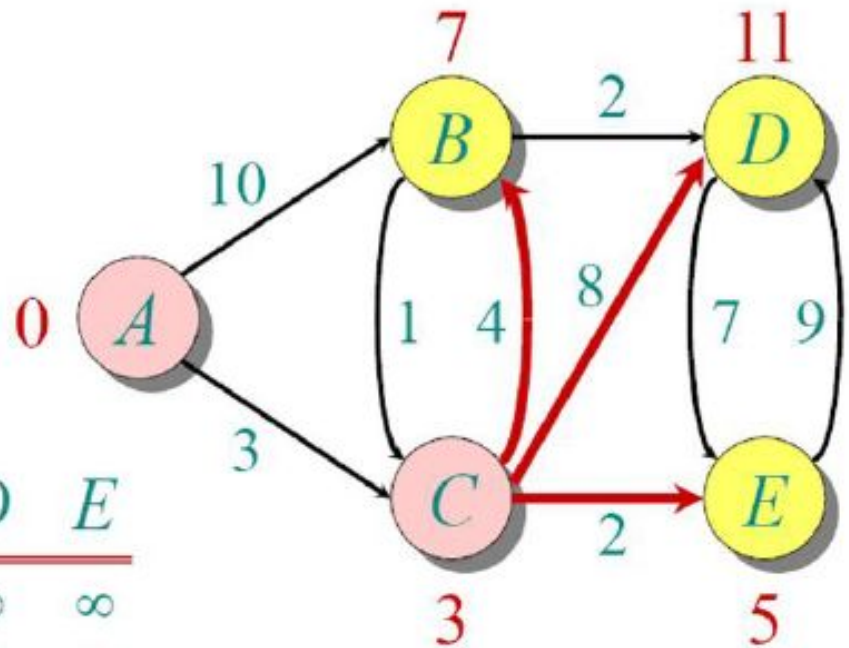
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A }



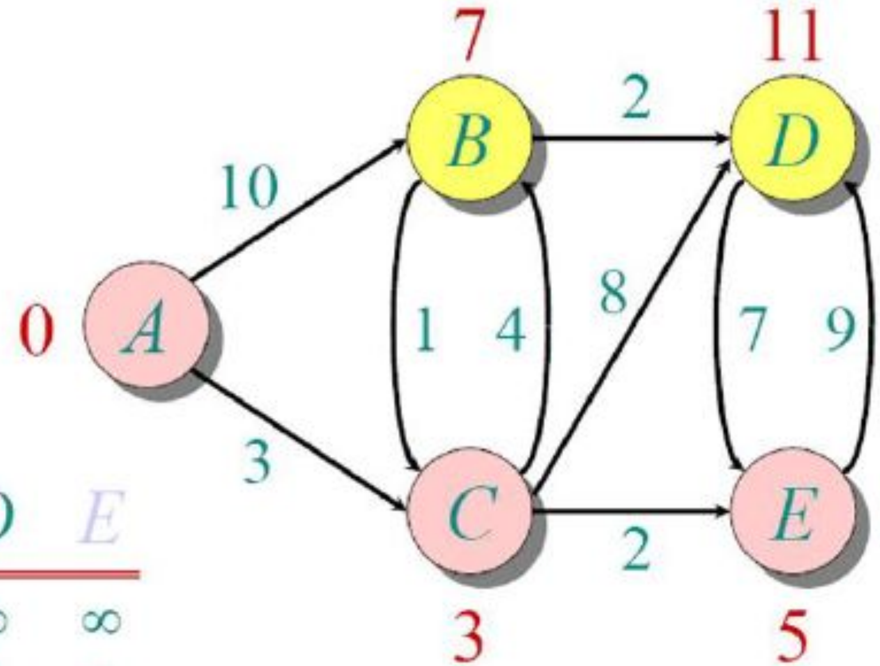
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A, C }



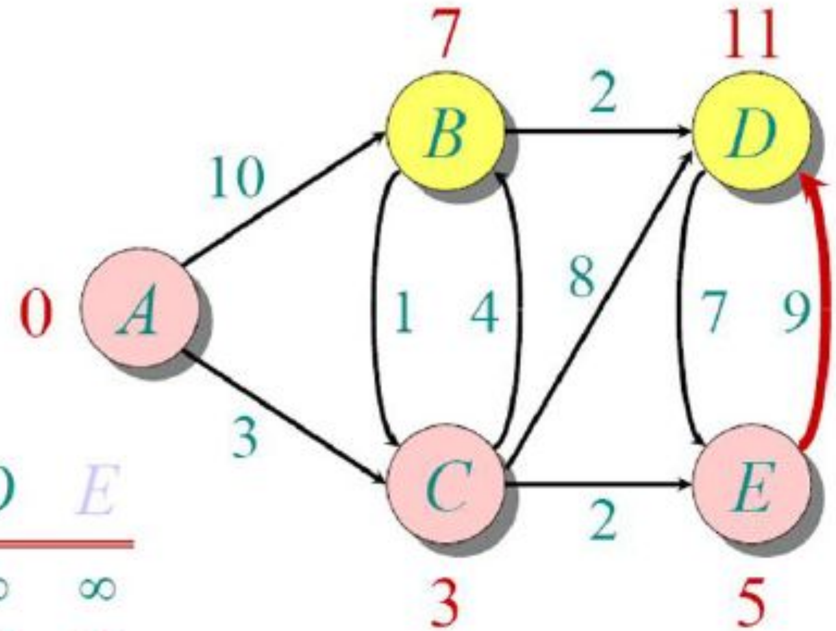
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { A, C }



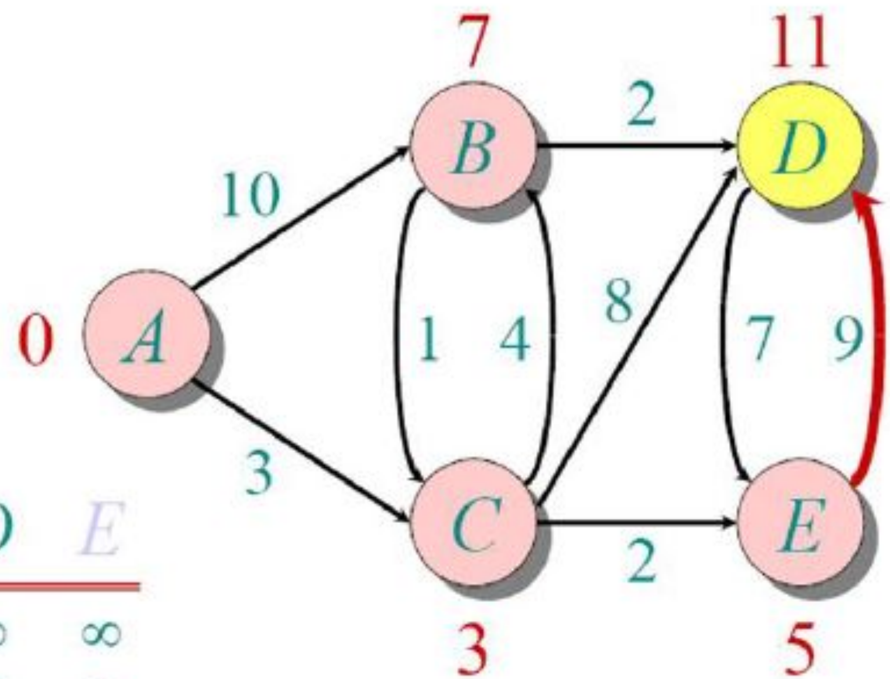
Q:	A	B	C	D	E
	$\boxed{0}$	∞	∞	∞	∞
		10	$\boxed{3}$	∞	∞
		7		11	$\boxed{5}$

S: { A, C, E }



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

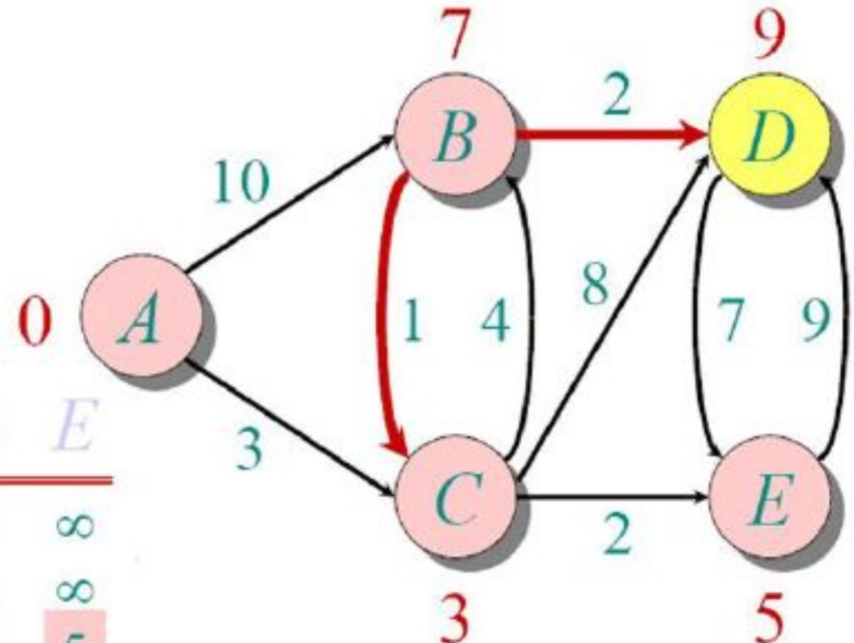
S: { *A*, *C*, *E* }



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

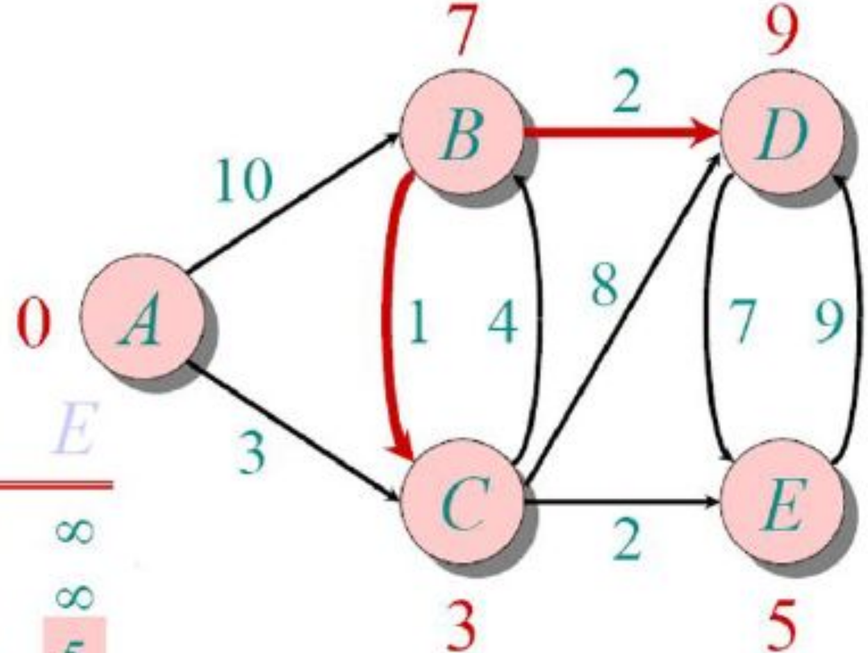
S: { A, C, E, B }

Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	



S: { A, C, E, B }

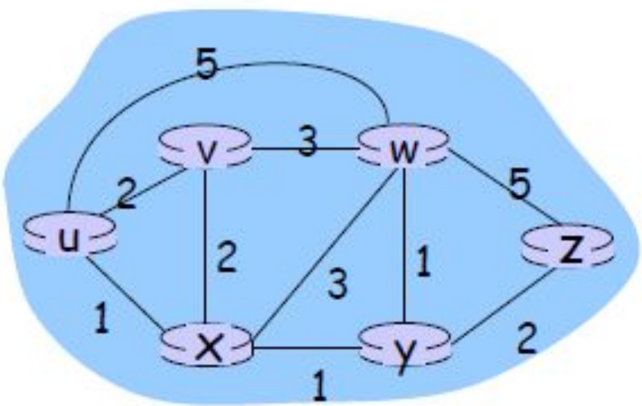
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	



S: { *A*, *C*, *E*, *B*, *D* }

Dijkstra's algorithm: example(2)

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



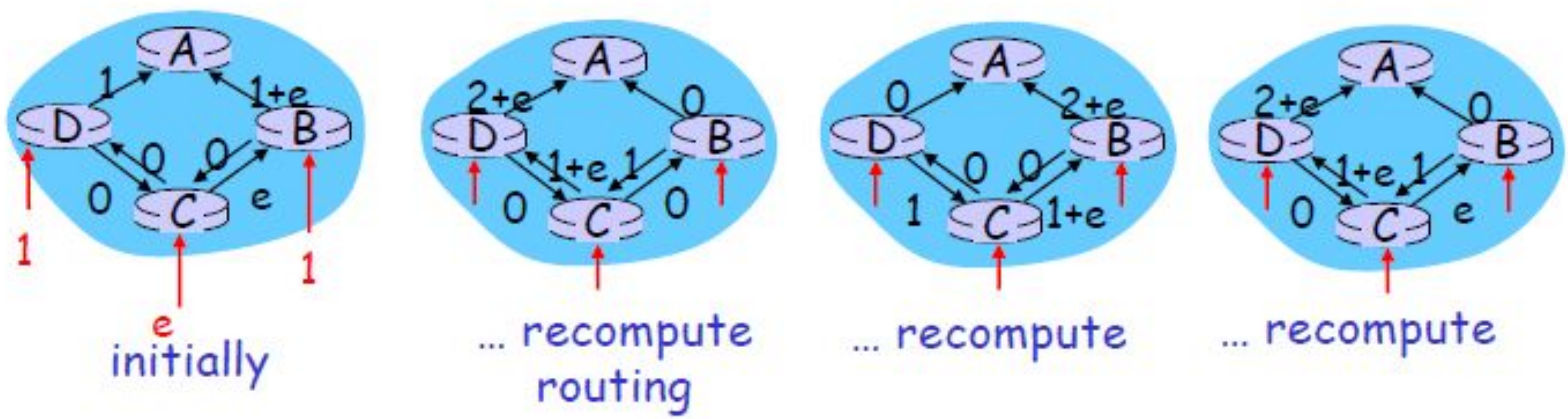
Dijkstra's algorithm, discussion

Algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons: $O(n^2)$
- more efficient implementations possible: $O(n\log n)$

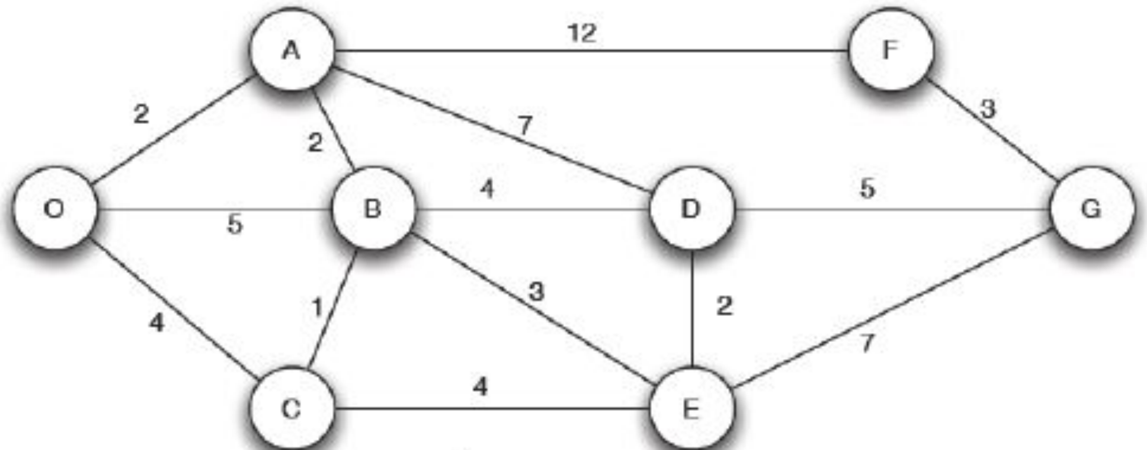
Oscillations possible:

- e.g., link cost = amount of carried traffic



Question..

Consider the network below and show the operation of Dijkastra (Link state) Algorithm for computing the least cost path from 0 to all destinations.

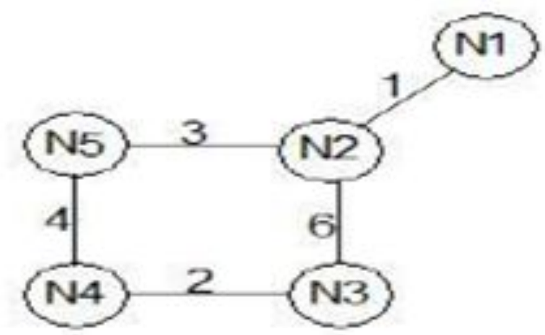


N	*D(A), P(A)*	*D(B), P(B)*	*D(C), P(C)*	*D(D), P(D)*	*D(E), P(E)*	*D(F), P(F)*	*D(E), P(E)*
0							

4-22

N	D(A), P(A)	D(B), P(B)	D(C), P(C)	D(D), P(D)	D(E), P(E)	D(F), P(F)	D(E), P(E)
O	2,O	5,O	4,O	Inf	Inf	Inf	Inf
OA	-	4,A	4,O	9,A	Inf	14,A	Inf
OAB	-	-	4,O	8,B	7,B	14,A	Inf
OABC	-	-	-	8,B	7,B	14,A	Inf
OABCE	-	-	-	8,B	-	14,A	14,E
OABCED	-	-	-	-	-	14,A	13,D
OABCEDG	-	-	-	-	-	14,A	-

2. Consider a network with five nodes, N1 to N5, as shown below. The network uses a Distance Vector Routing Distance Vector Routing protocol. Once the routes have stabilized, What are the distance vectors at different nodes: (show all iterations)



Each distance vector is the distance of the best known path at that instance to nodes, N1 to N5, where the distance to itself is 0. Also, all links are symmetric and the cost is identical in both directions. In each round, all nodes exchange their distance vectors with their respective neighbours. Then all nodes update their distance vectors. In between two rounds, any change in cost of a link will cause the two incident nodes to change only that entry in their distance vectors.
The cost of link N2-N3 reduces to 2 (in both directions). After the next round of update what will be the new distance vector at node, N3?

Solution:

Once the routes have stabilized, the distance vectors at different nodes are as following:

N1:(0, 1, 7, 8, 4)
N2:(1, 0, 6, 7, 3)
N3:(7, 6, 0, 2, 6)
N4:(8, 7, 2, 0, 4)
N5:(4, 3, 6, 4, 0)

Initialization

For N1

	N1	N2	N3	N4	N5
N1	0	1	∞	∞	∞
N2	∞	∞	∞	∞	∞
N3	—	—	—	—	—
N4	—	—	—	—	—
N5	—	—	—	—	—

For N2

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	1	0	6	∞	3
N3	—	—	—	—	—
N4	—	—	—	—	—
N5	—	—	—	—	—

For N5

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	—	—	—	—	—
N3	—	—	—	—	—
N4	—	—	—	—	—
N5	∞	3	∞	4	0

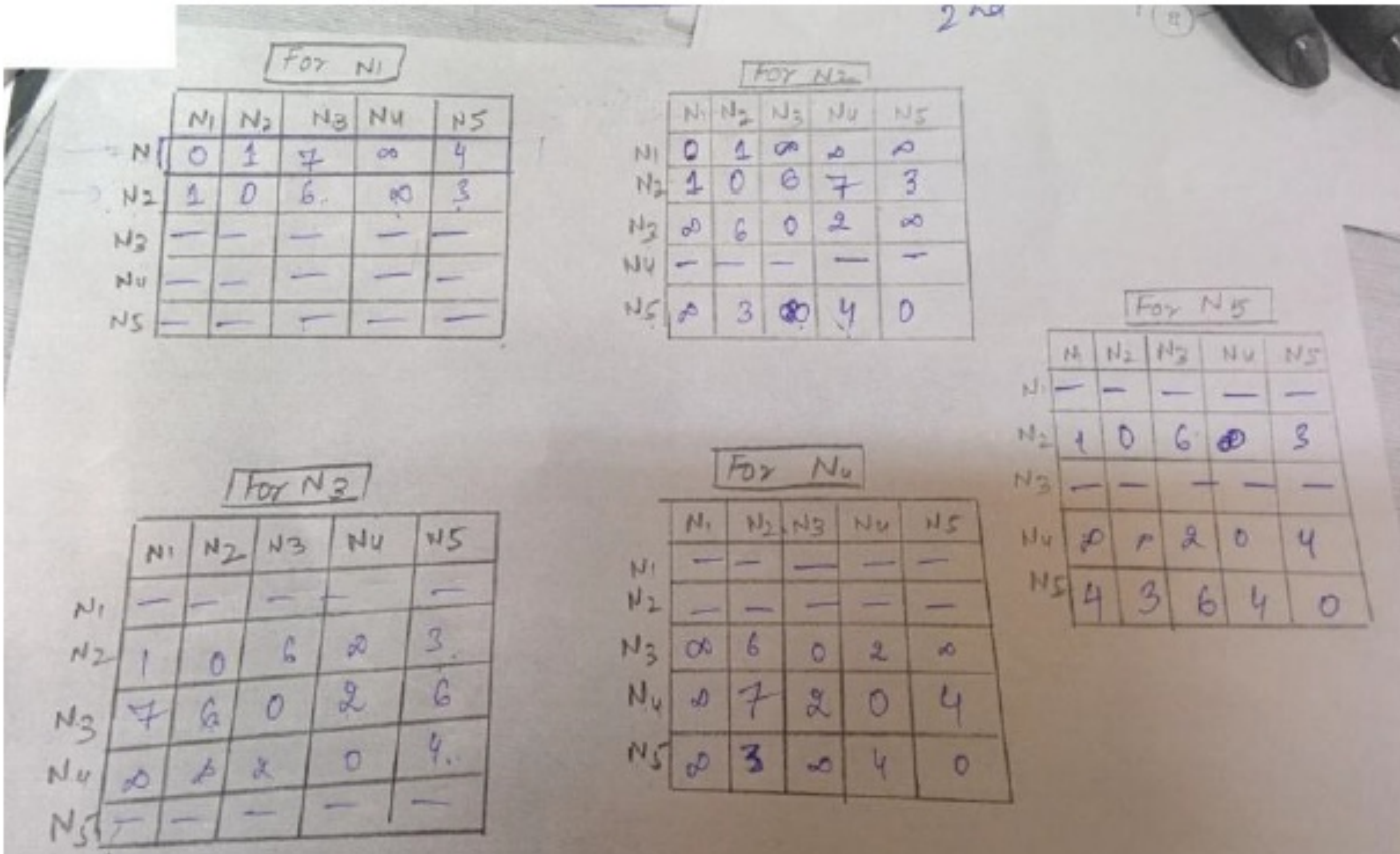
For N3

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	—	—	—	—	—
N3	∞	6	0	2	∞
N4	—	—	—	—	—
N5	—	—	—	—	—

For N4

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	—	—	—	—	—
N3	—	—	—	—	—
N4	∞	∞	2	0	4
N5	—	—	—	—	—

1st iteration



For N1

	N1	N2	N3	N4	N5
N1	0	1	7	∞	4
N2	1	0	6	∞	3
N3	—	—	—	—	—
N4	—	—	—	—	—
N5	—	—	—	—	—

For N2

	N1	N2	N3	N4	N5
N1	0	1	∞	∞	∞
N2	1	0	6	7	3
N3	∞	6	0	2	∞
N4	—	—	—	—	—
N5	∞	3	∞	4	0

For N3

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	1	0	6	∞	3
N3	7	6	0	2	6
N4	∞	∞	2	0	4
N5	—	—	—	—	—

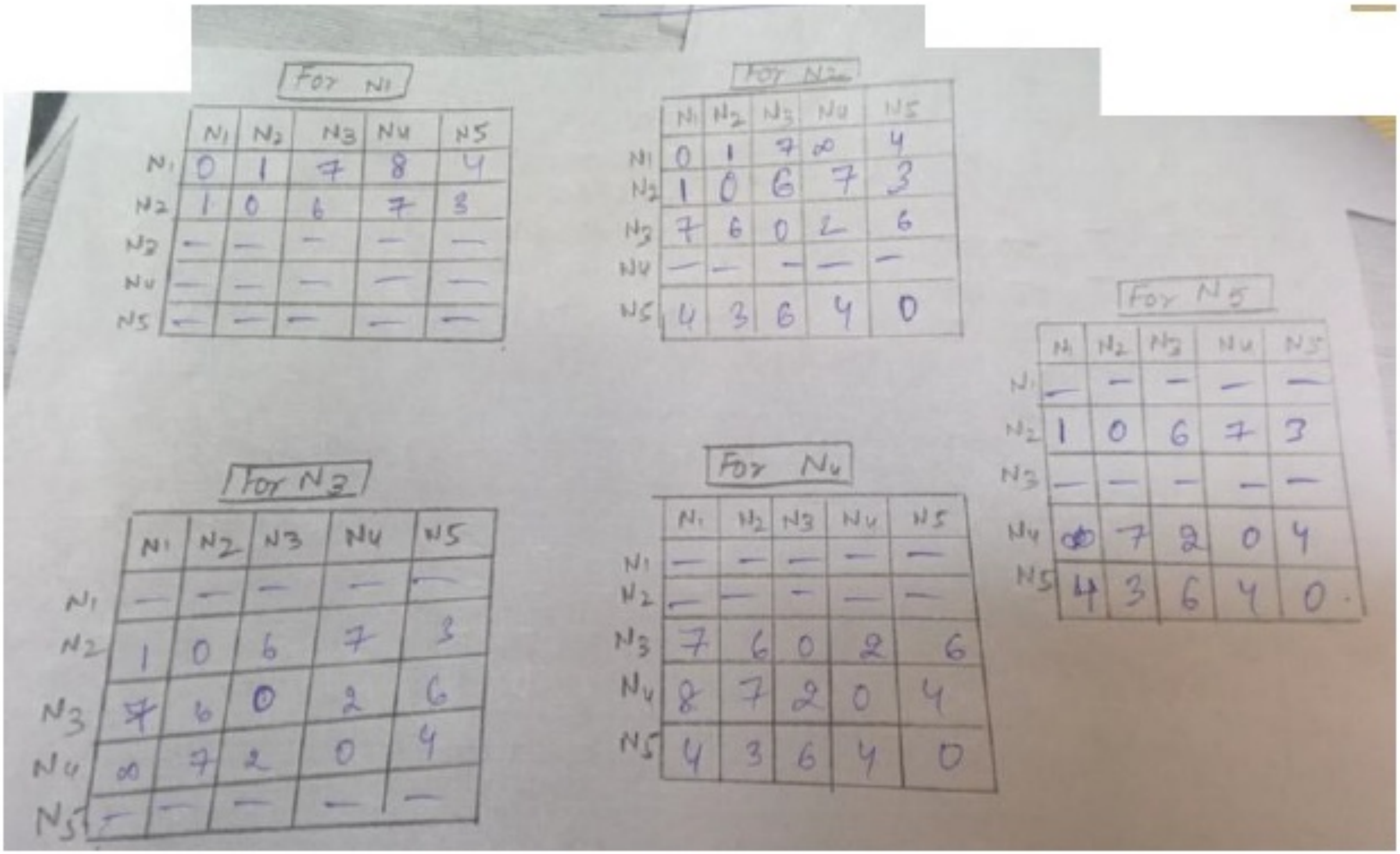
For N4

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	—	—	—	—	—
N3	∞	6	0	2	∞
N4	∞	7	2	0	4
N5	∞	3	∞	4	0

For N5

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	1	0	6	∞	3
N3	—	—	—	—	—
N4	∞	∞	2	0	4
N5	4	3	6	4	0

2nd iteration



For N1

	N1	N2	N3	N4	N5
N1	0	1	7	8	4
N2	1	0	6	7	3
N3	—	—	—	—	—
N4	—	—	—	—	—
N5	—	—	—	—	—

For N2

	N1	N2	N3	N4	N5
N1	0	1	7	∞	4
N2	1	0	6	7	3
N3	7	6	0	2	6
N4	—	—	—	—	—
N5	4	3	6	4	0

For N5

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	1	0	6	7	3
N3	—	—	—	—	—
N4	∞	7	2	0	4
N5	4	3	6	4	0

For N3

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	1	0	6	7	3
N3	7	6	0	2	6
N4	∞	7	2	0	4
N5	—	—	—	—	—

For N4

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	—	—	—	—	—
N3	7	6	0	2	6
N4	8	7	2	0	4
N5	4	3	6	4	0

3rd iteration

For N1

	N1	N2	N3	N4	N5
N1	0	1	7	8	4
N2	1	0	6	7	3
N3	—	—	—	—	—
N4	—	—	—	—	—
N5	—	—	—	—	—

For N2

	N1	N2	N3	N4	N5
N1	0	1	7	8	4
N2	1	0	6	7	3
N3	7	6	0	2	6
N4	—	—	—	—	—
N5	4	3	6	4	0

For N3

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	1	0	6	7	3
N3	7	6	0	2	6
N4	8	7	2	0	4
N5	—	—	—	—	—

For N4

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	—	—	—	—	—
N3	7	6	0	2	6
N4	8	7	2	0	4
N5	4	3	6	4	0

For N5

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	1	0	6	7	3
N3	—	—	—	—	—
N4	8	7	2	0	4
N5	4	3	6	4	0

4^{th} iteration

For N1

	N1	N2	N3	N4	N5
N1	0	1	7	8	4
N2	1	0	6	7	3
N3	—	—	—	—	—
N4	—	—	—	—	—
N5	—	—	—	—	—

For N2

	N1	N2	N3	N4	N5
N1	0	1	7	8	4
N2	1	0	6	7	3
N3	7	6	0	2	6
N4	—	—	—	—	—
N5	4	3	6	4	0

For N3

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	1	0	6	7	3
N3	7	6	0	2	6
N4	8	7	2	0	4
N5	—	—	—	—	—

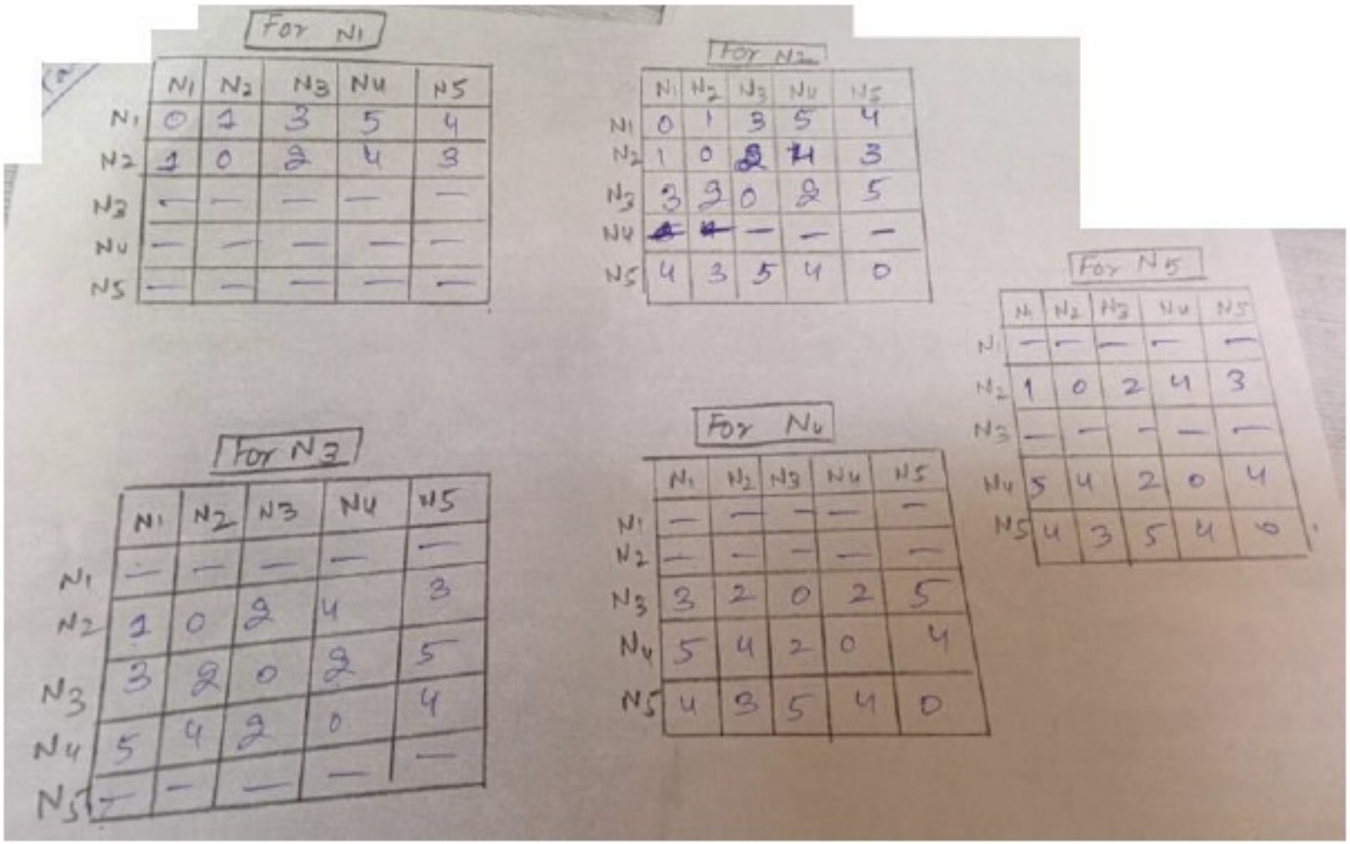
For N4

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	—	—	—	—	—
N3	7	6	0	2	6
N4	8	7	2	0	4
N5	4	3	6	4	0

For N5

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	1	0	6	7	3
N3	—	—	—	—	—
N4	8	7	2	0	4
N5	4	3	6	4	0

Link reduces to 2



For N1

	N1	N2	N3	N4	N5
N1	0	1	3	5	4
N2	1	0	2	4	3
N3	—	—	—	—	—
N4	—	—	—	—	—
N5	—	—	—	—	—

For N2

	N1	N2	N3	N4	N5
N1	0	1	3	5	4
N2	1	0	2	4	3
N3	3	2	0	2	5
N4	~~4~~	~~4~~	—	—	—
N5	4	3	5	4	0

For N3

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	1	0	2	4	3
N3	3	2	0	2	5
N4	5	4	2	0	4
N5	—	—	—	—	—

For N4

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	—	—	—	—	—
N3	3	2	0	2	5
N4	5	4	2	0	4
N5	4	3	5	4	0

For N5

	N1	N2	N3	N4	N5
N1	—	—	—	—	—
N2	1	0	2	4	3
N3	—	—	—	—	—
N4	5	4	2	0	4
N5	4	3	5	4	0

- James Kurose, Keith Ross," Computer Networking: A Top-Down Approach Featuring the Internet ", Addison Wesley
- Andrew S. Tanenbaum ,"Computer Networks ", Prentice-Hall Publishers
- Larry Peterson , Bruce Davie ,"Computer Networks a Systems Approach ", Morgan Kaufmann
- William Stallings ,"Data and Computer Communications", Prentice Hall
- David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Robert Barton, Jerome Henry, "IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things", CISCO Press, 2017
- Rajkumar Buyya, and Amir Vahid Dastjerdi, eds. Internet of Things: Principles and paradigms. Elsevier, 2016.