Link-State (LS) & Distance Vector (DV) Algorithms.

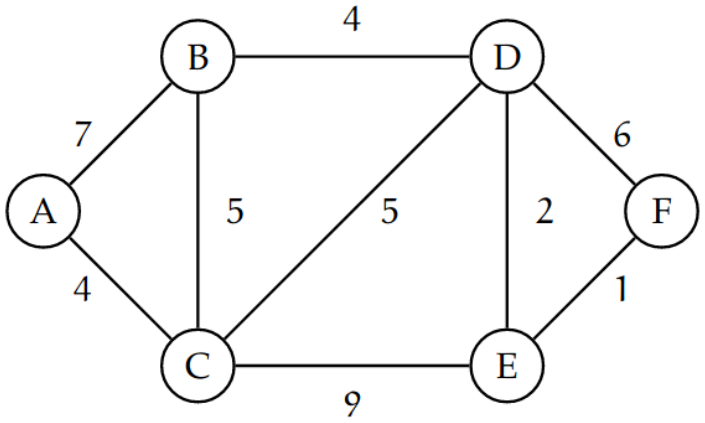
Worked through examples.

Link-State (LS) Algorithm for Source Node *u*

```
1  Initialization:
2   N' = {u}
3   for all nodes v
4     if v is a neighbor of u
5       then D(v) = c(u,v)
6     else D(v) = ∞
7
8  Loop
9   find w not in N' such that D(w) is a minimum
10  add w to N'
11  update D(v) for each neighbor v of w and not in N':
12        D(v) = min(D(v), D(w)+ c(w,v) )
13   /* new cost to v is either old cost to v or known
14    least path cost to w plus cost from w to v */
15 until N'= N
```

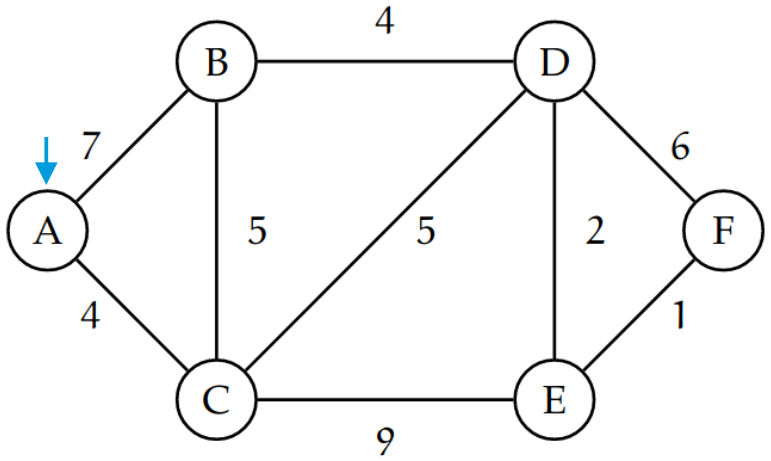
Re-exam 2021-22, Question 3.3.1

Question 3.3.1: Consider the network topology outlined in the graph below.



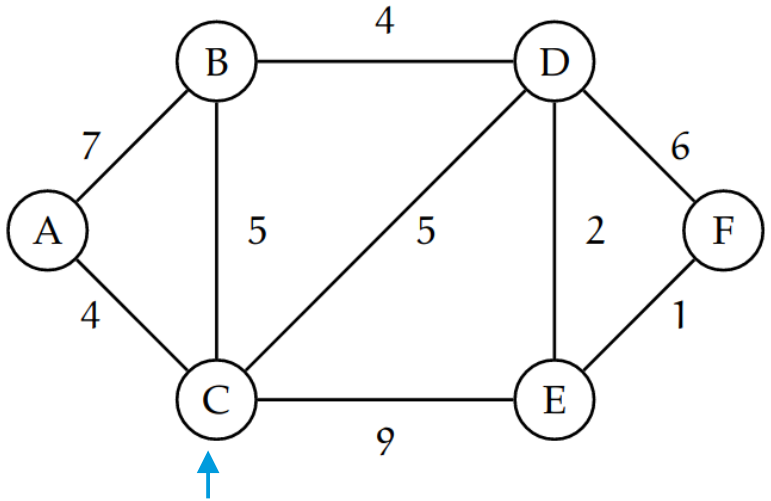
Apply the link state routing algorithm and compute the forwarding table on node **A** by filling out the following tables.

Step	N’	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E), p(E)	D(F), p(F)
0	A	7, A	4, A	∞	∞	∞



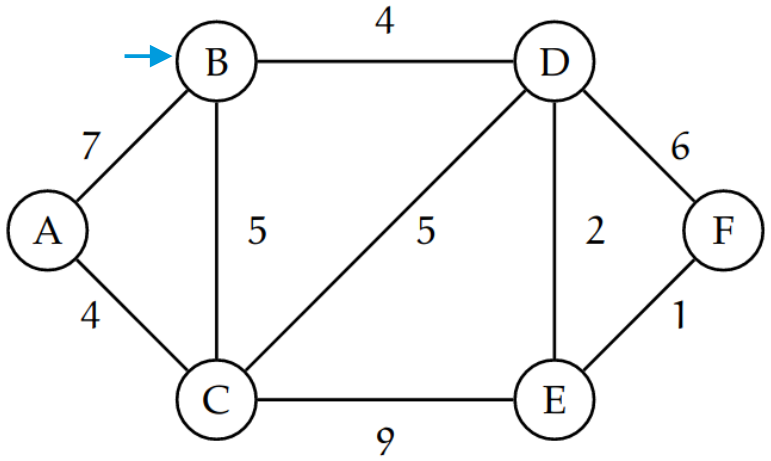
p(X) = predecessor, the node before X, this can and will change as we relax more nodes
D(X) = distance (cost) to X, from our initial node, in this case A

Step	N’	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E), p(E)	D(F), p(F)
0	A	7, A	4, A	∞	∞	∞
1	A, C	7, A	4, A	9, C	13, C	∞



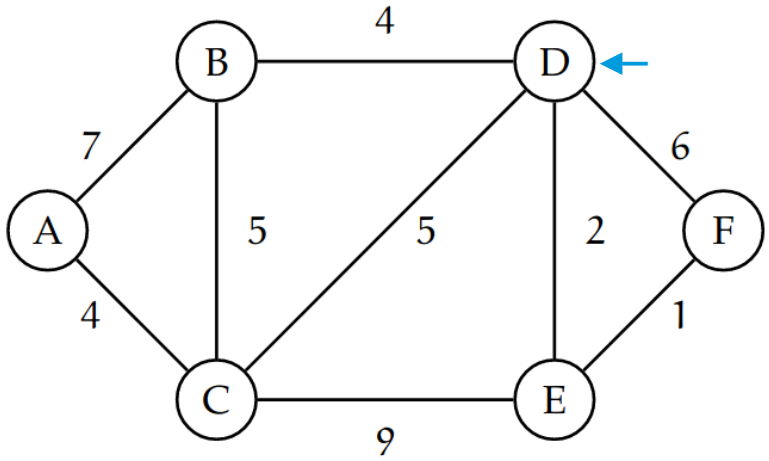
p(X) = predecessor, the node before X, this can and will change as we relax more nodes
D(X) = distance (cost) to X, from our initial node, in this case A

Step	N’	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E), p(E)	D(F), p(F)
0	A	7, A	4, A	∞	∞	∞
1	A, C	7, A	4, A	9, C	13, C	∞
2	A, C, B	7, A		9, C	13, C	∞



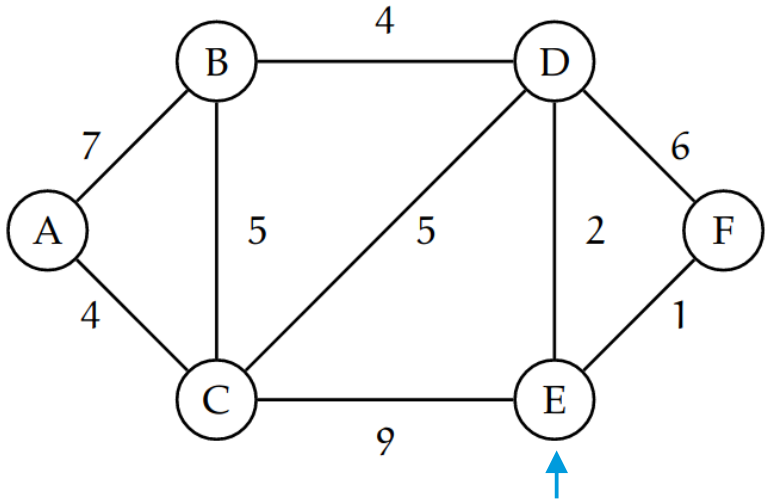
p(X) = predecessor, the node before X, this can and will change as we relax more nodes
D(X) = distance (cost) to X, from our initial node, in this case A

Step	N’	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E), p(E)	D(F), p(F)
0	A	7, A	4, A	∞	∞	∞
1	A, C	7, A	4, A	9, C	13, C	∞
2	A, C, B	7, A		9, C	13, C	∞
3	A, C, B, D			9, C	11, D	15, D



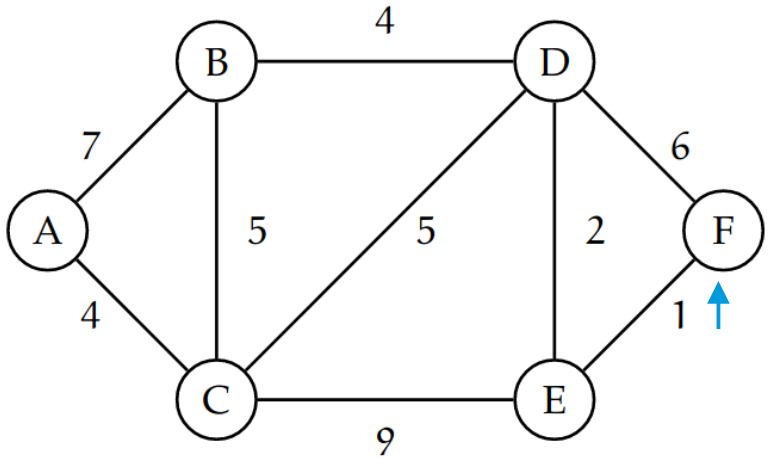
p(X) = predecessor, the node before X, this can and will change as we relax more nodes
D(X) = distance (cost) to X, from our initial node, in this case A

Step	N’	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E), p(E)	D(F), p(F)
0	A	7, A	4, A	∞	∞	∞
1	A, C	7, A	4, A	9, C	13, C	∞
2	A, C, B	7, A		9, C	13, C	∞
3	A, C, B, D			9, C	11, D	15, D
4	A, C, B, D, E				11, D	12, E



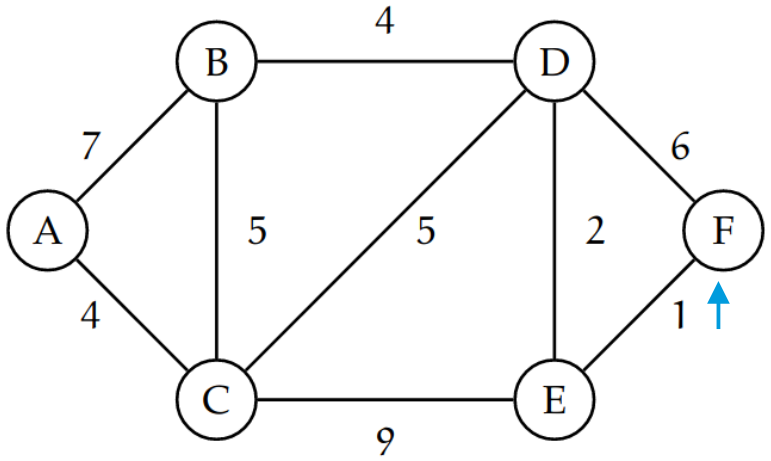
p(X) = predecessor, the node before X, this can and will change as we relax more nodes
D(X) = distance (cost) to X, from our initial node, in this case A

Step	N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E), p(E)	D(F), p(F)
0	A	7, A	4, A	∞	∞	∞
1	A, C	7, A	4, A	9, C	13, C	∞
2	A, C, B	7, A		9, C	13, C	∞
3	A, C, B, D			9, C	11, D	15, D
4	A, C, B, D, E				11, D	12, E
5	A, C, B, D, E, F					12, E



p(X) = predecessor, the node before X, this can and will change as we relax more nodes
D(X) = distance (cost) to X, from our initial node, in this case A

Step	N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E), p(E)	D(F), p(F)
0	A	7, A	4, A	∞	∞	∞
1	A, C	7, A	4, A	9, C	13, C	∞
2	A, C, B	7, A		9, C	13, C	∞
3	A, C, B, D			9, C	11, D	15, D
4	A, C, B, D, E				11, D	12, E
5	A, C, B, D, E, F					12, E



Forwarding table for A:

Destination	Edge
B	(A,B)
C	(A,C)
D	(A,C)
E	(A,C)
F	(A,C)

p(X) = predecessor, the node before X, this can and will change as we relax more nodes
D(X) = distance (cost) to X, from our initial node, in this case A

Distance-Vector (DV) Algorithm

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

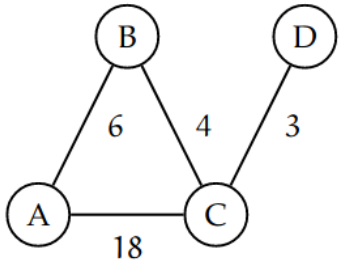
At each node, x:

```
1  Initialization:
2    for all destinations y in N:
3       D_x(y) = c(x,y) /* if y is not a neighbor then c(x,y) = ∞ */
4    for each neighbor w
5       D_w(y) = ? for all destinations y in N
6    for each neighbor w
7       send distance vector  D_x = [D_x(y): y in N] to w
8
9  loop
10    wait  (until I see a link cost change to some neighbor w or
11             until I receive a distance vector from some neighbor w)
12
13    for each y in N:
14        D_x(y) = min_v{c(x,v) + D_v(y)}
15
16 if D_x(y) changed for any destination y
17       send distance vector D_x = [D_x(y): y in N] to all neighbors
18
19 forever
```

P. 419 K&R.

Exam 2021-22, Question 3.3.1

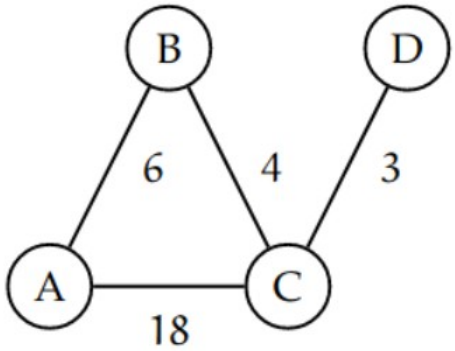
Consider the network topology outlined in the graph below



Question 3.3.1: Describe how to apply the Distance Vector Algorithm for Node A in the above diagram, so that it gets a complete routing table to all other Nodes. You do not need to describe each individual step on each Node in detail, but should be able to provide a generalised description that could apply to any Node. For this answer you can assume no connection costs change.

Start of the algorithm, nodes only know price to neighbor

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$



A

	A	B	C	D
A	0	6	18	∞
B	∞	∞	∞	∞
C	∞	∞	∞	∞
D	∞	∞	∞	∞

B

	A	B	C	D
A	∞	∞	∞	∞
B	6	0	4	∞
C	∞	∞	∞	∞
D	∞	∞	∞	∞

C

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	18	4	0	3
D	∞	∞	∞	∞

D

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	∞	∞	∞	∞
D	∞	∞	3	0

Start of the algorithm, nodes only know price to neighbor

Distance vectors are exchanged to neighboring nodes, and local distance vectors are updated

A

	A	B	C	D
A	0	6	18	∞
B	∞	∞	∞	∞
C	∞	∞	∞	∞
D	∞	∞	∞	∞

	A	B	C	D
A	0	6	**10**	**21**
B	**6**	**0**	**4**	∞
C	**18**	**4**	**0**	**3**
D	∞	∞	∞	∞

B

	A	B	C	D
A	∞	∞	∞	∞
B	6	0	4	∞
C	∞	∞	∞	∞
D	∞	∞	∞	∞

	A	B	C	D
A	**0**	**6**	**18**	∞
B	6	0	4	**7**
C	**18**	**4**	**0**	**3**
D	∞	∞	∞	∞

C

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	18	4	0	3
D	∞	∞	∞	∞

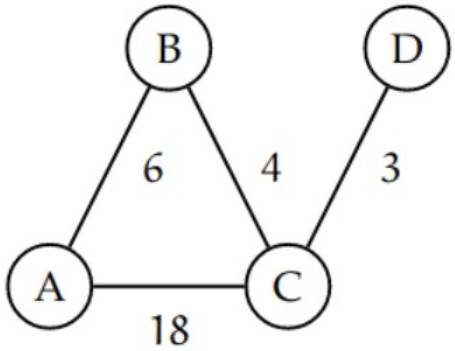
	A	B	C	D
A	**0**	**6**	**18**	∞
B	**6**	**0**	**4**	∞
C	**10**	4	0	3
D	∞	∞	**3**	∞

D

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	∞	∞	∞	∞
D	∞	∞	3	0

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	**18**	**4**	**0**	**3**
D	**21**	**7**	3	0

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

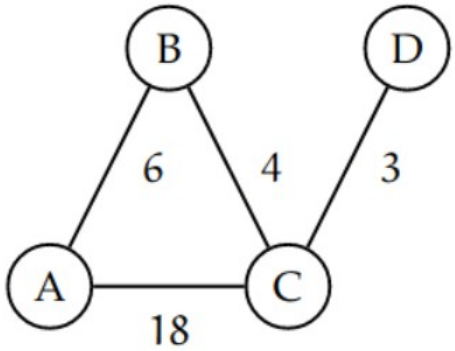


Start of the algorithm, nodes only know price to neighbor

Distance vectors are exchanged, and local distance vectors are updated

D_A(D) and D_D(A) get the final value 13

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$



A

	A	B	C	D
A	0	6	18	∞
B	∞	∞	∞	∞
C	∞	∞	∞	∞
D	∞	∞	∞	∞

	A	B	C	D
A	0	6	**10**	**21**
B	**6**	**0**	**4**	∞
C	**18**	**4**	**0**	**3**
D	∞	∞	∞	∞

	A	B	C	D
A	0	6	10	**13**
B	6	0	4	**7**
C	**10**	4	0	3
D	∞	∞	∞	∞

B

	A	B	C	D
A	∞	∞	∞	∞
B	6	0	4	∞
C	∞	∞	∞	∞
D	∞	∞	∞	∞

	A	B	C	D
A	**0**	**6**	**18**	∞
B	6	0	4	**7**
C	**18**	**4**	**0**	**3**
D	∞	∞	∞	∞

	A	B	C	D
A	0	6	**10**	**21**
B	6	0	4	7
C	**10**	4	0	3
D	∞	∞	∞	∞

C

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	18	4	0	3
D	∞	∞	∞	∞

	A	B	C	D
A	**0**	**6**	**18**	∞
B	**6**	**0**	**4**	∞
C	**10**	4	0	3
D	∞	∞	**3**	0

	A	B	C	D
A	0	6	**10**	**21**
B	6	0	4	**7**
C	10	4	0	3
D	**21**	**7**	3	0

D

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	∞	∞	∞	∞
D	∞	∞	3	0

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	**18**	**4**	**0**	**3**
D	**21**	**7**	3	0

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	**10**	4	0	3
D	**13**	7	3	0

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Start of the algorithm, nodes only know cost to their neighbor

Distance vectors are exchanged, and local distance vectors are updated

D_A(D) and D_D(A) get the final value 13

No more locale DVs to be exchanged. The algorithm has converged.

A

	A	B	C	D
A	0	6	18	∞
B	∞	∞	∞	∞
C	∞	∞	∞	∞
D	∞	∞	∞	∞

	A	B	C	D
A	0	6	**10**	**21**
B	**6**	**0**	**4**	∞
C	**18**	**4**	**0**	**3**
D	∞	∞	∞	∞

	A	B	C	D
A	0	6	10	**13**
B	6	0	4	**7**
C	**10**	4	0	3
D	∞	∞	∞	∞

	A	B	C	D
A	0	6	10	13
B	6	0	4	7
C	10	4	0	3
D	∞	∞	∞	∞

B

	A	B	C	D
A	∞	∞	∞	∞
B	6	0	4	∞
C	∞	∞	∞	∞
D	∞	∞	∞	∞

	A	B	C	D
A	**0**	**6**	**18**	∞
B	6	0	4	**7**
C	**18**	**4**	**0**	**3**
D	∞	∞	∞	∞

	A	B	C	D
A	0	6	**10**	**21**
B	6	0	4	7
C	**10**	4	0	3
D	∞	∞	∞	∞

	A	B	C	D
A	0	6	10	**13**
B	6	0	4	7
C	10	4	0	3
D	∞	∞	∞	∞

C

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	18	4	0	3
D	∞	∞	∞	∞

	A	B	C	D
A	**0**	**6**	**18**	∞
B	**6**	**0**	**4**	∞
C	**10**	4	0	3
D	∞	∞	**3**	∞

	A	B	C	D
A	0	6	**10**	**21**
B	6	0	4	**7**
C	10	4	0	3
D	**21**	**7**	3	∞

	A	B	C	D
A	0	6	10	**13**
B	6	0	4	7
C	10	4	0	3
D	**13**	7	3	0

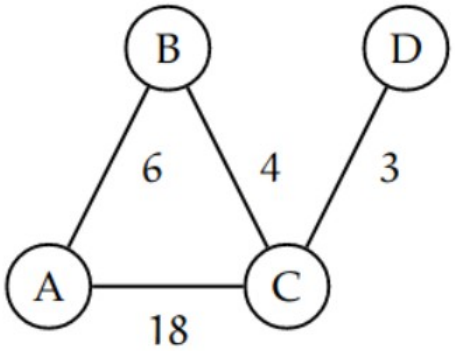
D

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	∞	∞	∞	∞
D	∞	∞	3	0

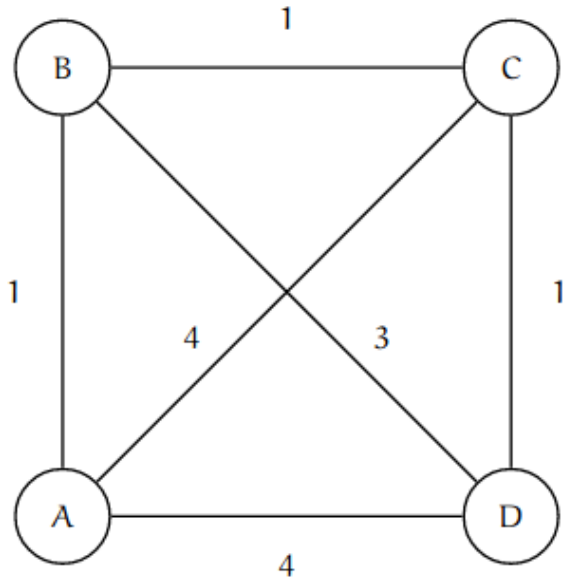
	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	**18**	**4**	**0**	**3**
D	**21**	**7**	3	0

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	**10**	4	0	3
D	**13**	7	3	0

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	10	4	0	3
D	13	7	3	0



Exam 2020-21, Question 3.3.3



Question 3.3.3: Now the cost of the edge between A and D changes to 1, would a re-run of the algorithm converge? If yes, how many iterations of the algorithm would be run before convergence and why? If no, explain your reasoning. *(Hint: it is recommended that you run the distance vector algorithm on paper to explore the answer.)*

Converged start state

A

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

B

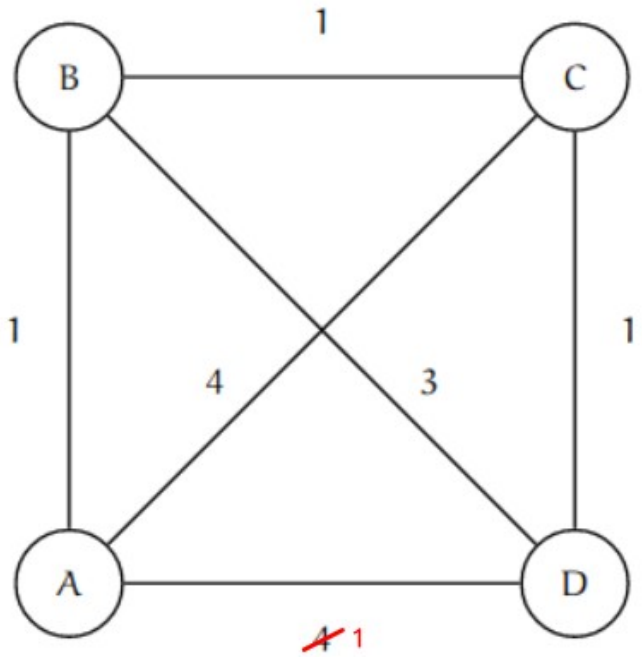
	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

C

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

D

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0



Converged start state

A detects link cost change

A

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	1
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

B

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

C

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

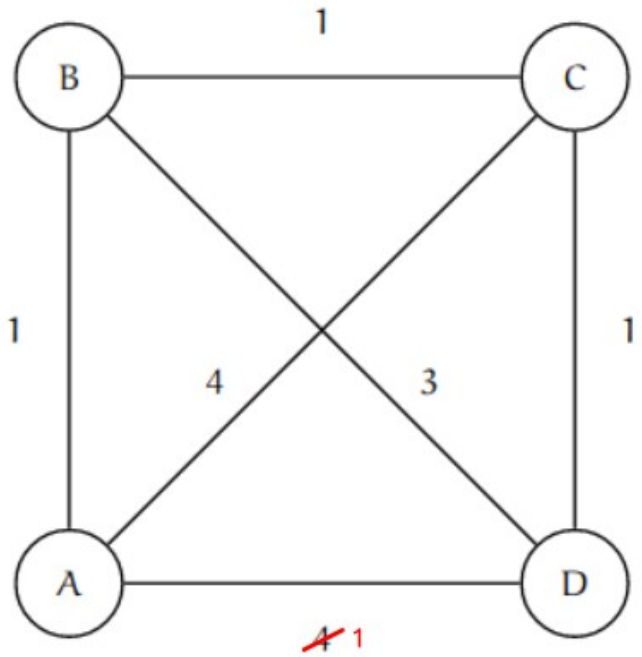
	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

D

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

Iteration 1



Converged start state

A detects link cost change

A propagates its changed distance vector. D updates is distance vector.

A

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	**1**
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	1
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

B

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	**1**
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

C

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	**1**
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

D

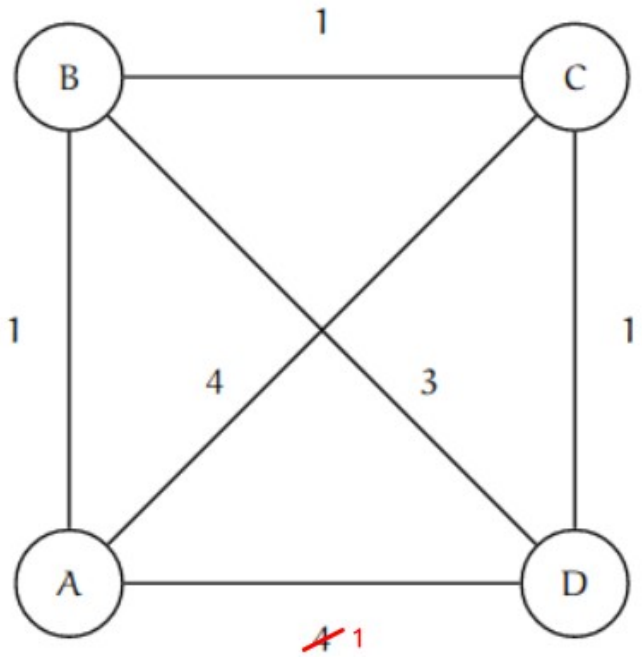
	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	**1**
B	1	0	1	2
C	2	1	0	1
D	**1**	2	1	0

Iteration 1

Iteration 2



Converged start state

A detects link cost change

A propagates its changed distance vector. D updates is distance vector.

D propagates its distance vector. Since no other nodes update their distance vector, no more distance vectors are sent. The algorithm has converged

A

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	**1**
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	1
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	1
B	1	0	1	2
C	2	1	0	1
D	**1**	2	1	0

B

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	**1**
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	1
B	1	0	1	2
C	2	1	0	1
D	**1**	2	1	0

C

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	**1**
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	1
B	1	0	1	2
C	2	1	0	1
D	**1**	2	1	0

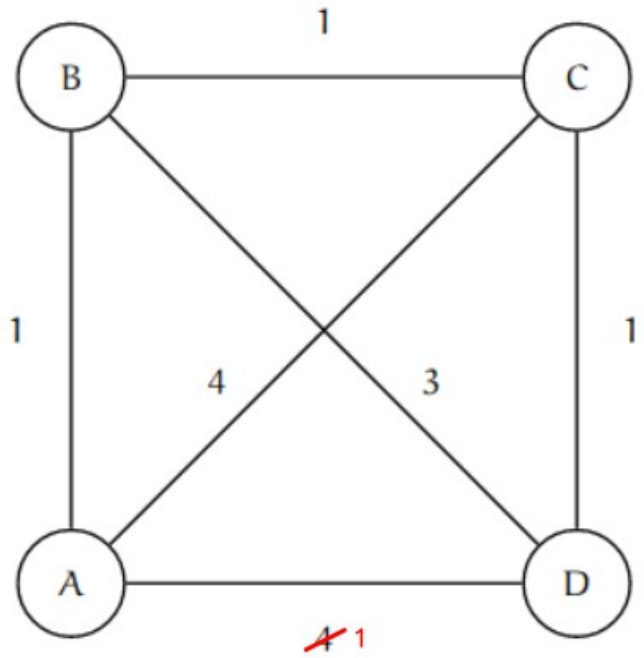
D

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	3
B	1	0	1	2
C	2	1	0	1
D	3	2	1	0

	A	B	C	D
A	0	1	2	**1**
B	1	0	1	2
C	2	1	0	1
D	**1**	2	1	0

	A	B	C	D
A	0	1	2	1
B	1	0	1	2
C	2	1	0	1
D	1	2	1	0

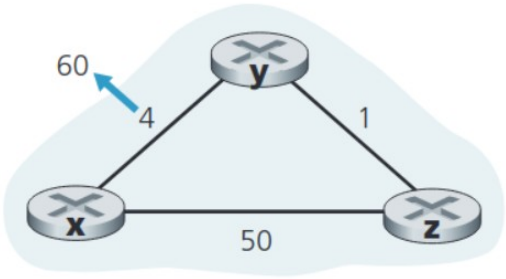


Iteration 1

Iteration 2

Iteration 3

DVA - Example with cost increase from K&R, p. 423,



b.

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

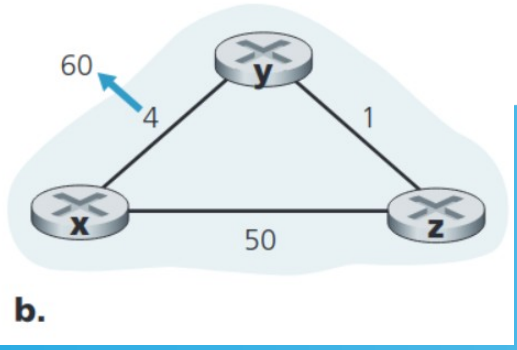
	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Iteration 1



b.

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Iteration 1

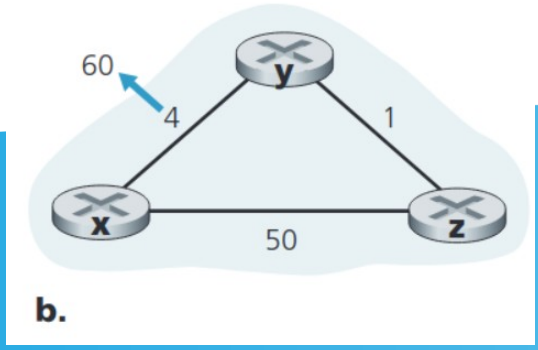
Y propogates its DV. Z calculates its D_z(x) based on the incorrect one received from Y.

	x	y	z
x	0	51	50
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	**6**	0	1
z	**7**	1	0

Iteration 2



b.

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Iteration 1

Y propogates its DV. Z calculates its D_z(x) based on the incorrect one received from Y.

	x	y	z
x	0	51	50
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	**7**	1	0

Iteration 2

Y updates its D_Y(x) based on the incorrect DV from Z

	x	y	z
x	0	51	50
y	6	0	1
z	**7**	1	0

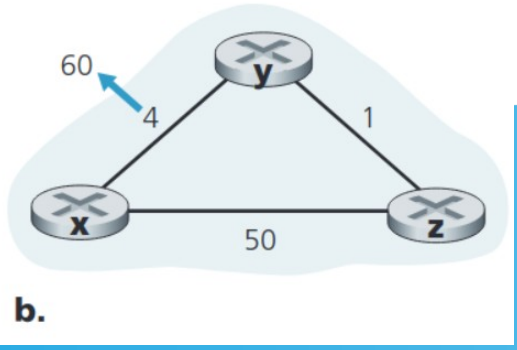
	x	y	z
x	0	51	50
y	**8**	0	1
z	**7**	1	0

	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

Iteration 3

And so on, back and forth.

. . .



$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Iteration 1

Y propogates its DV. Z calculates its D_z(x) based on the incorrect one received from Y.

	x	y	z
x	0	51	50
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	5	1	0

	x	y	z
x	0	51	50
y	6	0	1
z	**7**	1	0

Iteration 2

Y updates its D_Y(x) based on the incorrect DV from Z

	x	y	z
x	0	51	50
y	6	0	1
z	**7**	1	0

	x	y	z
x	0	51	50
y	**8**	0	1
z	**7**	1	0

	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

Iteration 3

And so on, back and forth.

. . .

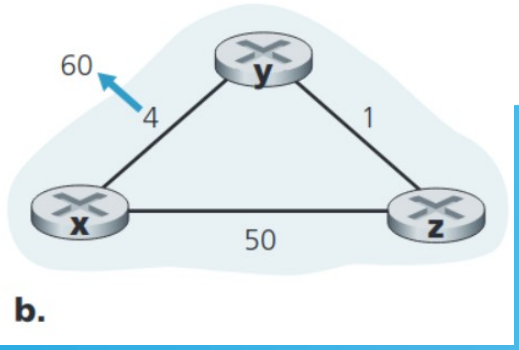
Until D_Y(x) is equal to the cost c(z,x)

	x	y	z
x	0	51	50
y	**50**	0	1
z	49	1	0

	x	y	z
x	0	51	50
y	50	0	1
z	49	1	0

	x	y	z
x	0	51	50
y	**50**	0	1
z	**50**	1	0

Iteration 46



b.

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Iteration 1

Y propogates its DV. Z calculates its D_z(x) based on the incorrect one received from Y.

	x	y	z
x	0	51	50
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	5	1	0

	x	y	z
x	0	51	50
y	6	0	1
z	**7**	1	0

Iteration 2

Y updates its D_Y(x) based on the incorrect DV from Z

	x	y	z
x	0	51	50
y	6	0	1
z	**7**	1	0

	x	y	z
x	0	51	50
y	**8**	0	1
z	**7**	1	0

	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

Iteration 3

And so on, back and forth.

. . .

Until D_Y(x) is equal to the cost c(z,x)

	x	y	z
x	0	51	50
y	**50**	0	1
z	49	1	0

	x	y	z
x	0	51	50
y	50	0	1
z	49	1	0

	x	y	z
x	0	51	50
y	**50**	0	1
z	**50**	1	0

Iteration 46

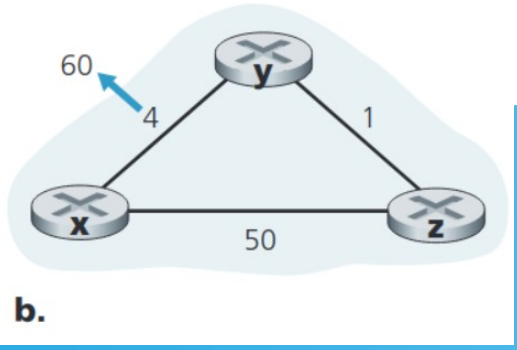
Z propagates its DV and Y updates its D_Y(x) to the correct value.

	x	y	z
x	0	51	50
y	50	0	1
z	**50**	1	0

	x	y	z
x	0	51	50
y	**51**	0	1
z	**50**	1	0

	x	y	z
x	0	51	50
y	50	0	1
z	50	1	0

Iteration 47



b.

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

X

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Iteration 1

Y propogates its DV. Z calculates its D_z(x) based on the incorrect one received from Y.

X

	x	y	z
x	0	51	50
y	**6**	0	1
z	5	1	0

Y

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	5	1	0

Z

	x	y	z
x	0	51	50
y	**6**	0	1
z	**7**	1	0

Iteration 2

Y updates its D_Y(x) based on the incorrect DV from Z

X

	x	y	z
x	0	51	50
y	6	0	1
z	**7**	1	0

Y

	x	y	z
x	0	51	50
y	**8**	0	1
z	**7**	1	0

Z

	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

Iteration 3

And so on, back and forth.

. . .

Until D_Y(x) is equal to the cost c(z,x)

X

	x	y	z
x	0	51	50
y	**50**	0	1
z	49	1	0

Y

	x	y	z
x	0	51	50
y	50	0	1
z	49	1	0

Z

	x	y	z
x	0	51	50
y	**50**	0	1
z	**50**	1	0

Iteration 46

Z propagates its DV and Y updates its D_Y(x) to the correct value.

X

	x	y	z
x	0	51	50
y	50	0	1
z	**50**	1	0

Y

	x	y	z
x	0	51	50
y	**51**	0	1
z	**50**	1	0

Z

	x	y	z
x	0	51	50
y	50	0	1
z	**50**	1	0

Iteration 47

After Y has propagated its DV, no other DVs are being updated. The algorithm has converged.

X

	x	y	z
x	0	51	50
y	**51**	0	1
z	50	1	0

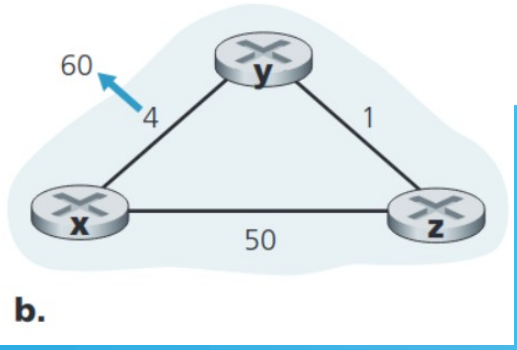
Y

	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

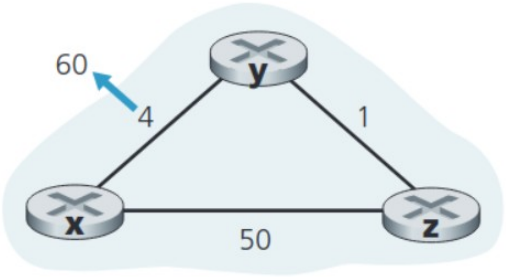
Z

	x	y	z
x	0	51	50
y	**51**	0	1
z	50	1	0

Iteration 48



Adding a poisoned reverse, to solve the looping problem.



b.

DVA with poisoned reverse.

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

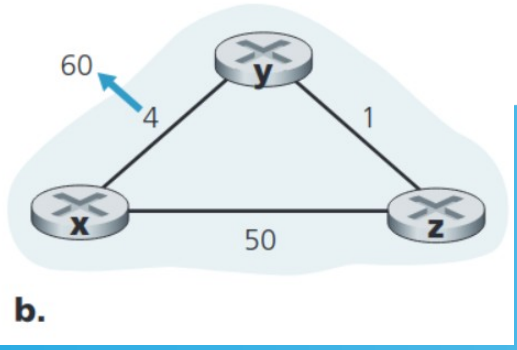
	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Iteration 1



b.

DVA with poisoned reverse.

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

Y propogates its DV. Z calculates its D_z(x) based on the incorrect one received from Y.

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

	x	y	z
x	0	51	50
y	**6**	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	5	1	0

Z

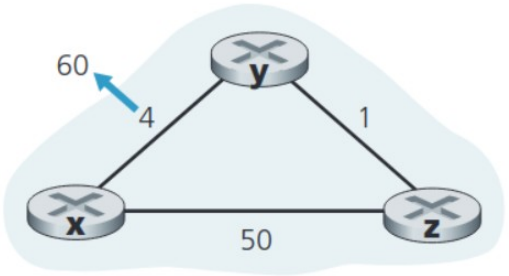
	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Iteration 1

	x	y	z
x	0	**51**	**50**
y	**6**	0	1
z	**7**	1	0

Iteration 2



b.

DVA with poisoned reverse.

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Iteration 1

Y propogates its DV. Z calculates its D_z(x) based on the incorrect one received from Y.

	x	y	z
x	0	51	50
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	**7**	1	0

Iteration 2

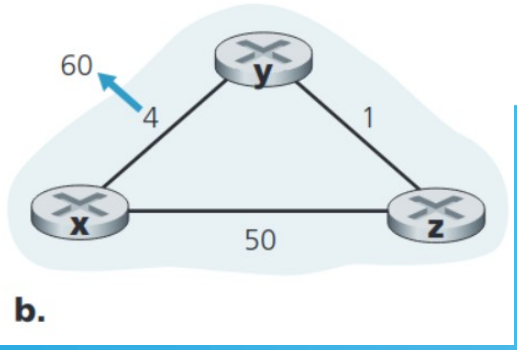
Because z is routing through y, it sets D_z(x) = ∞

	x	y	z
x	0	51	50
y	6	0	1
z	**7**	1	0

	x	y	z
x	0	51	50
y	**60**	0	1
z	∞	1	0

	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

Iteration 3



b.

DVA with poisoned reverse.

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

Y propogates its DV. Z calculates its D_z(x) based on the incorrect one received from Y.

Because z is routing through y, it sets D_z(x) = ∞

Z receives the correct DV from y and updates its DV to the right value.

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

	x	y	z
x	0	51	50
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	51	50
y	6	0	1
z	**7**	1	0

	x	y	z
x	0	51	50
y	**60**	0	1
z	**7**	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	5	1	0

	x	y	z
x	0	51	50
y	**60**	0	1
z	∞	1	0

	x	y	z
x	0	51	50
y	**8**	0	1
z	∞	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

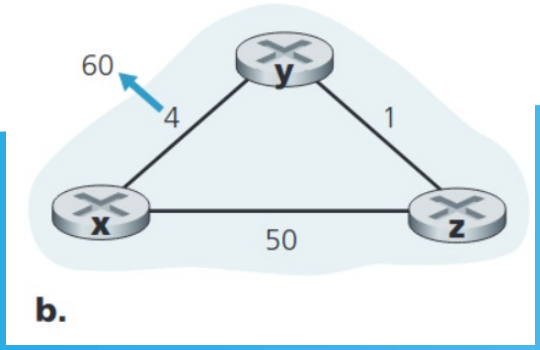
	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

	x	y	z
x	0	**51**	**50**
y	**6**	0	1
z	**7**	1	0

	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

	x	y	z
x	0	51	50
y	**60**	0	1
z	**50**	1	0

Iteration 1 | Iteration 2 | Iteration 3 | Iteration 4



DVA with poisoned reverse.

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

X

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Iteration 1

Y propogates its DV. Z calculates its D_z(x) based on the incorrect one received from Y.

X

	x	y	z
x	0	51	50
y	**6**	0	1
z	5	1	0

Y

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	5	1	0

Z

	x	y	z
x	0	**51**	**50**
y	**6**	0	1
z	**7**	1	0

Iteration 2

Because z is routing through y, it sets D_z(x) = ∞

X

	x	y	z
x	0	51	50
y	6	0	1
z	**7**	1	0

Y

	x	y	z
x	0	51	50
y	**60**	0	1
z	∞	1	0

Z

	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

Iteration 3

Z receives the correct DV from y and updates its DV to the right value.

X

	x	y	z
x	0	51	50
y	**60**	0	1
z	**7**	1	0

Y

	x	y	z
x	0	51	50
y	60	0	1
z	∞	1	0

Z

	x	y	z
x	0	51	50
y	**60**	0	1
z	**50**	1	0

Iteration 4

Z, no longer routing through y, propagates DV, and Y updates to the correct value.

X

	x	y	z
x	0	51	50
y	60	0	1
z	**50**	1	0

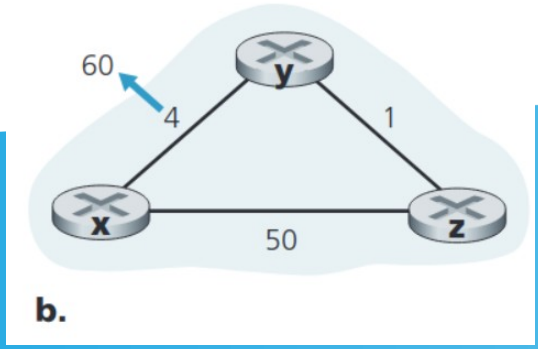
Y

	x	y	z
x	0	51	50
y	**51**	0	1
z	**50**	1	0

Z

	x	y	z
x	0	51	50
y	60	0	1
z	50	1	0

Iteration 5



DVA with poisoned reverse.

$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \} \quad \text{for each node } y \text{ in } N$$

Converged start state

X

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

X and Y detect link cost change. Y "thinks" there is a cheaper way to x based on D_z(x)

X

	x	y	z
x	0	**51**	**50**
y	4	0	1
z	5	1	0

Y

	x	y	z
x	0	4	5
y	**6**	0	1
z	5	1	0

Z

	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Iteration 1

Y propogates its DV. Z calculates its D_z(x) based on the incorrect one received from Y.

X

	x	y	z
x	0	51	50
y	**6**	0	1
z	5	1	0

Y

	x	y	z
x	0	**51**	**50**
y	6	0	1
z	5	1	0

Z

	x	y	z
x	0	**51**	**50**
y	**6**	0	1
z	**7**	1	0

Iteration 2

Because z is routing through y, it sets D_z(x) = ∞

X

	x	y	z
x	0	51	50
y	6	0	1
z	**7**	1	0

Y

	x	y	z
x	0	51	50
y	**60**	0	1
z	∞	1	0

Z

	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

Iteration 3

Z receives the correct DV from y and updates its DV to the right value.

X

	x	y	z
x	0	51	50
y	**60**	0	1
z	**7**	1	0

Y

	x	y	z
x	0	51	50
y	**8**	0	1
z	∞	1	0

Z

	x	y	z
x	0	51	50
y	**60**	0	1
z	**50**	1	0

Iteration 4

Z, no longer routing through y, propagates DV, and Y updates to the correct value.

X

	x	y	z
x	0	51	50
y	60	0	1
z	**50**	1	0

Y

	x	y	z
x	0	51	50
y	**51**	0	1
z	50	1	0

Z

	x	y	z
x	0	51	50
y	60	0	1
z	50	1	0

Iteration 5

Y propagetes, and no more DVs are being updated. The algorithm has converged after only 6 iterations.

X

	x	y	z
x	0	51	50
y	**51**	0	1
z	50	1	0

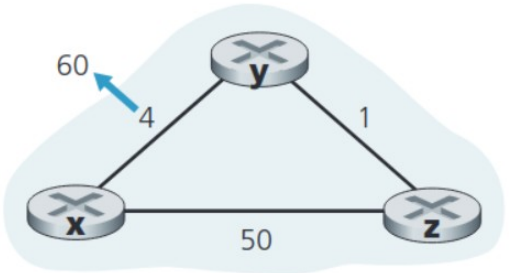
Y

	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

Z

	x	y	z
x	0	51	50
y	**51**	0	1
z	50	1	0

Iteration 6



b.