Networks: Tutorial 09

Shinnazar Seytnazarov, PhD

Innopolis University

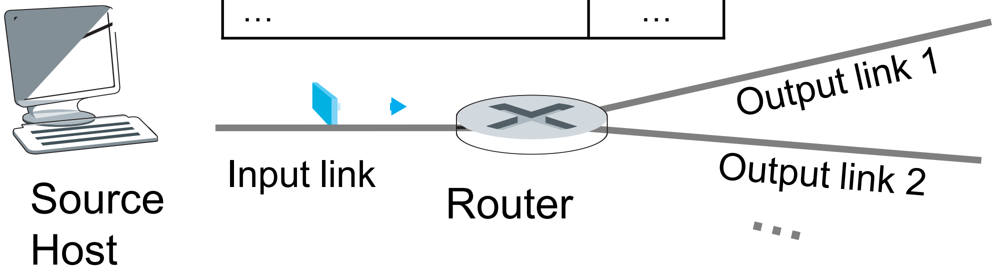
s.seytnazarov@innopolis.ru

February 14, 2022

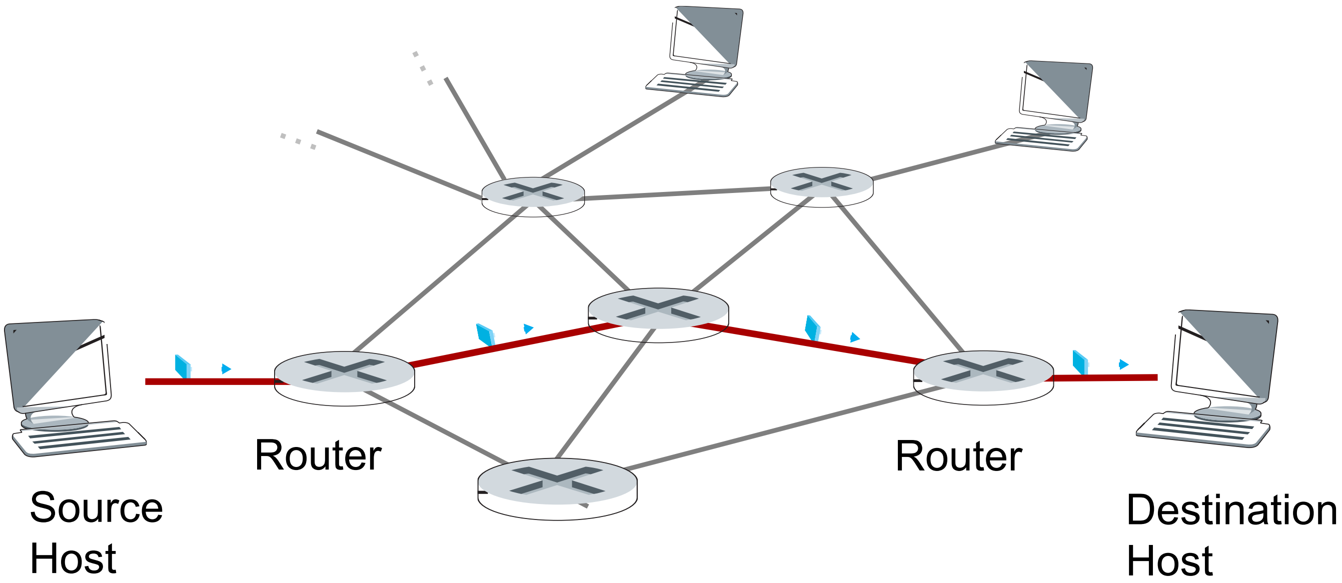
Recap

Forwarding – a local action at a router level (to forward packets to some output port)

Destination Address Range	Output Link
Address Prefix 1	3
Address Prefix 2	2
…	…

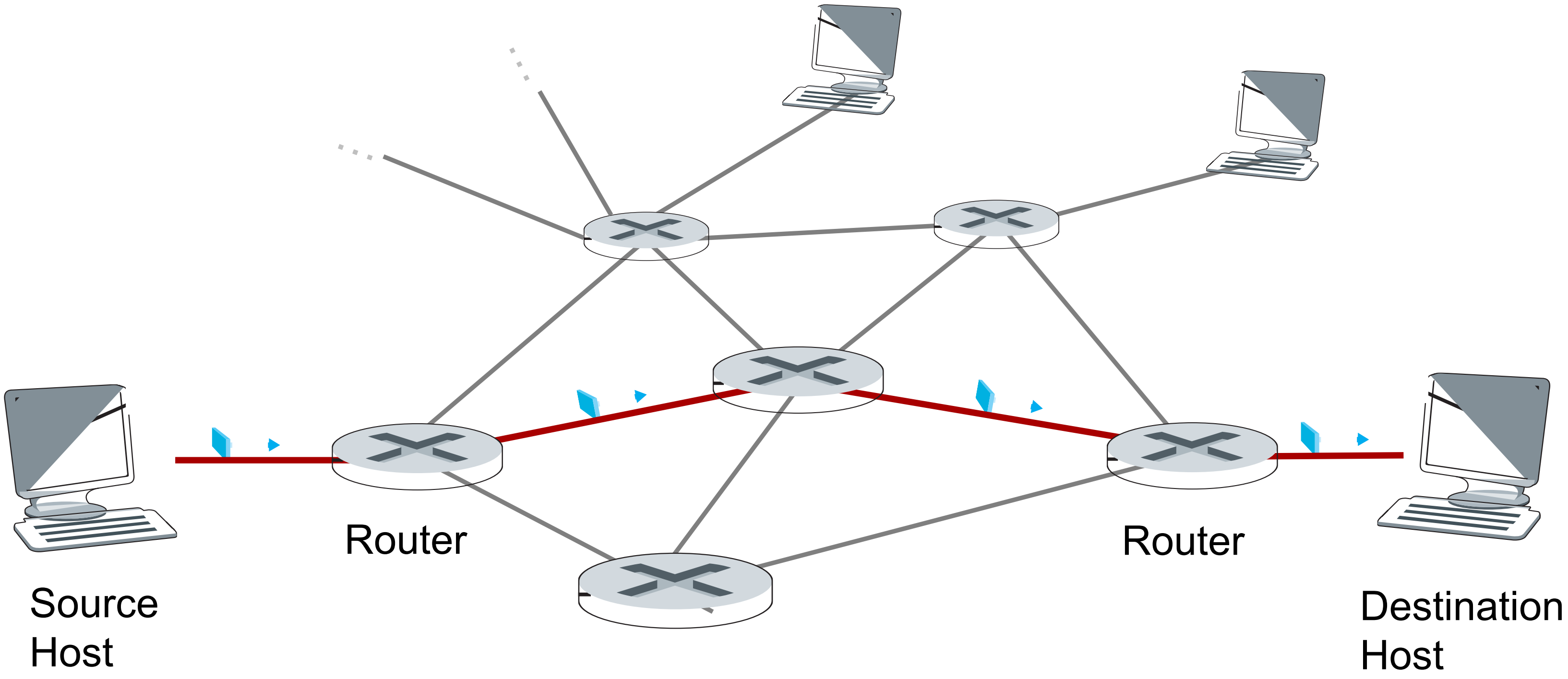


Routing – a network-wide action, to compute the least-cost path



Routing problem – to compute the least-cost path between network hosts

Routing algorithm – the solution to this problem



Routing algorithms represent a network as a graph:

- graph nodes are routers, and
- edges are the communication links between routers

A routing algorithm computes the least-cost paths between network hosts,
to compute forwarding tables at each router

The Classification of Routing Algorithms

Characteristic	Description
Global / Decentralized	
Static / Dynamic	
Sensitivity to Network Load	

A routing algorithm computes the least-cost paths between network hosts, to allow forwarding tables at each router

The Classification of Routing Algorithms

Characteristic	Description
Global / Decentralized	**Global** (or centralized), e.g. the Link-state alg.: - Each node knows the topology of an entire network; - Each node communicates to any other network node (via broadcast messages)
Static / Dynamic	
Sensitivity to Network Load	

A routing algorithm computes the least-cost paths between network hosts,
to allow forwarding tables at each router

The Classification of Routing Algorithms

Characteristic	Description
Global / Decentralized	**Global** (or centralized), e.g. the Link-state alg.: - Each node knows the topology of an entire network; - Each node communicates to any other network node (via broadcast messages) **Decentralized** (or distributed), e.g. the dynamic-vector alg.: - A distributed manner of computation, that is each node contributes to the final result; - Each node communicates to its neighbors only
Static / Dynamic	
Sensitivity to Network Load	

A routing algorithm computes the least-cost paths between network hosts, to allow forwarding tables at each router

The Classification of Routing Algorithms

Characteristic	Description
Global / Decentralized	**Global** (or centralized), e.g. the Link-state alg.: - Each node knows the topology of an entire network; - Each node communicates to any other network node (via broadcast messages) **Decentralized** (or distributed), e.g. the dynamic-vector alg.: - A distributed manner of computation, that is each node contributes to the final result; - Each node communicates to its neighbors only
Static / Dynamic	**Static** – routes are recomputed very rarely over time, and usually by hand (changes in a forwarding table);
Sensitivity to Network Load	

A routing algorithm computes the least-cost paths between network hosts, to allow forwarding tables at each router

The Classification of Routing Algorithms

Characteristic	Description
Global / Decentralized	**Global** (or centralized), e.g. the Link-state alg.: - Each node knows the topology of an entire network; - Each node communicates to any other network node (via broadcast messages) **Decentralized** (or distributed), e.g. the dynamic-vector alg.: - A distributed manner of computation, that is each node contributes to the final result; - Each node communicates to its neighbors only
Static / Dynamic	**Static** – routes are recomputed very rarely over time, and usually by hand (changes in a forwarding table); **Dynamic** – routes are updated frequently, to reflect changes in a network load (congestion) or topology
Sensitivity to Network Load	

A routing algorithm computes the least-cost paths between network hosts, to allow forwarding tables at each router

The Classification of Routing Algorithms

Characteristic	Description
Global / Decentralized	**Global** (or centralized), e.g. the Link-state alg.: - Each node knows the topology of an entire network; - Each node communicates to any other network node (via broadcast messages) **Decentralized** (or distributed), e.g. the dynamic-vector alg.: - A distributed manner of computation, that is each node contributes to the final result; - Each node communicates to its neighbors only
Static / Dynamic	**Static** – routes are recomputed very rarely over time, and usually by hand (changes in a forwarding table); **Dynamic** – routes are updated frequently, to reflect changes in a network load (congestion) or topology
Sensitivity to Network Load	**Load-sensitive** – link costs may change dynamically, mainly to reflect network congestion;

A routing algorithm computes the least-cost paths between network hosts, to allow forwarding tables at each router

The Classification of Routing Algorithms

Characteristic	Description
Global / Decentralized	**Global** (or centralized), e.g. the Link-state alg.: - Each node knows the topology of an entire network; - Each node communicates to any other network node (via broadcast messages) **Decentralized** (or distributed), e.g. the dynamic-vector alg.: - A distributed manner of computation, that is each node contributes to the final result; - Each node communicates to its neighbors only
Static / Dynamic	**Static** – routes are recomputed very rarely over time, and usually by hand (changes in a forwarding table); **Dynamic** – routes are updated frequently, to reflect changes in a network load (congestion) or topology
Sensitivity to Network Load	**Load-sensitive** – link costs may change dynamically, mainly to reflect network congestion; **Load-insensitive** – no dynamic change of links costs (most of the routing algorithms)

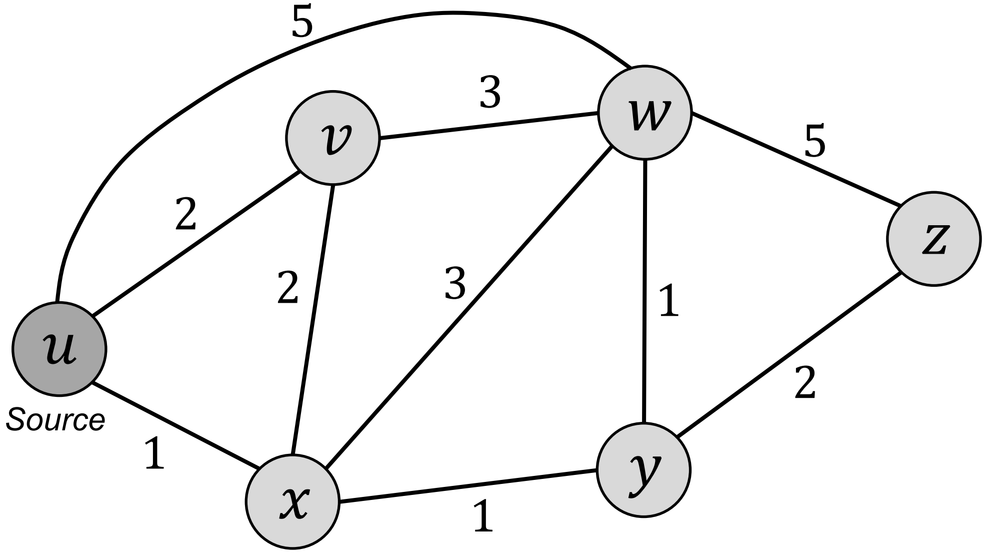
A routing algorithm computes the least-cost paths between network hosts, to allow forwarding tables at each router

The Classification of Routing Algorithms

Characteristic	Description
Global / Decentralized	**Global** (or centralized), e.g. the Link-state alg.: - Each node knows the topology of an entire network; - Each node communicates to any other network node (via broadcast messages) **Decentralized** (or distributed), e.g. the distance vector alg.: - A distributed manner of computation, that is each node contributes to the final result; - Each node communicates to its neighbors only
Static / Dynamic	**Static** – routes are recomputed very rarely over time, and usually by hand (changes in a forwarding table); **Dynamic** – routes are updated frequently, to reflect changes in a network load (congestion) or topology
Sensitivity to Network Load	**Load-sensitive** – link costs may change dynamically, mainly to reflect network congestion; **Load-insensitive** – no dynamic change of links costs (most of the routing algorithms)

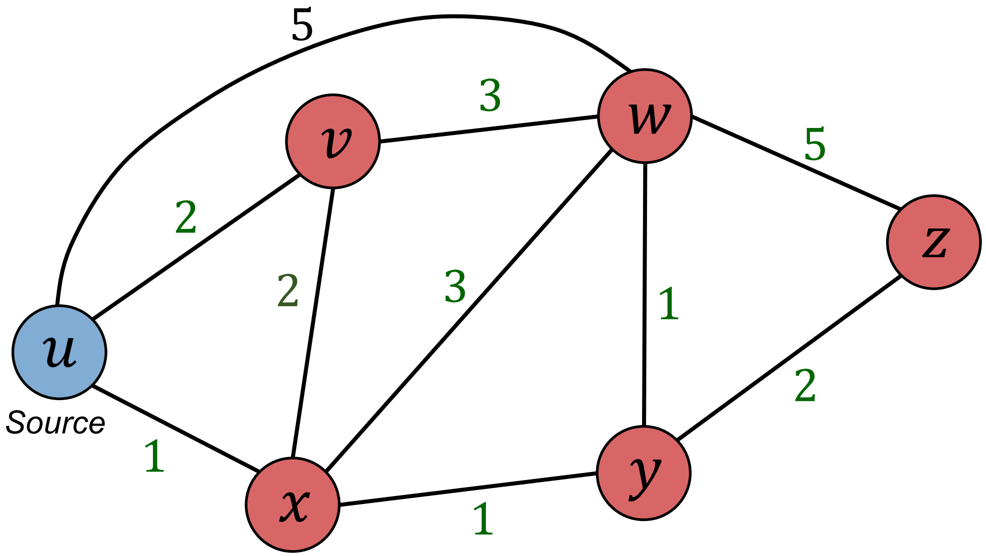
Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z

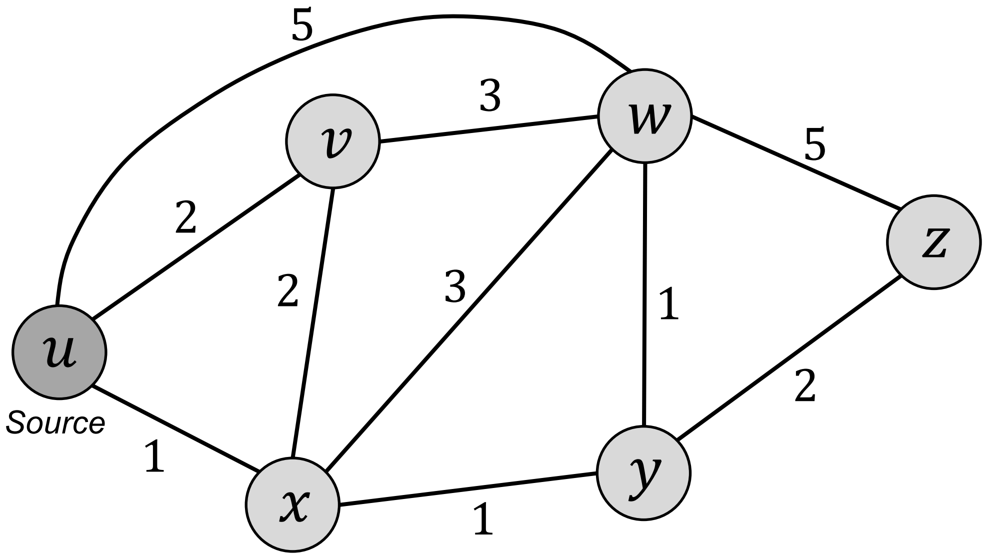


Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

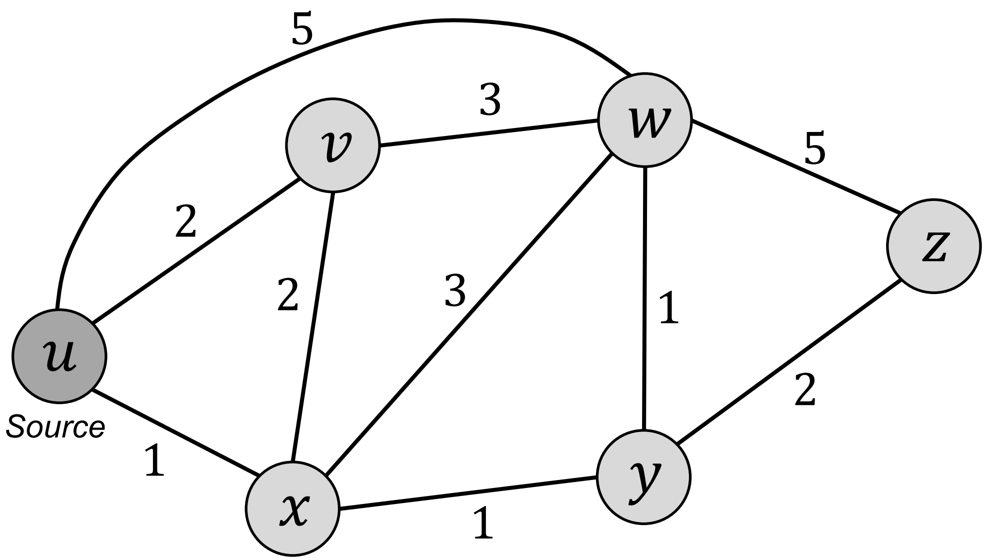
N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Steps of Dijkstra's algorithm:

	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$

Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Steps of Dijkstra's algorithm:

	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0						

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

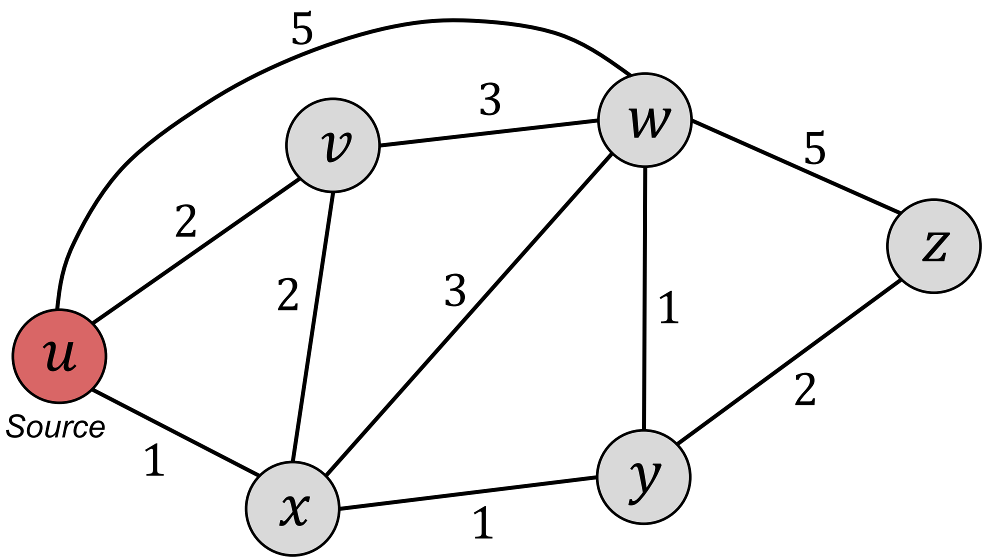
N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

```
1: $N' \leftarrow \{u\}$
2: For all nodes $i \in \{v, x, w, y, z\}$ do
3:     If $i$ – the neighbour of $u$ then
4:         $D(i) \leftarrow c(u, i)$
5:     Else   $D(i) \leftarrow \infty$
```

Link-State Routing Algorithm: Dijkstra’s Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

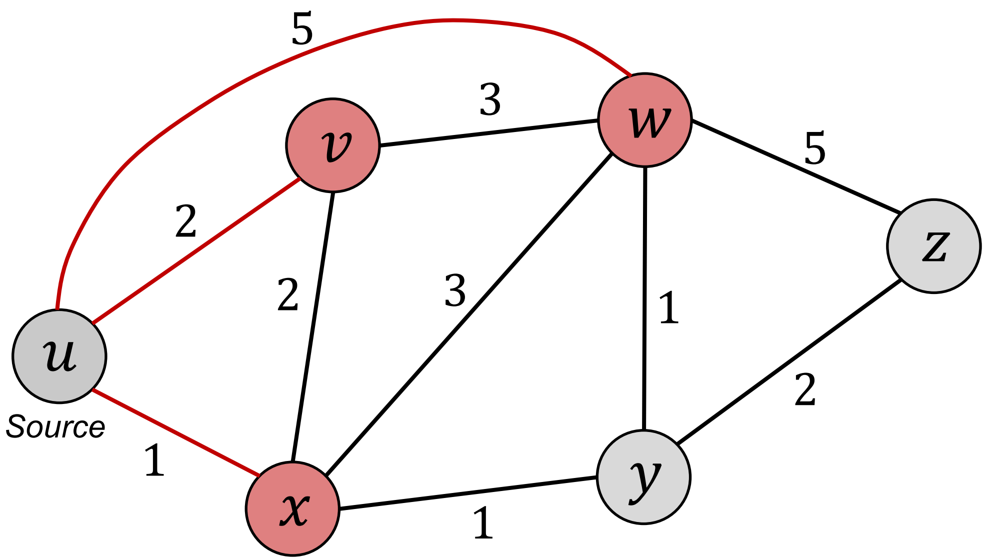
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else  $D(i) \leftarrow \infty$
```

Steps of Dijkstra’s algorithm:

	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	$\{\boldsymbol{u}\}$					

Link-State Routing Algorithm: Dijkstra’s Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

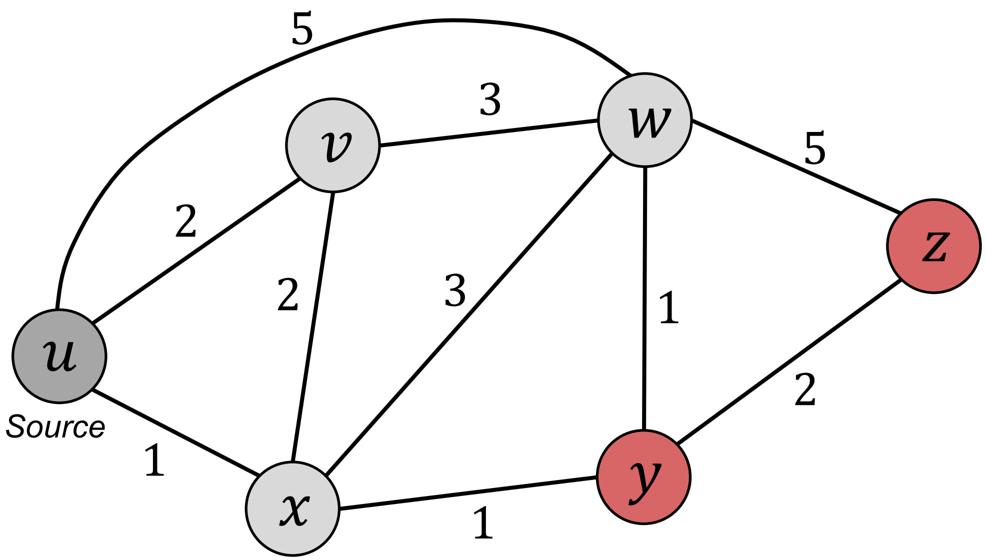
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else  $D(i) \leftarrow \infty$
```

Steps of Dijkstra’s algorithm:

	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	$\{u\}$	$\mathbf{2, u}$	$\mathbf{5, u}$	$\mathbf{1, u}$		

Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

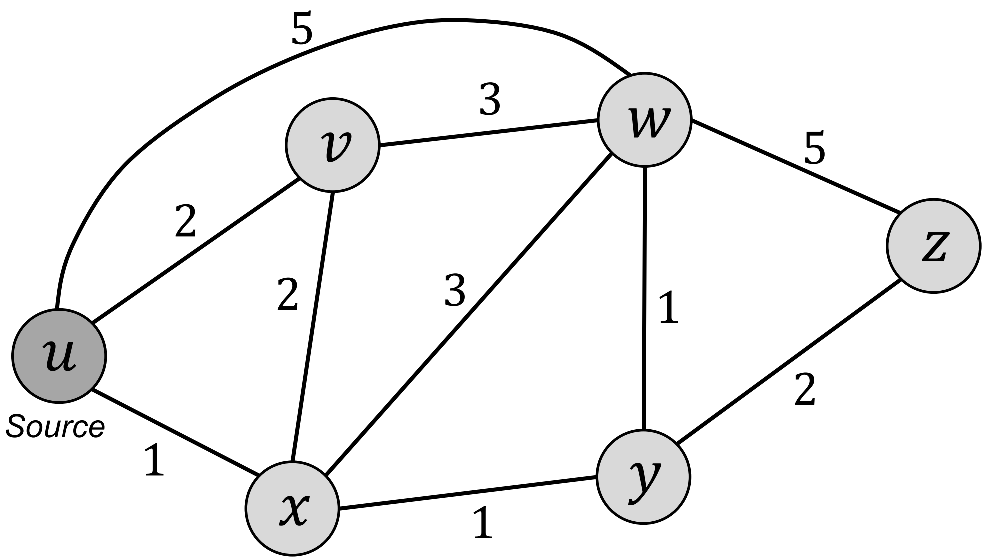
```
1: $N' \leftarrow \{u\}$
2: For all nodes $i \in \{v, x, w, y, z\}$ do
3:     If $i$ – the neighbour of $u$ then
4:         $D(i) \leftarrow c(u, i)$
5:     Else   $D(i) \leftarrow \infty$
```

Steps of Dijkstra's algorithm:

	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	$\{u\}$	$2, u$	$5, u$	$1, u$	∞	∞

Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Steps of Dijkstra's algorithm:

	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	$\{u\}$	$2, u$	$5, u$	$1, u$	∞	∞

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

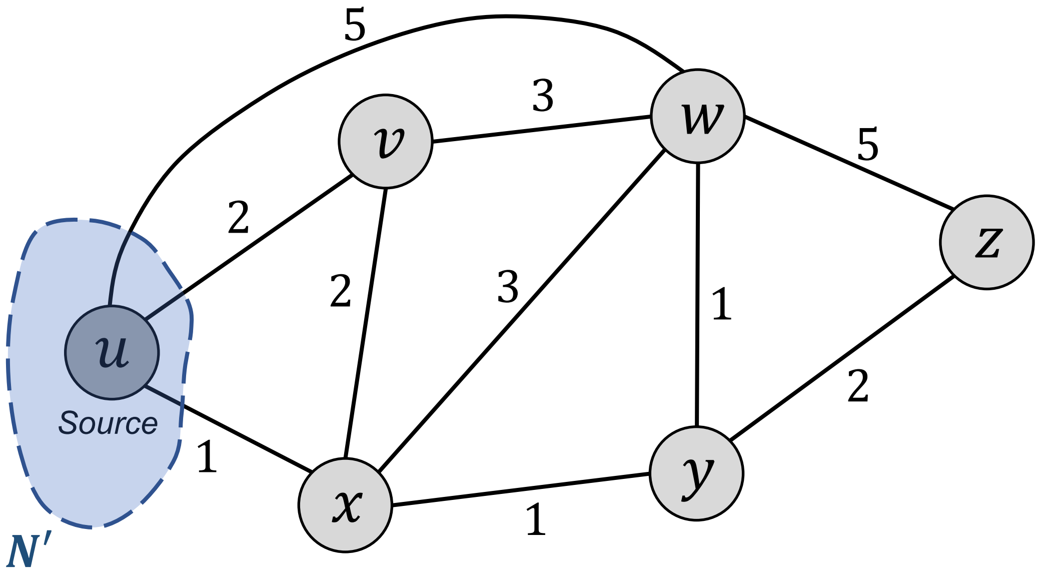
```
$N' \leftarrow \{u\}$
**For** all nodes $i \in \{v, x, w, y, z\}$ do
    **If** $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    **Else**  $D(i) \leftarrow \infty$
```

Iterations:

```
**Loop**
    **Find** $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    **For** each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
**Until**  $N' = N$
```

Link-State Routing Algorithm: Dijkstra’s Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Steps of Dijkstra’s algorithm:

	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	$\{u\}$	$2, u$	$5, u$	$1, u$	∞	∞

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

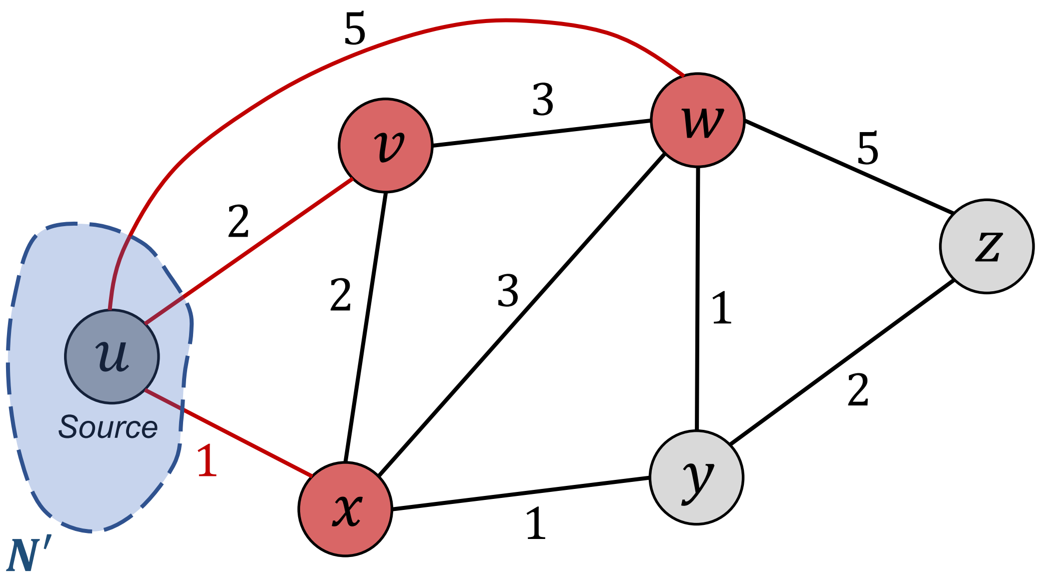
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else  $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
Until  $N' = N$
```

Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Candidate nodes are highlighted in red

Steps of Dijkstra's algorithm:

	N'	$D(v),p(v)$	$D(w),p(w)$	$D(x),p(x)$	$D(y),p(y)$	$D(z),p(z)$
0	$\{u\}$	$2,u$	$5,u$	$1,u$	∞	∞

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i,k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

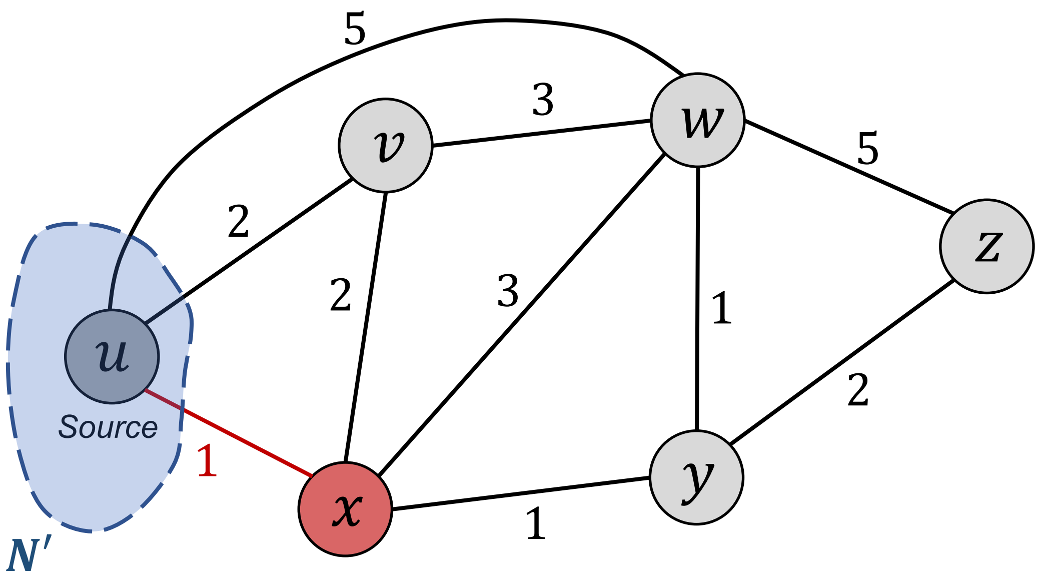
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u,i)$
    Else  $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i,k)\}$
Until  $N' = N$
```

Link-State Routing Algorithm: Dijkstra’s Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



$D(x)$ is minimal

Steps of Dijkstra’s algorithm:

	N'	$D(v),p(v)$	$D(w),p(w)$	$D(x),p(x)$	$D(y),p(y)$	$D(z),p(z)$
0	$\{u\}$	$2,u$	$5,u$	$1,u$	∞	∞

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

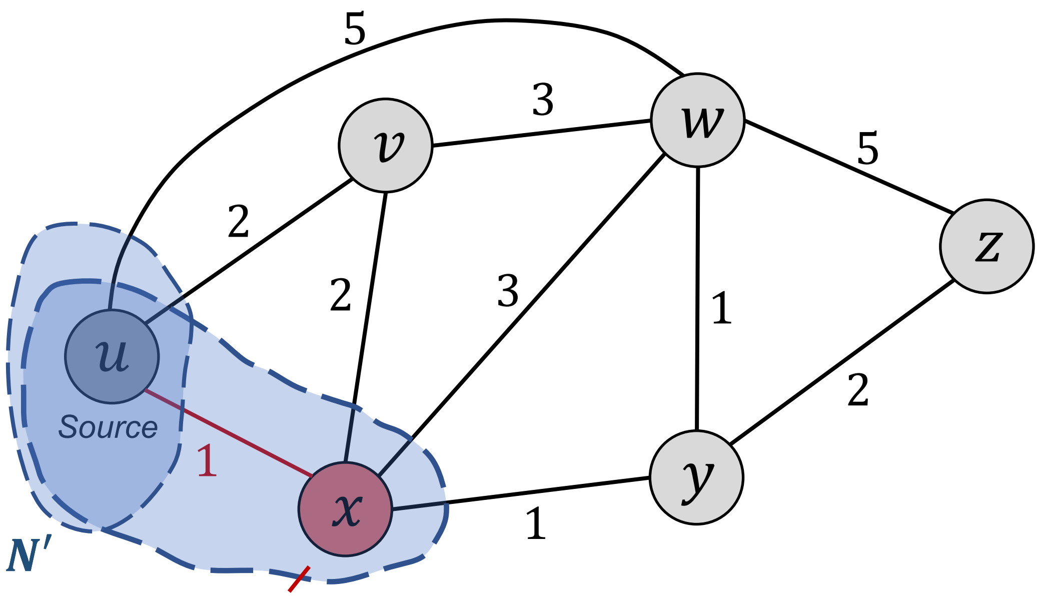
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else   $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
Until  $N' = N$
```

Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Steps of Dijkstra's algorithm:

	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	$\{\boldsymbol{u}, \boldsymbol{x}\}$					

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

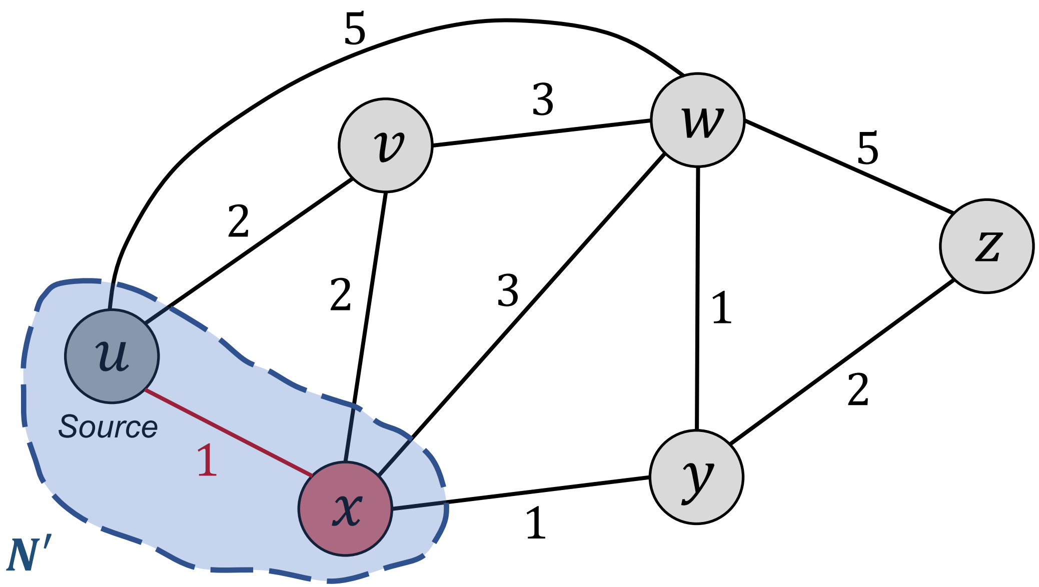
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else  $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
Until  $N' = N$
```

Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Steps of Dijkstra's algorithm:

	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	$\{\boldsymbol{u}, \boldsymbol{x}\}$					

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

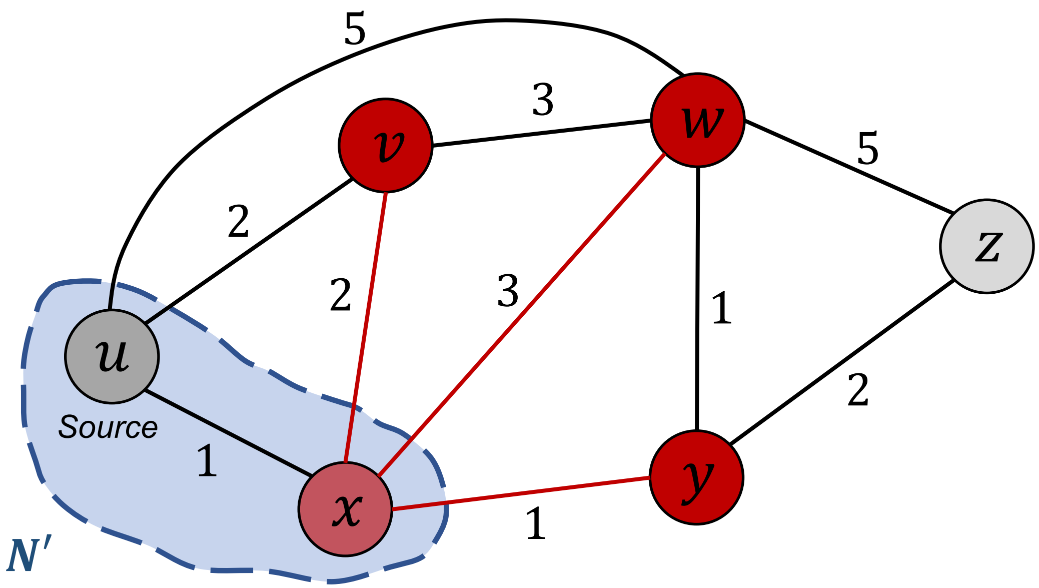
```
$N' \leftarrow \{u\}$
**For** all nodes $i \in \{v, x, w, y, z\}$ do
    **If** $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    **Else**  $D(i) \leftarrow \infty$
```

Iterations:

```
**Loop**
    **Find** $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    **For each neighbour** $k$ **of** $i$, **with** $k \notin N'$ **do**
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
**Until**  $N' = N$
```

Link-State Routing Algorithm: Dijkstra’s Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Steps of Dijkstra’s algorithm:

	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	$\{u, x\}$					

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

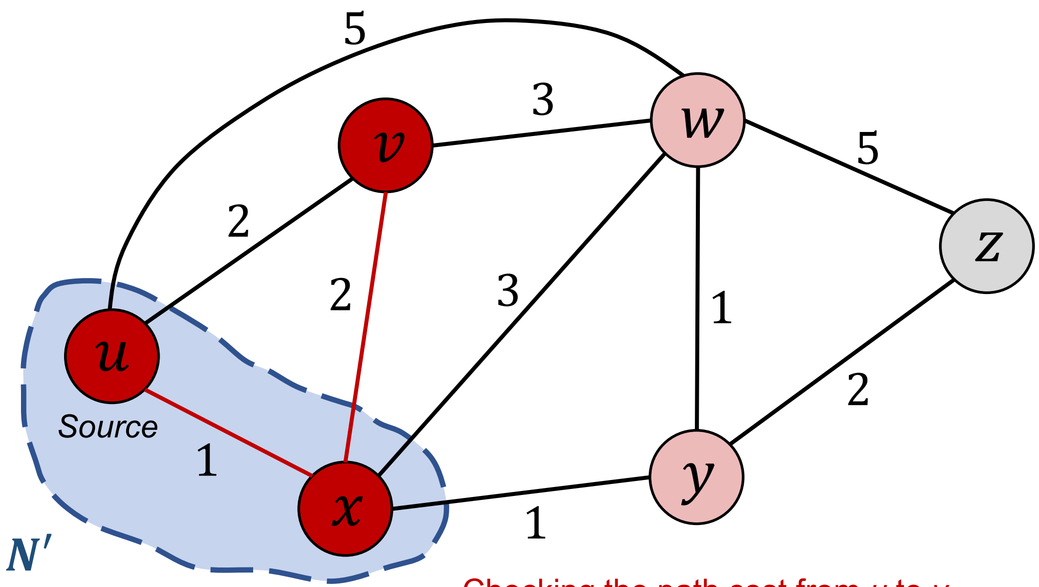
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else  $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
Until  $N' = N$
```

Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Checking the path cost from *u* to *v* through *x;* the length of a new path is 1+2=3; keeping the old path as the least-cost

Steps of Dijkstra's algorithm:

	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	$\boldsymbol{\{u, x\}}$	$\boldsymbol{2, u}$				

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

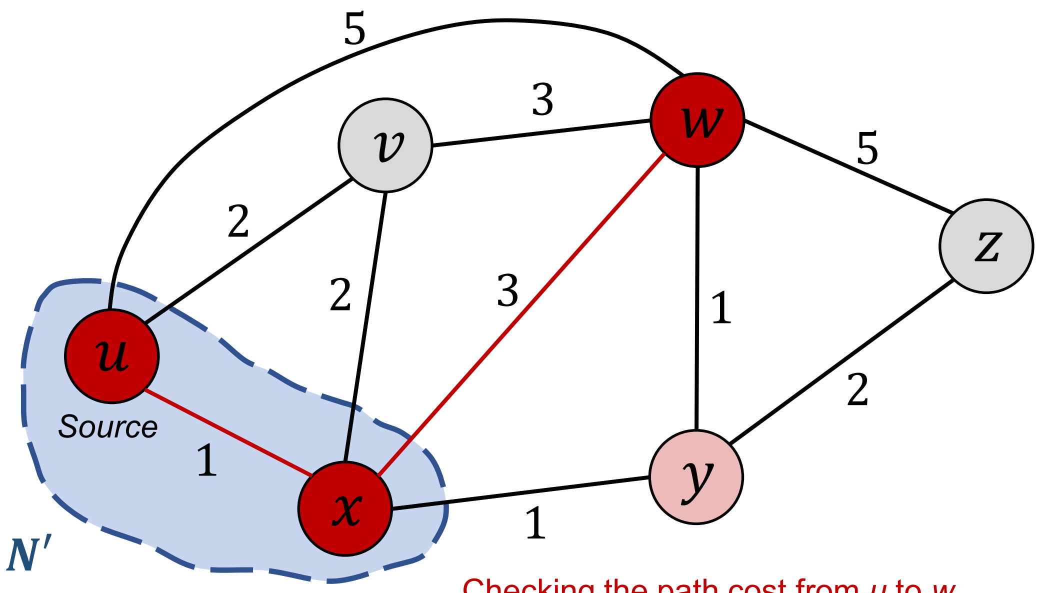
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
Until $N' = N$
```

Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Checking the path cost from *u* to *w* through *x;* the length of a new path is 1+3=4; updating the least-cost path

Steps of Dijkstra's algorithm:

	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	$\{\boldsymbol{u}, \boldsymbol{x}\}$	$\mathbf{2}, \boldsymbol{u}$	$\mathbf{4}, \boldsymbol{x}$			

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

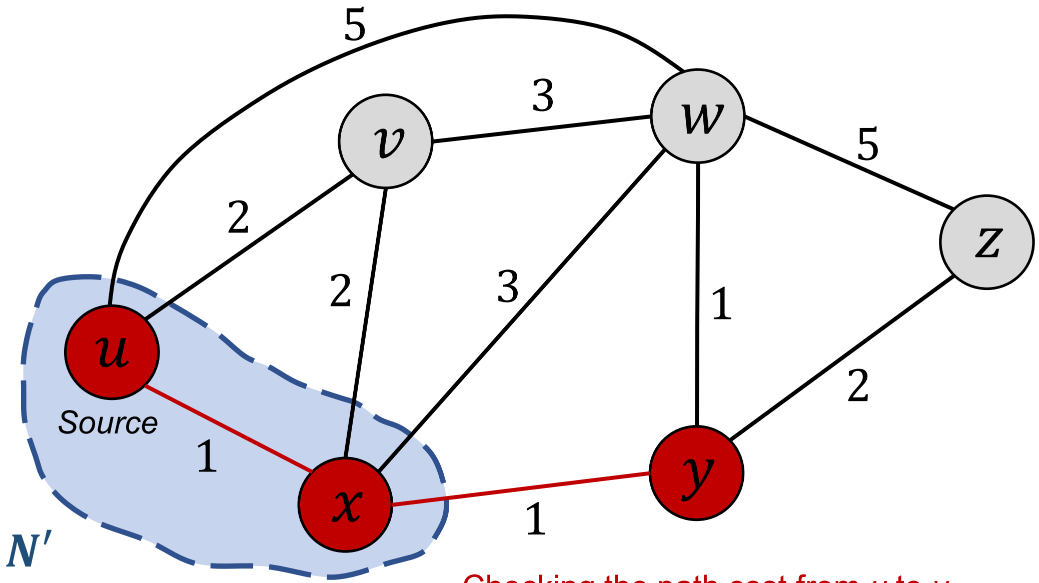
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else  $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
Until  $N' = N$
```

Example of Using Link-State Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Checking the path cost from *u* to *y* through *x;* the length of a new path is 1+1=2; updating the least-cost path

Steps of Dijkstra's algorithm:

	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	$\{\boldsymbol{u}, \boldsymbol{x}\}$	$\mathbf{2}, \boldsymbol{u}$	$\mathbf{4}, \boldsymbol{x}$		$\mathbf{2}, \boldsymbol{x}$	

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

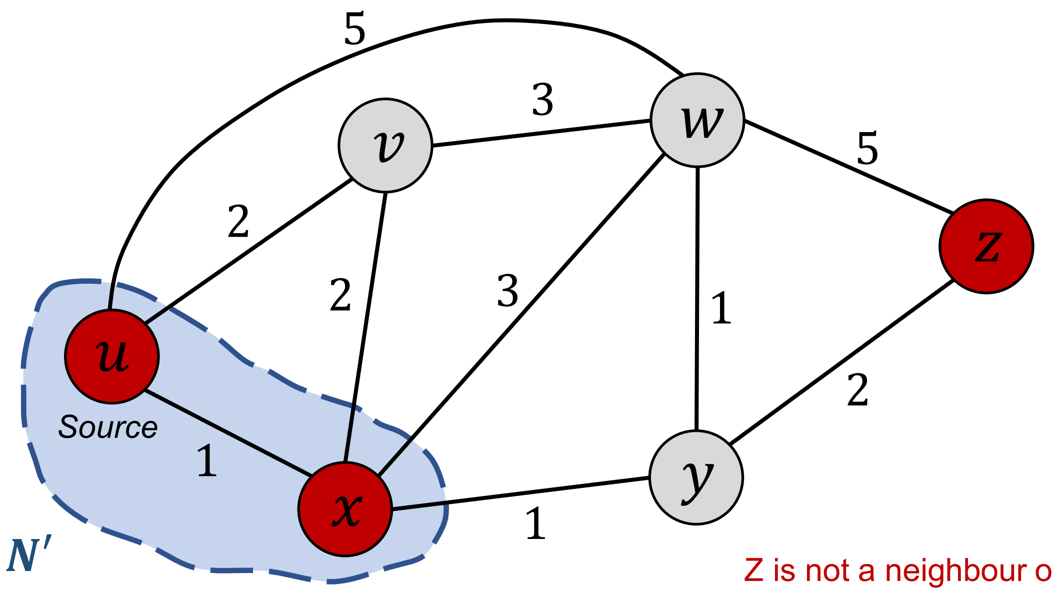
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else   $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
Until  $N' = N$
```

Link-State Routing Algorithm: Dijkstra’s Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Z is not a neighbour of *x*; no updates

Steps of Dijkstra’s algorithm:

	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	$\{\boldsymbol{u}, \boldsymbol{x}\}$	$\boldsymbol{2, u}$	$\boldsymbol{4, x}$		$\boldsymbol{2, x}$	∞

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

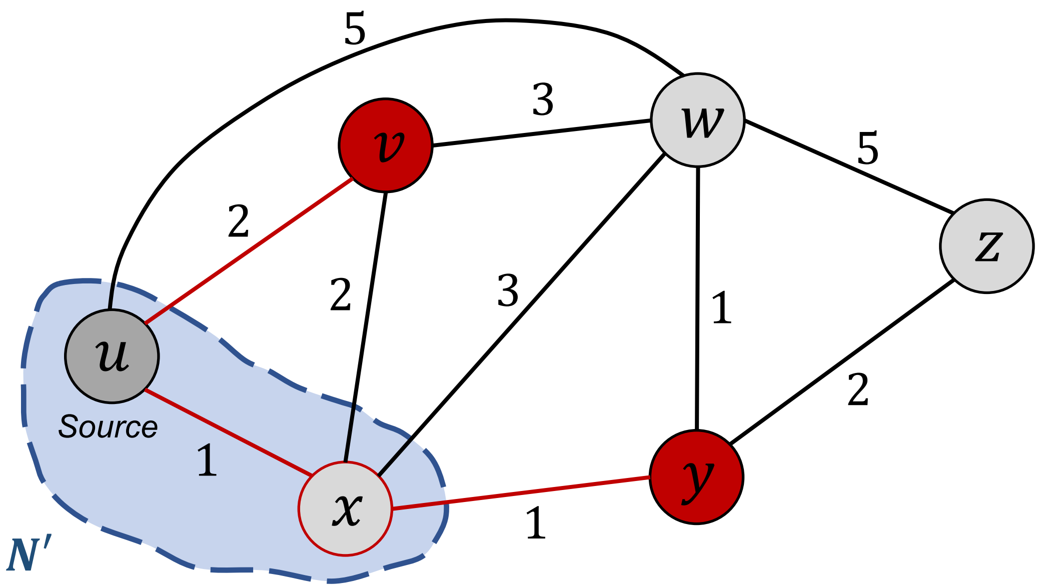
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else  $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
Until  $N' = N$
```

Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Steps of Dijkstra's algorithm:

	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	$\{u, x\}$	$2, u$	$4, x$		$2, x$	∞
2						
3						
4						
5						

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

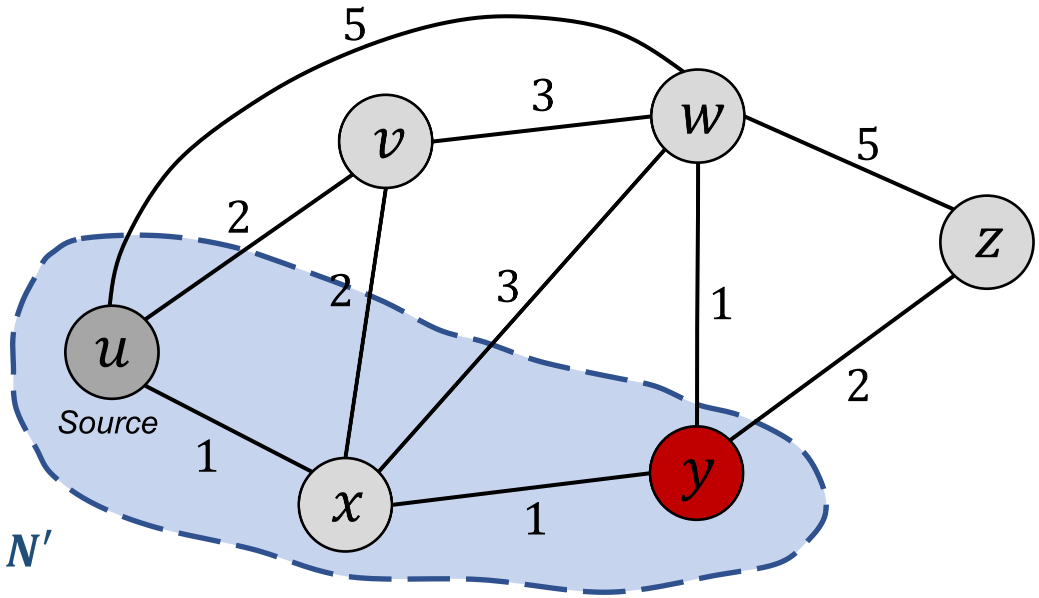
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else  $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
Until  $N' = N$
```

Link-State Routing Algorithm: Dijkstra’s Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Steps of Dijkstra’s algorithm:

	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	$\{u, x\}$	$2, u$	$4, x$		$2, x$	∞
2	$\{\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{y}\}$					
3						
4						
5						

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

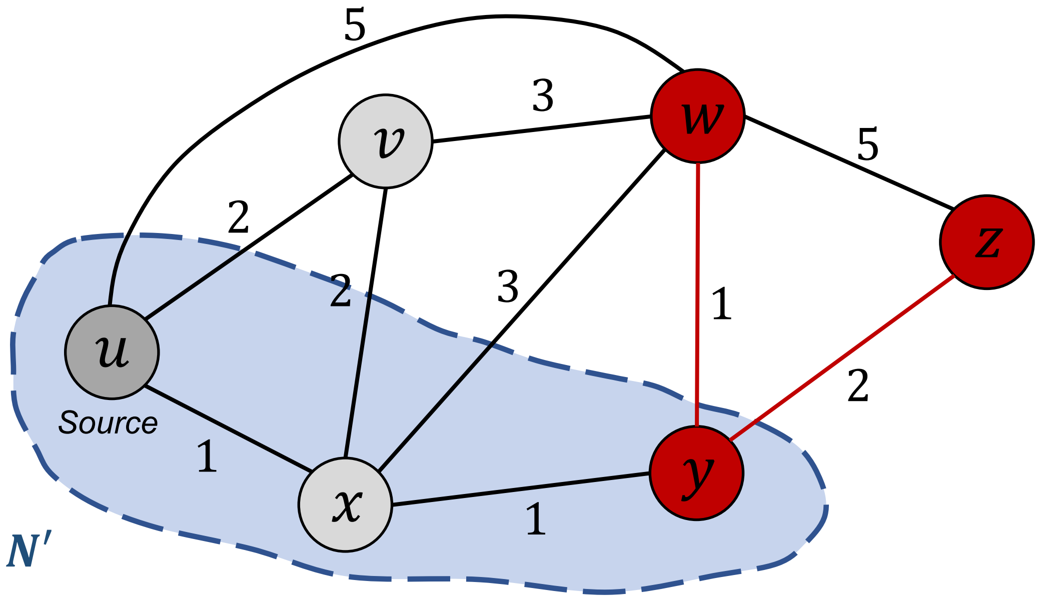
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else  $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
Until  $N' = N$
```

Link-State Routing Algorithm: Dijkstra’s Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Steps of Dijkstra’s algorithm:

	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	$\{u,x\}$	$2,u$	$4,x$		$2,x$	∞
2	$\{u,x,y\}$	$2,y$	$3,y$			$4,y$
3						
4						
5						

Given:

u	The source node
v,x,w,y,z	Destination nodes
$c(i,k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

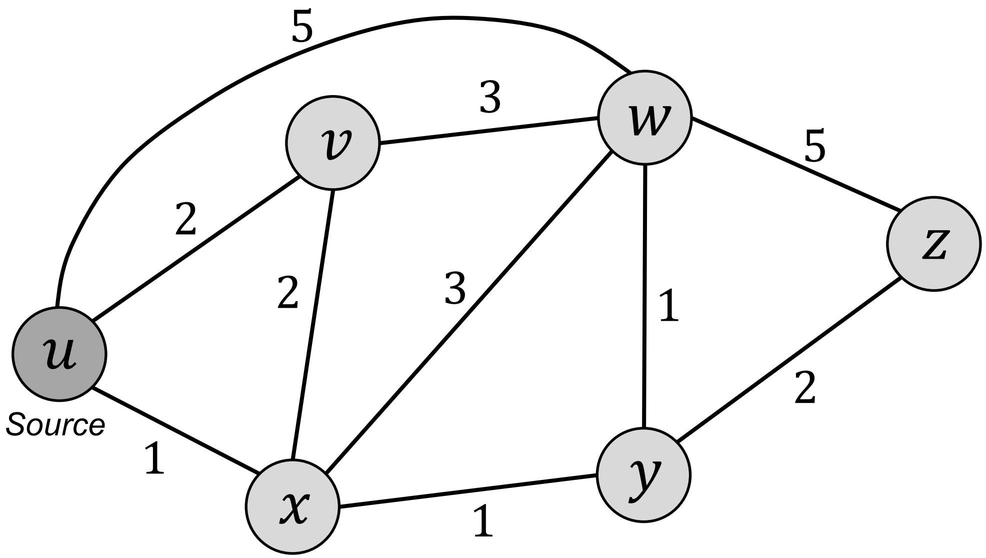
```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v,x,w,y,z\}$ do
   If $i$ – the neighbour of $u$ then
         $D(i) \leftarrow c(u,i)$
   Else   $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
   Find $i \notin N'$ such that $D(i)$ is minimal
   add $i$ to $N'$
   For each neighbour $k$ of $i$, with $k \notin N'$ do
         $D(k) \leftarrow \min\{D(k), D(i) + c(i,k)\}$
Until  $N' = N$
```

Link-State Routing Algorithm: Dijkstra's Algorithm

Problem: To find the shortest paths from the source node u to destination nodes v, x, w, y, z



Steps of Dijkstra's algorithm:

	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	$\{u, x\}$	$2, u$	$4, x$		$2, x$	∞
2	$\{u, x, y\}$	$2, u$	$3, y$			$4, y$
3	$\{u, x, y, v\}$		$3, y$			$4, y$
4	$\{u, x, y, v, z\}$		$3, y$			
5	$\{u, x, y, v, z, w\}$					

Given:

u	The source node
v, x, w, y, z	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each graph edge)

Algorithm variables:

N'	A subset of nodes, to which optimal paths have been found by a given iteration
$D(i)$	The cost of the least-cost path to node *i*, found at this iteration of the algorithm
$p(i)$	The previous node in the currently found least-cost path from node *u* to *i*

Initialization (*Step 0*):

```
$N' \leftarrow \{u\}$
For all nodes $i \in \{v, x, w, y, z\}$ do
    If $i$ – the neighbour of $u$ then
        $D(i) \leftarrow c(u, i)$
    Else  $D(i) \leftarrow \infty$
```

Iterations:

```
Loop
    Find $i \notin N'$ such that $D(i)$ is minimal
    add $i$ to $N'$
    For each neighbour $k$ of $i$, with $k \notin N'$ do
        $D(k) \leftarrow \min\{D(k), D(i) + c(i, k)\}$
Until  $N' = N$
```

Bellman-Ford Equation for the Distance-Vector Algorithm:

- Specifies the relation between costs of the least-cost paths of neighboring nodes
- The basis to distance-vector algorithms

$$d_x(y) = \min_v\{\, c(x,v) + d_v(y) \,\}$$

The cost of the least-cost path between nodes *x* and *y*

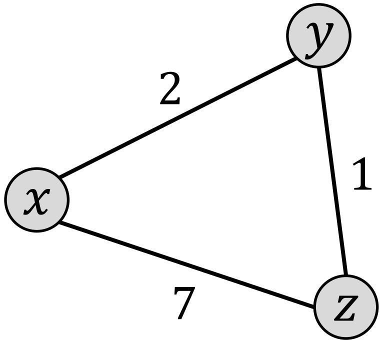
The minimum over all neighbours *v* of node *x*

The cost of the link between direct nodes

The cost of the least-cost path between nodes *v* and *y*

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Algorithm variables:

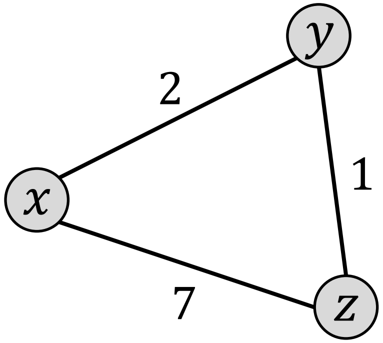
$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i:* $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i, k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:

Cost to

Node x

From

	x	y	z
x			
y			
z			

Algorithm variables:

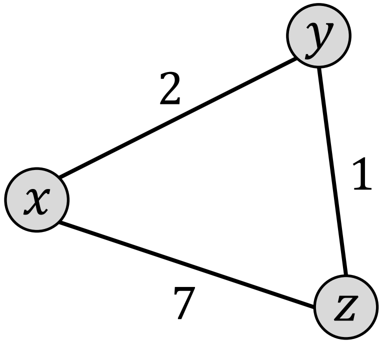
$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i:* $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

```
For all nodes $k \in N$ do
      If $k$ – the neighbour of $i$ then
            $D_i(k) \leftarrow c(i, k)$
      Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
      Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:

Node x

From	Cost to x	Cost to y	Cost to z
x	0	2	7
y			
z			

Algorithm variables:

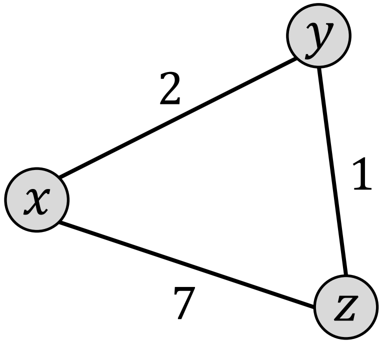
$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i:* $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i, k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:

Node x

	Cost to		
From	x	y	z
x	0	2	7
y			
z			

Algorithm variables:

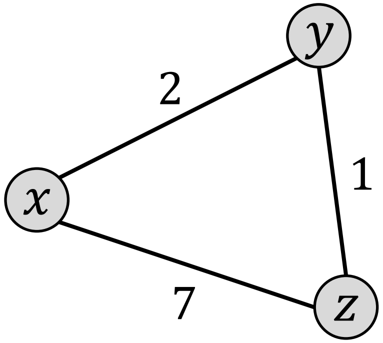
$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i:* $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i, k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Algorithm variables:

$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i:* $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i, k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

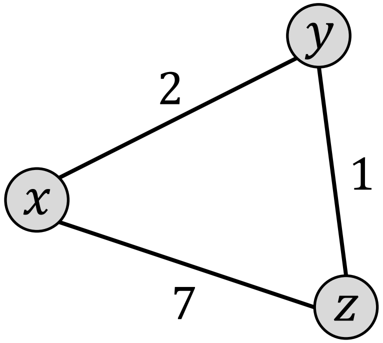
Steps of the Distance-Vector algorithm:

Node x

From \ Cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:

			Cost to		
			x	y	z
Node x	From	x	0	2	7
		y	∞	∞	∞
		z	∞	∞	∞

Send to other nodes

Algorithm variables:

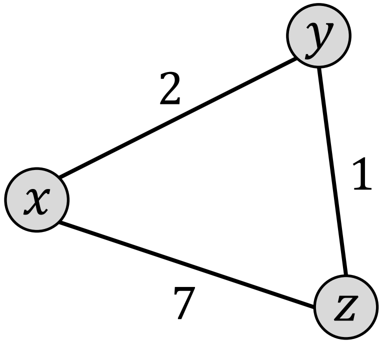
$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i:* $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i, k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:

Cost to

Node x	From	x	y	z
	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

Node y	From	x	y	z
	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

Node z	From	x	y	z
	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0

Algorithm variables:

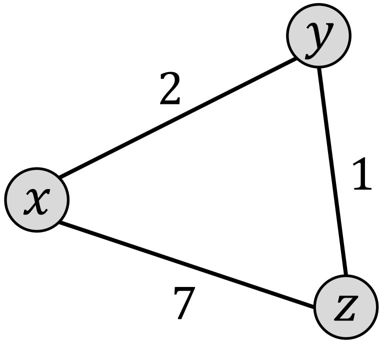
$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i:* $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i, k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Algorithm variables:

$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i*: $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i,k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Iterations at node $i \in \{x, y, z\}$:

```
Loop
    Wait for change in local link cost or msg
        from neighbor
    For each node $k \in N$ and neighbour $v$ do
        $D_i(k) \leftarrow \min_v\{\ c(i,v) + D_v(k)\ \}$
    If $D_i(k)$ changed
        Send $\mathbb{D}_i$ to all neighbours $v \in N$
Forever
```

Steps of the Distance-Vector algorithm:

Cost to

Node x — From

	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

Node y — From

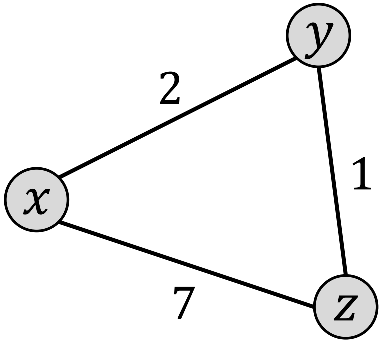
	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Node z — From

	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:

Node x

From	Cost to x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

From	Cost to x	y	z
x			
y	2	0	1
z	7	1	0

Node y

From	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Node z

From	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

Algorithm variables:

$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i:* $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

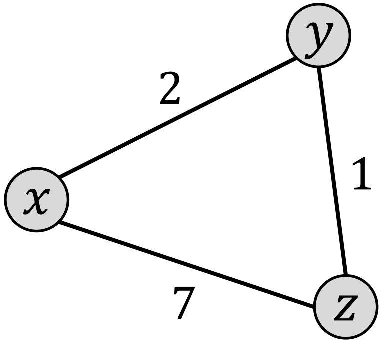
```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i, k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Iterations at node $i \in \{x, y, z\}$:

```
Loop
    Wait for change in local link cost or msg
        from neighbor
    For each node $k \in N$ and neighbour $v$ do
        $D_i(k) \leftarrow \min_v\{\ c(i, v) + D_v(k)\ \}$
    If $D_i(k)$ changed
        Send $\mathbb{D}_i$ to all neighbours $v \in N$
Forever
```

Example of Using the Distance-Vector Algorithm

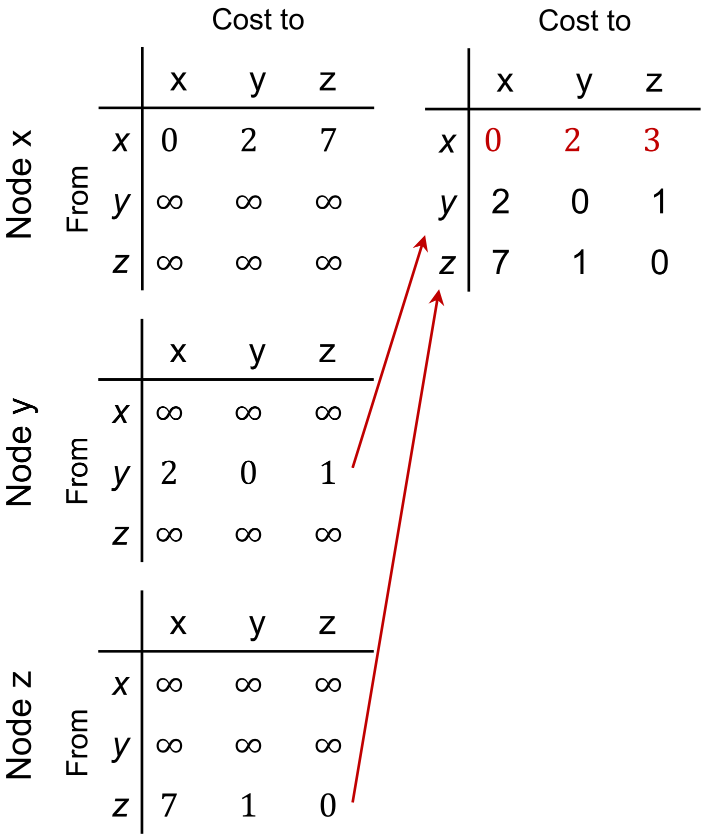
Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i,k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:



Node x

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

From	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Node z

From	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

Algorithm variables:

$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i:* $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

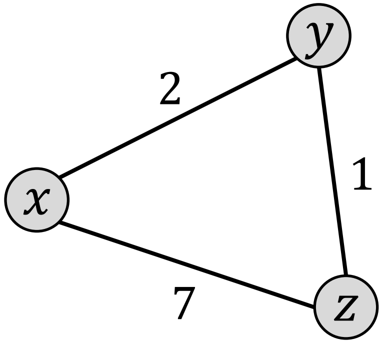
```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i,k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Iterations at node $i \in \{x, y, z\}$:

```
Loop
    Wait for change in local link cost or msg
        from neighbor
    For each node $k \in N$ and neighbour $v$ do
        $D_i(k) \leftarrow \min_v\{\ c(i,v) + D_v(k)\ \}$
    If $D_i(k)$ changed
        Send $\mathbb{D}_i$ to all neighbours $v \in N$
Forever
```

Example of Using the Distance-Vector Algorithm

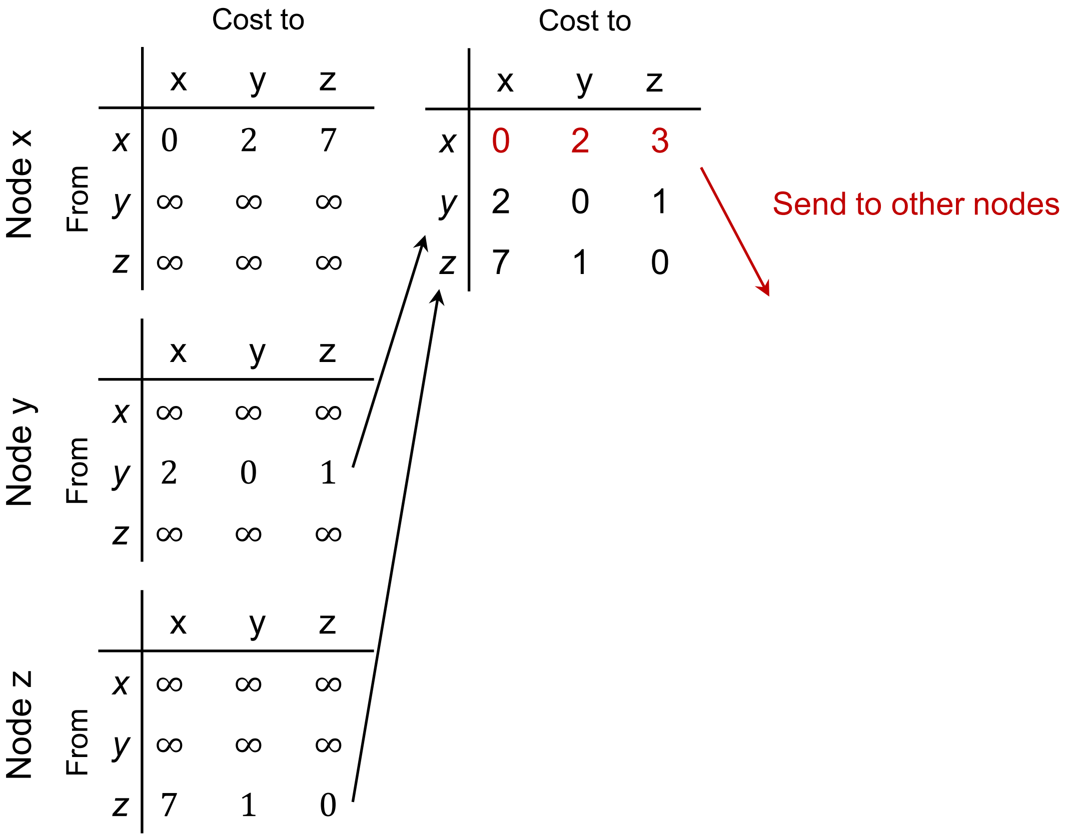
Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i,k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:



Node x

From	Cost to x	Cost to y	Cost to z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

From	Cost to x	Cost to y	Cost to z
x	0	2	3
y	2	0	1
z	7	1	0

Send to other nodes

Node y

From	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Node z

From	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

Algorithm variables:

$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i:* $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

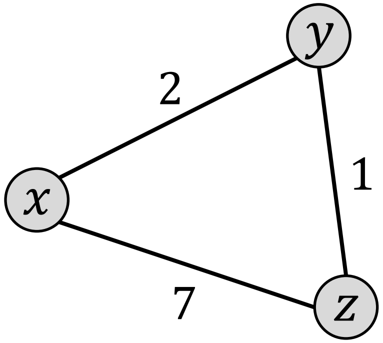
```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i,k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Iterations at node $i \in \{x, y, z\}$:

```
Loop
    Wait for change in local link cost or msg
        from neighbor
    For each node $k \in N$ and neighbour $v$ do
        $D_i(k) \leftarrow \min_v\{\ c(i,v) + D_v(k)\ \}$
    If $D_i(k)$ changed
        Send $\mathbb{D}_i$ to all neighbours $v \in N$
Forever
```

Example of Using the Distance-Vector Algorithm

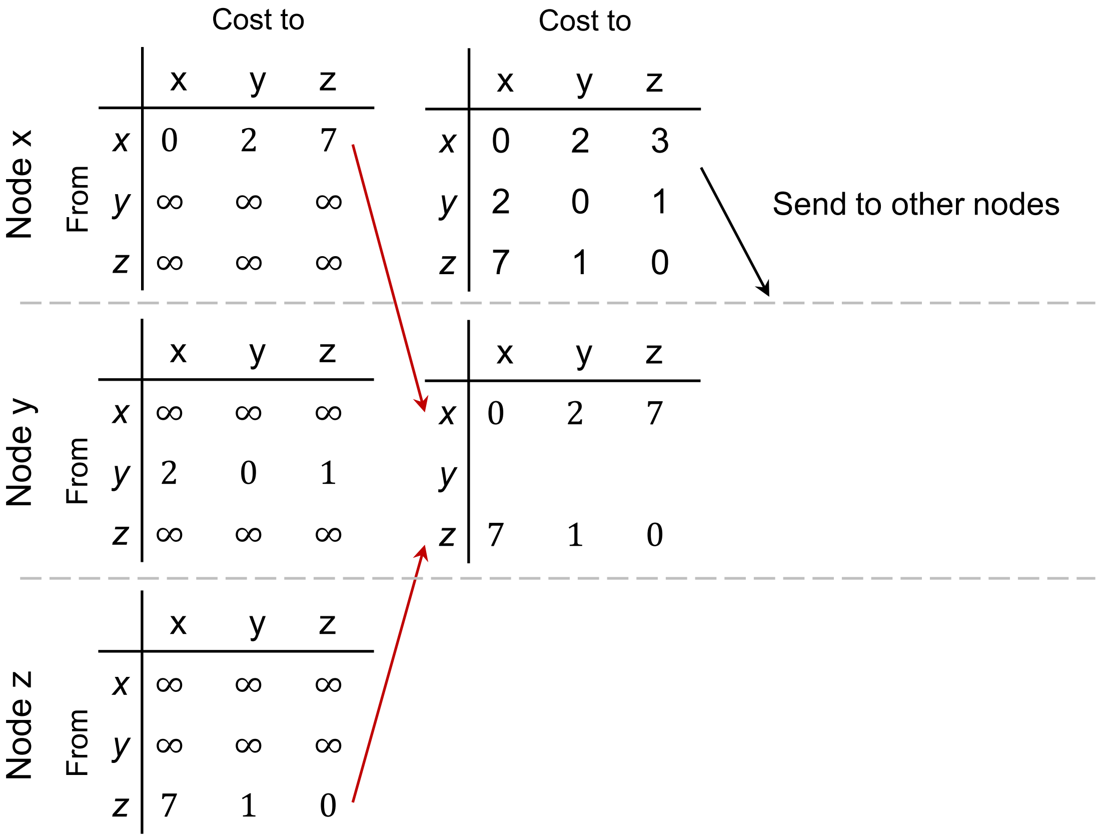
Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:



Node x

Cost to

From	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

Cost to

	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

Send to other nodes

Node y

From	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

	x	y	z
x	0	2	7
y			
z	7	1	0

Node z

From	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

Algorithm variables:

$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i*: $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

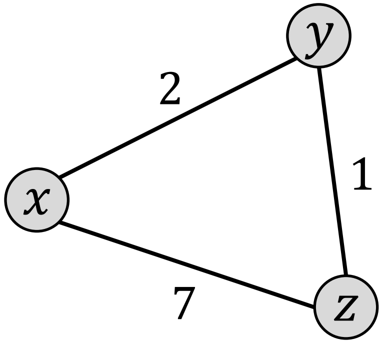
```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i, k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Iterations at node $i \in \{x, y, z\}$:

```
Loop
    Wait for change in local link cost or msg
        from neighbor
    For each node $k \in N$ and neighbour $v$ do
        $D_i(k) \leftarrow \min_v\{\ c(i, v) + D_v(k)\ \}$
    If $D_i(k)$ changed
        Send $\mathbb{D}_i$ to all neighbours $v \in N$
Forever
```

Example of Using the Distance-Vector Algorithm

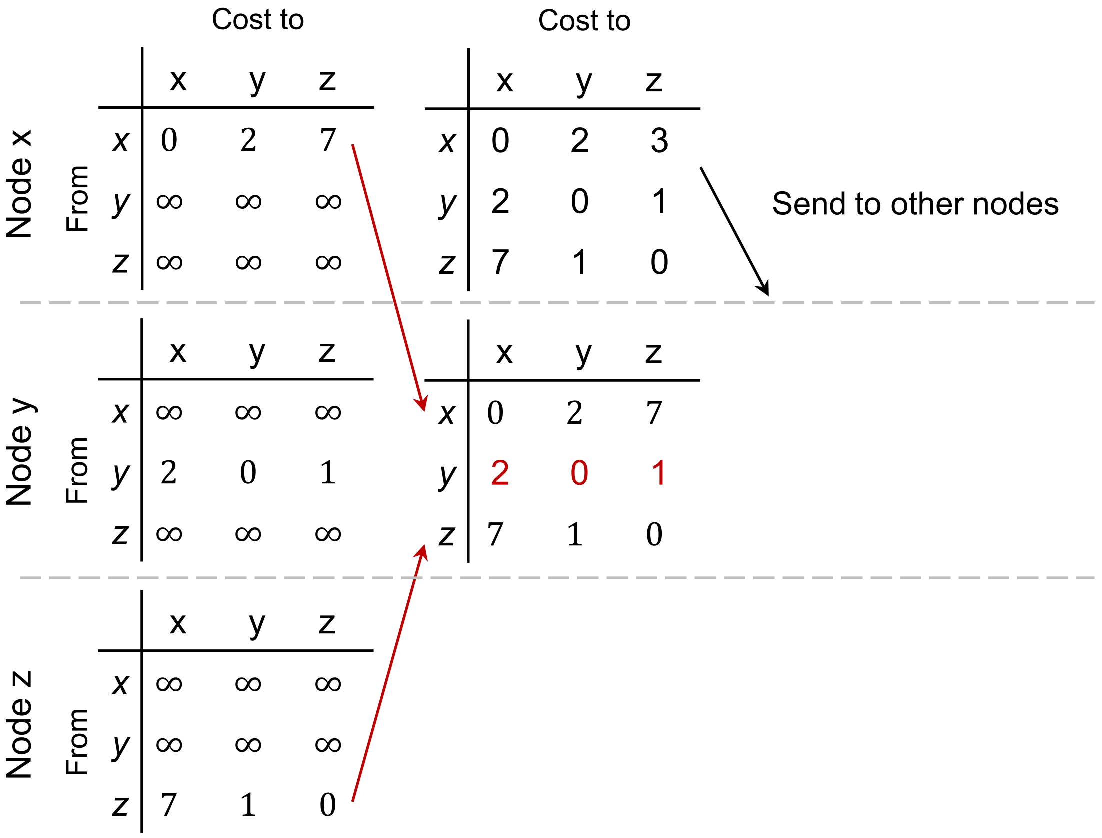
Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i,k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:



Node x

Cost to

From	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

Cost to

From	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

Send to other nodes

Node y

From	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

From	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

Node z

From	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

Algorithm variables:

$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i*: $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

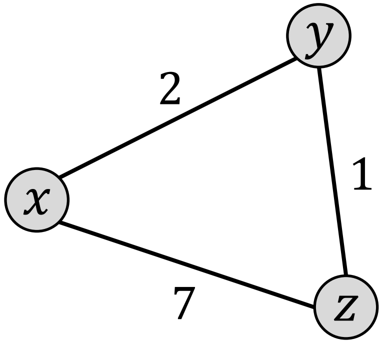
```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i,k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Iterations at node $i \in \{x, y, z\}$:

```
Loop
    Wait for change in local link cost or msg from neighbor
    For each node $k \in N$ and neighbour $v$ do
        $D_i(k) \leftarrow \min_v\{ c(i,v) + D_v(k) \}$
    If $D_i(k)$ changed
        Send $\mathbb{D}_i$ to all neighbours $v \in N$
Forever
```

Example of Using the Distance-Vector Algorithm

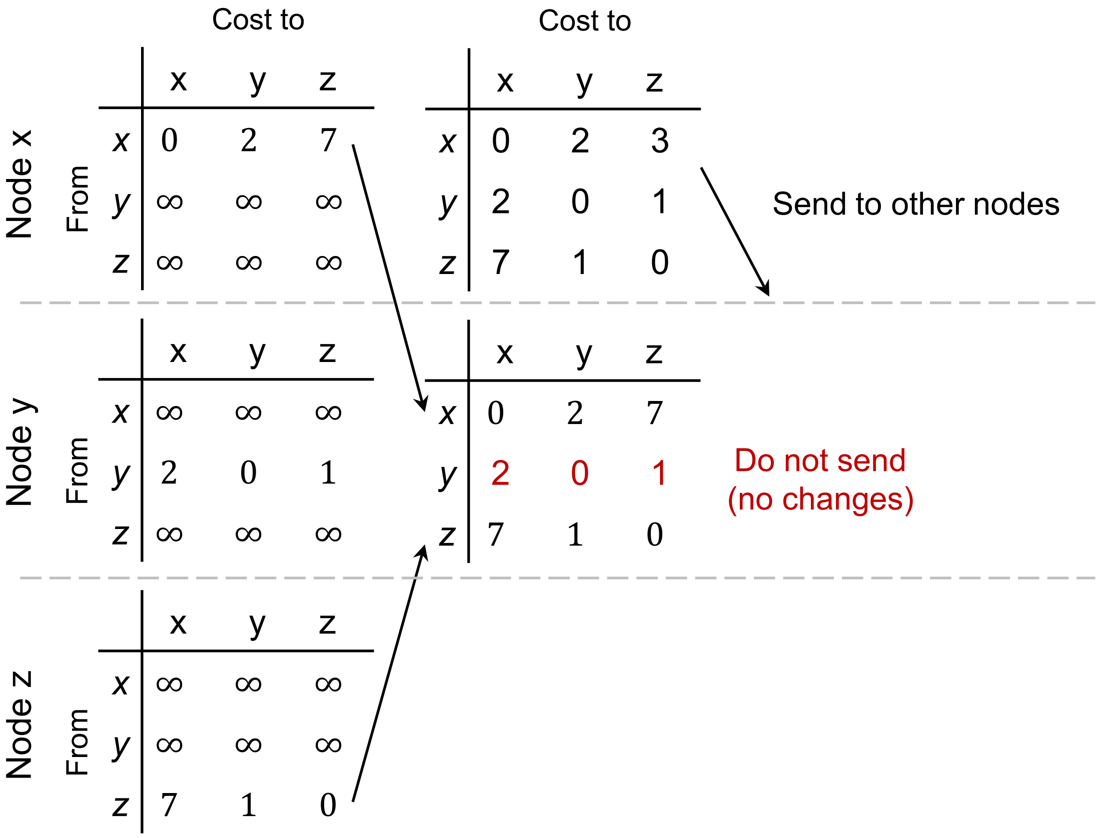
Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:



Node x

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

Send to other nodes

Node y

	x	y	z
From x	∞	∞	∞
From y	2	0	1
From z	∞	∞	∞

	x	y	z
From x	0	2	7
From y	2	0	1
From z	7	1	0

Do not send (no changes)

Node z

	x	y	z
From x	∞	∞	∞
From y	∞	∞	∞
From z	7	1	0

Algorithm variables:

$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i*: $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

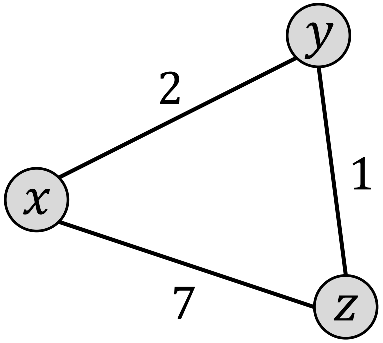
```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i, k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Iterations at node $i \in \{x, y, z\}$:

```
Loop
    Wait for change in local link cost or msg
        from neighbor
    For each node $k \in N$ and neighbour $v$ do
        $D_i(k) \leftarrow \min_v\{\ c(i, v) + D_v(k)\ \}$
    If $D_i(k)$ changed
        Send $\mathbb{D}_i$ to all neighbours $v \in N$
Forever
```

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Steps of the Distance-Vector algorithm:

Node x

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

From	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

From	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

Node z

From	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

From	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

Algorithm variables:

$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i:* $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

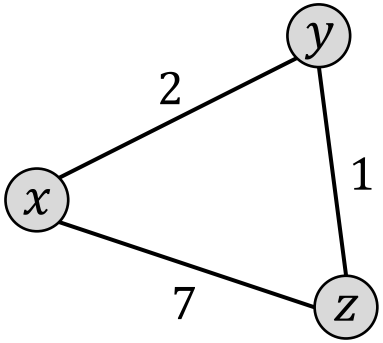
```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i, k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Iterations at node $i \in \{x, y, z\}$:

```
Loop
    Wait for change in local link cost or msg
        from neighbor
    For each node $k \in N$ and neighbour $v$ do
        $D_i(k) \leftarrow \min_v\{ c(i, v) + D_v(k) \}$
    If $D_i(k)$ changed
        Send $\mathbb{D}_i$ to all neighbours $v \in N$
Forever
```

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Algorithm variables:

$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i*: $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

```
For all nodes $k \in N$ do
    If $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i,k)$
    Else $D_i(k) \leftarrow \infty$
For each neighbour $v$ of $i$ do
    Wait $\mathbb{D}_v \leftarrow ?$
Send $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Iterations at node $i \in \{x, y, z\}$:

```
Loop
    Wait for change in local link cost or msg
        from neighbor
    For each node $k \in N$ and neighbour $v$ do
        $D_i(k) \leftarrow \min_v\{\ c(i,v) + D_v(k)\ \}$
    If $D_i(k)$ changed
        Send $\mathbb{D}_i$ to all neighbours $v \in N$
Forever
```

Steps of the Distance-Vector algorithm:

Cost to

Node x

From

	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Node y

From

	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Node z

From

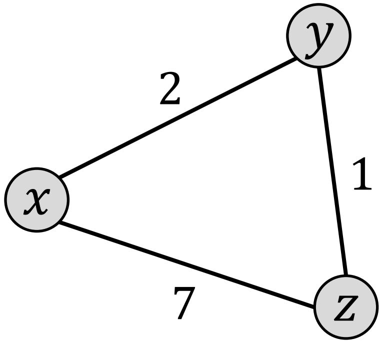
	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Example of Using the Distance-Vector Algorithm

Problem: To find the shortest paths between any two nodes



Given:

$N = \{x, y, z\}$	Destination nodes
$c(i, k)$	Link cost between nodes i and k (specified near each edge)

Algorithm variables:

$D_i(k)$	An estimated cost of the least-cost path from node *i* to *k*, found at this iteration of the algorithm
$\mathbb{D}_i$	The distance vector of node *i*: $\mathbb{D}_i = [D_i(x),\ D_i(y),\ D_i(z)]$

Initialization at node $i \in \{x, y, z\}$ (*Step 0*):

```
**For** all nodes $k \in N$ do
    **If** $k$ – the neighbour of $i$ then
        $D_i(k) \leftarrow c(i, k)$
    **Else** $D_i(k) \leftarrow \infty$
**For** each neighbour $v$ of $i$ do
    **Wait** $\mathbb{D}_v \leftarrow ?$
**Send** $\mathbb{D}_i$ to each neighbour $v$ of $i$
```

Iterations at node $i \in \{x, y, z\}$:

```
**Loop**
    **Wait** for change in local link cost or msg
        from neighbor
    **For** each node $k \in N$ and neighbour $v$ do
        $D_i(k) \leftarrow \min_v\{\, c(i, v) + D_v(k) \,\}$
    **If** $D_i(k)$ changed
        **Send** $\mathbb{D}_i$ to all neighbours $v \in N$
**Forever**
```

Steps of the Distance-Vector algorithm:

Node x

Cost to

From	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

Cost to

From	x	y	z
x	**0**	**2**	**3**
y	2	0	1
z	7	1	0

Cost to

From	x	y	z
x	**0**	**2**	**3**
y	2	0	1
z	3	1	0

Node y

From	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

From	x	y	z
x	0	2	7
y	**2**	**0**	**1**
z	7	1	0

From	x	y	z
x	0	2	3
y	**2**	**0**	**1**
z	3	1	0

Node z

From	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

From	x	y	z
x	0	2	7
y	2	0	1
z	**3**	**1**	**0**

From	x	y	z
x	0	2	3
y	2	0	1
z	**3**	**1**	**0**

Acknowledgment

These slides are prepared with the help of Artem Burmyakov and Muhammad Fahim