CSCI 466: Networks

Network Layer – Routing (Control Plane)

Reese Pearsall

Fall 2024

Announcements

No class on Friday
- **Workday for PA2 (I'll be in my office still if you need help)**

PA2 due on Sunday at 11:59 PM
- Make sure everything is inside of a /PA2 folder in your repo
- Everyone needs to submit video demo link to D2L

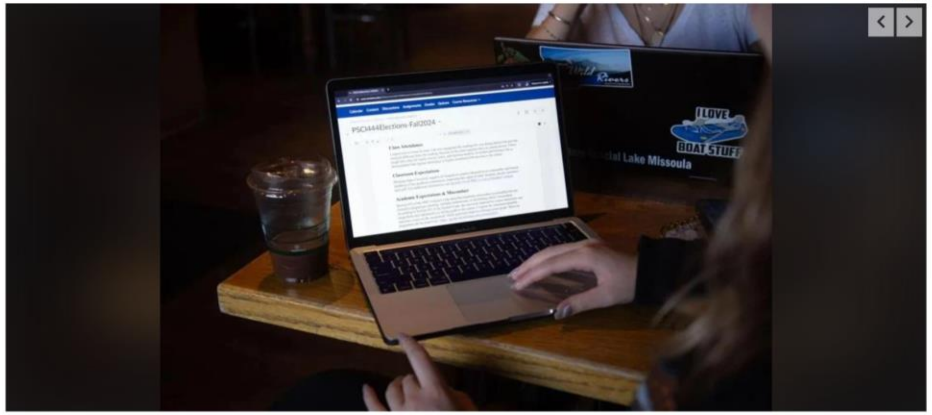
ACM is hosting a guest speaker tonight at 5PM in Barnard Hall 347
- Speaker has experience working at HP, Microsoft, Amazon
- Great opportunity to learn about industry, careers, etc

https://www.bozemandailychronicle.com/news/montana_state_university/exponent/kind-of-scary-montana-state-officials-confirm-learning-platform-tracks-student-locations/article_740f212e-859d-11ef-a536-771135b8a5d8.html

'Kind of scary': Montana State officials confirm learning platform tracks student locations

Dom Lucero The MSU Exponent 22 hrs ago

1 of 2



A student accesses D2L while at a coffee shop.

Maddi Hohner/The MSU Exponent

Buy Now

Quizzes › Quiz 3 › Attempt Logs

Attempt Logs

Overview | **Detailed**

2471 items in the list.

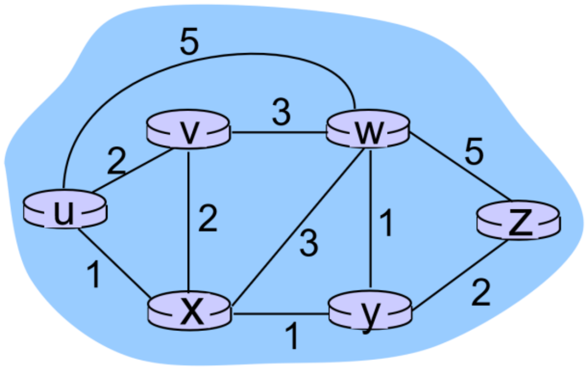
Attempt	Event	Modified by	IP Address	Date ▲
Reese Pearsall (Attempt: 1)	Quiz Entry	Reese Pearsall	153.90.118.85	Oct 3, 2024 3:13 PM
Reese Pearsall (Attempt: 1)	Response to Question 1 Saved	Reese Pearsall	153.90.118.85	Oct 3, 2024 3:13 PM
Reese Pearsall (Attempt: 1)	Response to Question 2 Saved	Reese Pearsall	153.90.118.85	Oct 3, 2024 3:14 PM
Reese Pearsall (Attempt: 1)	Response to Question 3 Saved	Reese Pearsall	153.90.118.85	Oct 3, 2024 3:14 PM
Reese Pearsall (Attempt: 1)	Response to Question 4 Saved	Reese Pearsall	153.90.118.85	Oct 3, 2024 3:14 PM
Reese Pearsall (Attempt: 1)	Response to Question 5 Saved	Reese Pearsall	153.90.118.85	Oct 3, 2024 3:14 PM
Reese Pearsall (Attempt: 1)	Response to Question 6 Saved	Reese Pearsall	153.90.118.85	Oct 3, 2024 3:14 PM
Reese Pearsall (Attempt: 1)	Response to Question 7 Saved	Reese Pearsall	153.90.118.85	Oct 3, 2024 3:14 PM
Reese Pearsall (Attempt: 1)	Response to Question 8 Saved	Reese Pearsall	153.90.118.85	Oct 3, 2024 3:14 PM
Reese Pearsall (Attempt: 1)	Response to Question 9 Saved	Reese Pearsall	153.90.118.85	Oct 3, 2024 3:14 PM
Reese Pearsall (Attempt: 1)	Response to Question 10 Saved	Reese Pearsall	153.90.118.85	Oct 3, 2024 3:14 PM

Forwarding refers to moving packets from a **router's input** to appropriate **router output**, and is implemented in the data plane.

Address range	Interface (output link)
128.11.52.0 – 128.11.52.255	1
153.90.2.0 – 153.90.2.255	2
153.90.2.87 – 153.90.2.89	3

Ideally, this output links are the most optimal path to get to the destination

Routing refers to determining the route taken by packets from **source** to **destination**, and is implemented in the control plane.



What is the best way to get from **u** to **z**?

Routing Metrics:
- Shortest Path
- Highest Throughput Path
- *Minimum Number of Hops*
- Lowest Congested Path

Routing tables are filled via **routing algorithms**

There are two types of routing algorithms

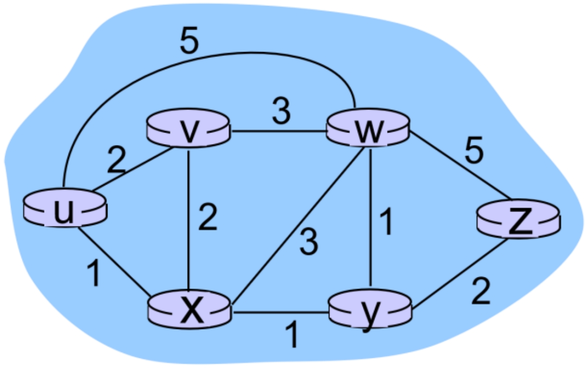
Routing tables are filled via **routing algorithms**

There are two types of routing algorithms

Centralized/Global- we know the edge costs of the network

Link State algorithms

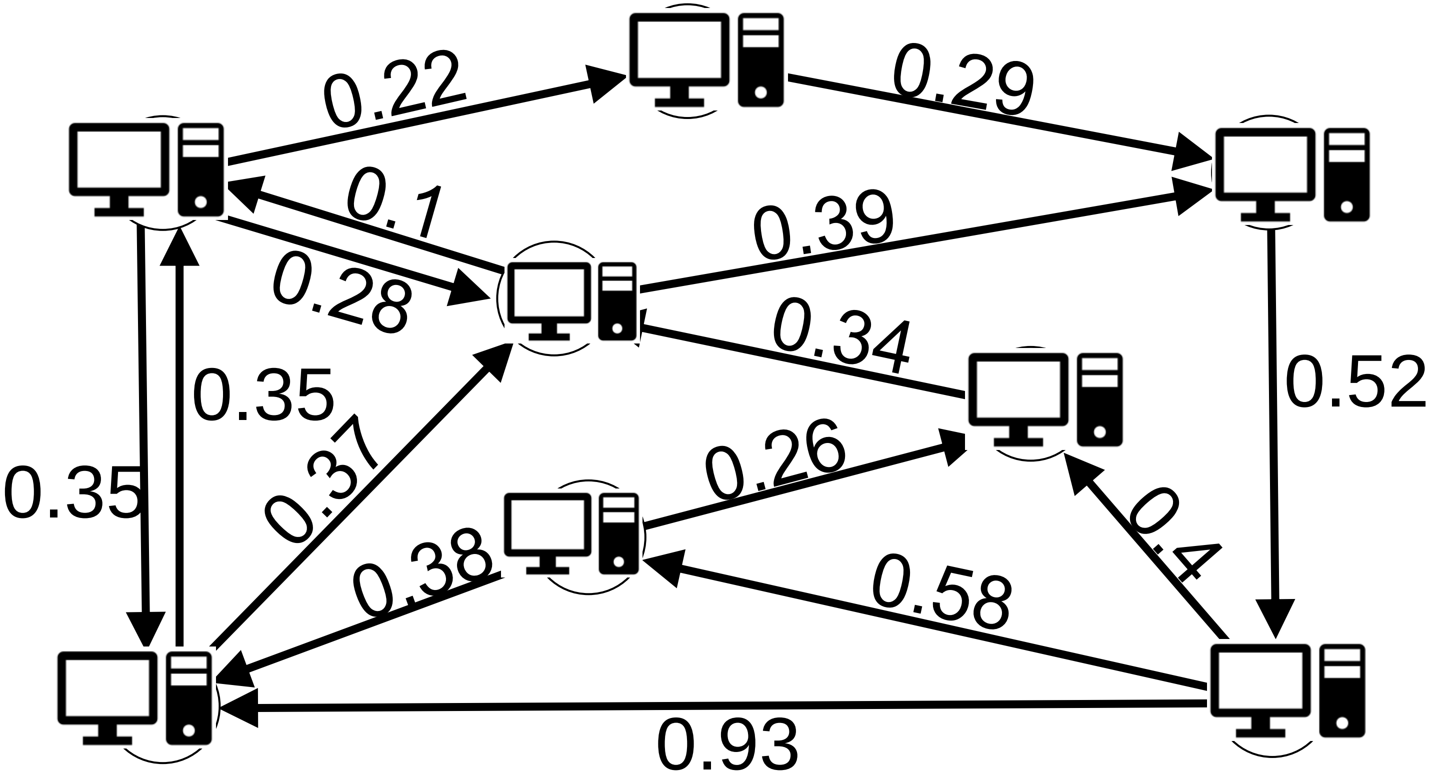
(Dijkstra's Algorithm)



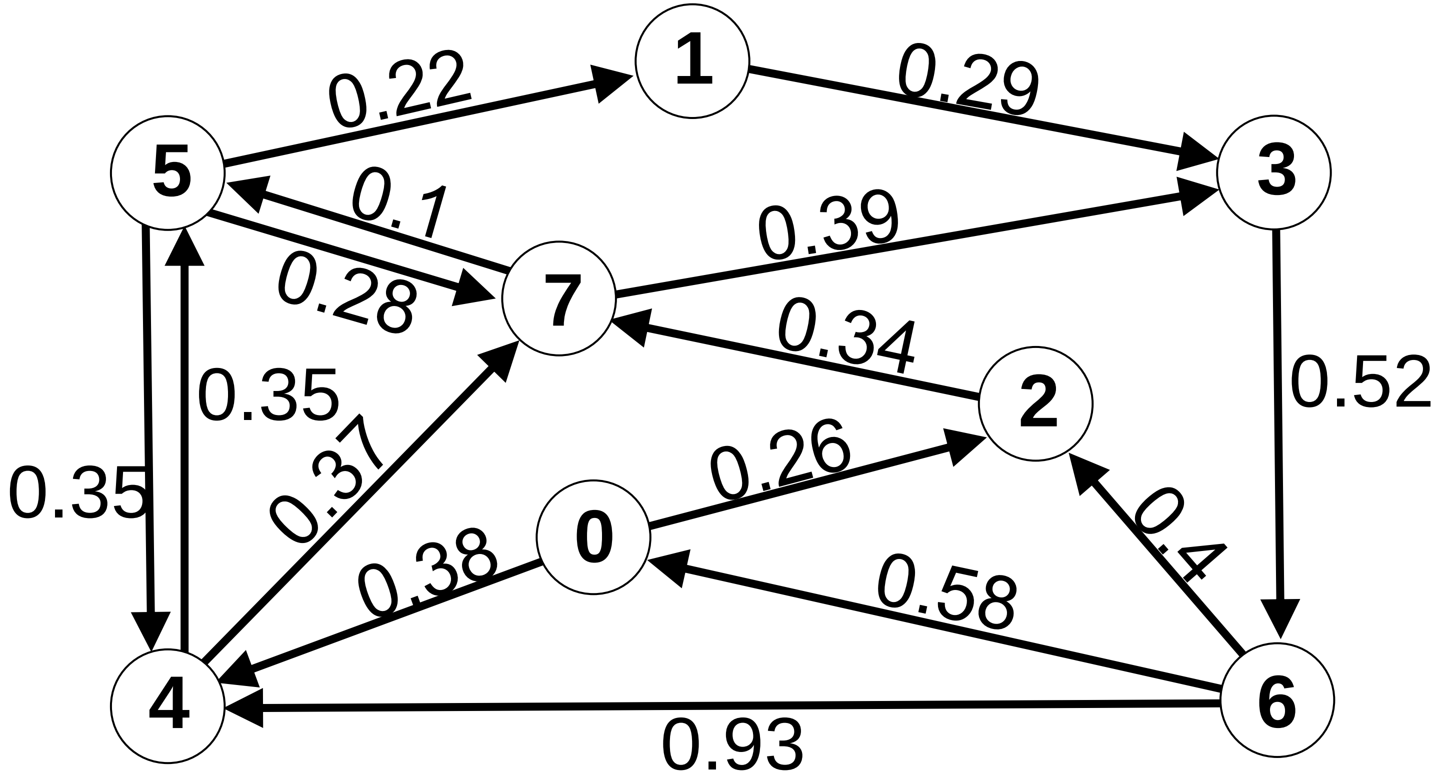
We can compute the shortest path from one node, to all other nodes in polynomial time.

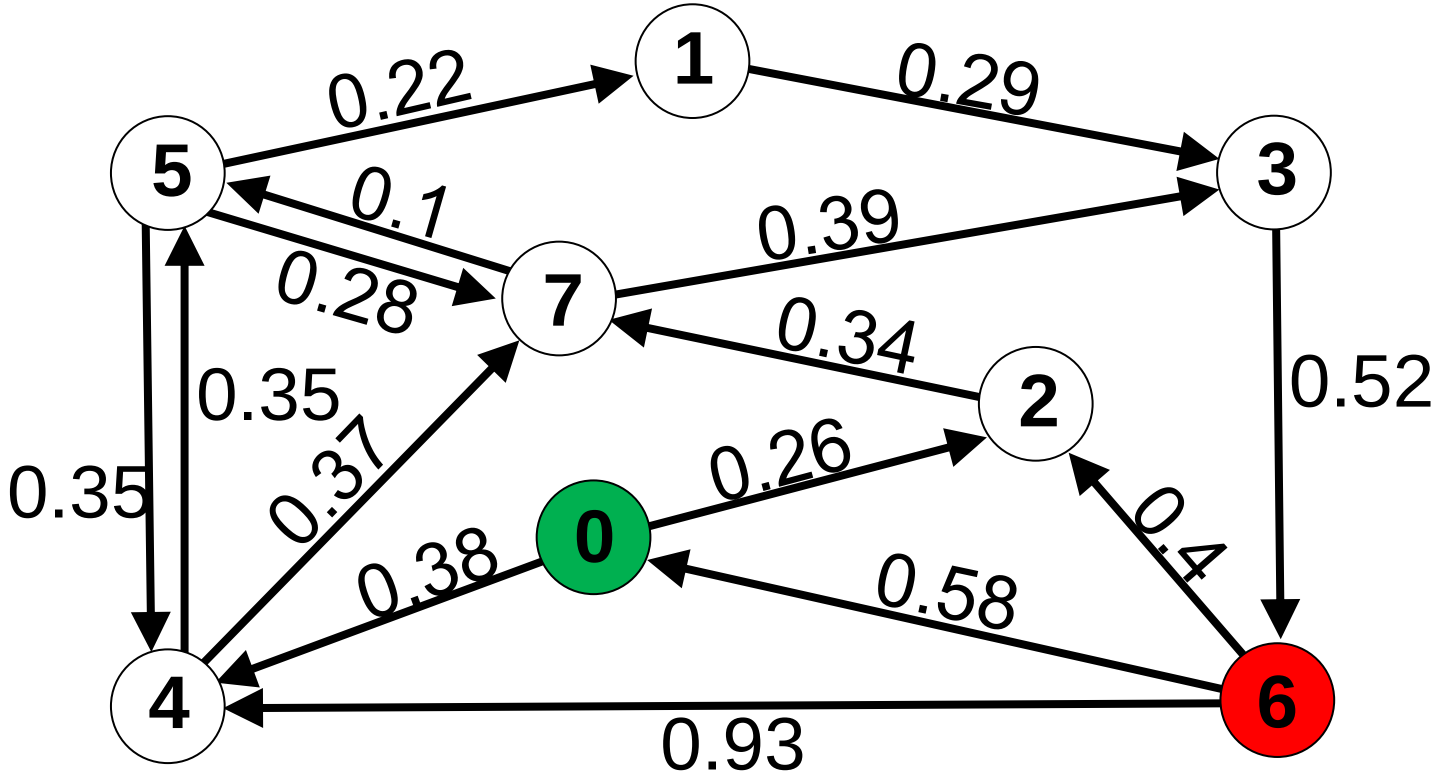
Once we know the shortest path from A to B, we can update routing tables to reflect that shortest path

Shortest Path



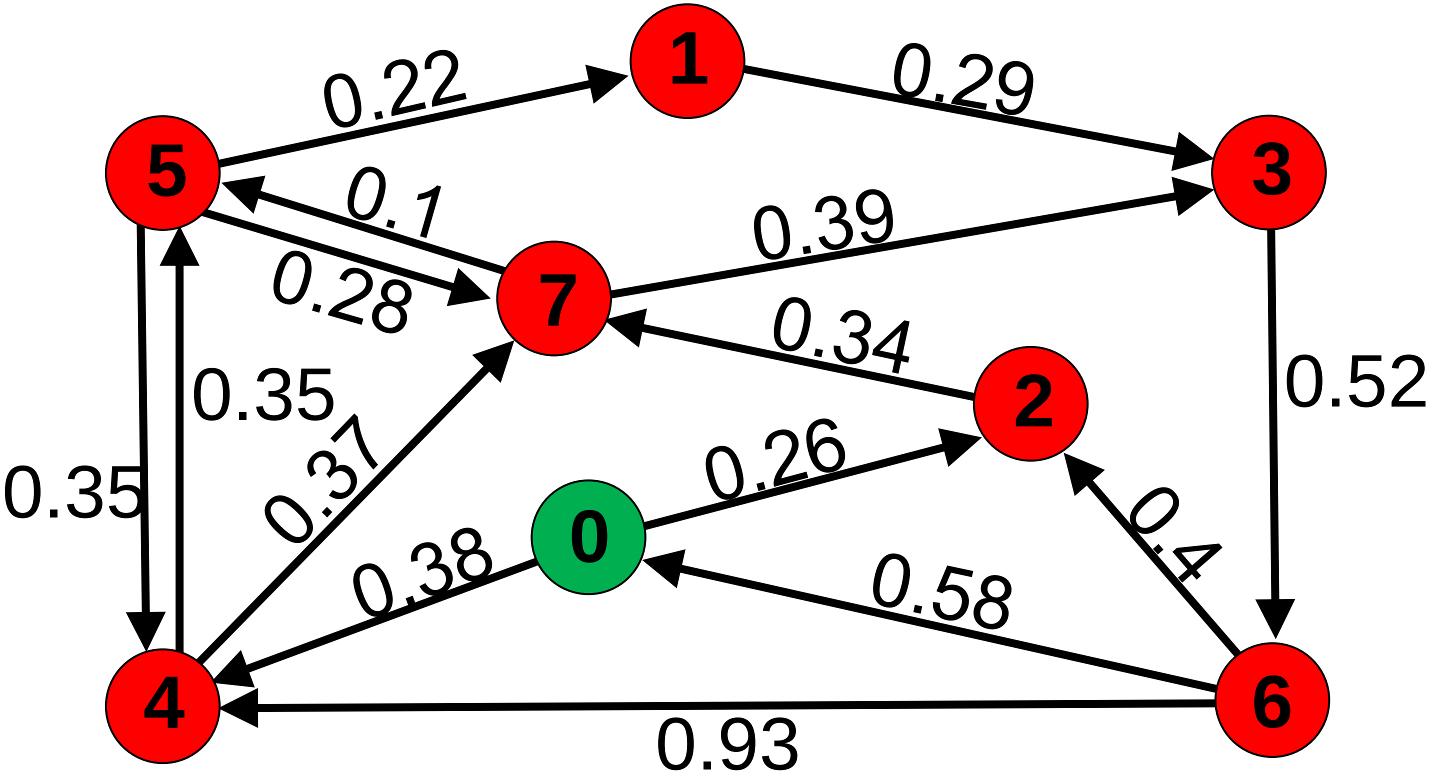
Shortest Path



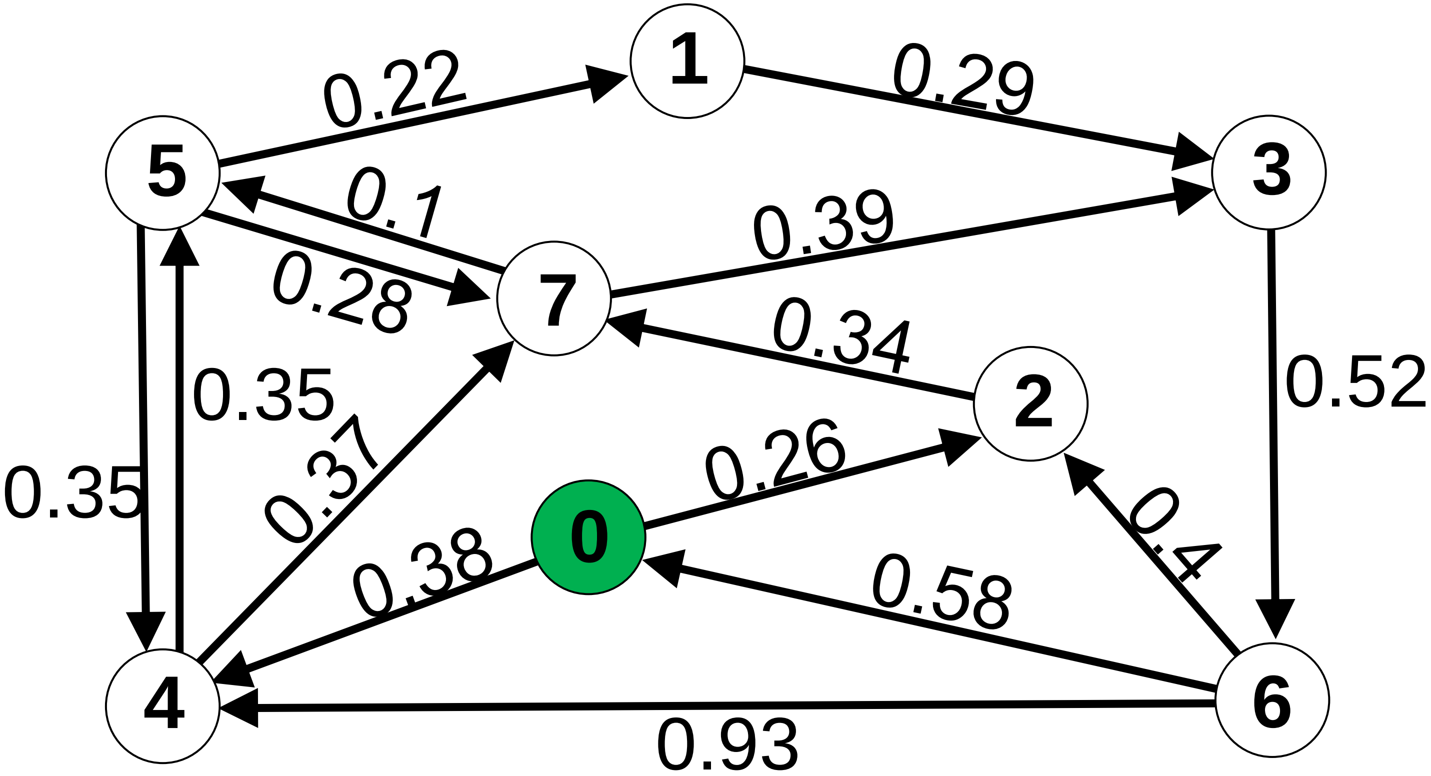


Path with the smallest sum of edge weights.

What is the shortest path between **vertex 0** and **vertex 6**?

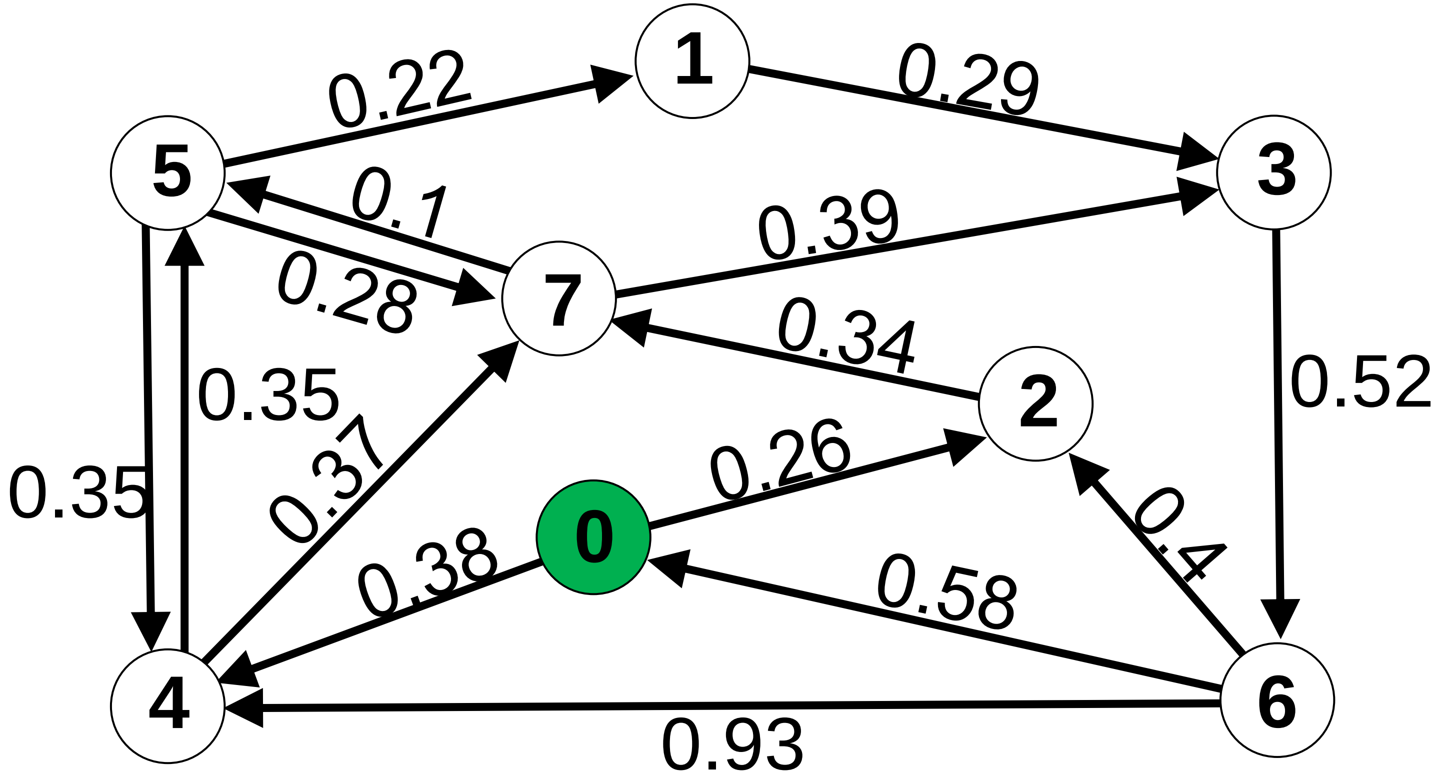


We are going to find the shortest path between vertex 0 and every other vertex, flooding out from 0.



Distance from 0

0	?
1	?
2	?
3	?
4	?
5	?
6	?
7	?

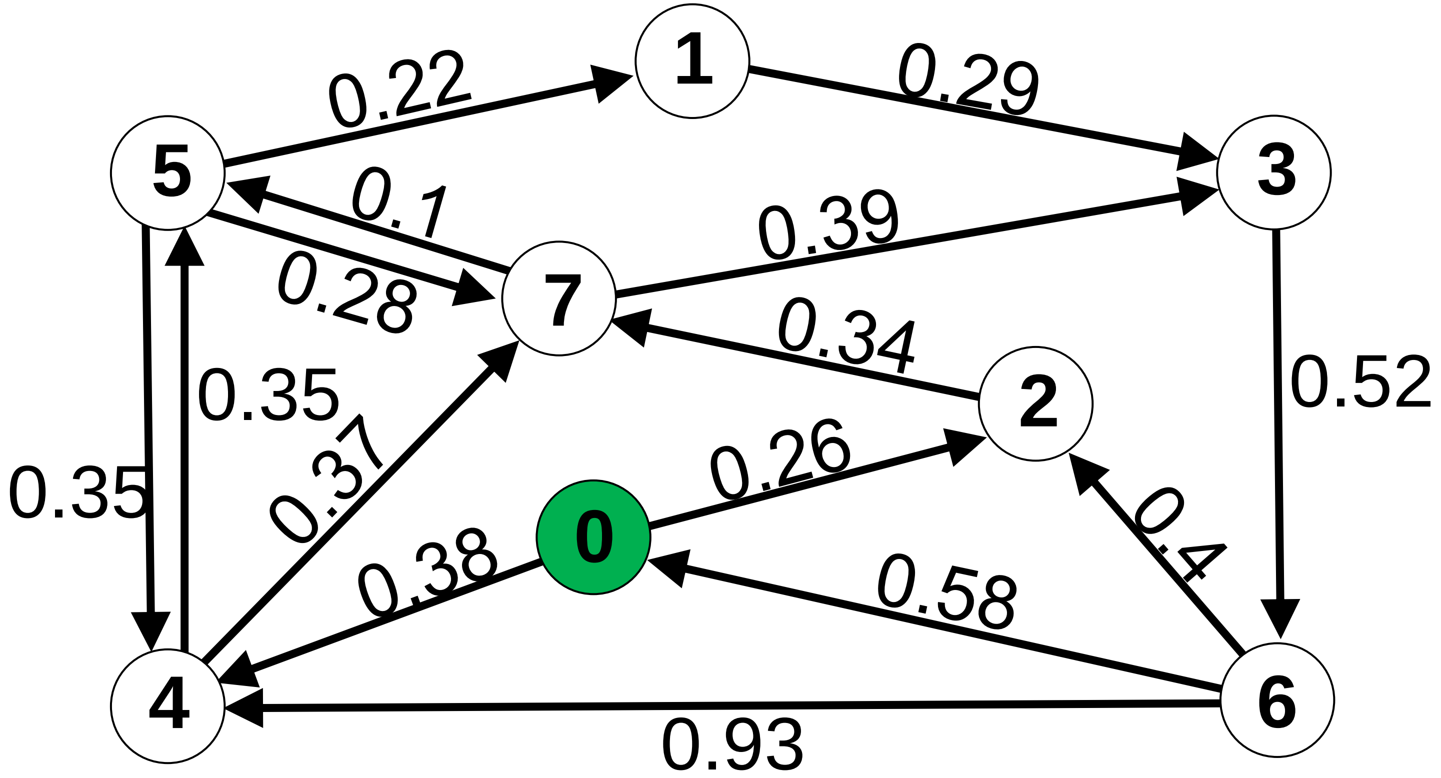


Distance from 0

0	0
1	∞
2	∞
3	∞
4	∞
5	∞
6	∞
7	∞

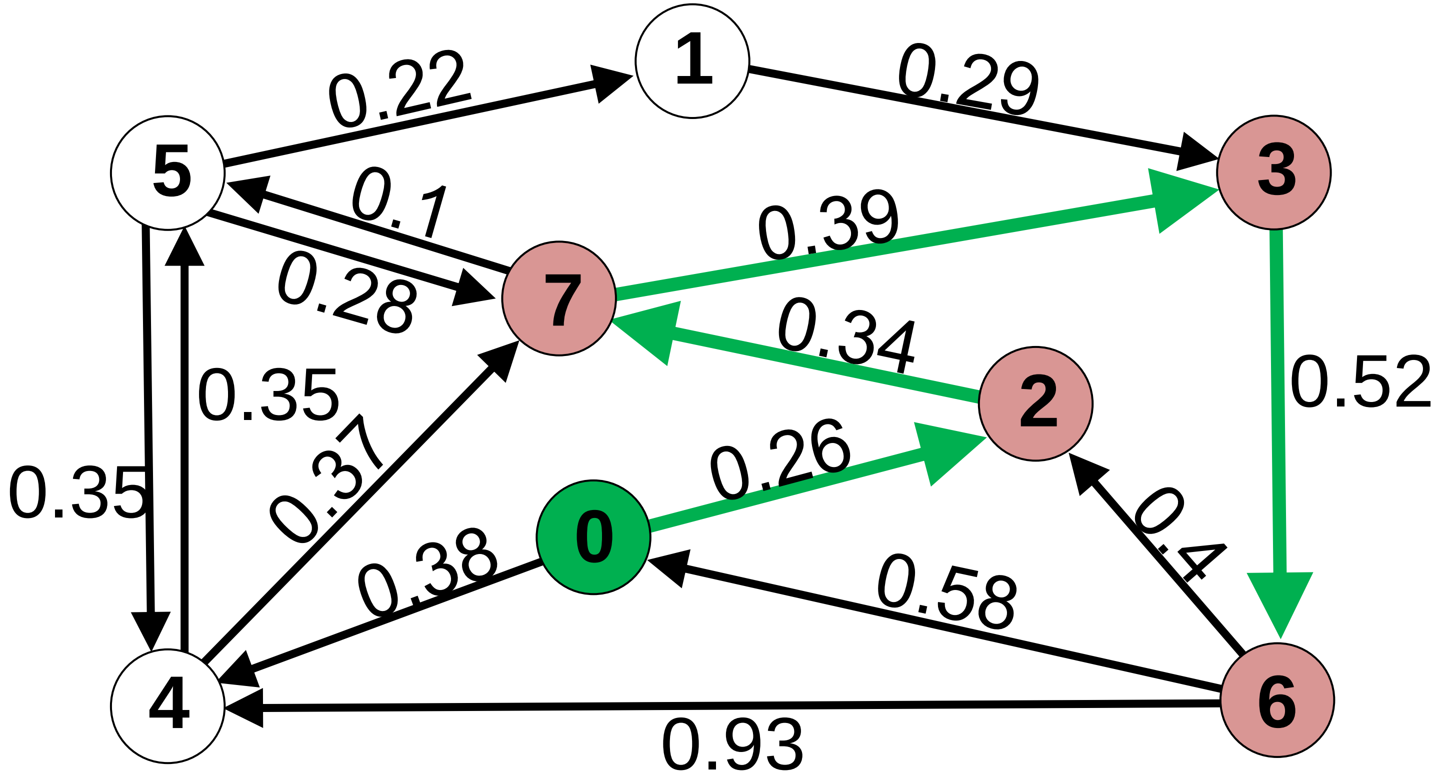
How can we keep track of routes?

Shortest Path



	Distance from 0	Previous vertex
0	0	-
1	∞	
2	∞	
3	∞	
4	∞	
5	∞	
6	∞	
7	∞	

How can we keep track of routes?



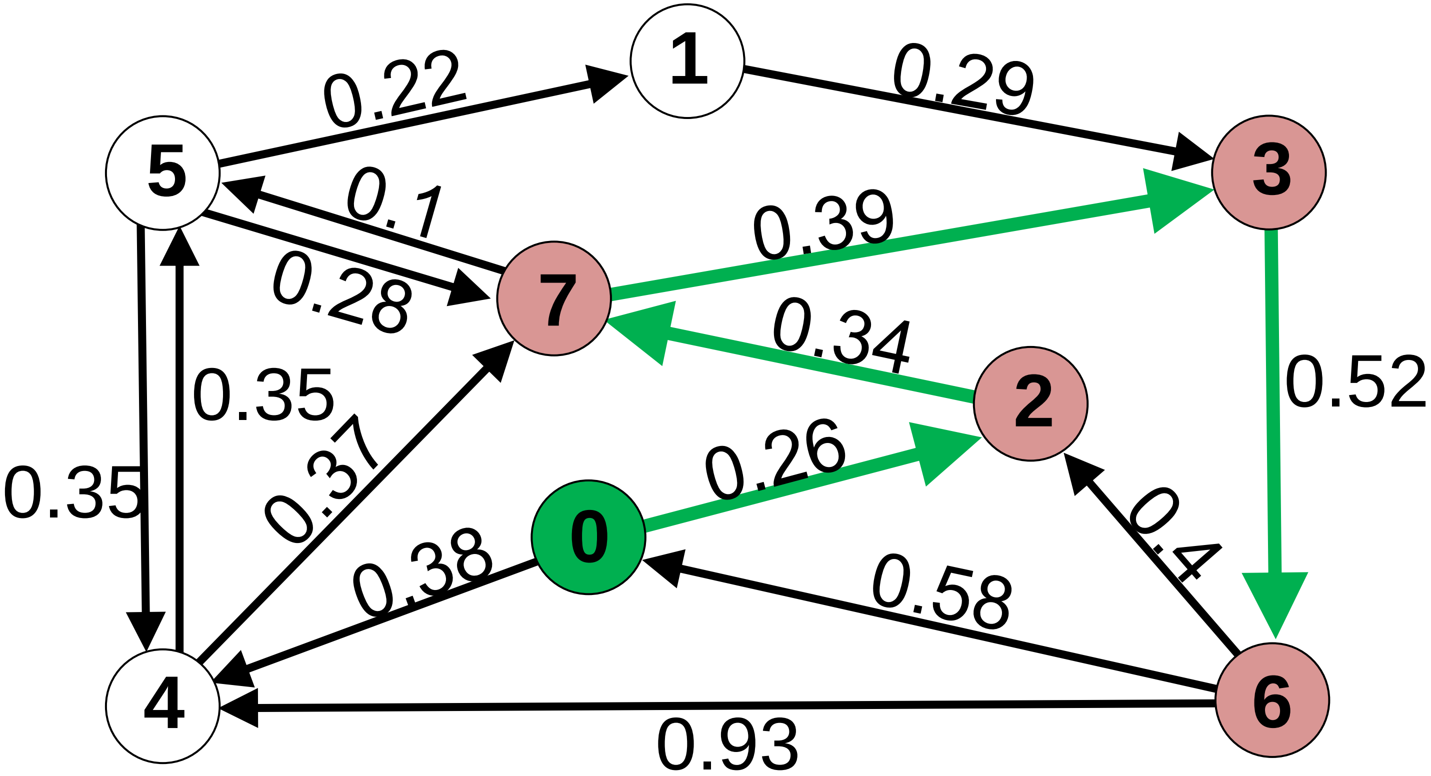
Distance from 0

0	0
1	∞
2	0.26
3	0.99
4	∞
5	∞
6	1.51
7	0.60

Previous vertex

0	-
1	
2	0
3	7
4	
5	
6	3
7	2

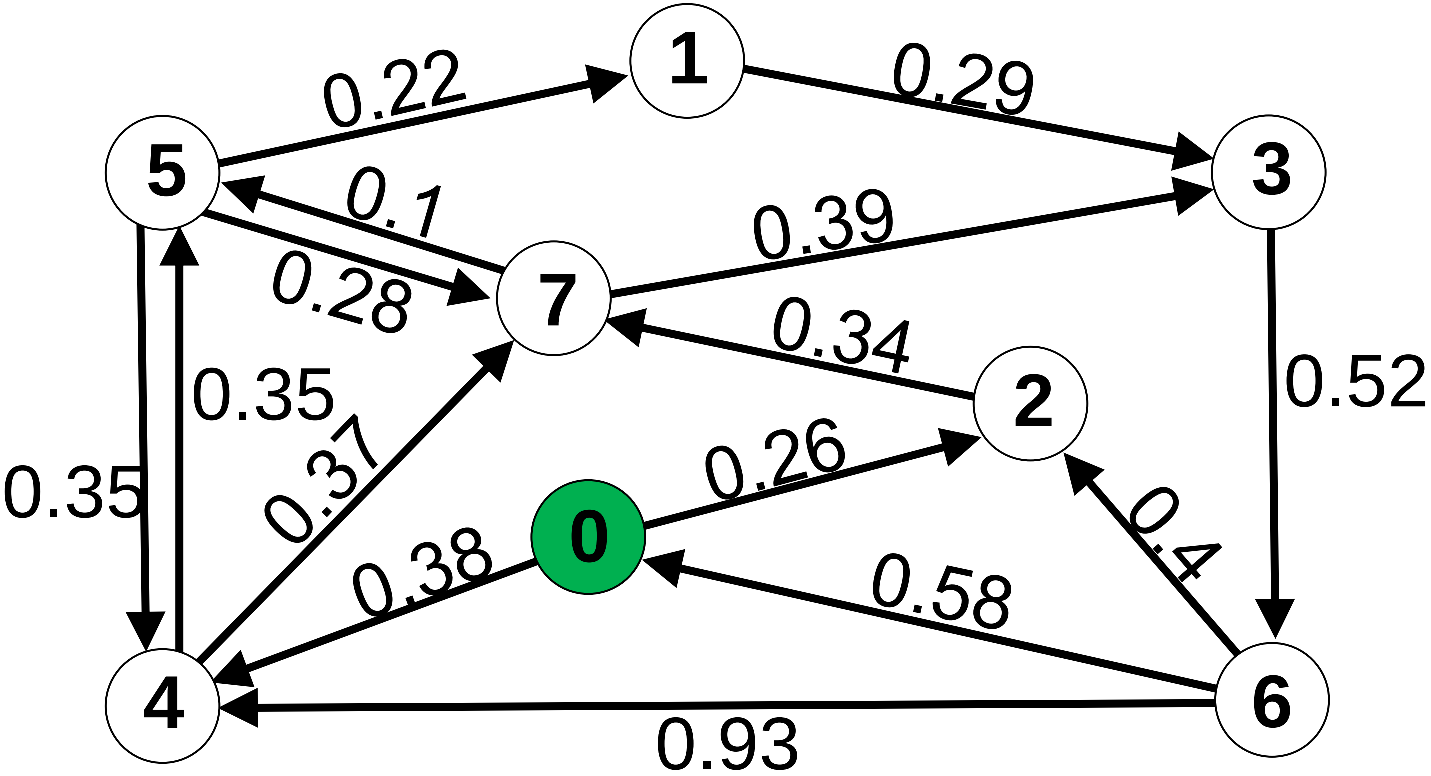
How can we keep track of routes?



	Distanc e from 0	Previou s vertex
0	0	-
1	∞	
2	∞	0
3	∞	7
4	∞	
5	∞	
6	∞	3
7	∞	2

If this is the shortest path from 0 to 6, what can we say about the shortest path from 0 to 3?

Shortest Path

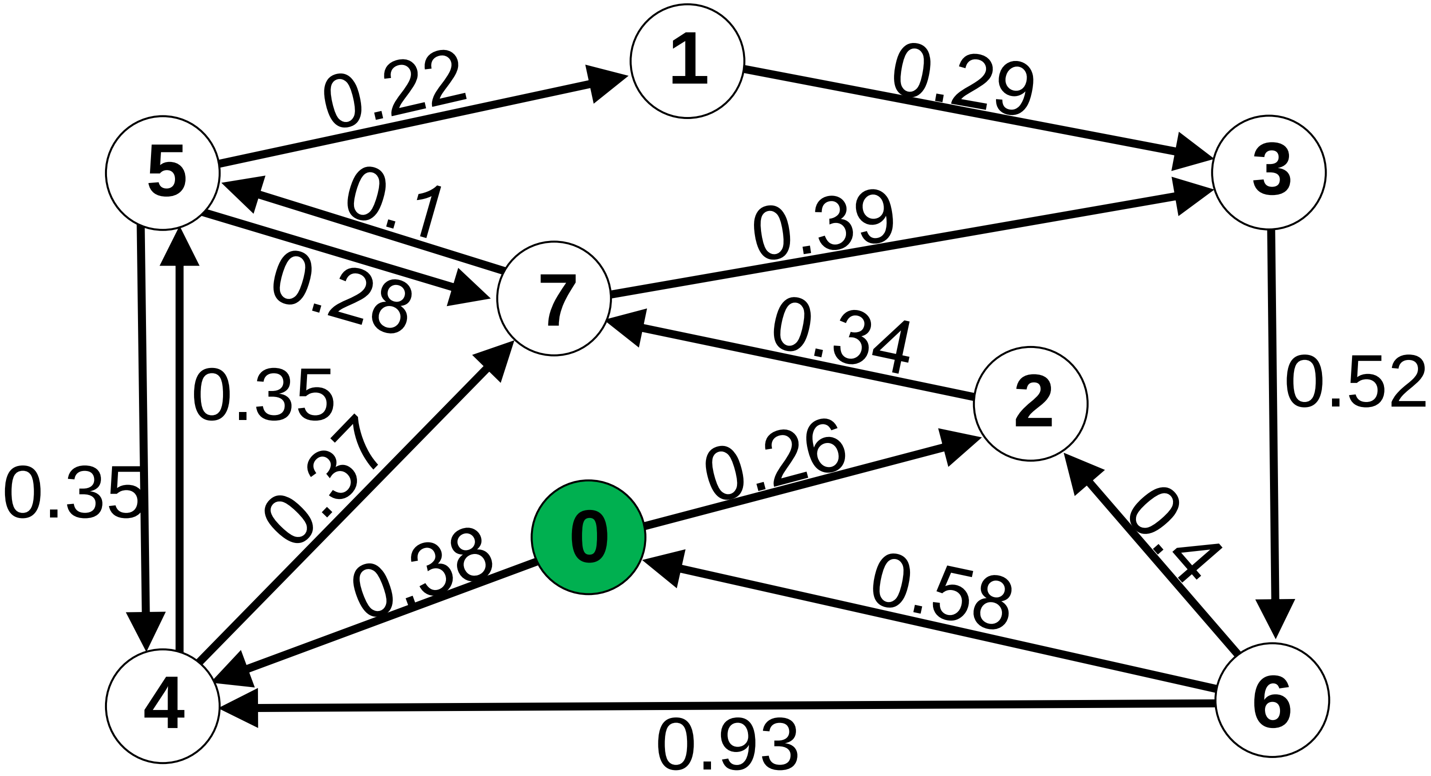


	Distance from 0	Previous vertex
0	0	-
1	∞	
2	∞	
3	∞	
4	∞	
5	∞	
6	∞	
7	∞	

Priority queue

vertex (distance)

What can we reach from connected vertices and at what distance (from 0)?



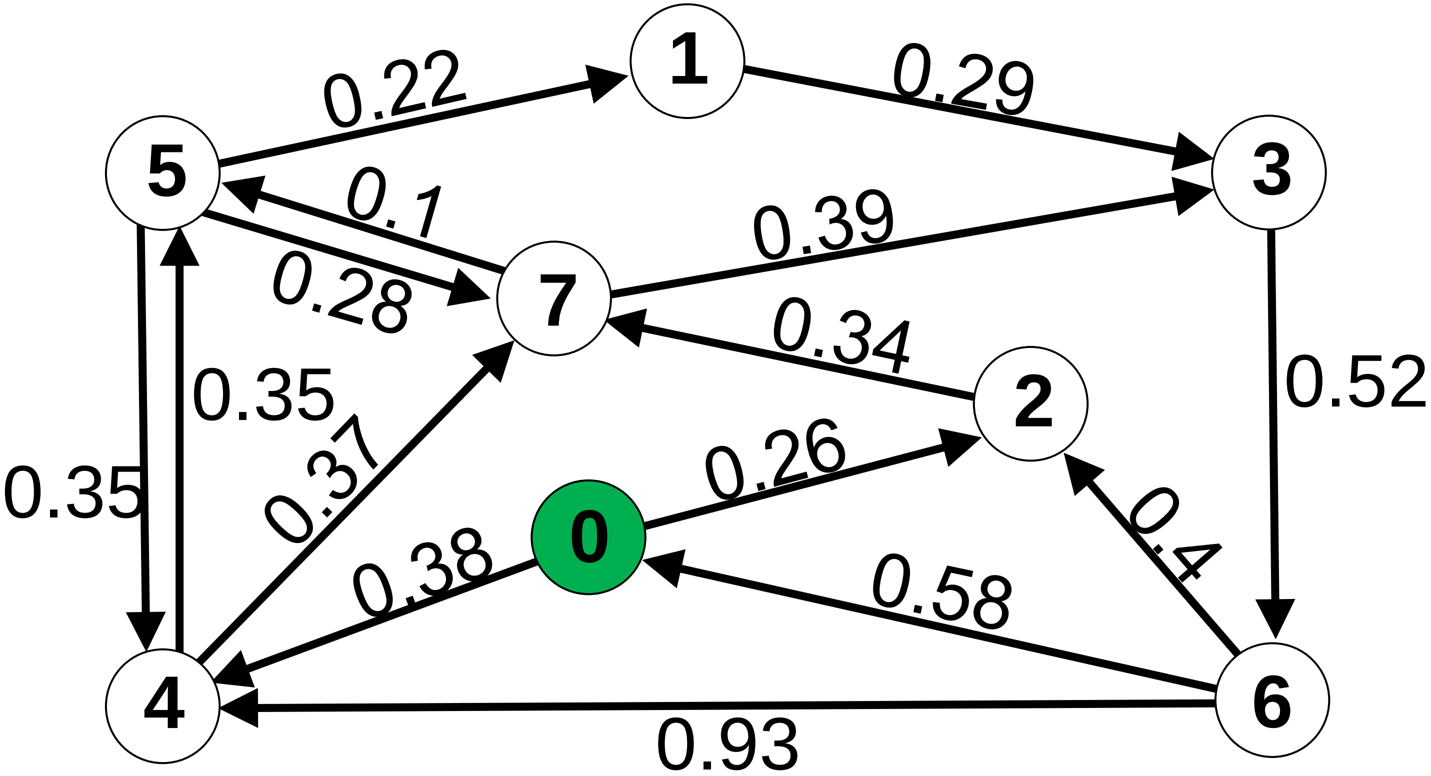
	Distance from 0	Previous vertex
0	0	-
1	∞	
2	∞	
3	∞	
4	∞	
5	∞	
6	∞	
7	∞	

Priority queue

2 (0.26)
4 (0.38)

vertex (distance)

What can we reach from connected vertices and at what distance (from 0)?



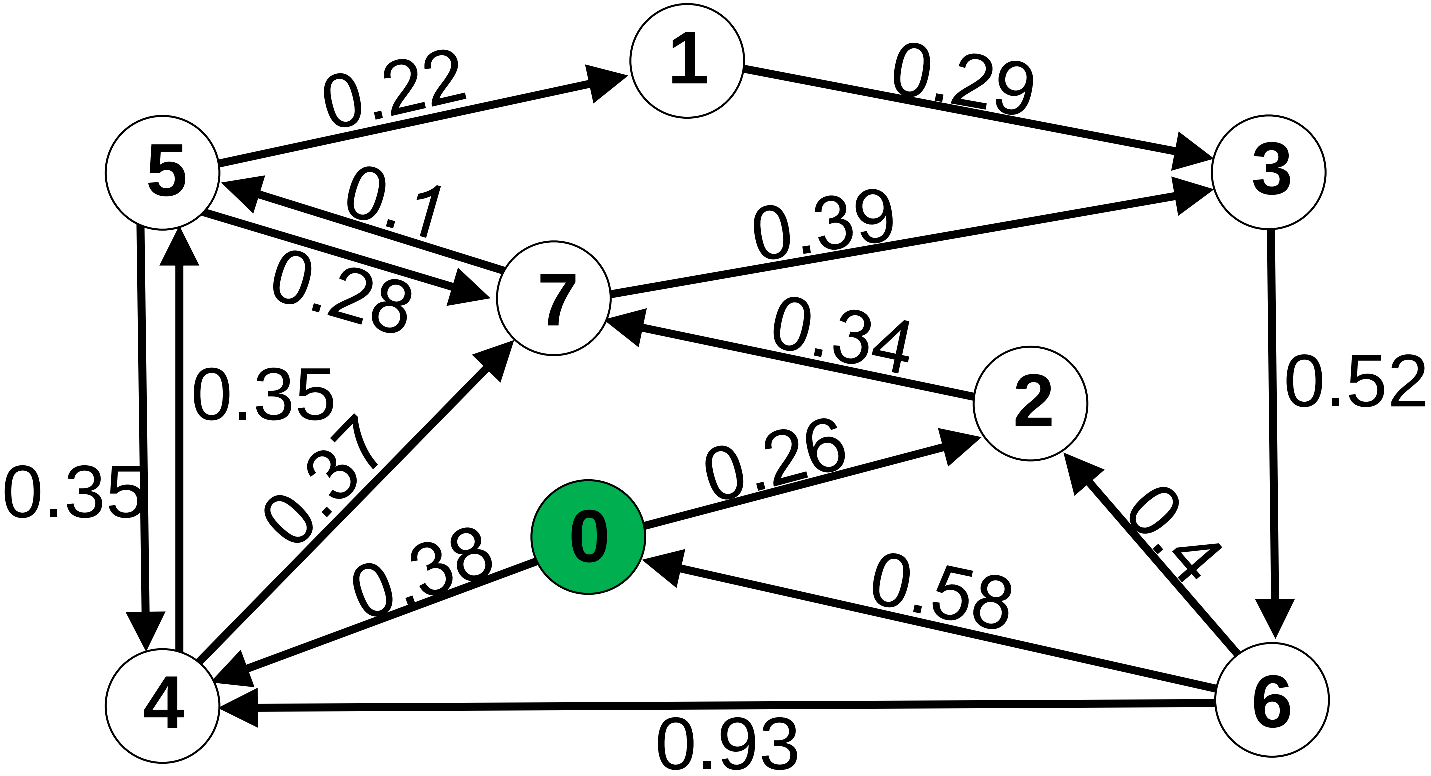
	Distance from 0
0	0
1	∞
2	∞
3	∞
4	∞
5	∞
6	∞
7	∞

	Previous vertex
0	-
1	
2	0
3	
4	0
5	
6	
7	

Priority queue
2 (0.26)
4 (0.38)

vertex (distance)

What can we reach from connected vertices and at what distance (from 0)?



	Distance from 0	Previous vertex
0	0	-
1	∞	
2	0.26	0
3	∞	
4	0.38	0
5	∞	
6	∞	
7	∞	

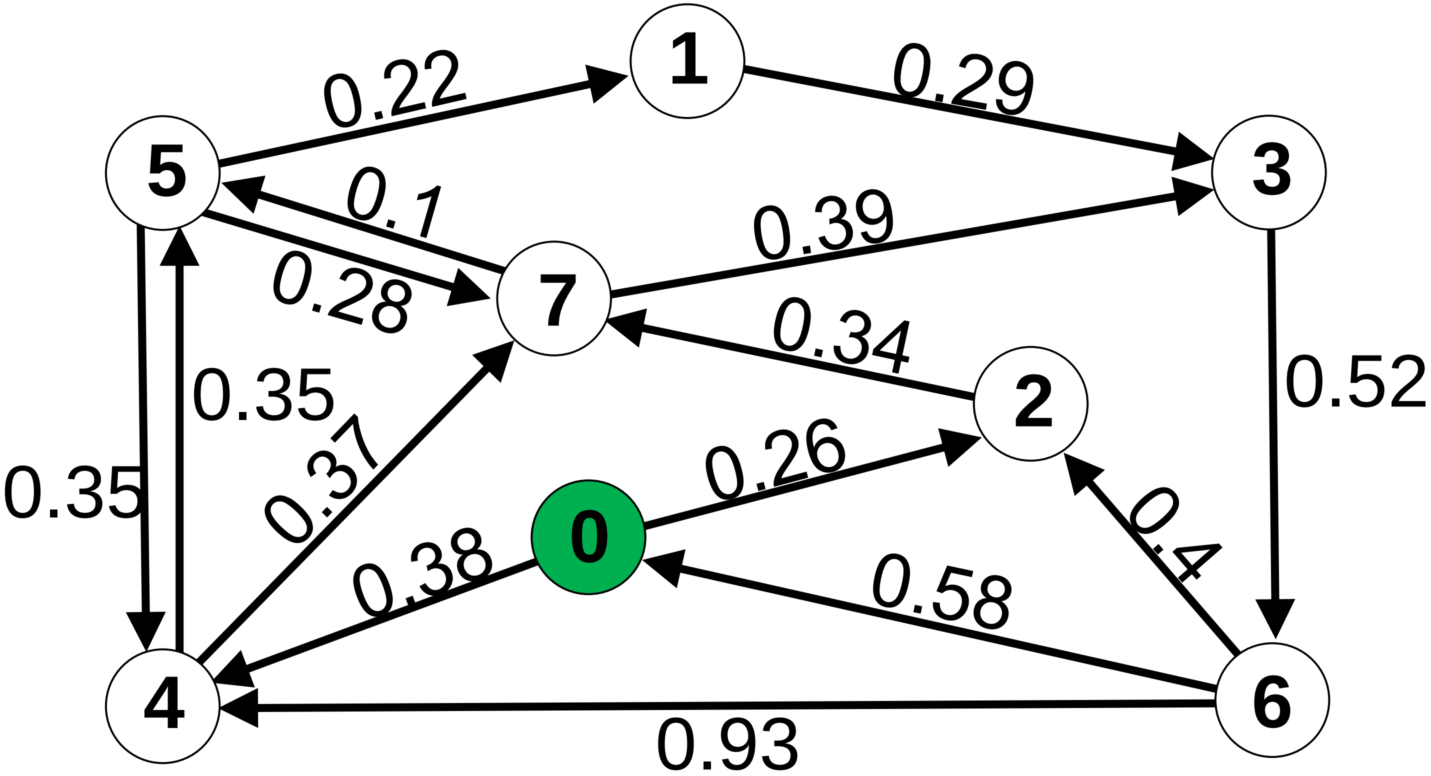
Priority queue

2 (0.26)
4 (0.38)

vertex (distance)

What can we reach from connected vertices and at what distance (from 0)?

queue top = 2 (0.26)

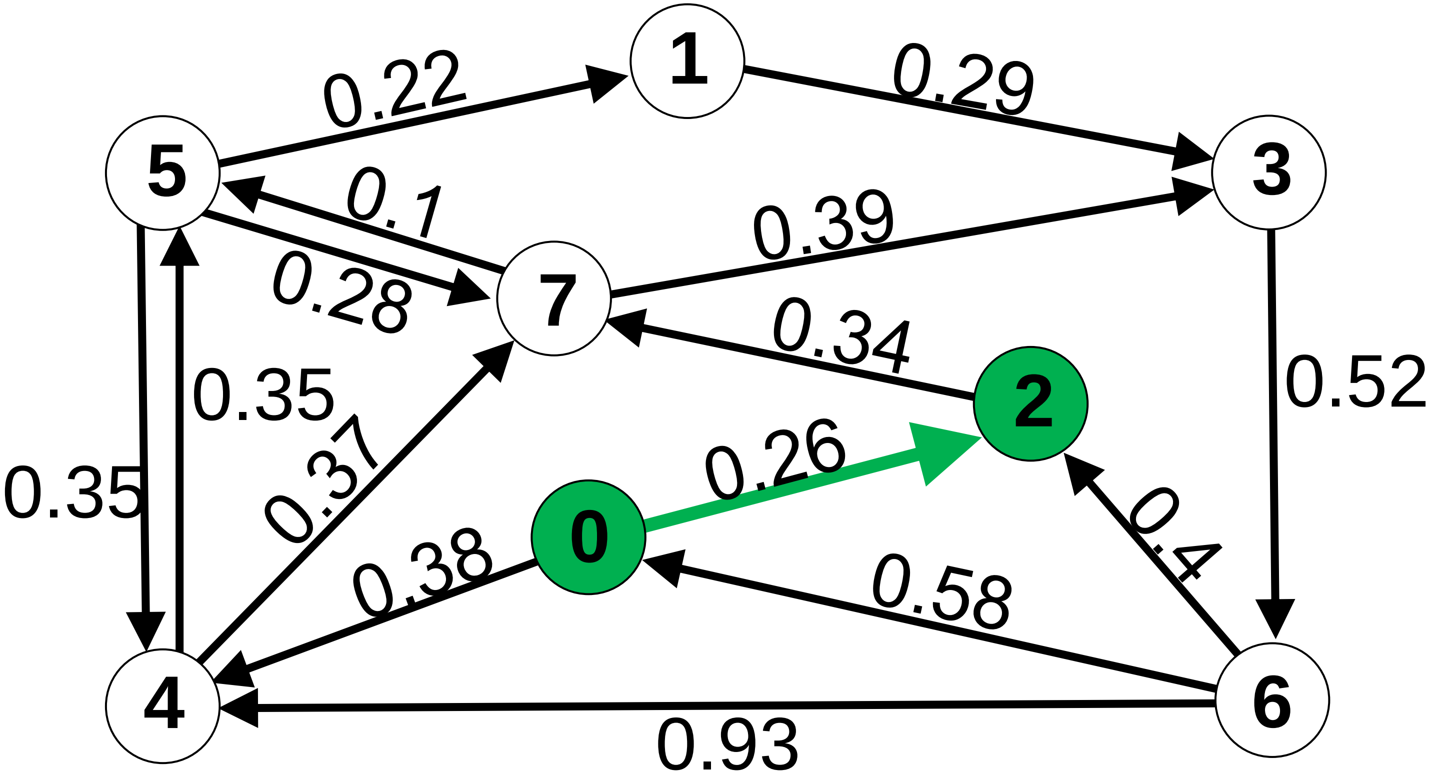


	Distance from 0	Previous vertex	Priority queue
0	0	-	4 (0.38)
1	∞		
2	0.26	0	
3	∞		
4	0.38	0	
5	∞		
6	∞		
7	∞		

vertex (distance)

What can we reach from connected vertices and at what distance (from 0)?

queue top = 2 (0.26)



	Distance from 0	Previous vertex
0	0	-
1	∞	
2	0.26	0
3	∞	
4	0.38	0
5	∞	
6	∞	
7	∞	

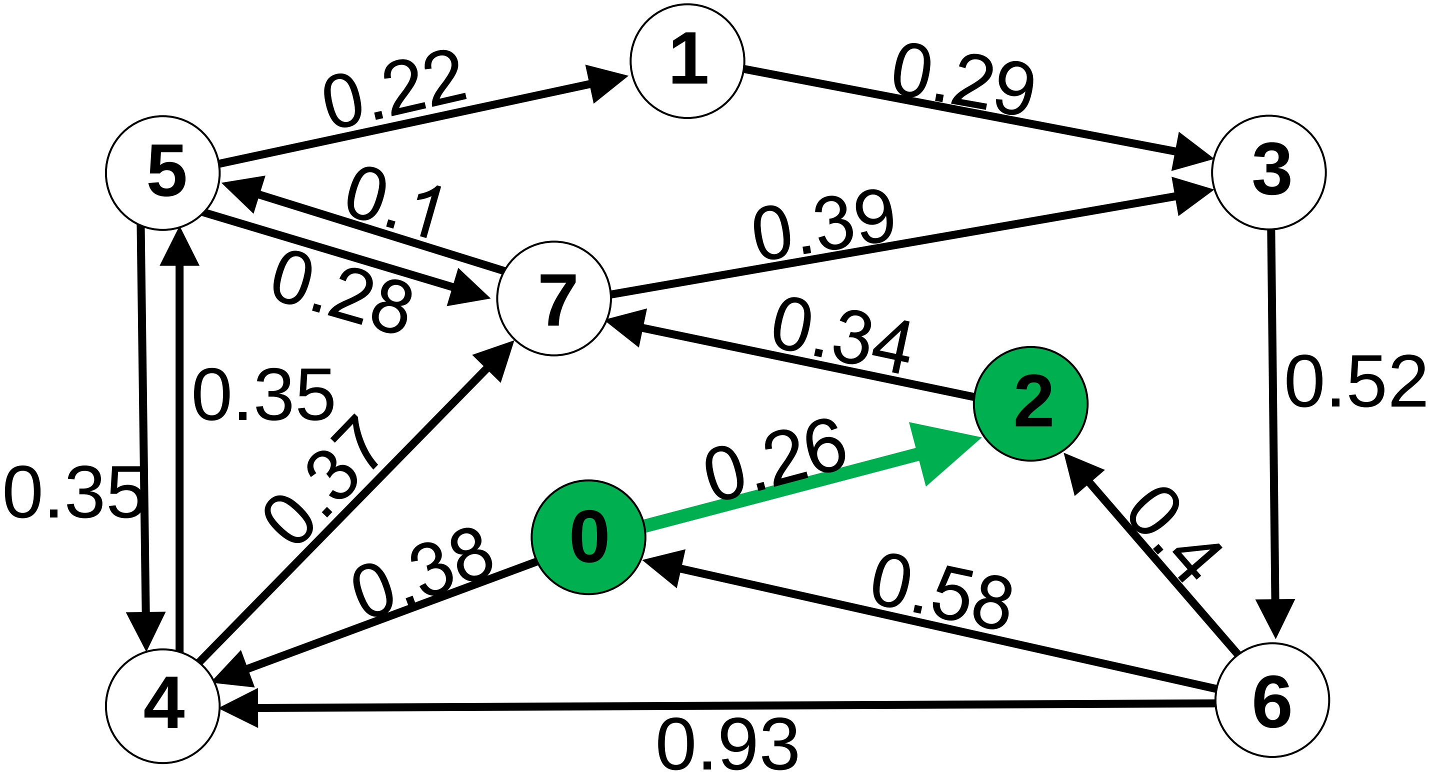
Priority queue

4 (0.38)

vertex (distance)

What can we reach from connected vertices and at what distance (from 0)?

queue top = 2 (0.26)



Distance from 0	
0	0
1	∞
2	0.26
3	∞
4	0.38
5	∞
6	∞
7	0.60

Previous vertex	
0	-
1	
2	0
3	
4	0
5	
6	
7	2

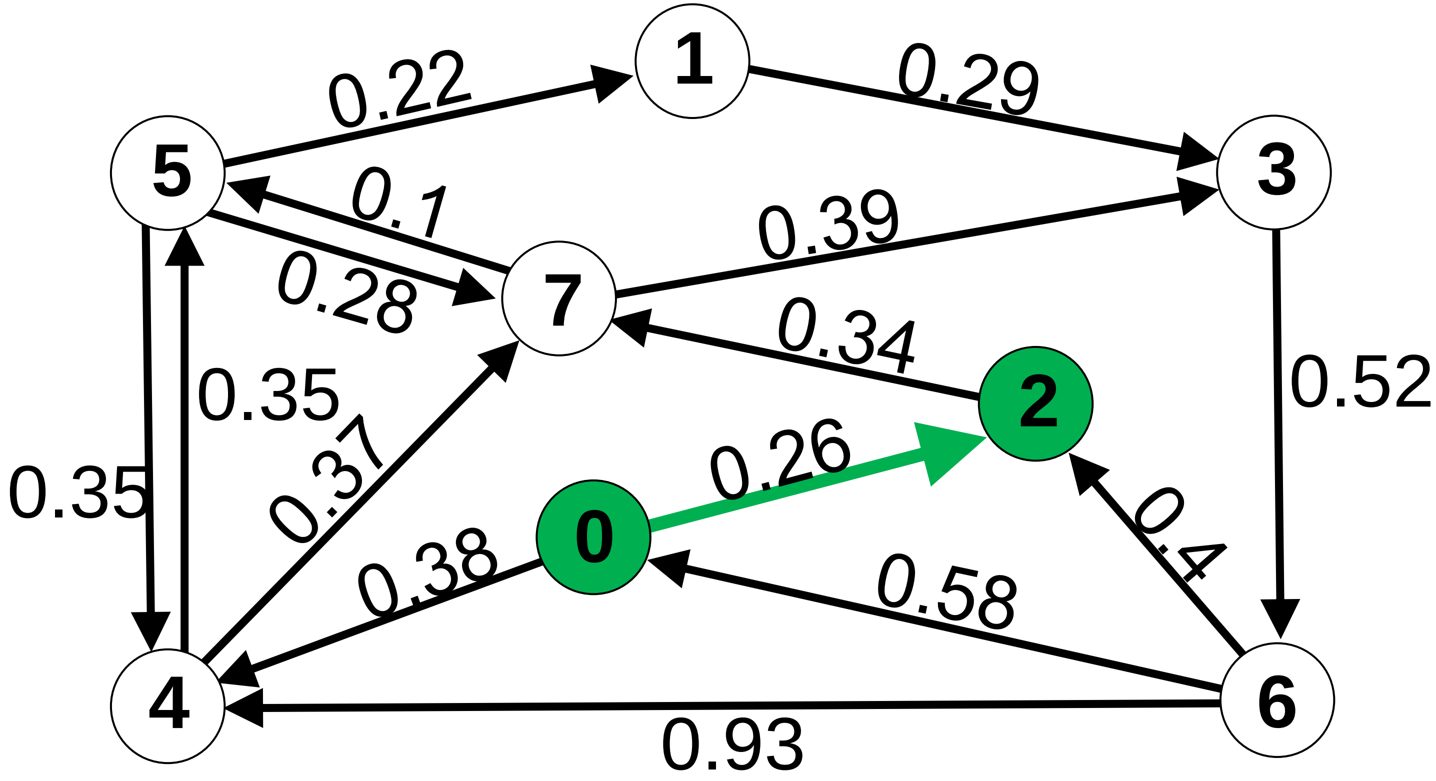
Priority queue

4 (0.38)
7 (0.60)

vertex (distance)

What can we reach from connected vertices and at what distance (from 0)?

queue top = 4 (0.38)



Repeat.

	Distance from 0
0	0
1	∞
2	0.26
3	∞
4	0.38
5	∞
6	∞
7	0.60

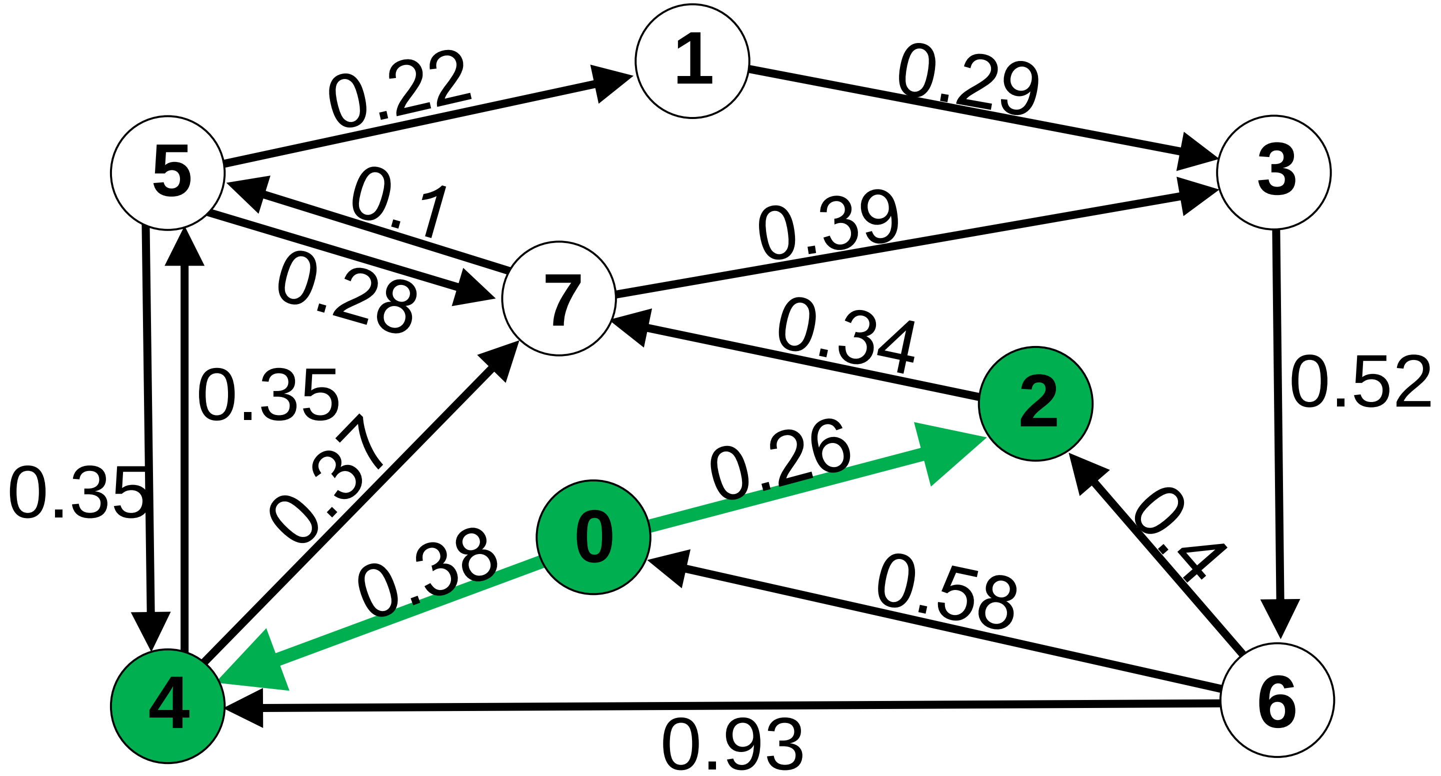
	Previous vertex
0	-
1	
2	0
3	
4	0
5	
6	
7	2

Priority queue

7 (0.60)

vertex (distance)

queue top = 4 (0.38)

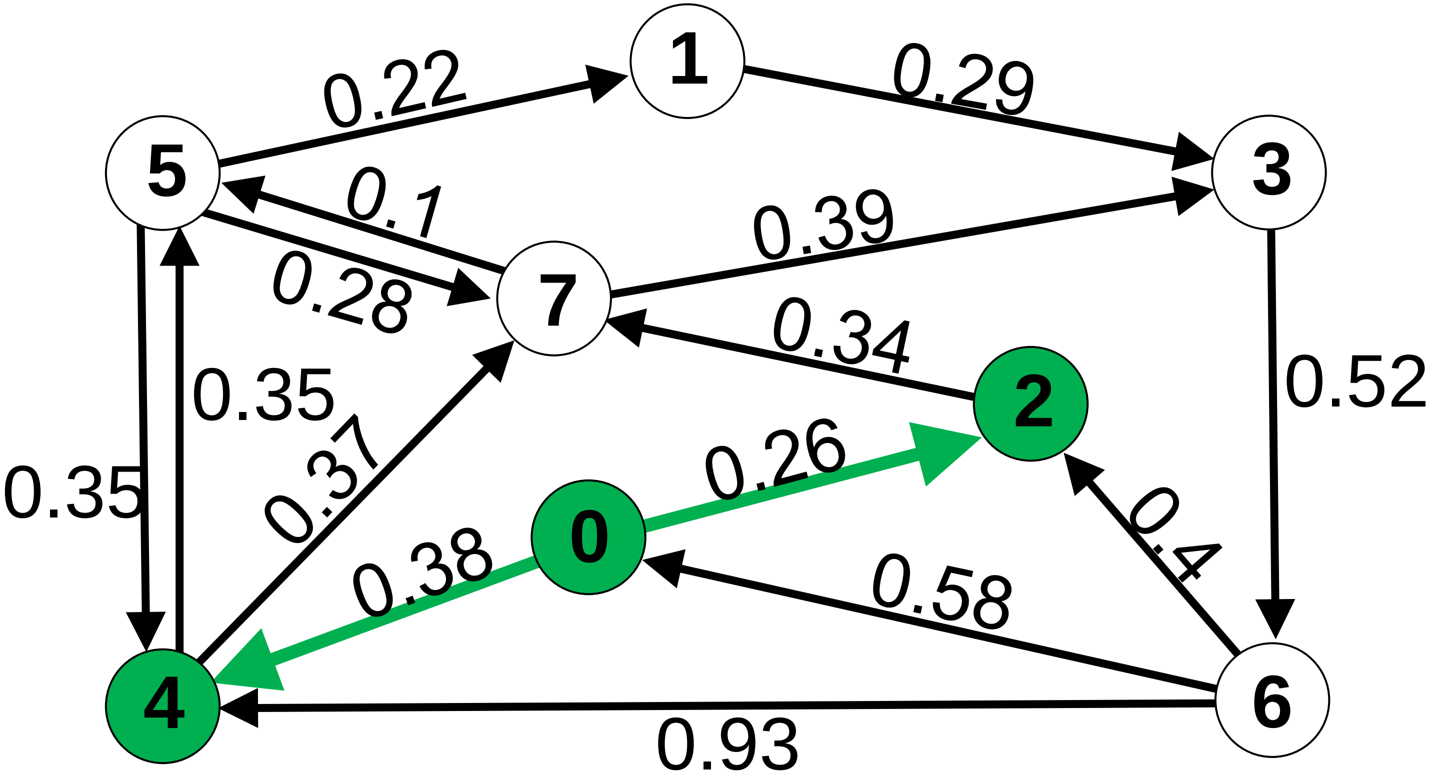


	Distance from 0	Previous vertex	Priority queue
0	0	-	7 (0.60)
1	∞		
2	0.26	0	
3	∞		
4	0.38	0	
5	∞		
6	∞		
7	0.60	2	

vertex (distance)

Add neighbors to queue/previous.

queue top	= 4 (0.38)



Distance from 0

0	0
1	∞
2	0.26
3	∞
4	0.38
5	0.73
6	∞
7	0.60

Previous vertex

0	-
1	
2	0
3	
4	0
5	4
6	
7	2

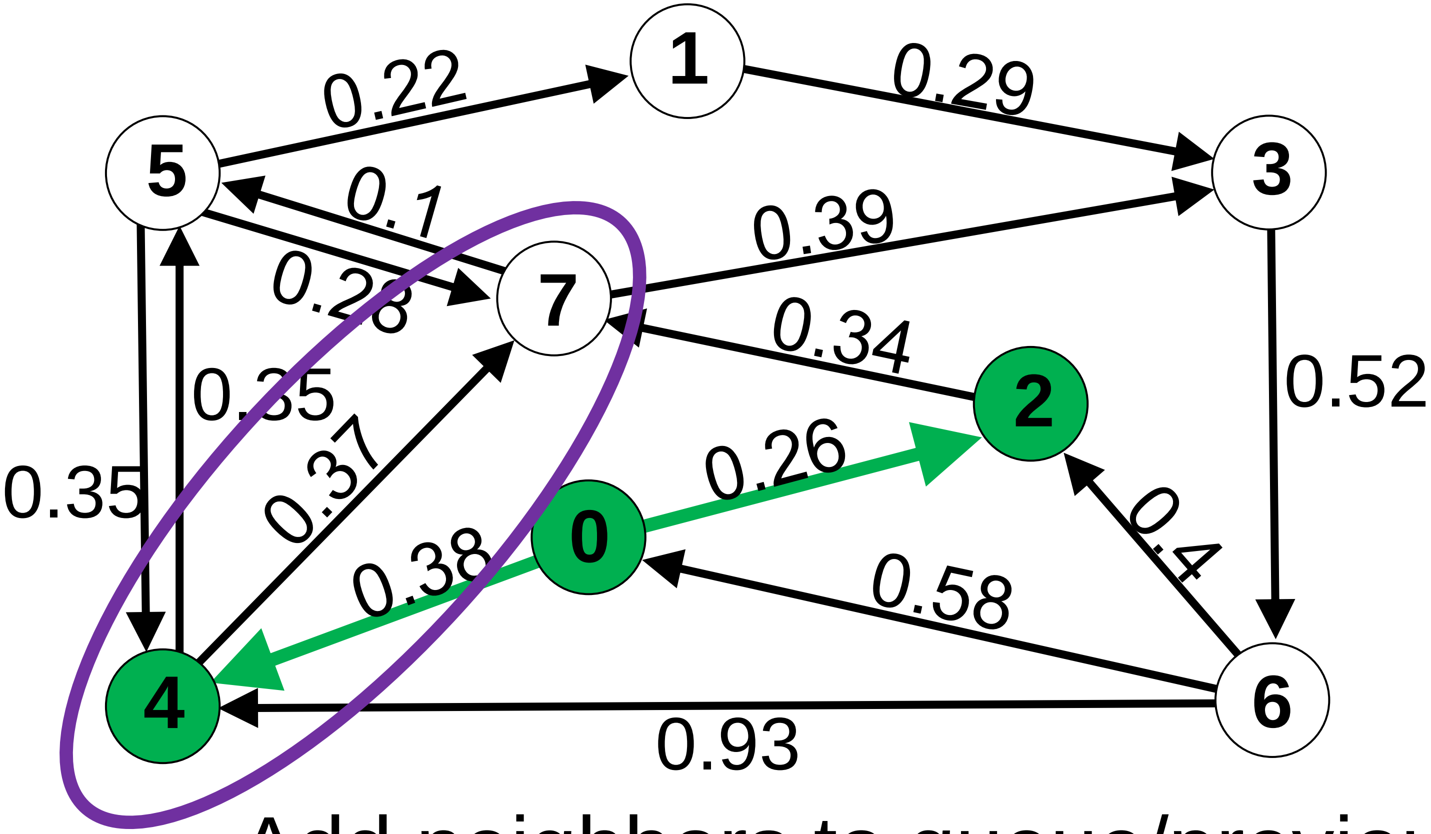
Priority queue

7 (0.60)
5 (0.73)

vertex (distance)

Add neighbors to queue/previous.

queue top	= 4 (0.38)



Add neighbors to queue/previous.

We have another route to 7!

	Distance from 0
0	0
1	∞
2	0.26
3	∞
4	0.38
5	0.73
6	∞
7	0.60

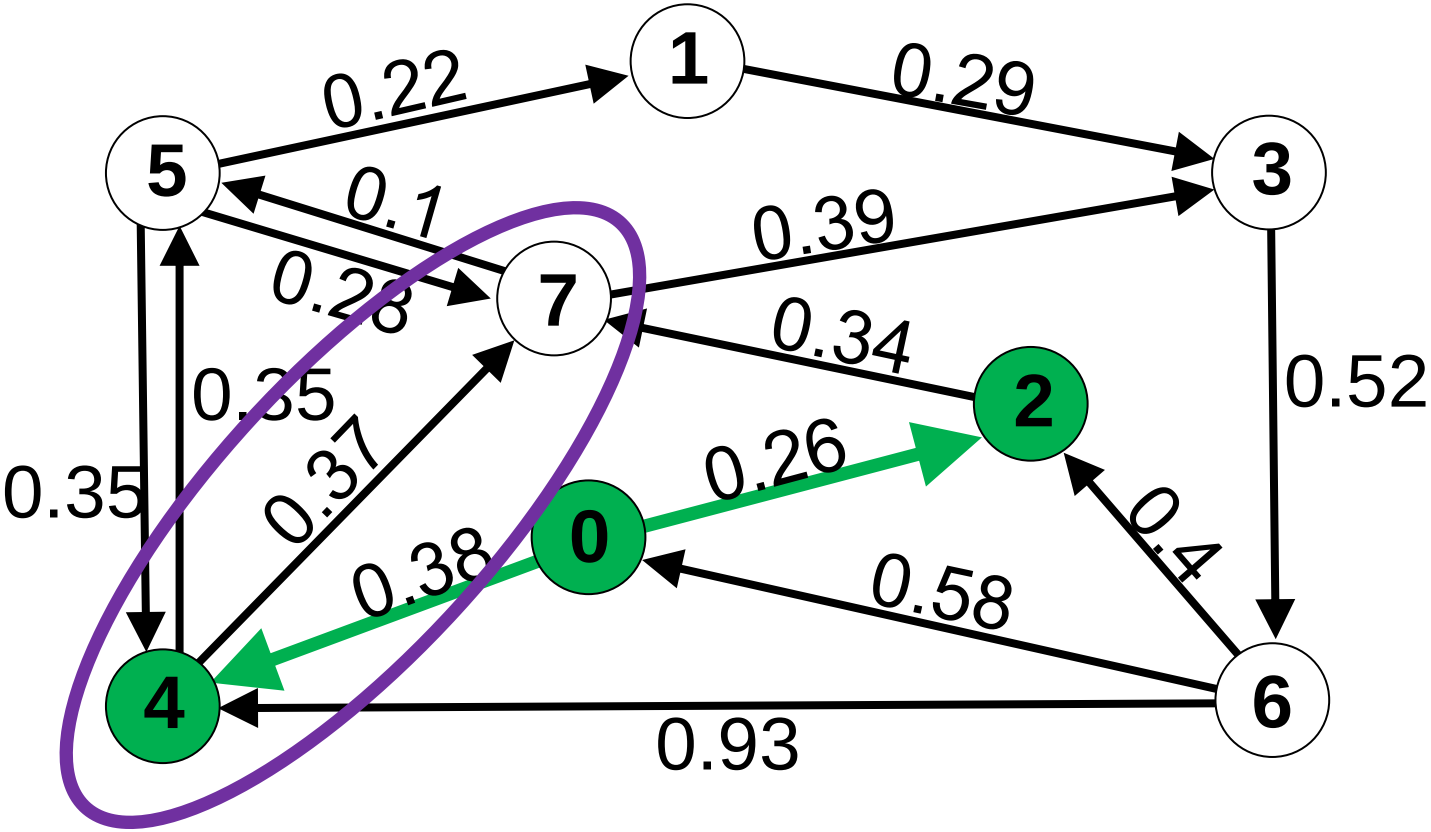
	Previous vertex
0	-
1	
2	0
3	
4	0
5	4
6	
7	2

Priority queue

7 (0.60)
5 (0.73)

vertex (distance)

queue top = 4 (0.38)



	Distance from 0
0	0
1	∞
2	0.26
3	∞
4	0.38
5	0.73
6	∞
7	0.60

	Previous vertex
0	-
1	
2	0
3	
4	0
5	4
6	
7	2

Priority queue

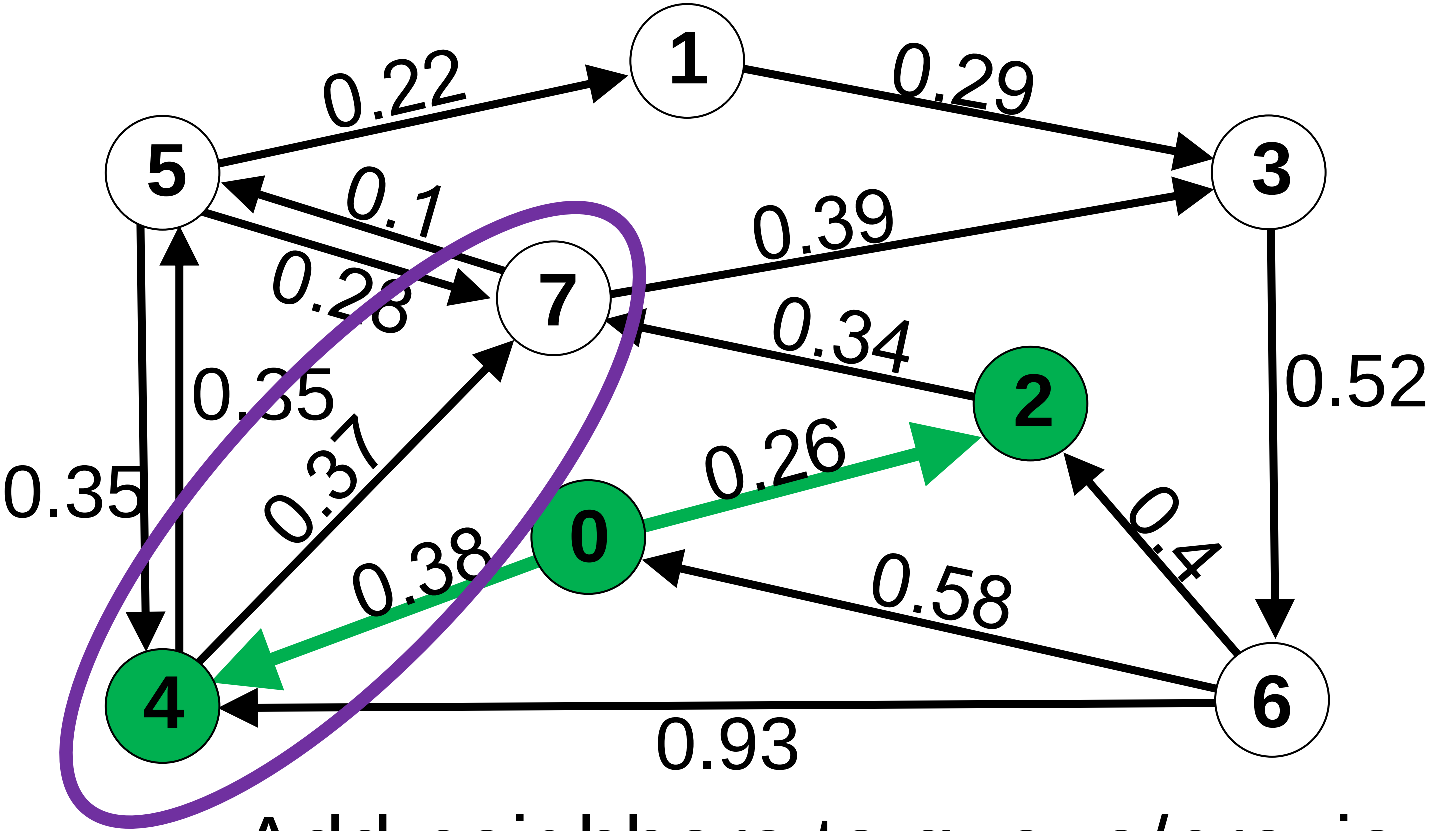
7 (0.60)
5 (0.73)

vertex (distance)

Add neighbors to queue/previous.

We have another route to 7! Check to see if it is shorter!

queue top = 4 (0.38)



Distance from 0	
0	0
1	∞
2	0.26
3	∞
4	0.38
5	0.73
6	∞
7	0.60

Previous vertex	
0	-
1	
2	0
3	
4	0
5	4
6	
7	2

Priority queue

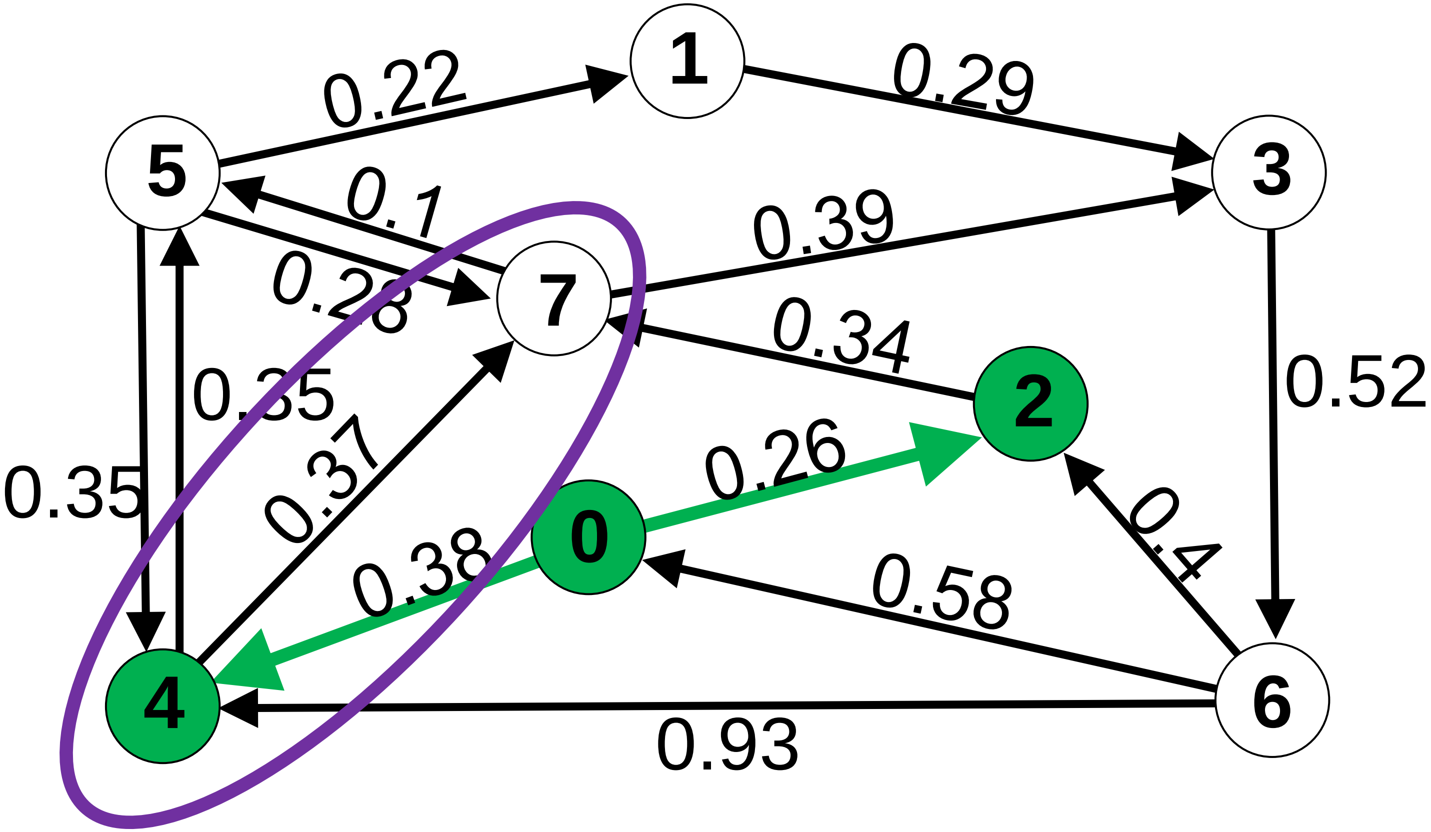
7 (0.60)
5 (0.73)

vertex (distance)

Add neighbors to queue/previous.

We have another route to 7! Check to see if it is shorter! It's not (0.38 + 0.37 = 0.75 > 0.60).

queue top	= 4 (0.38)



	Distance from 0
0	0
1	∞
2	0.26
3	∞
4	0.38
5	0.73
6	∞
7	0.60

	Previous vertex
0	-
1	
2	0
3	
4	0
5	4
6	
7	2

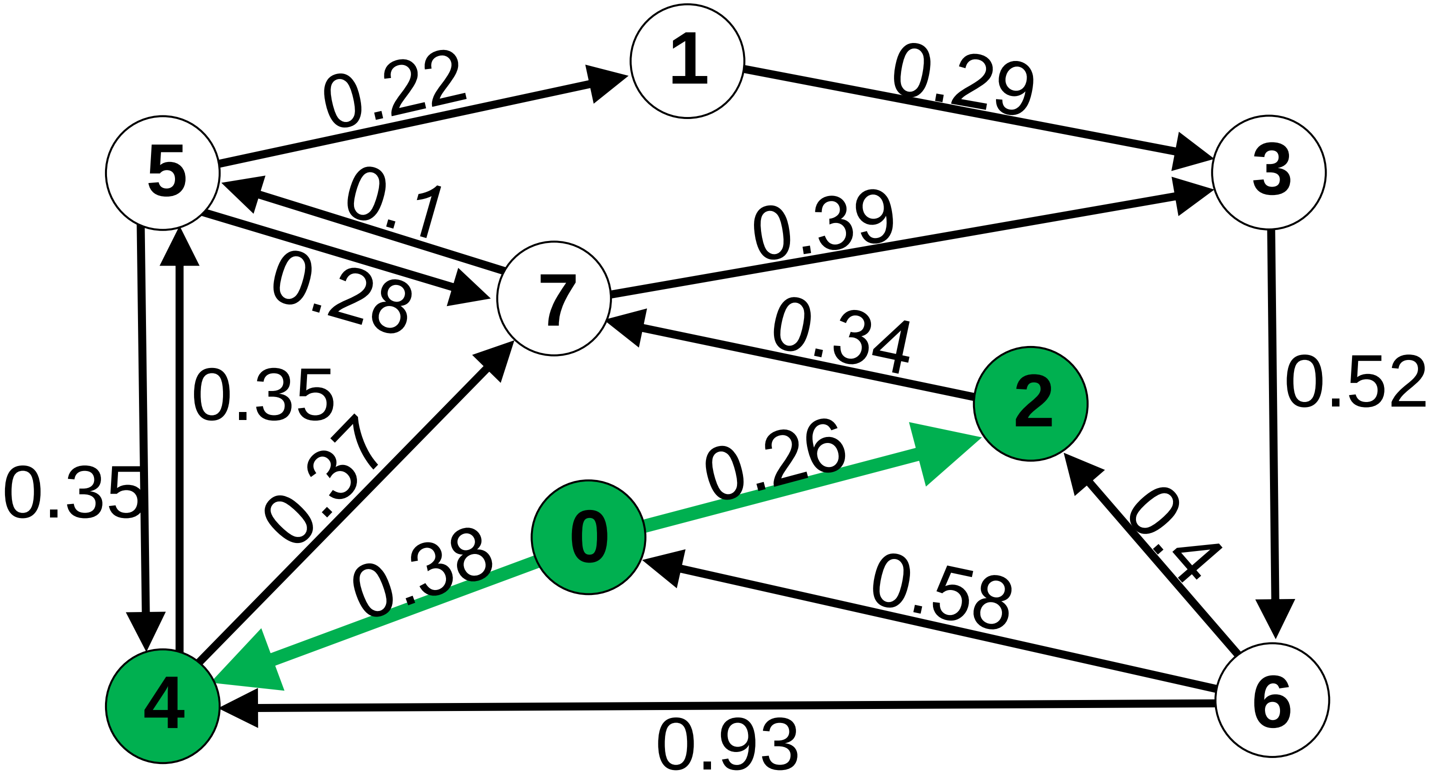
Priority queue
7 (0.60)
5 (0.73)

vertex (distance)

Rule: When processing vertex v, only add/modify queue for neighbor u if and only if:

`distance[v] + weight(v, u) < distance[u]`

`queue top` =



Repeat.

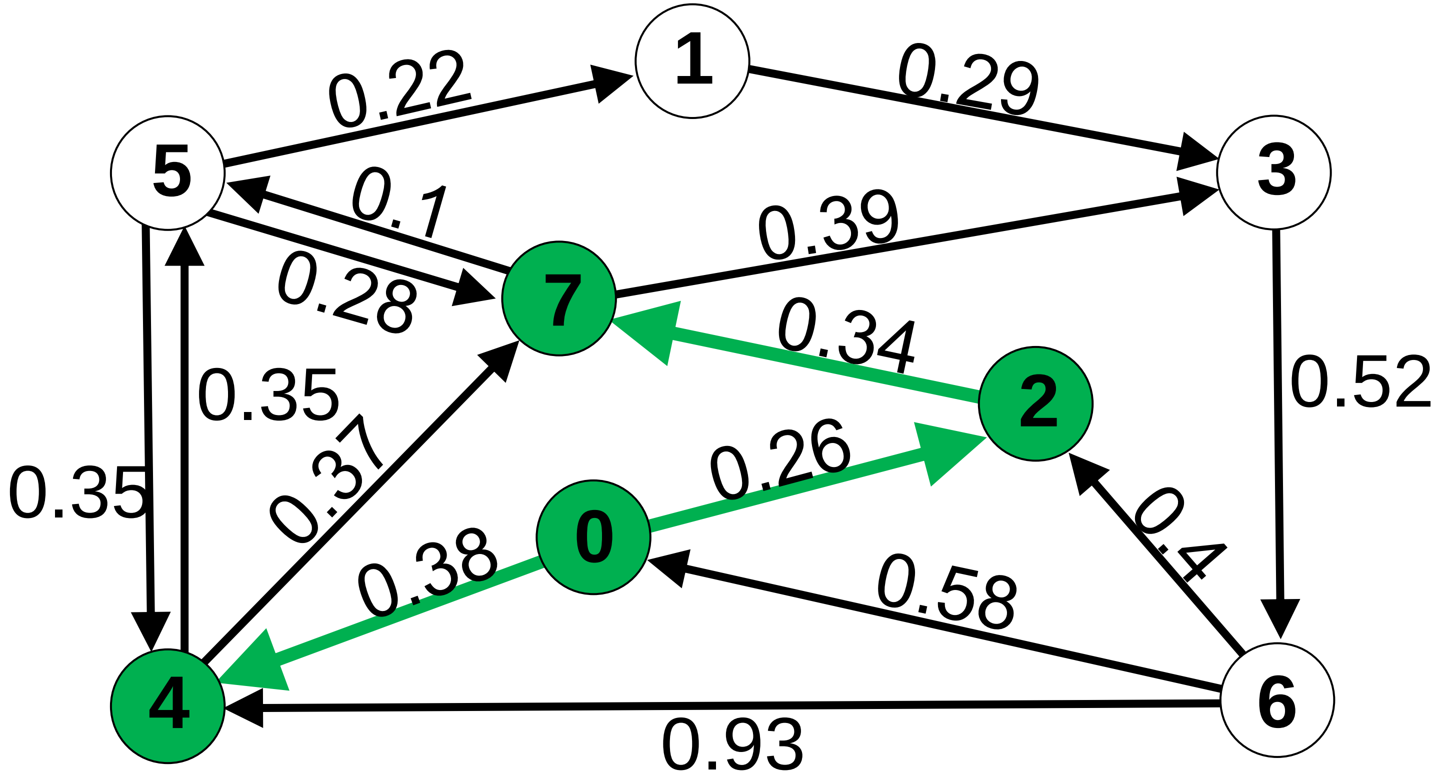
	Distance from 0
0	0
1	∞
2	0.26
3	∞
4	0.38
5	0.73
6	∞
7	0.60

	Previous vertex
0	-
1	
2	0
3	
4	0
5	4
6	
7	2

Priority queue
7 (0.60)
5 (0.73)

vertex (distance)

queue top = 7 (0.60)



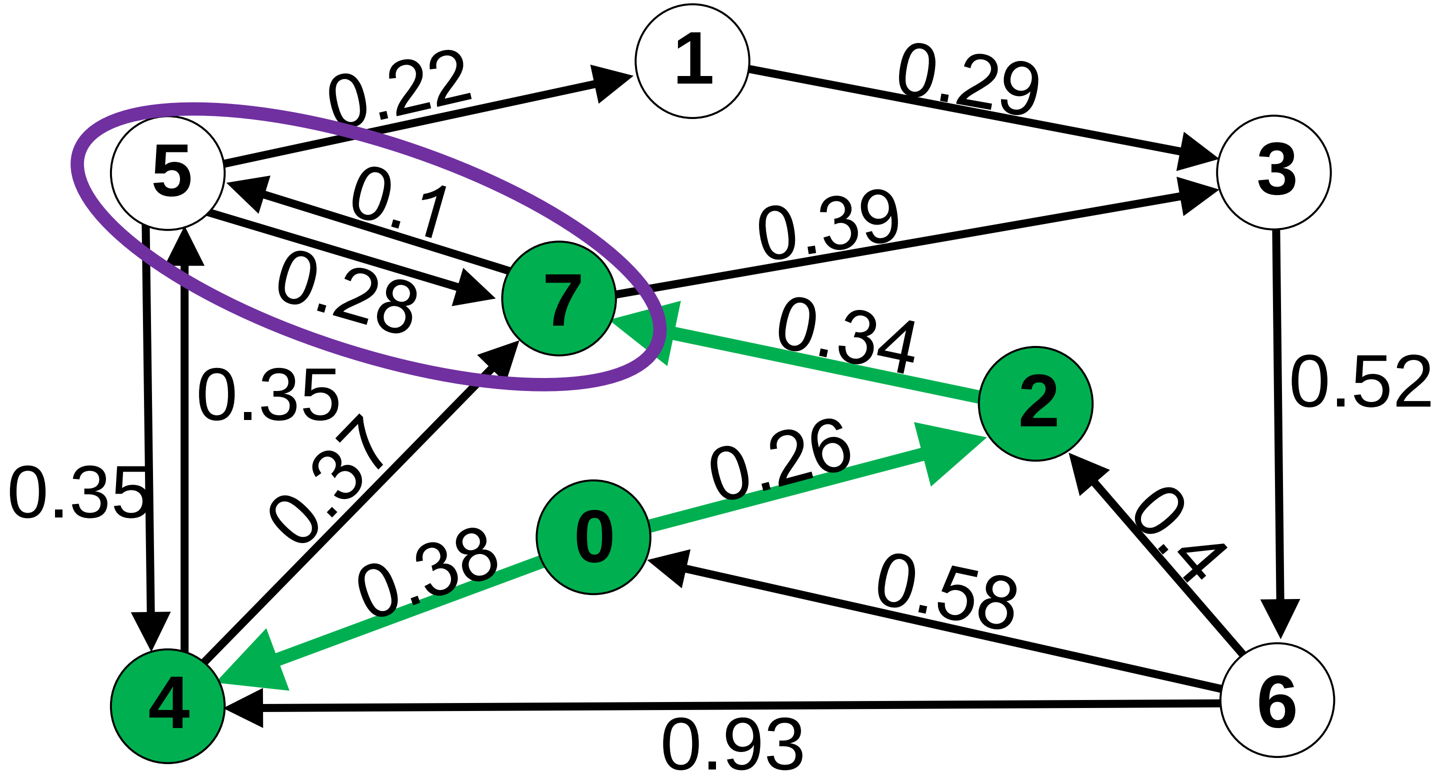
Repeat.

Vertex	Distance from 0	Previous vertex
0	0	-
1	∞	
2	0.26	0
3	0.99	7
4	0.38	0
5	0.73	4
6	∞	
7	0.60	2

Priority queue:
5 (0.73)
3 (0.99)

vertex (distance)

queue top = 7 (0.60)



Distance from 0	
0	0
1	∞
2	0.26
3	0.99
4	0.38
5	0.73
6	∞
7	0.60

Previous vertex	
0	-
1	
2	0
3	7
4	0
5	4
6	
7	2

Priority queue

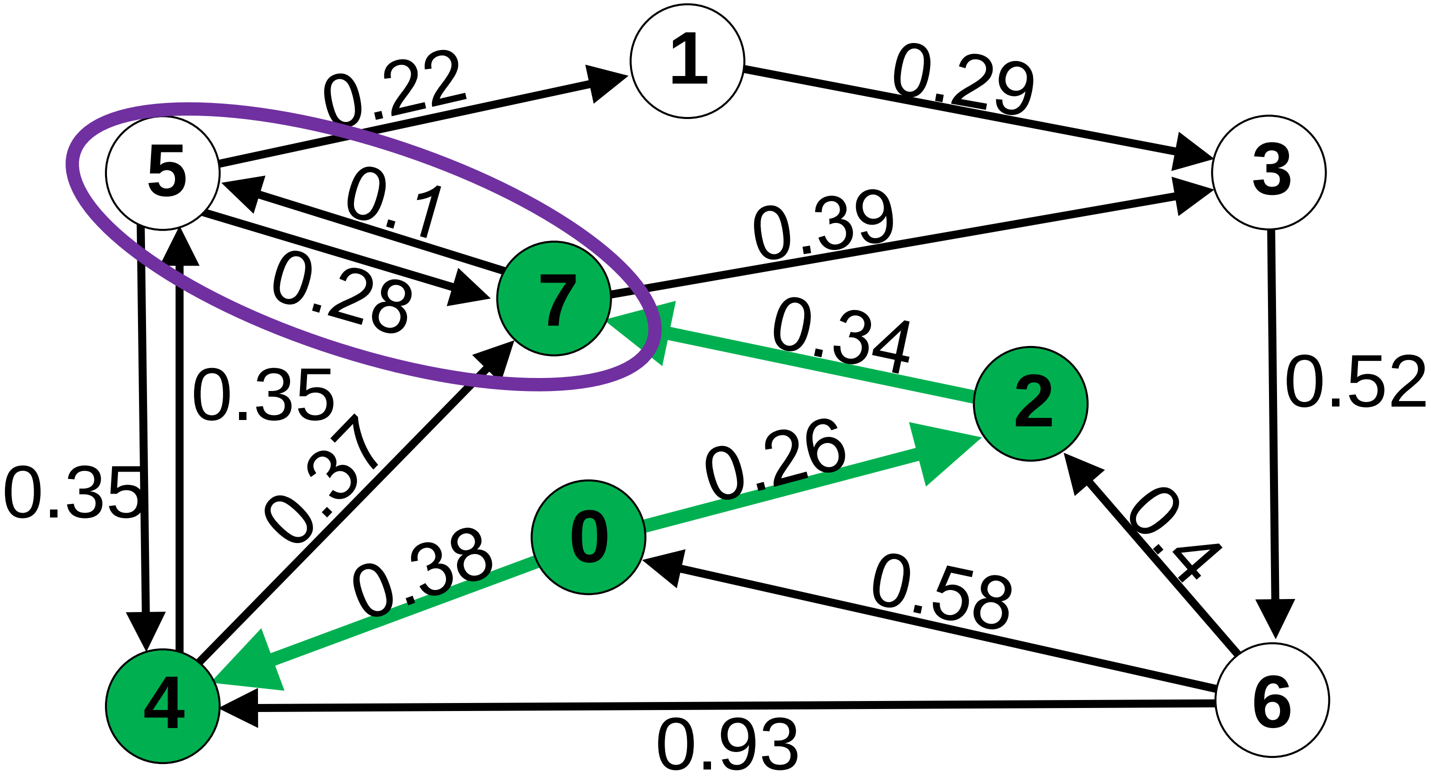
5 (0.73)
3 (0.99)

vertex (distance)

Repeat.

We have another route to 5, and at cost 0.7 <

queue top	= 7 (0.60)



	Distance from 0
0	0
1	∞
2	0.26
3	0.99
4	0.38
5	0.73
6	∞
7	0.60

	Previous vertex
0	-
1	
2	0
3	7
4	0
5	4
6	
7	2

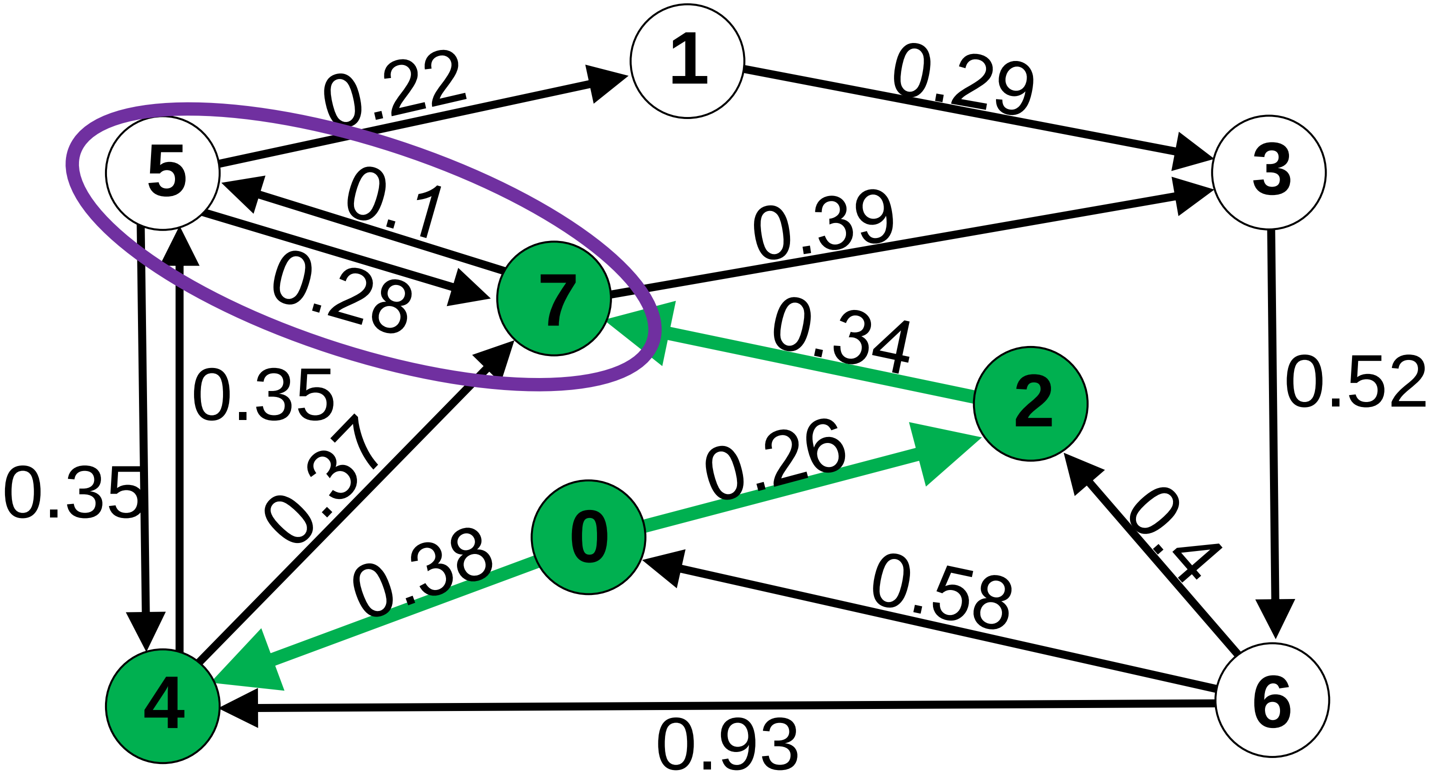
Priority queue
5 (0.73)
3 (0.99)

vertex (distance)

Repeat. **We have another route to 5, and at cost 0.7 < 0.73.**

i.e., `distance[v] + weight(v, u) < distance[u]`

queue top = 7 (0.60)



	Distance from 0	Previous vertex
0	0	-
1	∞	
2	0.26	0
3	0.99	7
4	0.38	0
5	~~0.73~~ 0.70	4 7
6	∞	
7	0.60	2

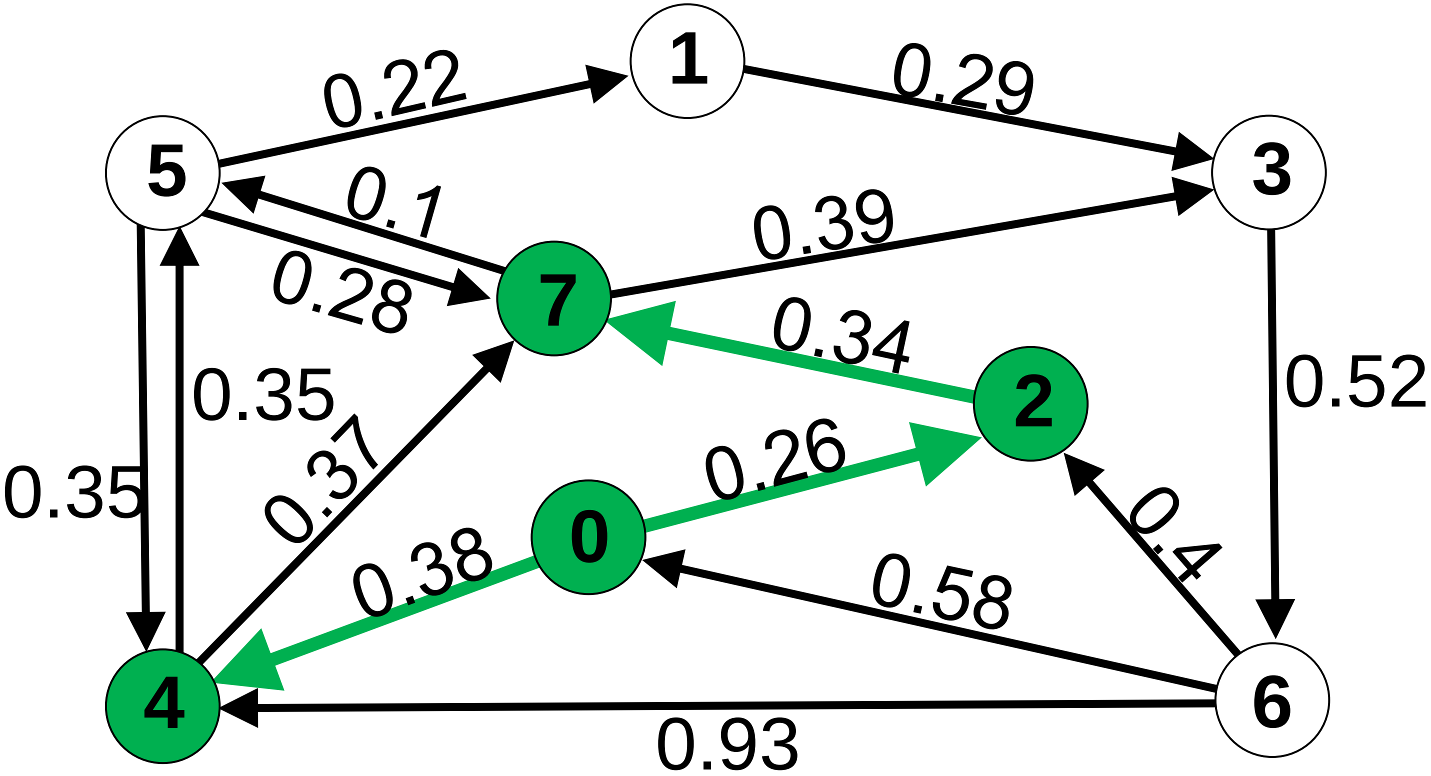
Priority queue

5 (~~0.73~~) 0.70
3 (0.99)

vertex (distance)

Repeat. **We have another route to 5, and at cost 0.7 < 0.73. So updated queue/previous/distance.**

queue top	= 7 (0.60)



	Distance from 0		Previous vertex
0	0	0	-
1	∞	1	
2	0.26	2	0
3	0.99	3	7
4	0.38	4	0
5	0.70	5	7
6	∞	6	
7	0.60	7	2

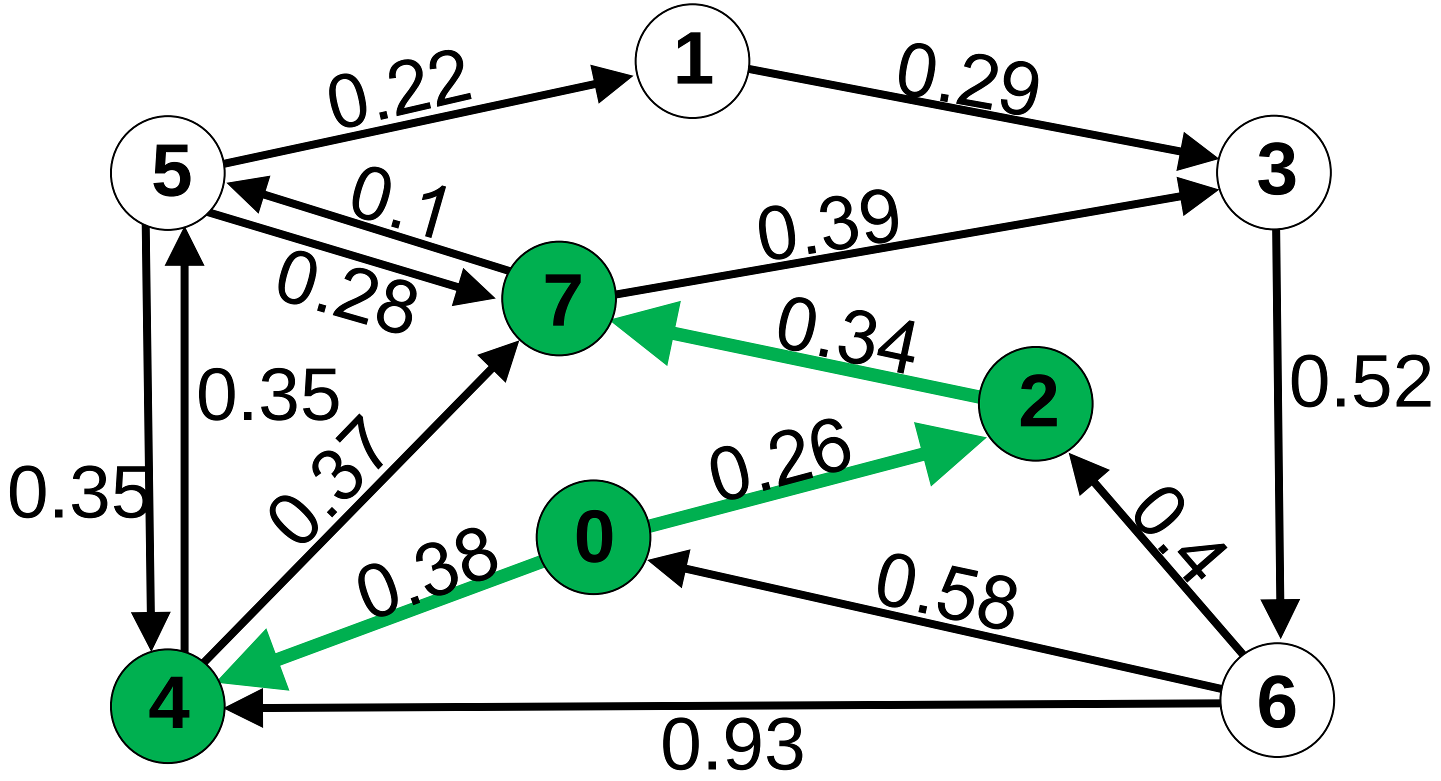
Priority queue

5 (0.70)
3 (0.99)

vertex (distance)

Repeat. **We have another route to 5, and at cost 0.7 < 0.73. So updated queue/previous/distance.**

queue top = 7 (0.60)



Repeat.

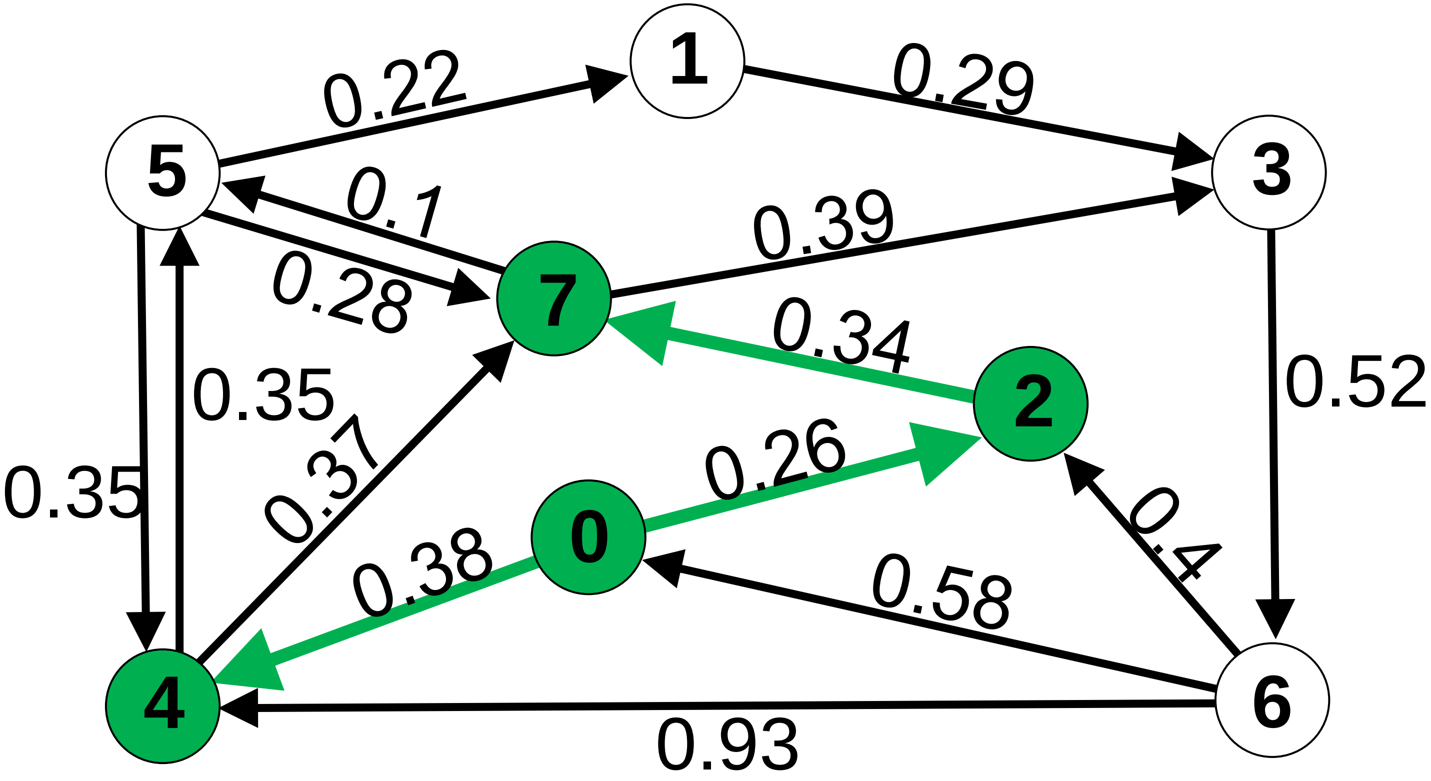
	Distance from 0	Previous vertex
0	0	-
1	∞	
2	0.26	0
3	0.99	7
4	0.38	0
5	0.70	7
6	∞	
7	0.60	2

Priority queue

5 (0.70)
3 (0.99)

vertex (distance)

queue top = 5 (0.70)



Repeat.

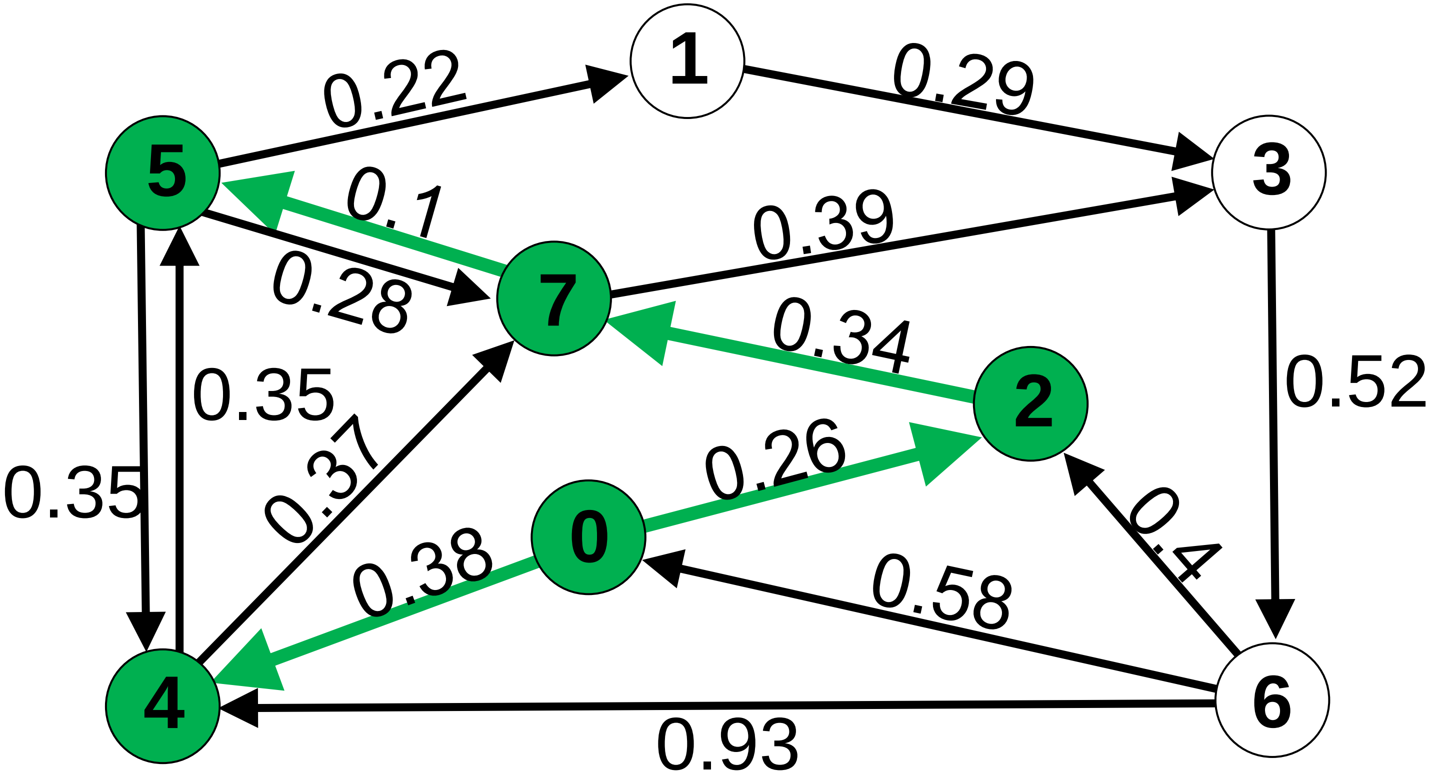
	Distance from 0	Previous vertex
0	0	-
1	∞	
2	0.26	0
3	0.99	7
4	0.38	0
5	0.70	7
6	∞	
7	0.60	2

Priority queue

3 (0.99)

vertex (distance)

Shortest Path



queue top	= 5 (0.70)

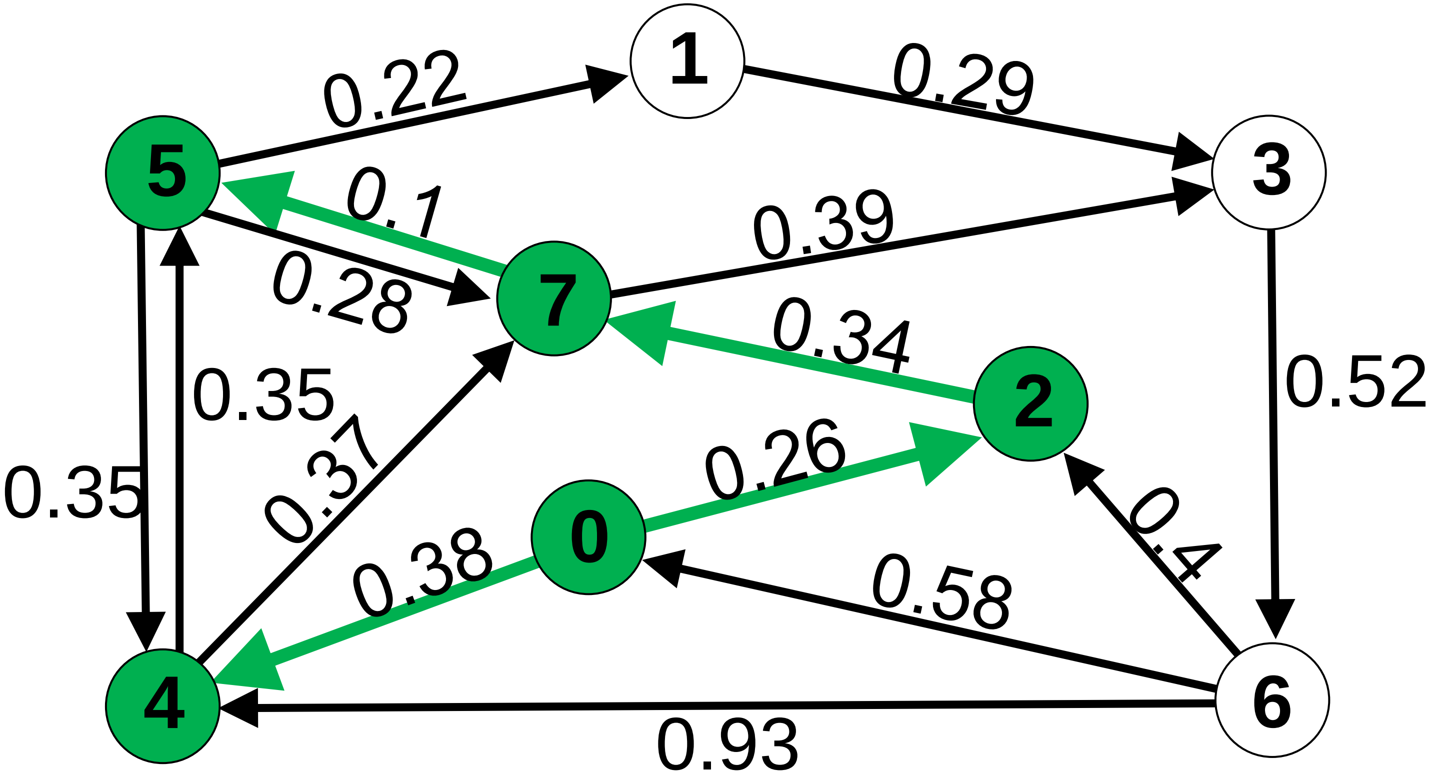
	Distance from 0	Previous vertex
0	0	-
1	∞	
2	0.26	0
3	0.99	7
4	0.38	0
5	0.70	7
6	∞	
7	0.60	2

Priority queue
3 (0.99)

vertex (distance)

Repeat.

queue top	= 5 (0.70)



Repeat.

	Distance from 0
0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	∞
7	0.60

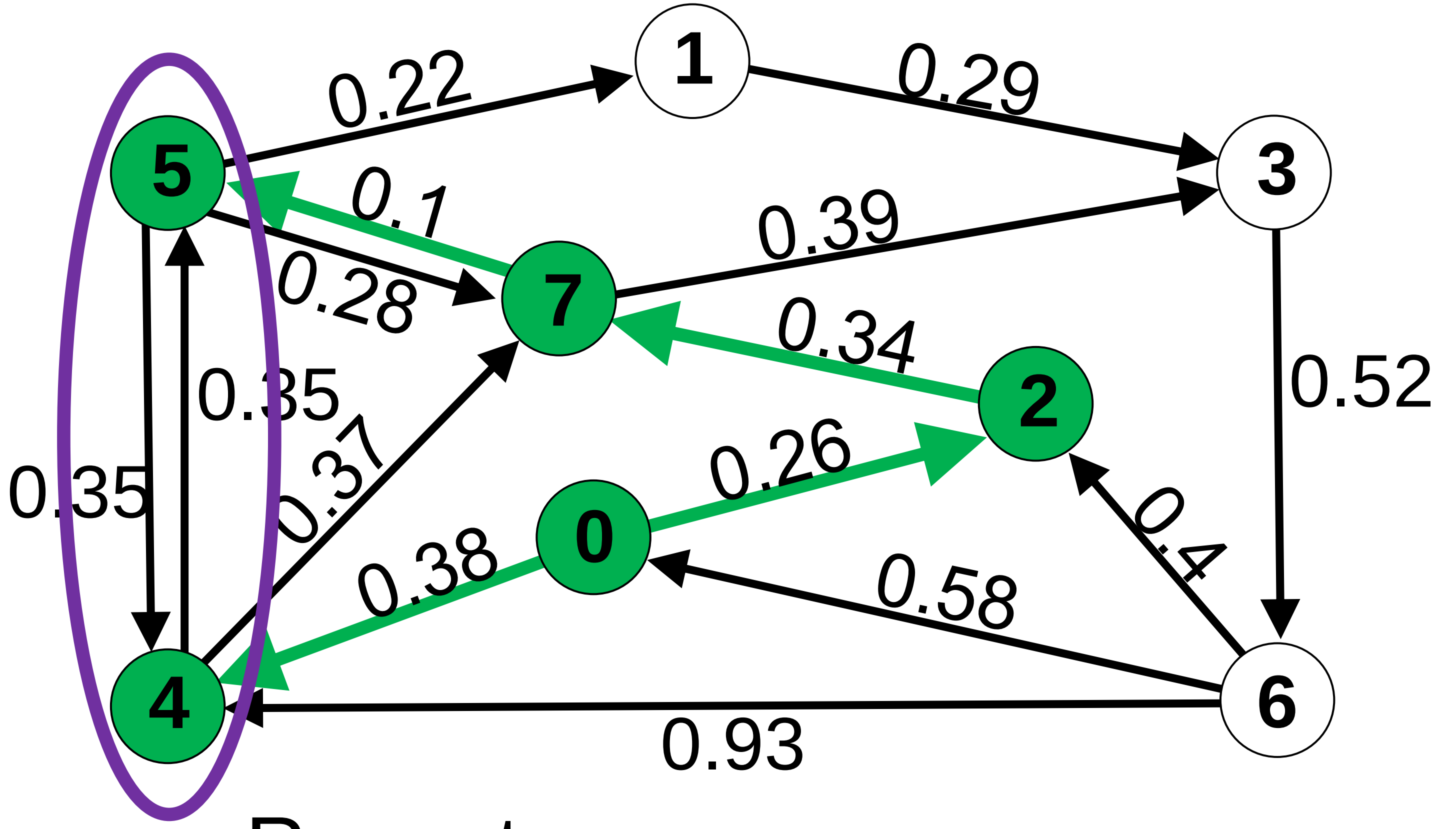
	Previous vertex
0	-
1	5
2	0
3	7
4	0
5	7
6	
7	2

Priority queue

1 (0.92)
3 (0.99)

vertex (distance)

queue top = 5 (0.70)



	Distance from 0
0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	∞
7	0.60

	Previous vertex
0	-
1	5
2	0
3	7
4	0
5	7
6	
7	2

Priority queue

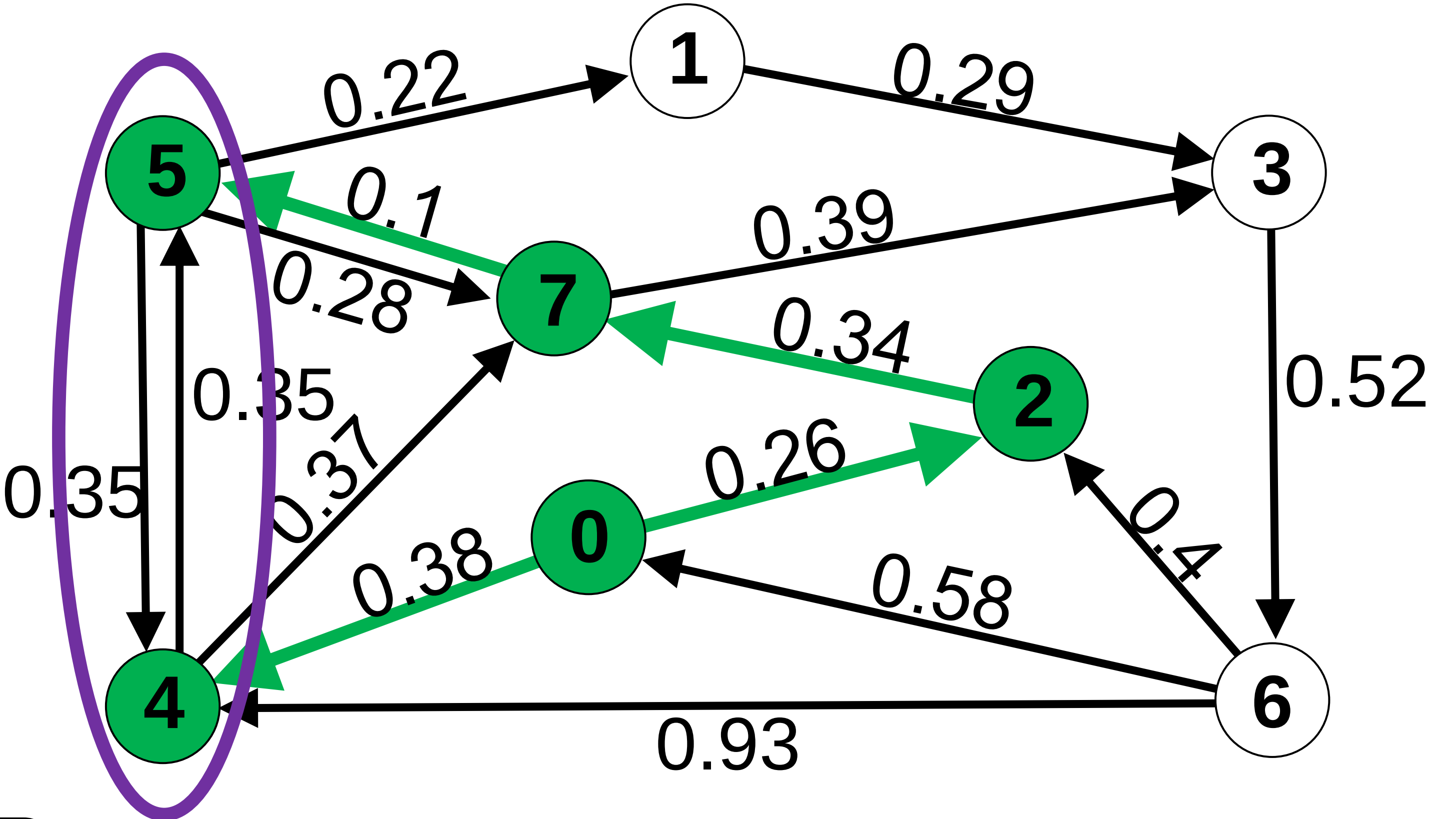
1 (0.92)
3 (0.99)

vertex (distance)

Repeat.

What about neighbor 4?

queue top = 5 (0.70)



Distance from 0

0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	∞
7	0.60

Previous vertex

0	-
1	5
2	0
3	7
4	0
5	7
6	
7	2

Priority queue

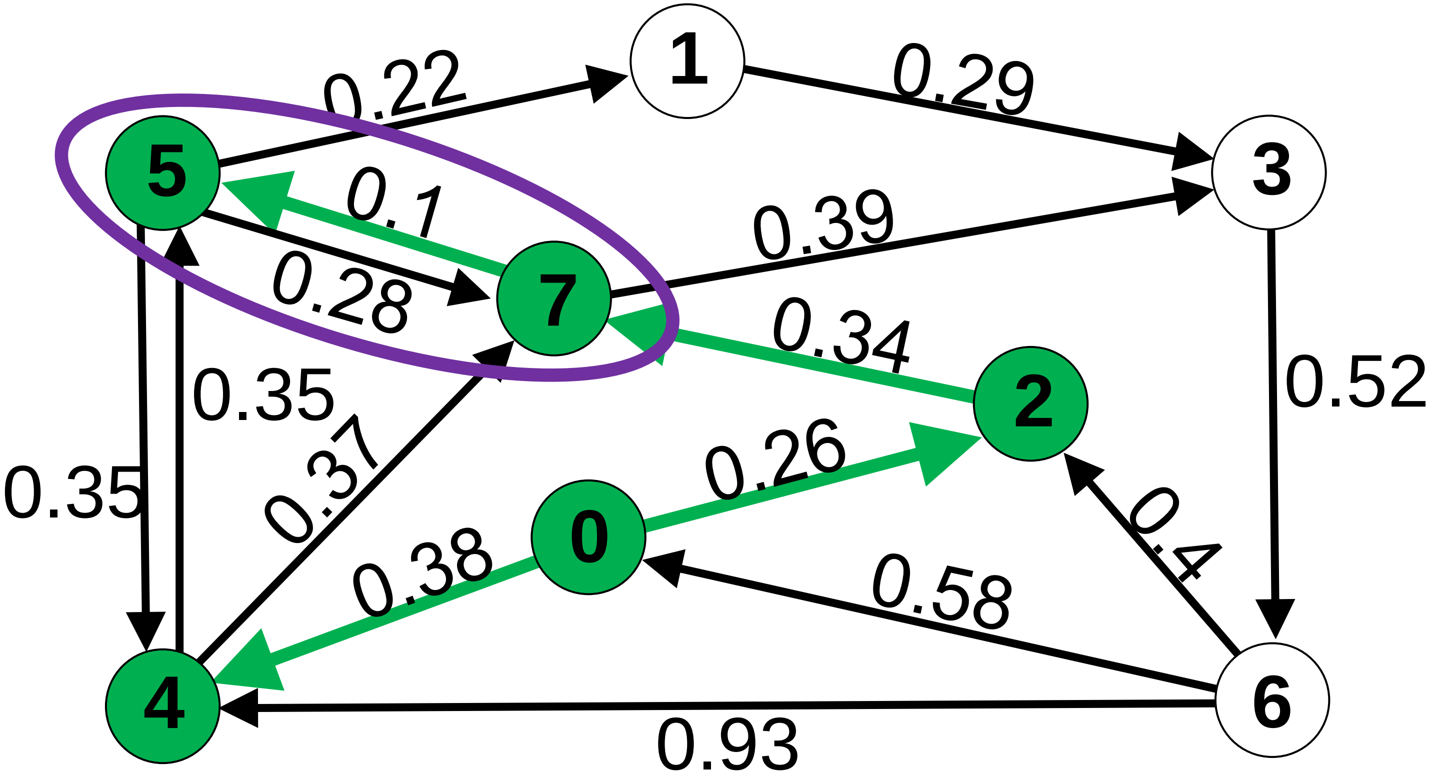
1 (0.92)
3 (0.99)

vertex (distance)

Repeat.

What about neighbor 4? distance[5] + weight(5, 4) = 0.70 + 0.35 = 1.05 $\not<$ 0.38 = distance[4]

queue top = 5 (0.70)



	Distance from 0
0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	∞
7	0.60

	Previous vertex
0	-
1	5
2	0
3	7
4	0
5	7
6	
7	2

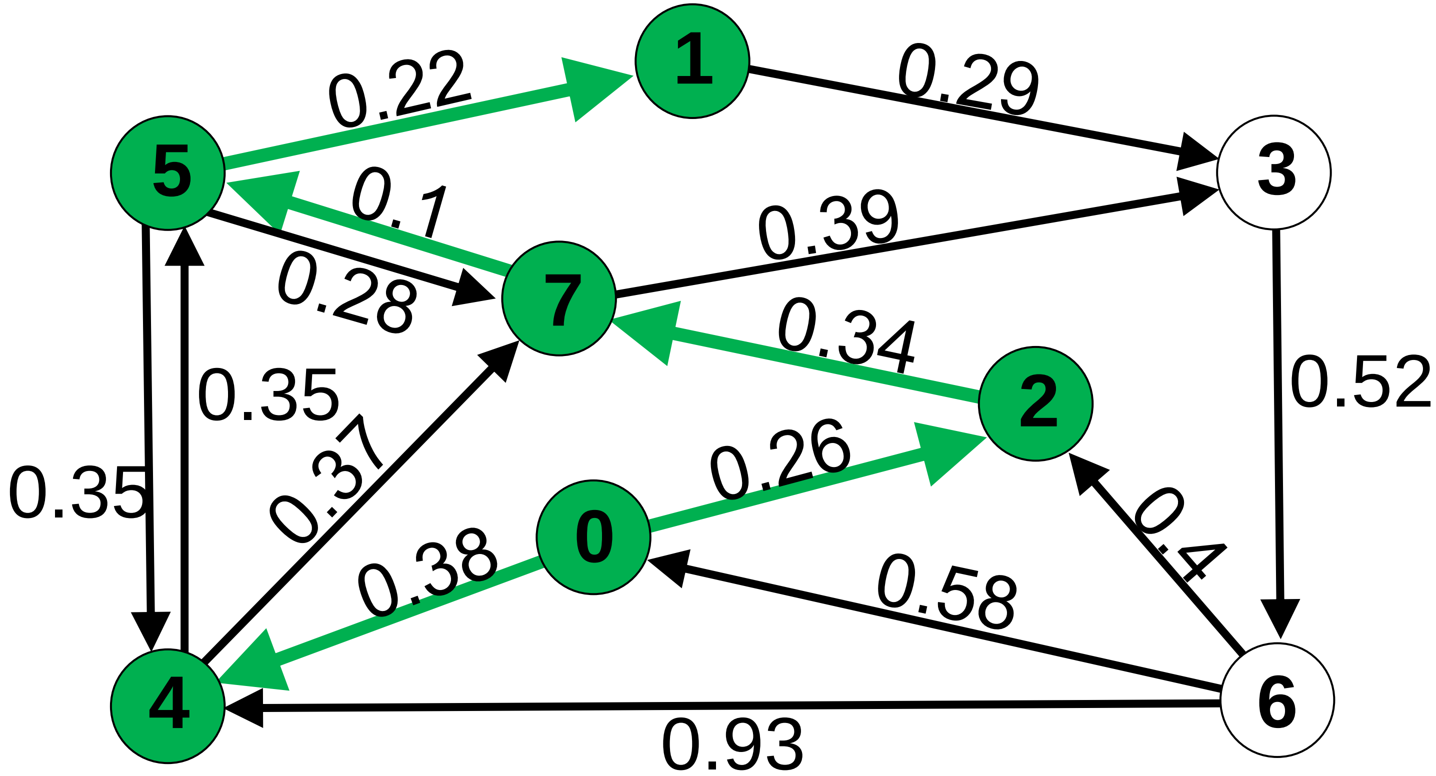
Priority queue
1 (0.92)
3 (0.99)

vertex (distance)

Repeat. **What about neighbor 7?**

distance[5] + weight(5, 7) = 0.70 + 0.28 = 0.98 $\not<$ 0.60 = distance[7]

queue top	= 1 (0.92)



Repeat.

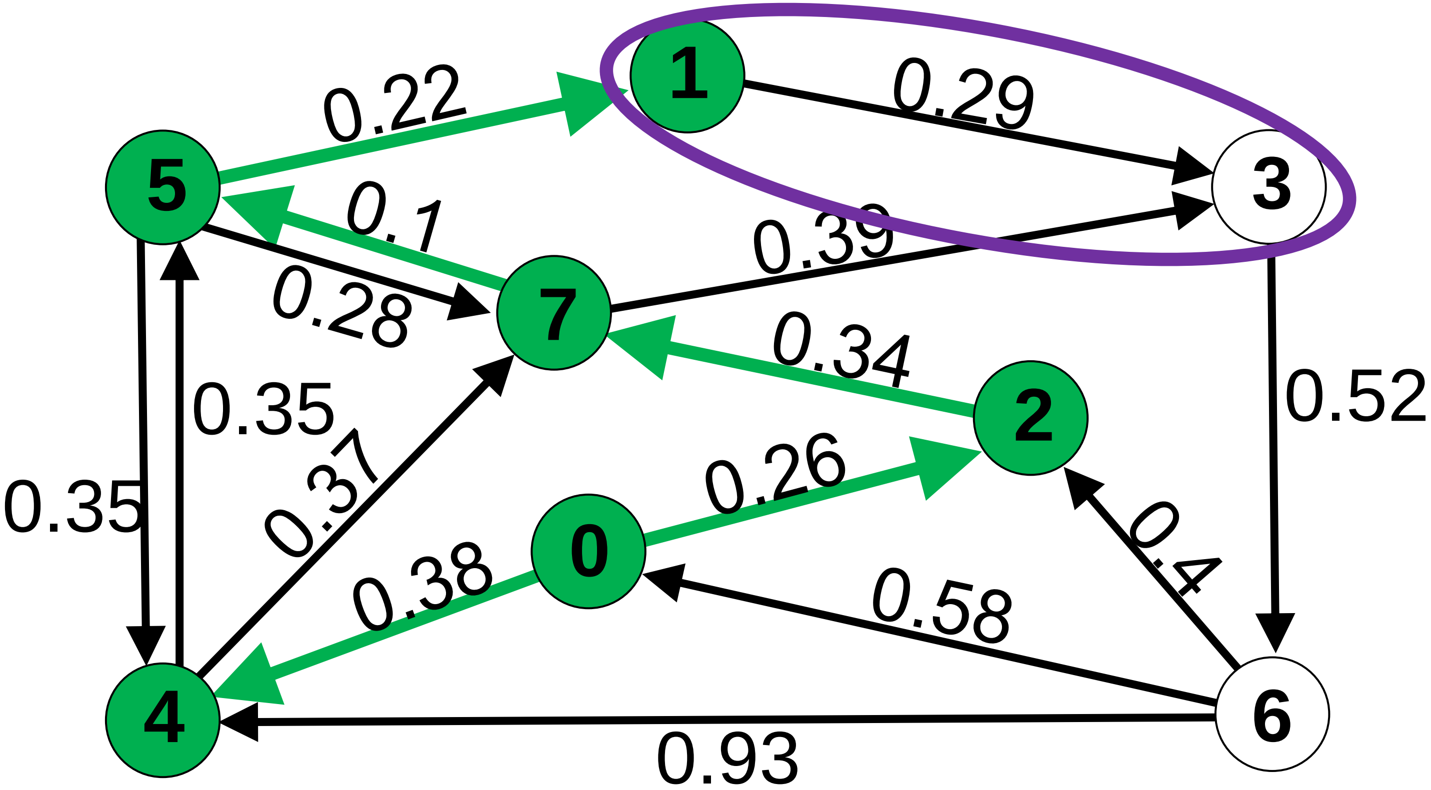
	Distance from 0
0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	∞
7	0.60

	Previous vertex
0	-
1	5
2	0
3	7
4	0
5	7
6	
7	2

Priority queue
3 (0.99)

vertex (distance)

queue top = 1 (0.92)



	Distance from 0	Previous vertex
0	0	-
1	0.92	5
2	0.26	0
3	0.99	7
4	0.38	0
5	0.70	7
6	∞	
7	0.60	2

Priority queue

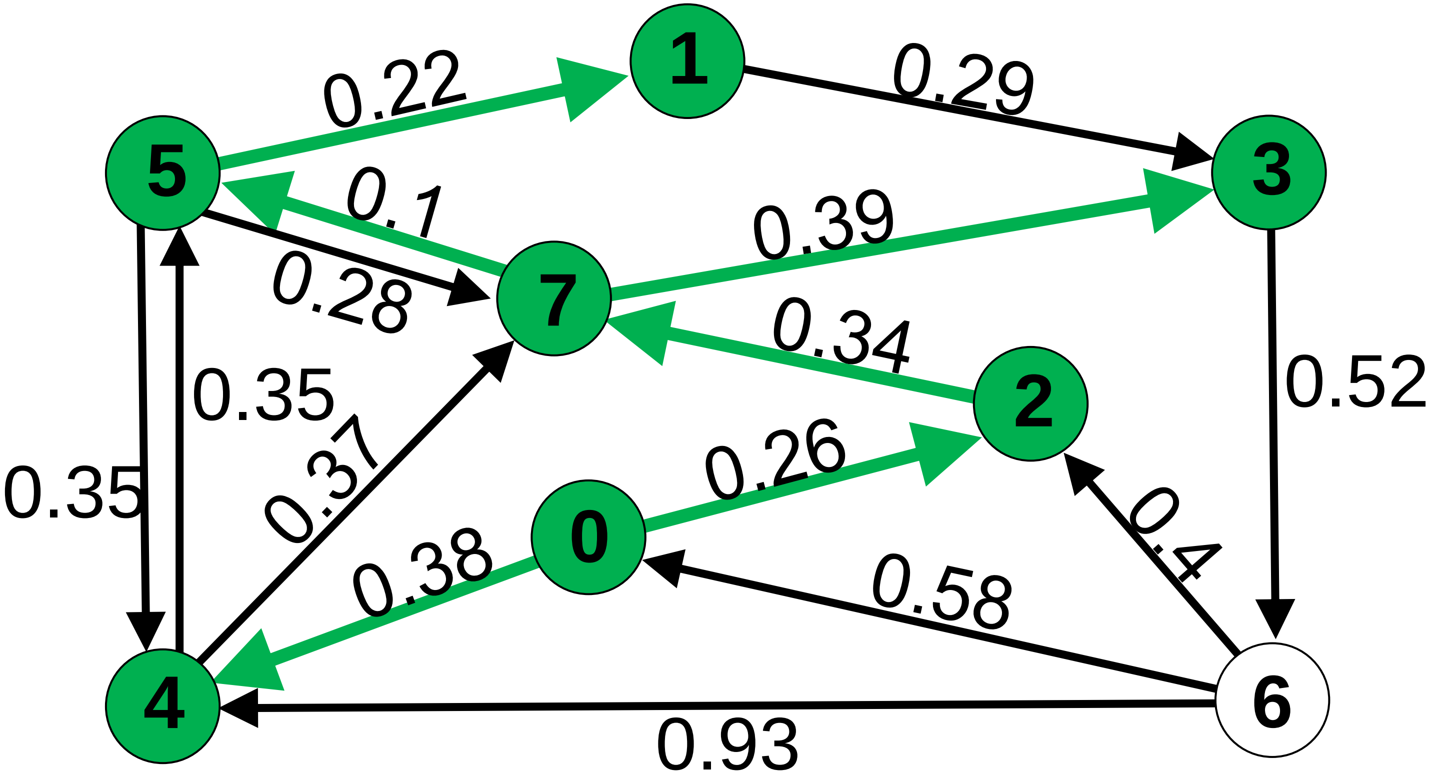
3 (0.99)

vertex (distance)

Repeat.

What about neighbor 3? 0.92 + 0.29 = 1.21 > 0.99

queue top = 3 (0.99)



Repeat.

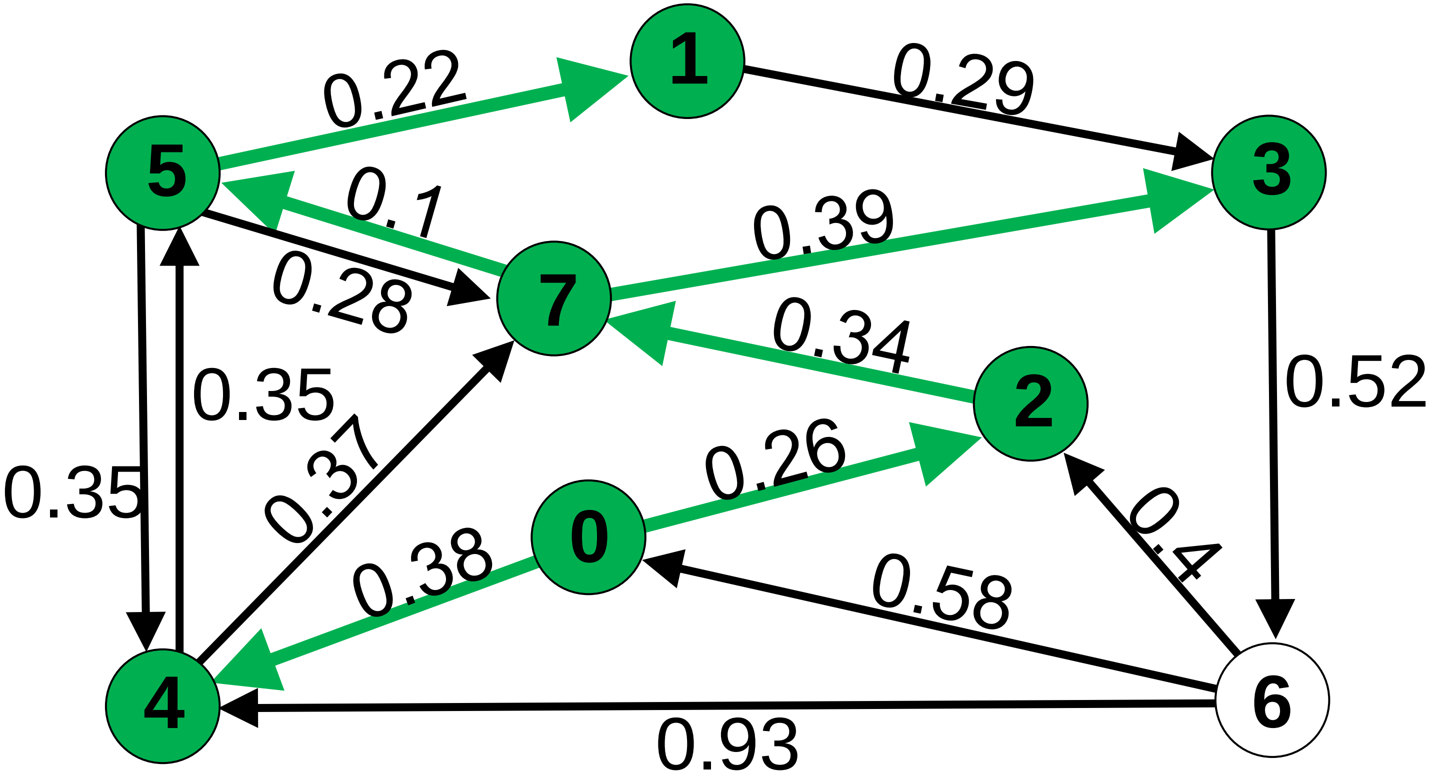
Distance from 0	
0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	∞
7	0.60

Previous vertex	
0	-
1	5
2	0
3	7
4	0
5	7
6	
7	2

Priority queue

vertex (distance)

queue top = 3 (0.99)



Repeat.

Distance from 0

0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	1.51
7	0.60

Previous vertex

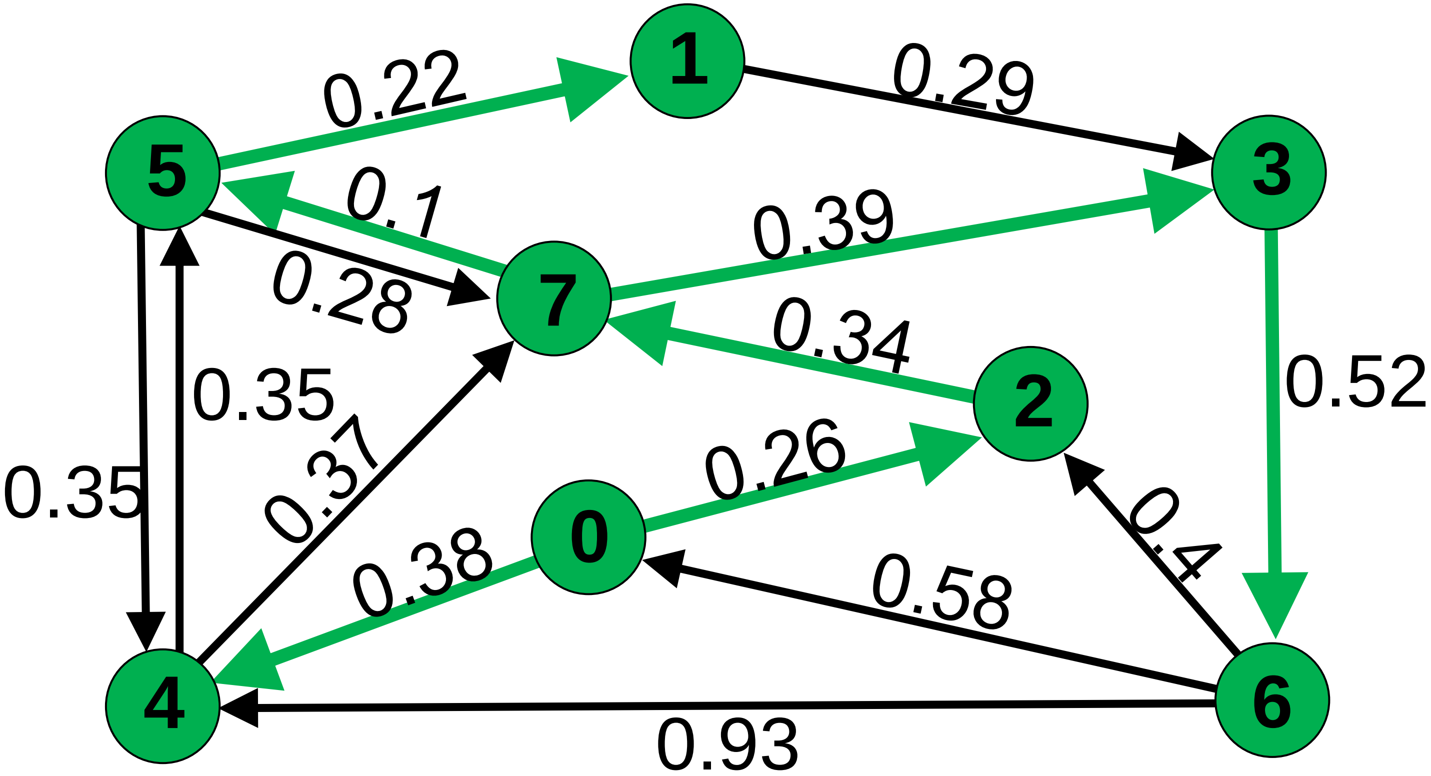
0	-
1	5
2	0
3	7
4	0
5	7
6	3
7	2

Priority queue

6 (1.51)

vertex (distance)

queue top	= 6 (1.51)



Repeat.

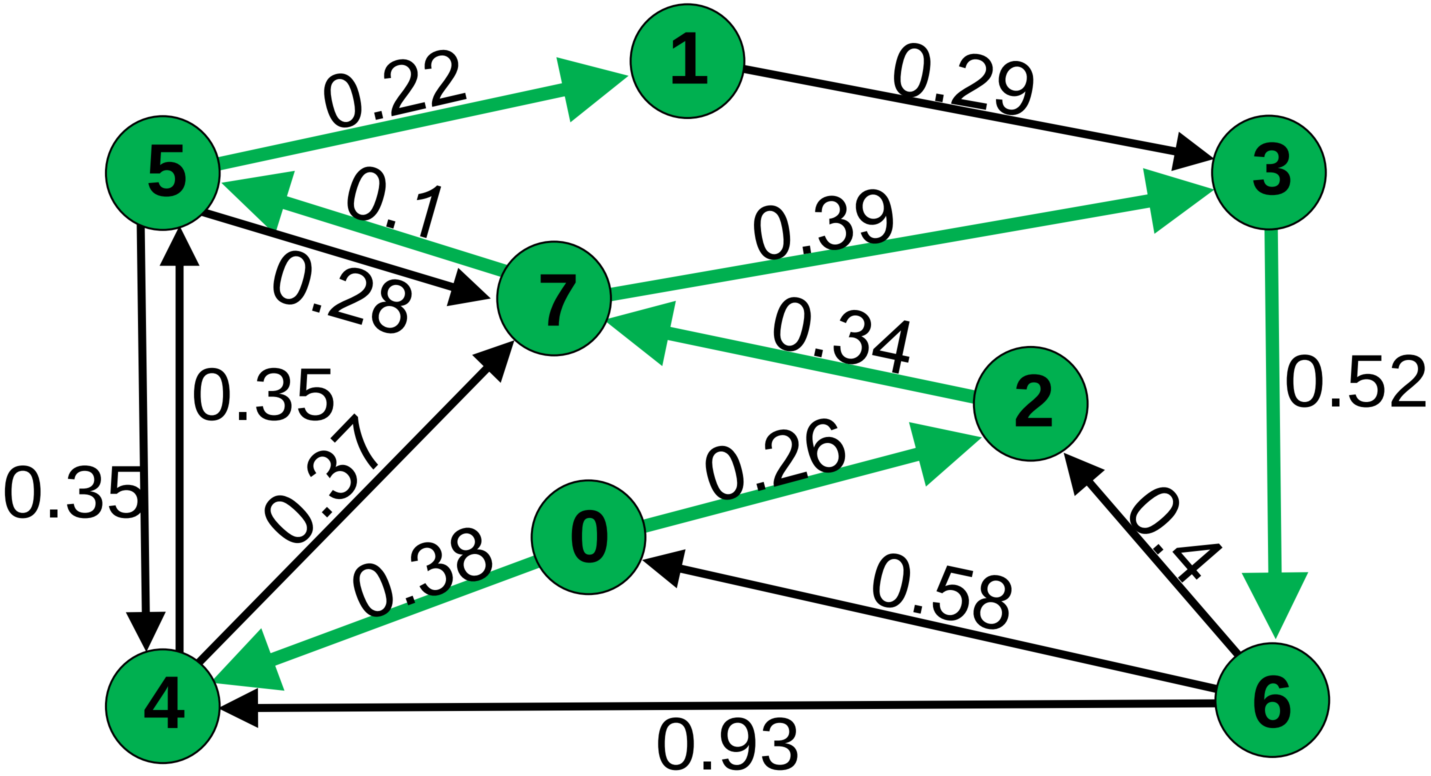
	Distance from 0
0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	1.51
7	0.60

	Previous vertex
0	-
1	5
2	0
3	7
4	0
5	7
6	3
7	2

Priority queue

vertex (distance)

queue top = 6 (1.51)



Repeat?

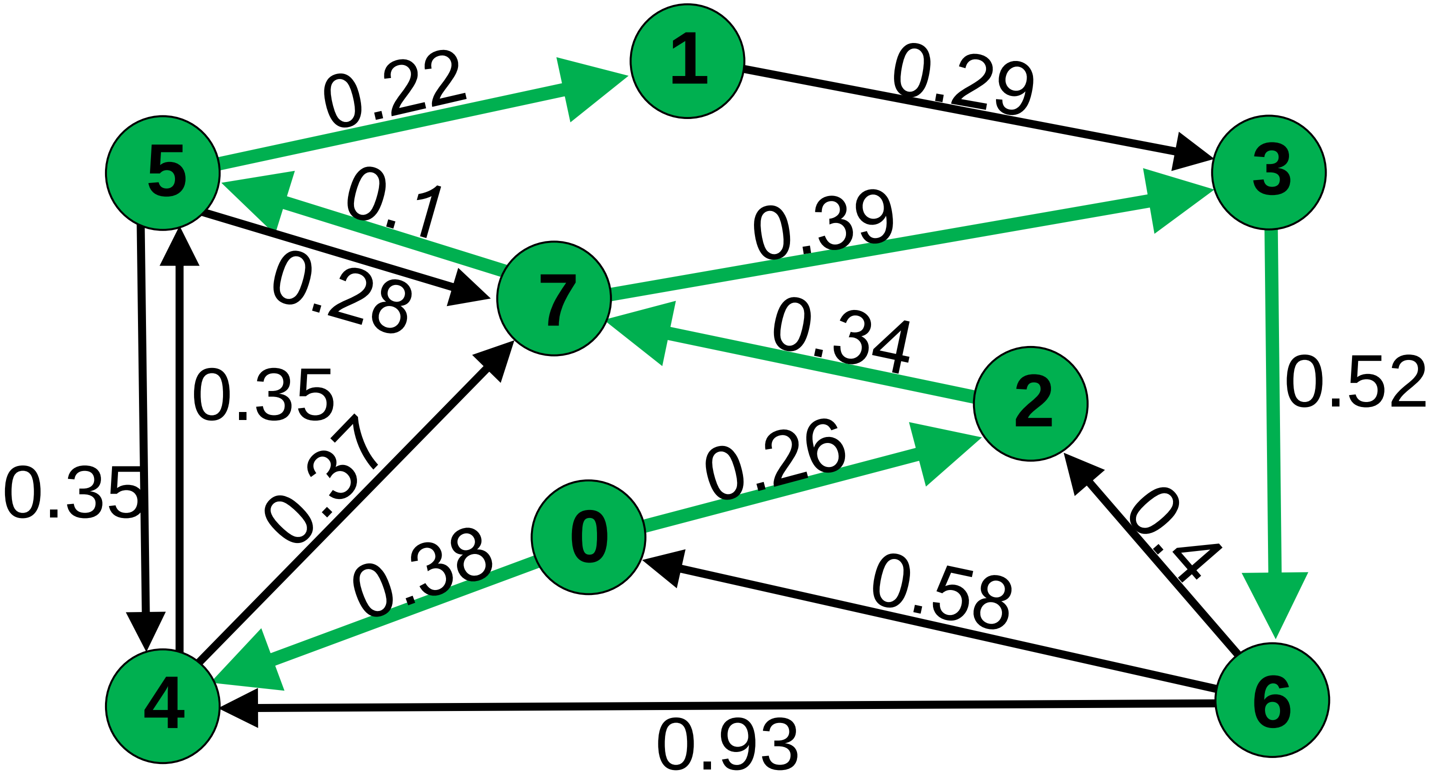
	Distance from 0
0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	1.51
7	0.60

	Previous vertex
0	-
1	5
2	0
3	7
4	0
5	7
6	3
7	2

Priority queue

vertex (distance)

`queue top`	= 6 (1.51)



	Distance from 0
0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	1.51
7	0.60

	Previous vertex
0	-
1	5
2	0
3	7
4	0
5	7
6	3
7	2

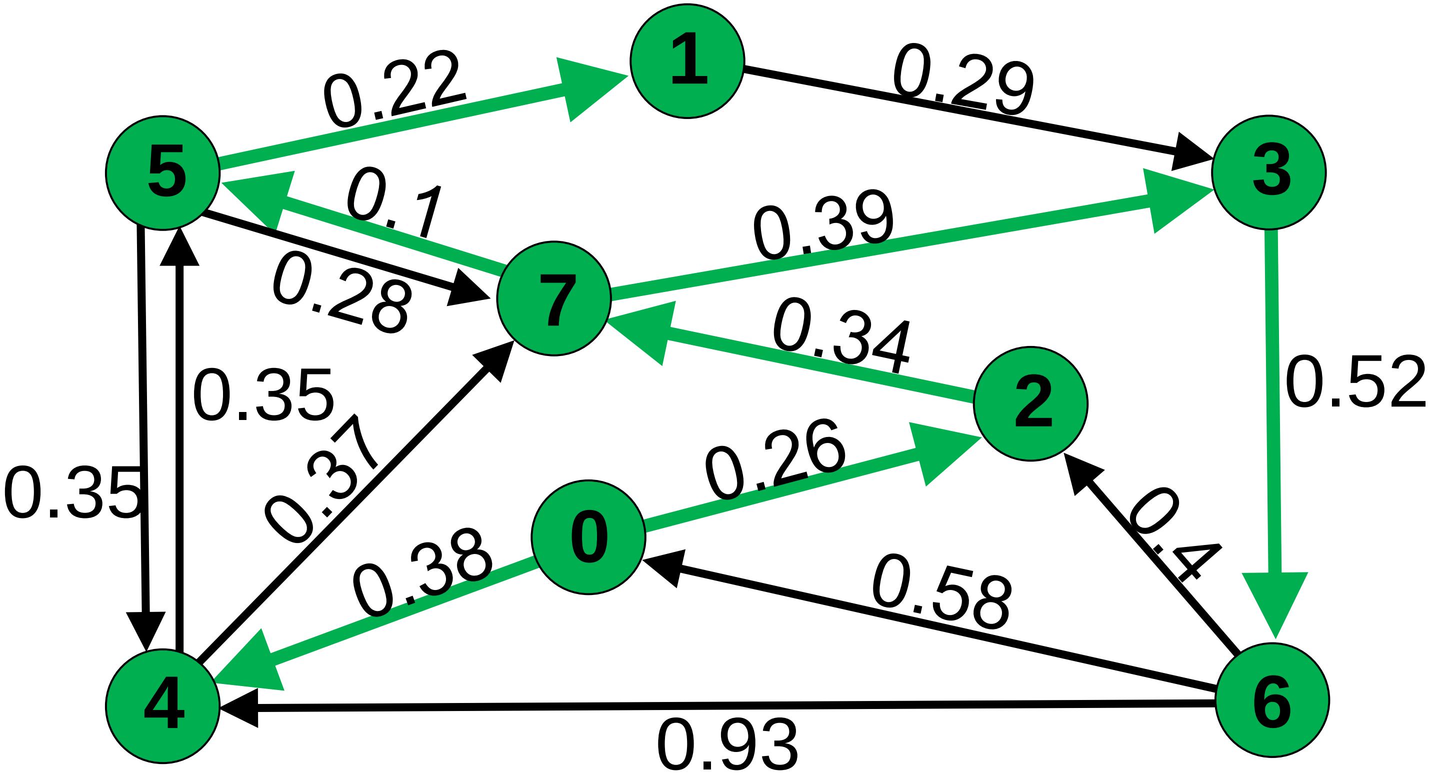
Priority queue

vertex (distance)

Repeat?

Neighbor 4? 1.51 + 0.93 > 0.38

queue top	= 6 (1.51)



Repeat?

	Distance from 0
0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	1.51
7	0.60

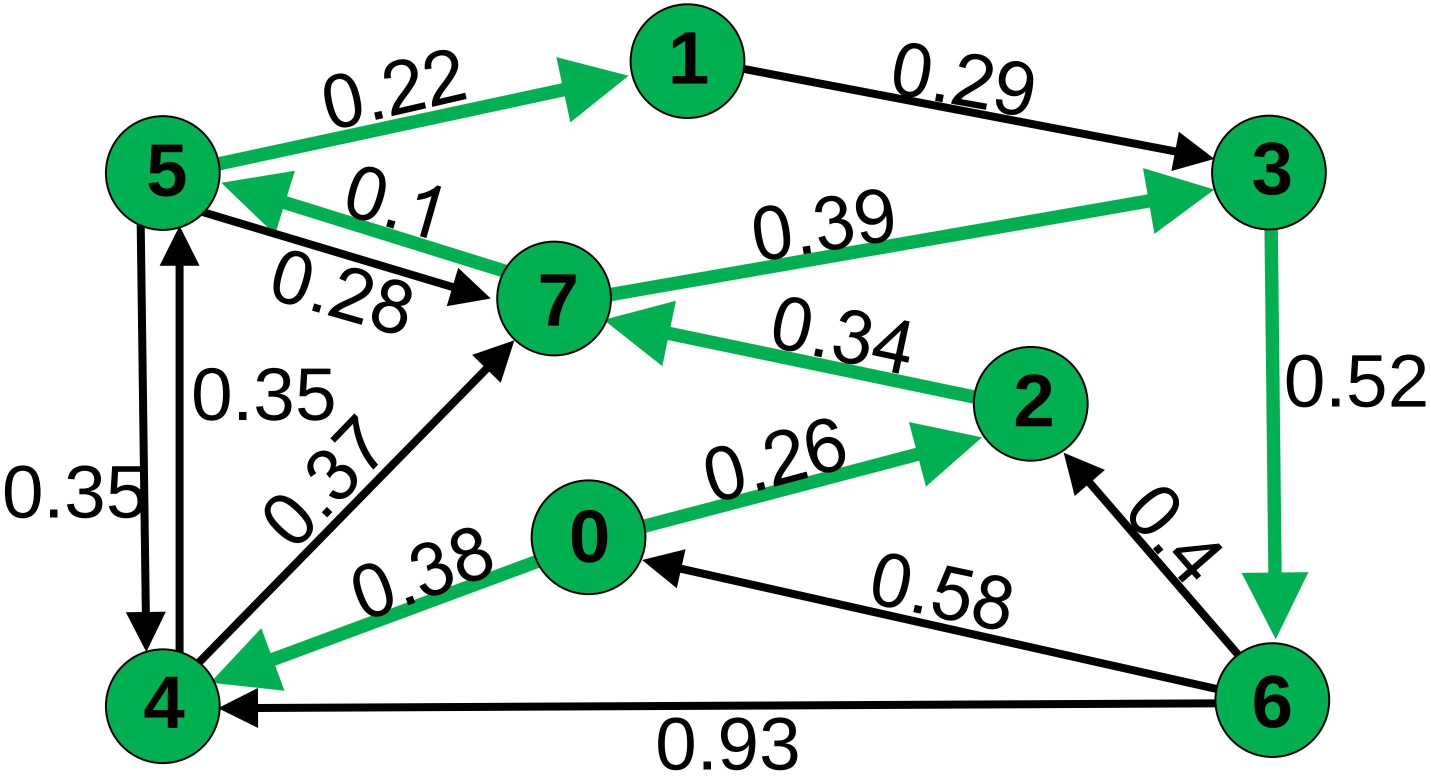
	Previous vertex
0	-
1	5
2	0
3	7
4	0
5	7
6	3
7	2

Priority queue

vertex (distance)

Neighbor 0? 1.51 + 0.58 > 0

queue top	= 6 (1.51)



	Distance from 0
0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	1.51
7	0.60

	Previous vertex
0	-
1	5
2	0
3	7
4	0
5	7
6	3
7	2

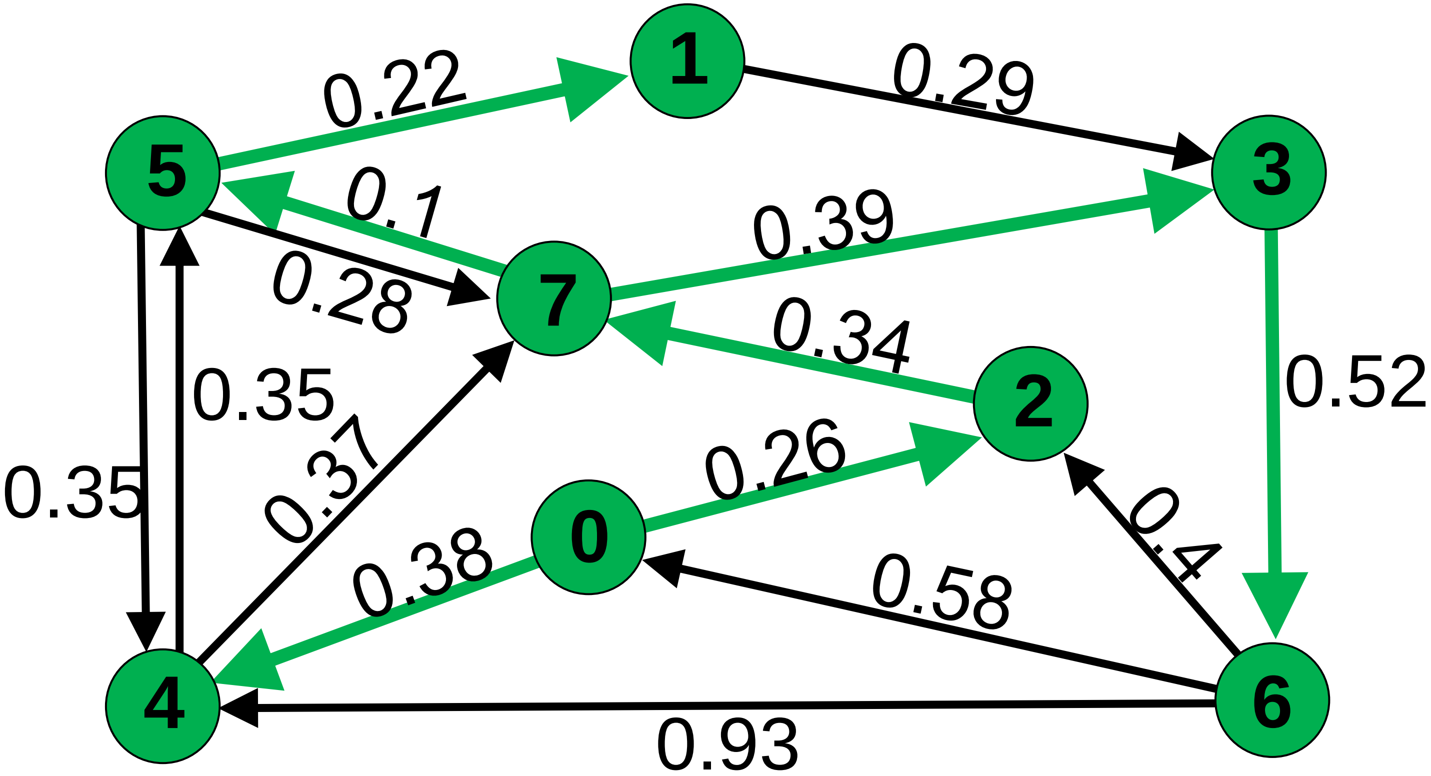
Priority queue

vertex (distance)

Repeat?

Neighbor 2? 1.51 + 0.4 > 0.26

queue top	= 6 (1.51)



When are we done?

Distance from 0

0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	1.51
7	0.60

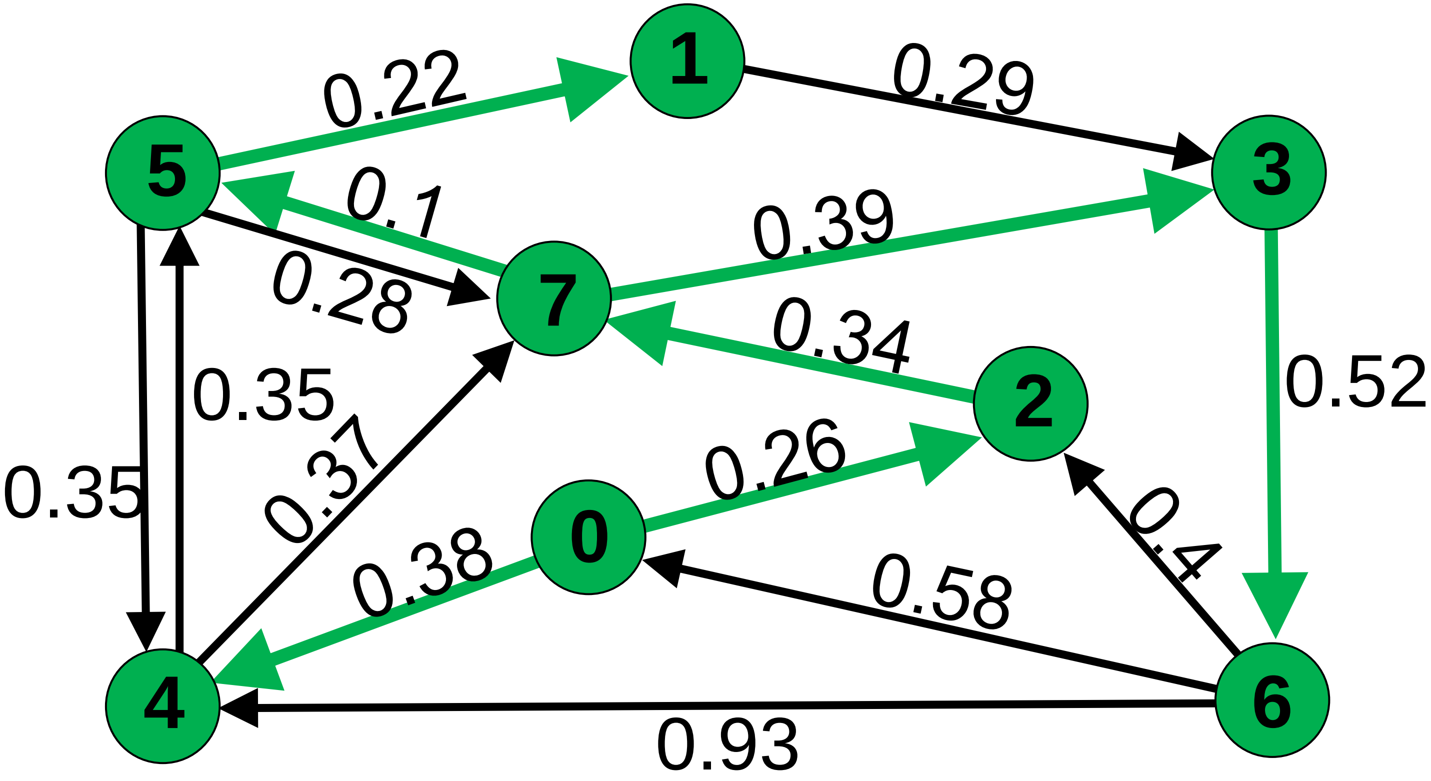
Previous vertex

0	-
1	5
2	0
3	7
4	0
5	7
6	3
7	2

Priority queue

vertex (distance)

queue top = 6 (1.51)



Distance from 0	
0	0
1	0.92
2	0.26
3	0.99
4	0.38
5	0.70
6	1.51
7	0.60

Previous vertex	
0	-
1	5
2	0
3	7
4	0
5	7
6	3
7	2

Priority queue

vertex (distance)

When are we done?

When the queue is empty!

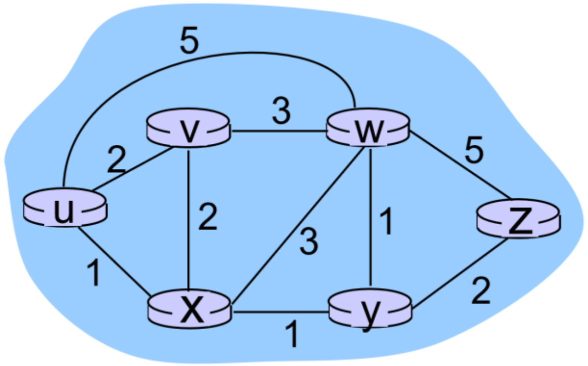
Routing tables are filled via **routing algorithms**

There are two types of routing algorithms

Centralized/Global- we know the edge costs of the network

Link State algorithms

(Dijkstra's Algorithm)

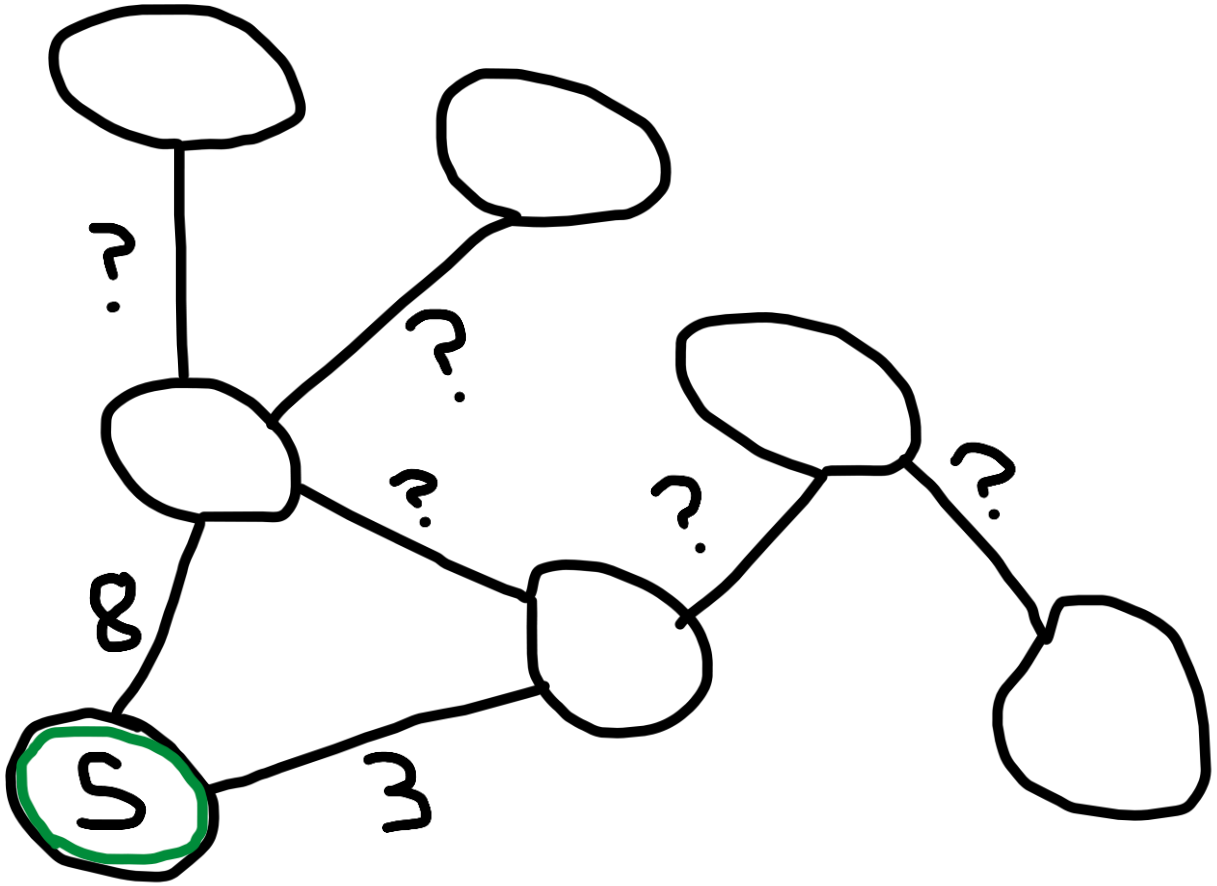


We can compute the shortest path from one node, to all other nodes in polynomial time.

Once we know the shortest path from A to B, we can update routing tables to reflect that shortest path

Routing tables are filled via **routing algorithms**

There are two types of routing algorithms



Decentralized- we do not know the edge costs of the entire network.
Only know edge costs to neighbors

Distance Vector algorithms

Routing tables are filled via **routing algorithms**

There are two types of routing algorithms

Based on *Bellman-Ford* (BF) equation (dynamic programming):

Bellman-Ford equation

Let $D_x(y)$: cost of least-cost path from x to y.

Then:

$$D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$$

v's estimated least-cost-path cost to y

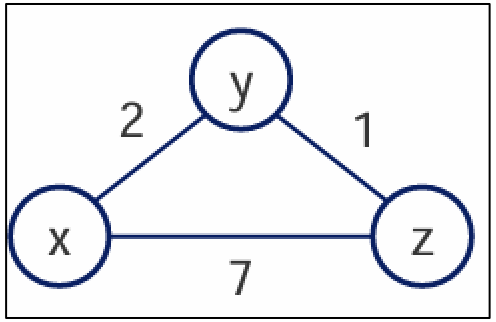
min taken over all neighbors v of x

direct cost of link from x to v

X		Cost to		
		X	Y	Z
Cost From	X	0	2	7
	Y			
	Z			

Y		Cost to		
		X	Y	Z
Cost From	X			
	Y	2	0	1
	Z			

Z		Cost to		
		X	Y	Z
Cost From	X			
	Y			
	Z	7	1	0

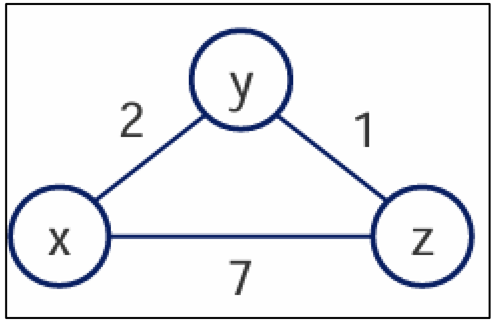


X		Cost to		
		X	Y	Z
Cost From	X	0	2	7
	Y			
	Z			

Y		Cost to		
		X	Y	Z
Cost From	X			
	Y	2	0	1
	Z			

Z		Cost to		
		X	Y	Z
Cost From	X			
	Y			
	Z	7	1	0

X		Cost to		
		X	Y	Z
Cost From	X	0	2	**3**
	Y	2	0	1
	Z	7	1	0



X learns that Y can reach Z with only a cost of 1

Meaning:

The cost to get to Y from X (2) plus the cost to get from Y to Z (1) is lower than my direct path to Y (7), so I have now learned a new shortest path!

X		Cost to		
		X	Y	Z
Cost	X	0	2	7
From	Y			
	Z			

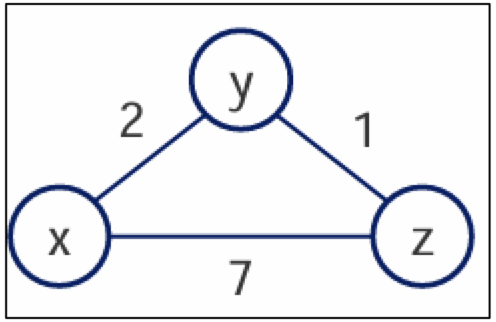
X		Cost to		
		X	Y	Z
Cost	X	0	2	3
From	Y	2	0	1
	Z	7	1	0

Y		Cost to		
		X	Y	Z
Cost	X			
From	Y	2	0	1
	Z			

Y		Cost to		
		X	Y	Z
Cost	X	0	2	7
From	Y	2	0	1
	Z	7	1	0

Z		Cost to		
		X	Y	Z
Cost	X			
From	Y			
	Z	7	1	0

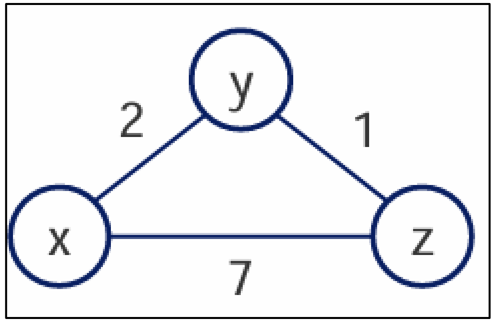
Z		Cost to		
		X	Y	Z
Cost	X	0	2	7
From	Y	2	0	1
	Z	3	1	0



X		Cost to		
		X	Y	Z
Cost From	X	0	2	7
	Y			
	Z			

X		Cost to		
		X	Y	Z
Cost From	X	0	2	3
	Y	2	0	1
	Z	7	1	0

X		Cost to		
		X	Y	Z
Cost From	X	0	2	3
	Y	2	0	1
	Z	3	1	0



Y		Cost to		
		X	Y	Z
Cost From	X			
	Y	2	0	1
	Z			

Y		Cost to		
		X	Y	Z
Cost From	X	0	2	7
	Y	2	0	1
	Z	7	1	0

Y		Cost to		
		X	Y	Z
Cost From	X	0	2	3
	Y	2	0	1
	Z	3	1	0

Z		Cost to		
		X	Y	Z
Cost From	X			
	Y			
	Z	7	1	0

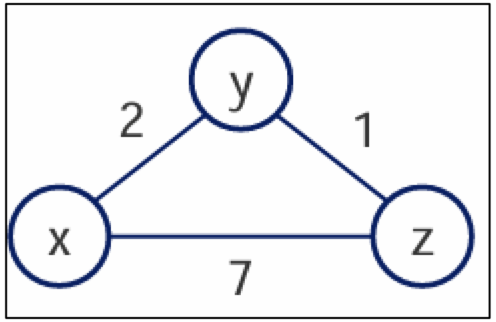
Z		Cost to		
		X	Y	Z
Cost From	X	0	2	7
	Y	2	0	1
	Z	3	1	0

Z		Cost to		
		X	Y	Z
Cost From	X	0	2	3
	Y	2	0	1
	Z	3	1	0

X		Cost to		
		X	Y	Z
Cost From	X	0	2	7
	Y			
	Z			

X		Cost to		
		X	Y	Z
Cost From	X	0	2	3
	Y	2	0	1
	Z	7	1	0

X		Cost to		
		X	Y	Z
Cost From	X	0	2	3
	Y	2	0	1
	Z	3	1	0



Y		Cost to		
		X	Y	Z
Cost From	X			
	Y	2	0	1
	Z			

Y		Cost to		
		X	Y	Z
Cost From	X	0	2	7
	Y	2	0	1
	Z	7	1	0

Y		Cost to		
		X	Y	Z
Cost From	X	0	2	3
	Y	2	0	1
	Z	3	1	0

We can update our routing table and then make sure we forward packets to the correct destination in the path

Z		Cost to		
		X	Y	Z
Cost From	X			
	Y			
	Z	7	1	0

Z		Cost to		
		X	Y	Z
Cost From	X	0	2	7
	Y	2	0	1
	Z	3	1	0

Z		Cost to		
		X	Y	Z
Cost From	X	0	2	3
	Y	2	0	1
	Z	3	1	0

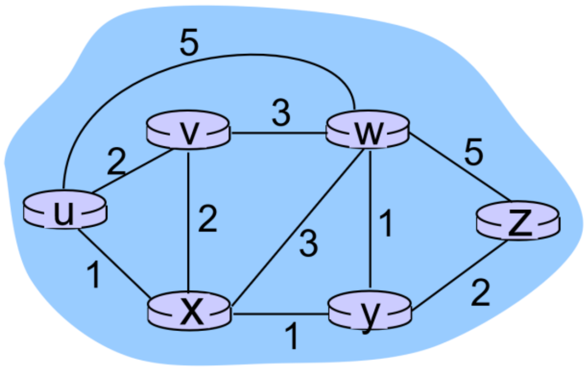
Routing tables are filled via **routing algorithms**

There are two types of routing algorithms

Centralized/Global- we know the edge costs of the network

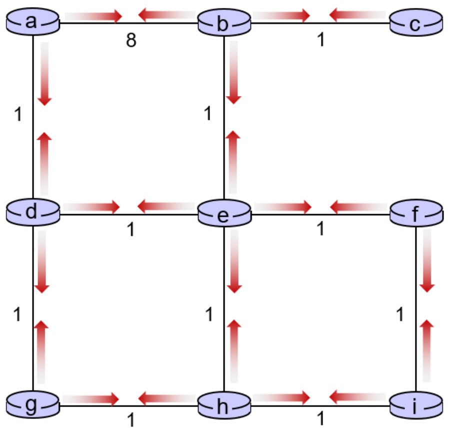
Link State algorithms

(Dijkstra's Algorithm)



Decentralized- we do not know the edge costs of the entire network.
Only know edge costs to neighbors

Distance Vector algorithms



Distance vector algorithm:

each node:

wait for (change in local link cost or msg from neighbor)

recompute DV estimates using DV received from neighbor

if DV to any destination has changed, *notify* neighbors

iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed, self-stopping: each node notifies neighbors *only* when its DV changes

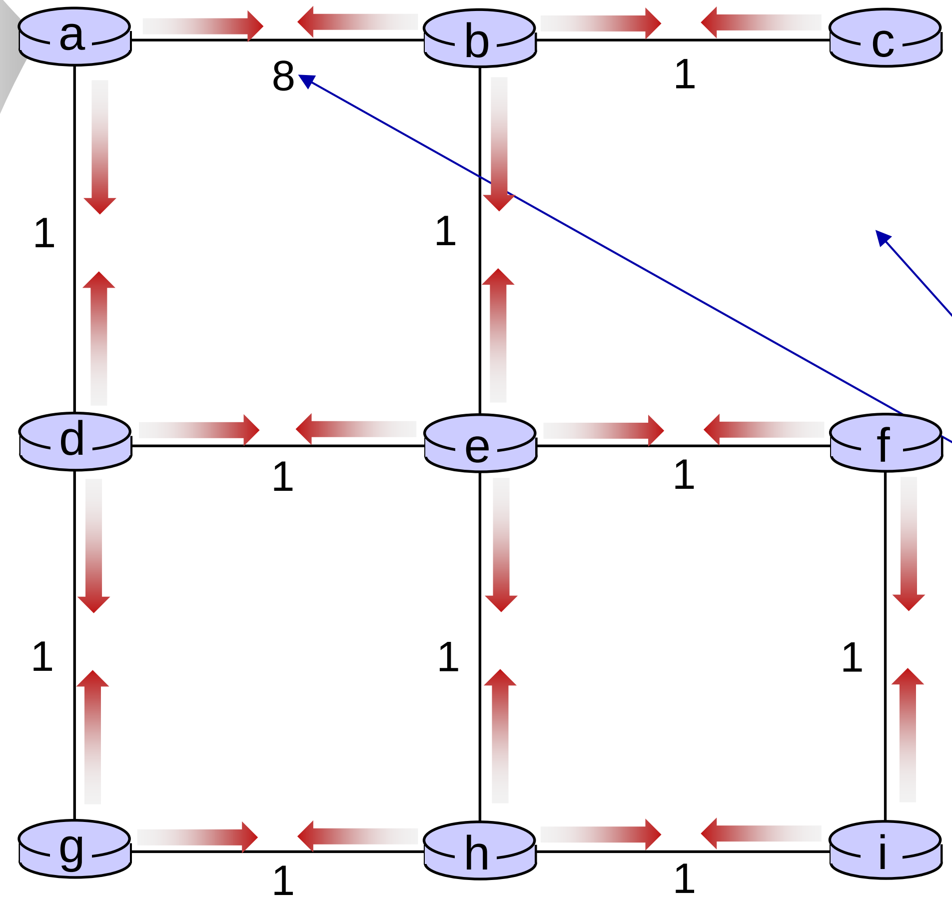
- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

Distance vector: example

t=0

- All nodes have distance estimates to nearest neighbors (only)
- All nodes send their local distance vector to their neighbors

DV in a:
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

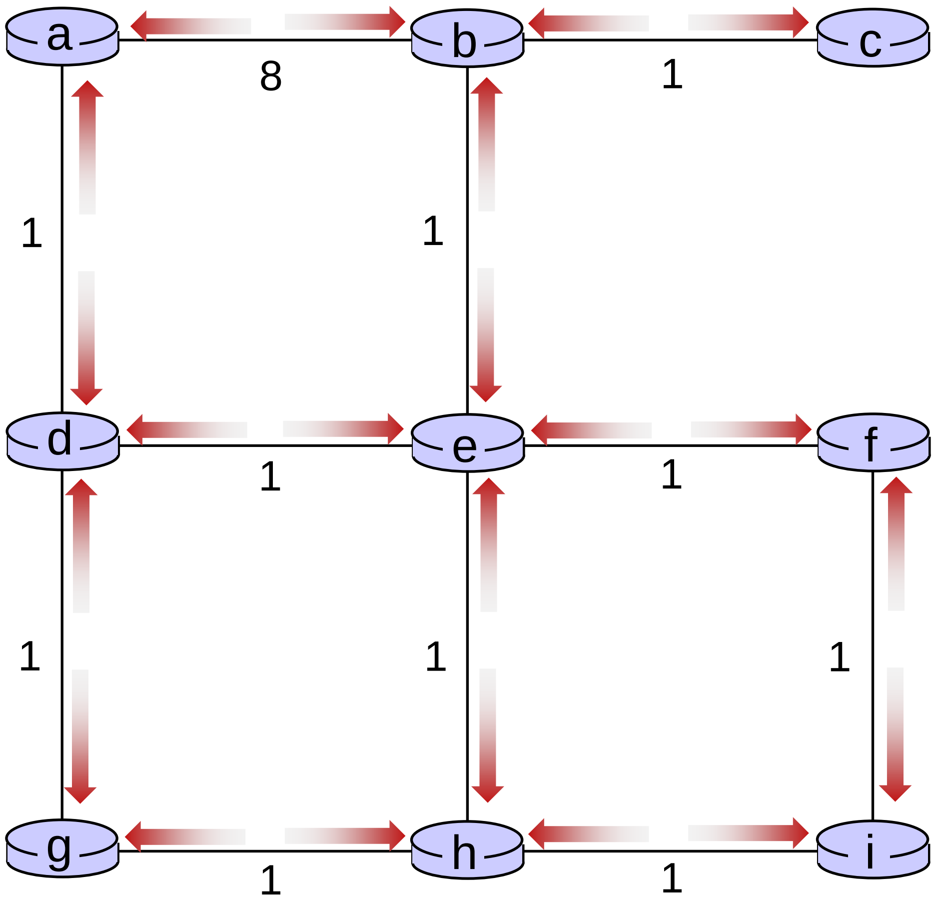


A few asymmetries:
- missing link
- larger cost

Distance vector example: iteration

t=1

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

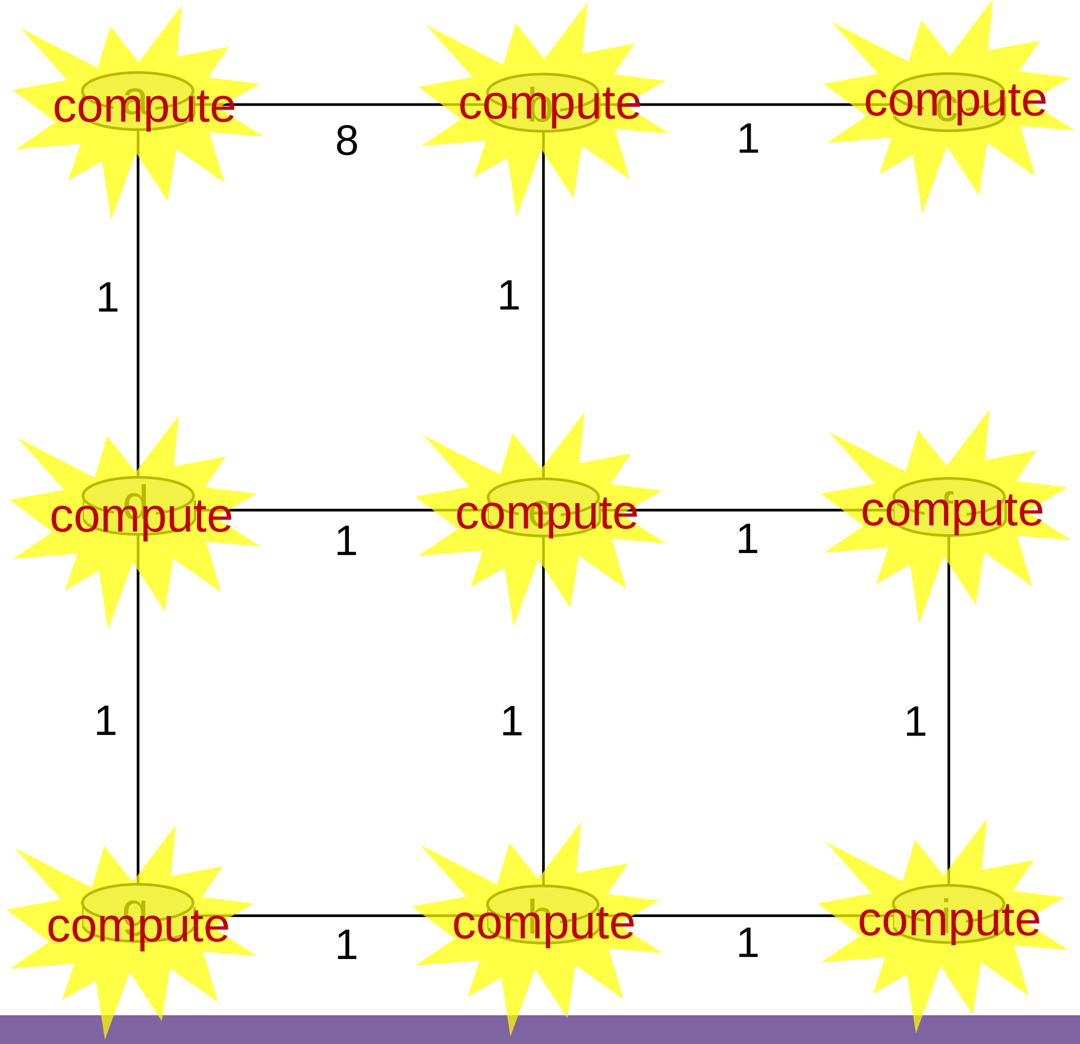


Distance vector example: iteration

t=1

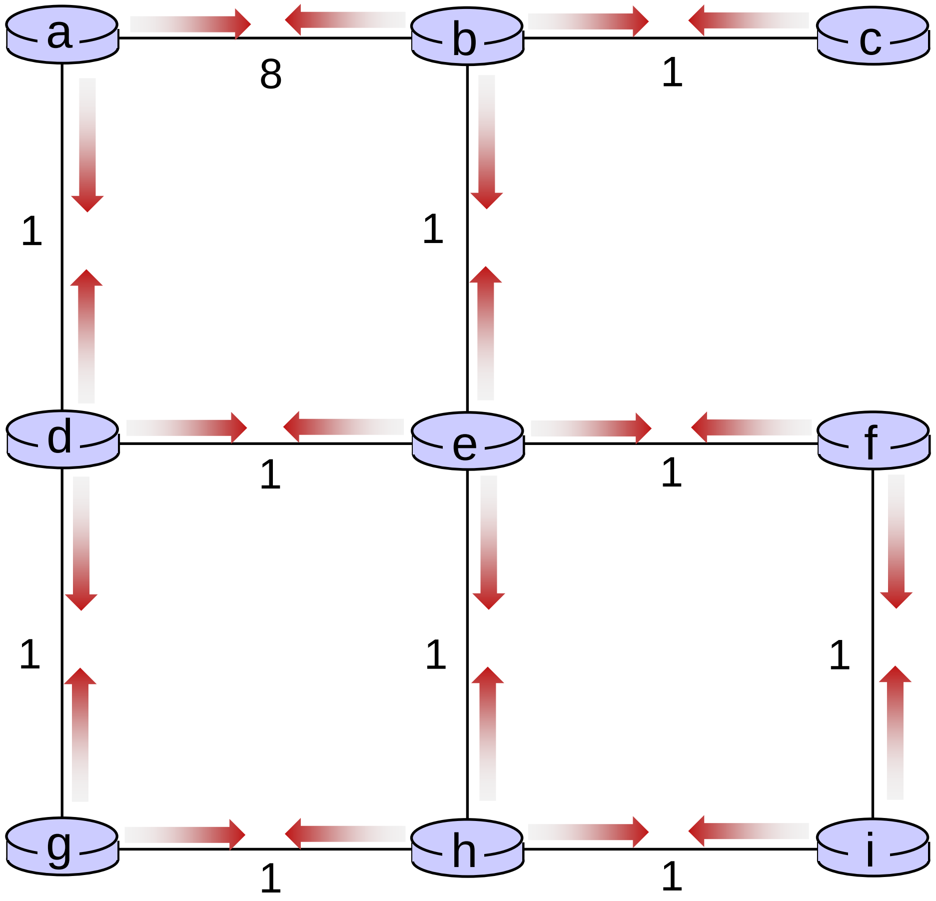
All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

t=1



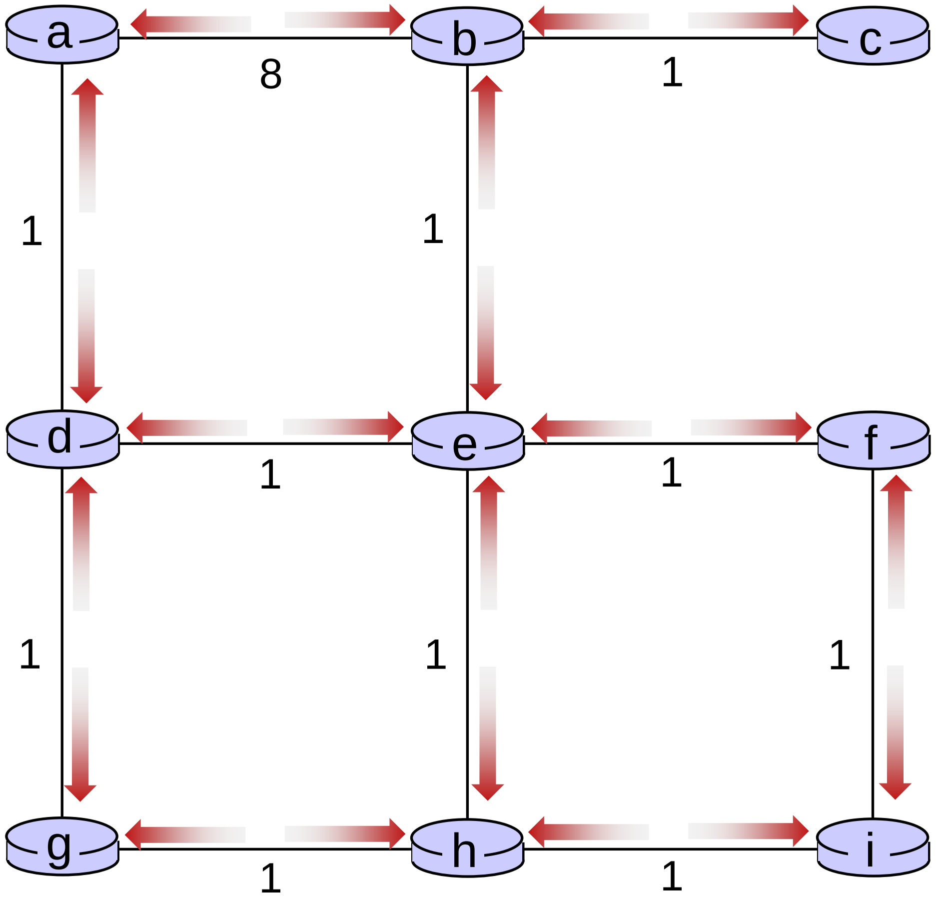
All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

Distance vector example: iteration

t=2

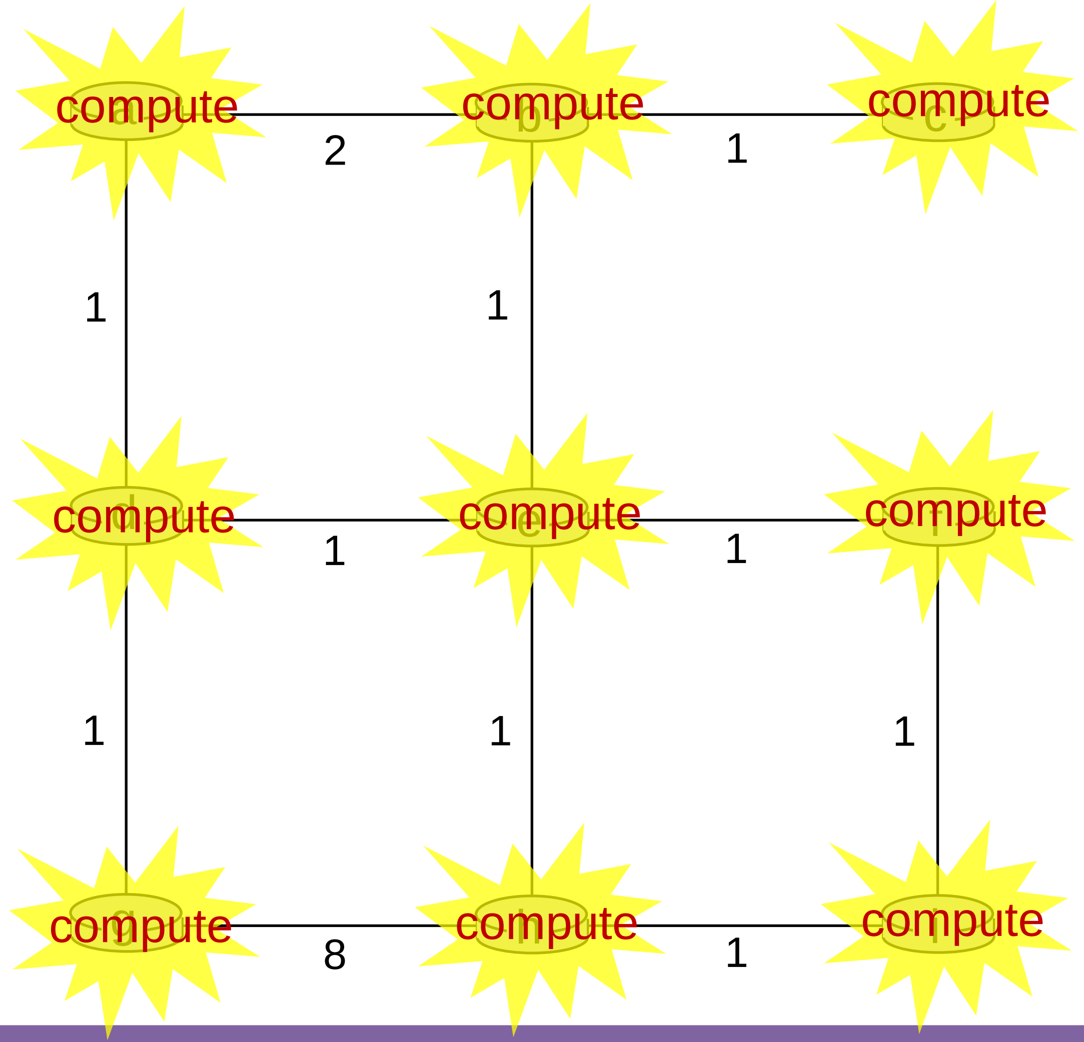
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



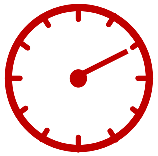
Distance vector example: iteration

t=2

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



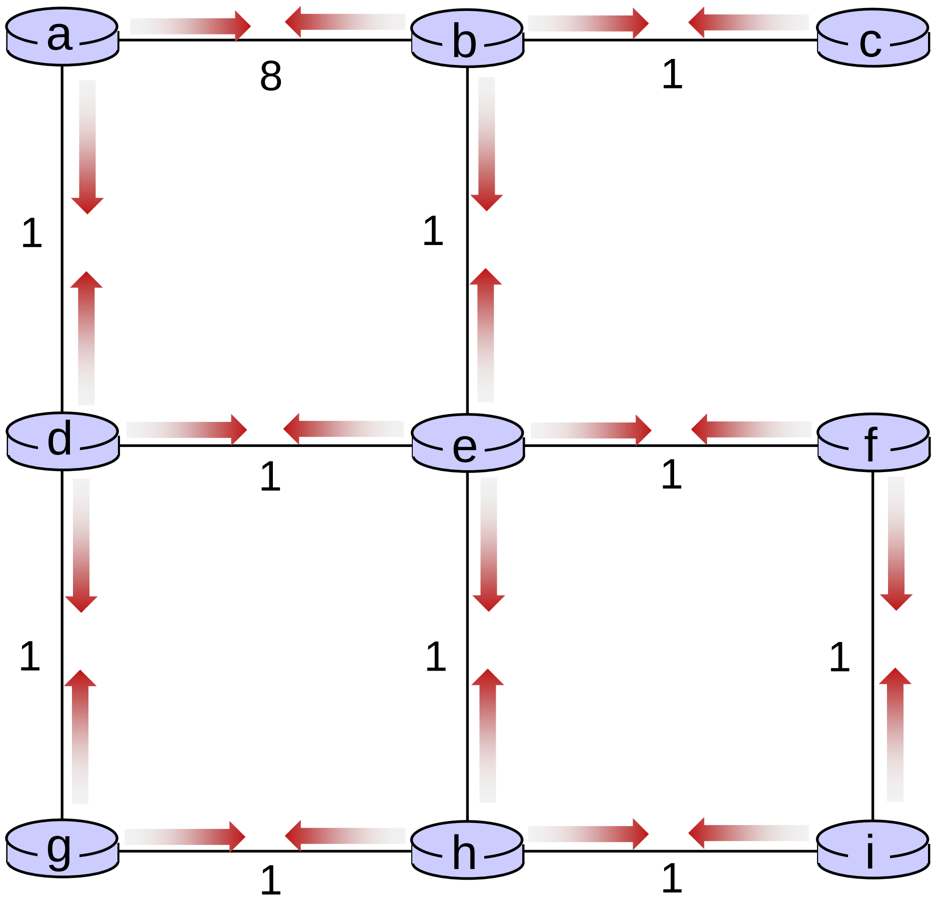
Distance vector example: iteration



t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



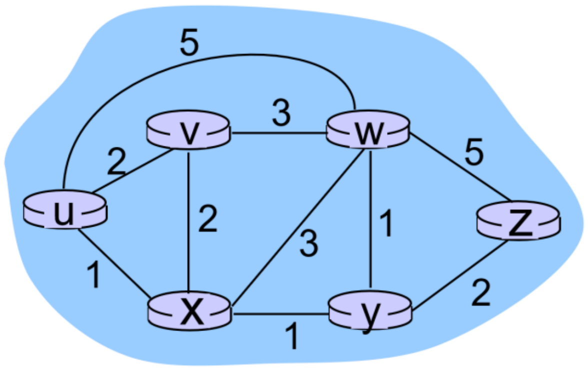
Routing tables are filled via **routing algorithms**

There are two types of routing algorithms

Centralized/Global- we know the edge costs of the network

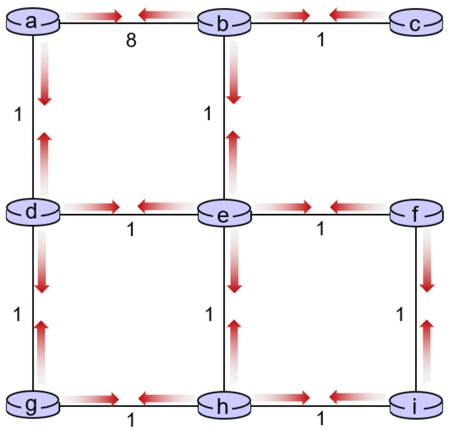
Link State algorithms

(Dijkstra's Algorithm)



Decentralized- we do not know the edge costs of the entire network.
Only know edge costs to neighbors

Distance Vector algorithms



These are not network protocols, these are simply general routing/shortest path algorithms

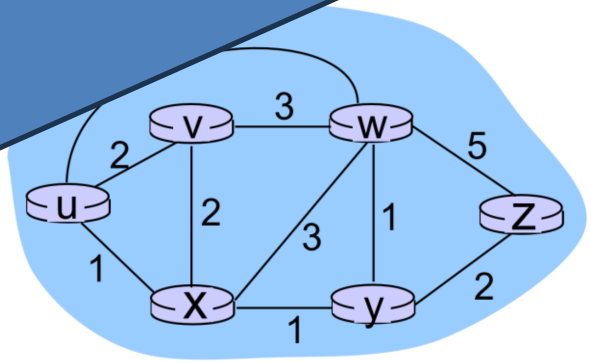
Routing tables are filled via **routing algorithms**

There are two types of routing algorithms

Centralized/Global- we know
the edge costs of the n k

Link Stat

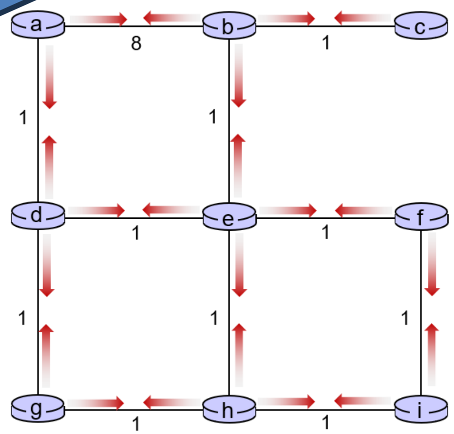
OSPF



Decentralized- we do not know
the edge costs of the e
network.
Only know
neig

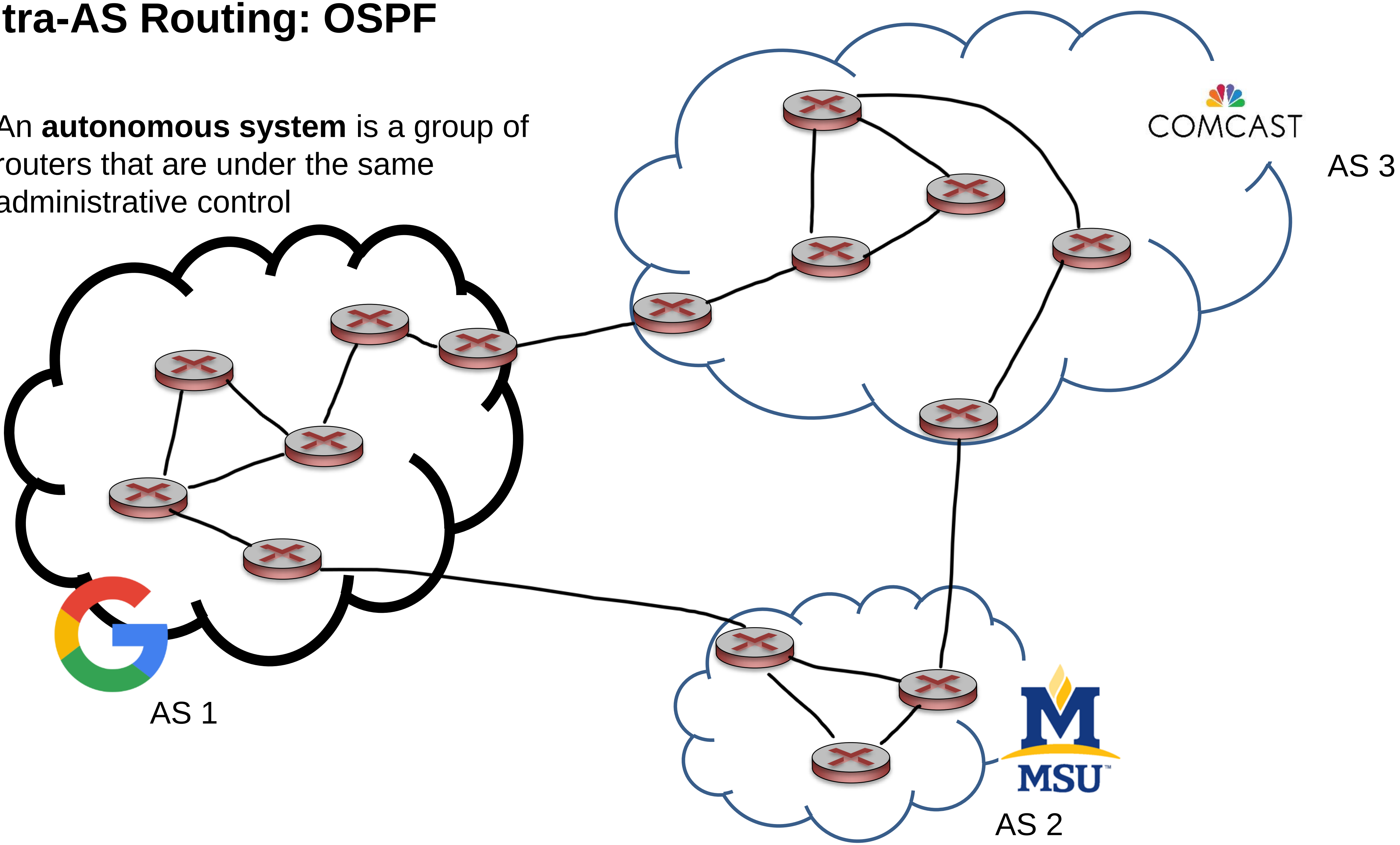
BGP

vector algorithms



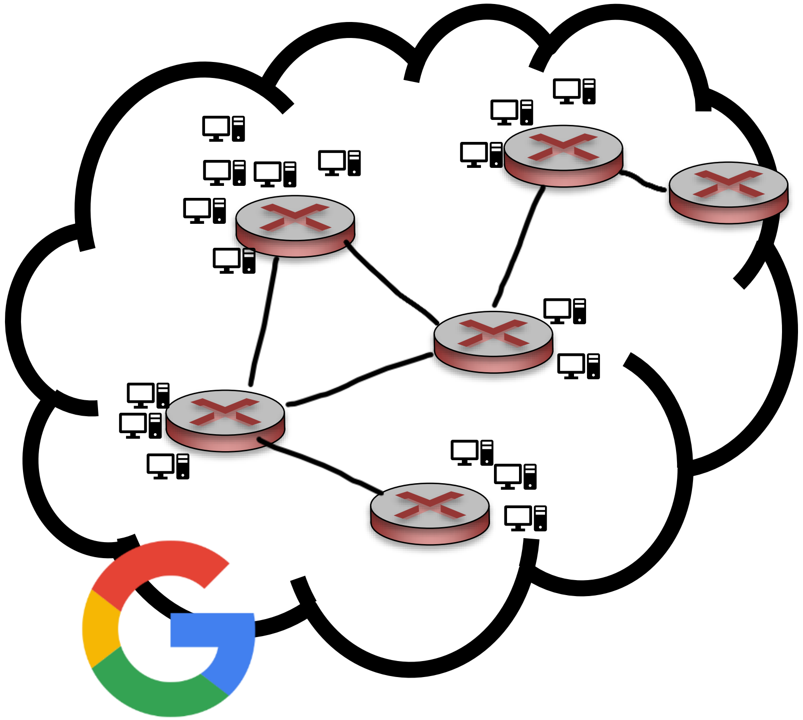
Intra-AS Routing: OSPF

An **autonomous system** is a group of routers that are under the same administrative control



Intra-AS Routing: OSPF

An **autonomous system** is a group of routers that are under the same administrative control



Open Shortest Path First

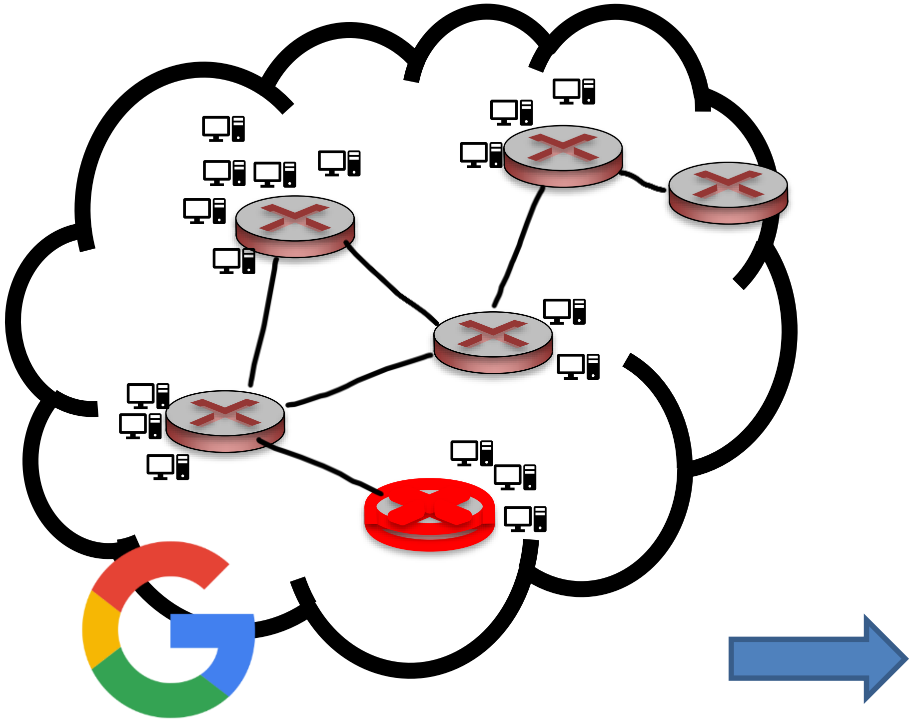
OSPF is a link-state protocol that uses flooding of link-state information and Dijkstra's least-cost algorithm

Intra-AS Routing: OSPF

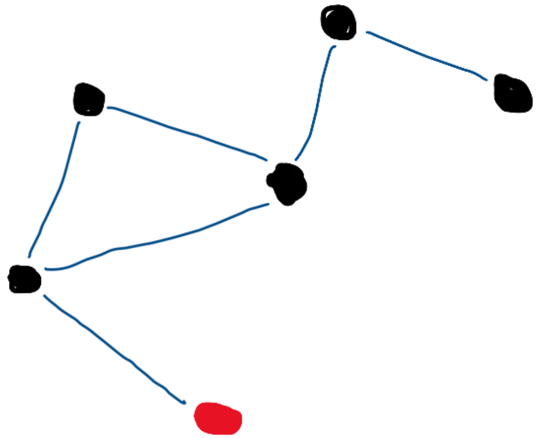
Open Shortest Path First

An **autonomous system** is a group of routers that are under the same administrative control

OSPF is a link-state protocol that uses flooding of link-state information and Dijkstra's least-cost algorithm



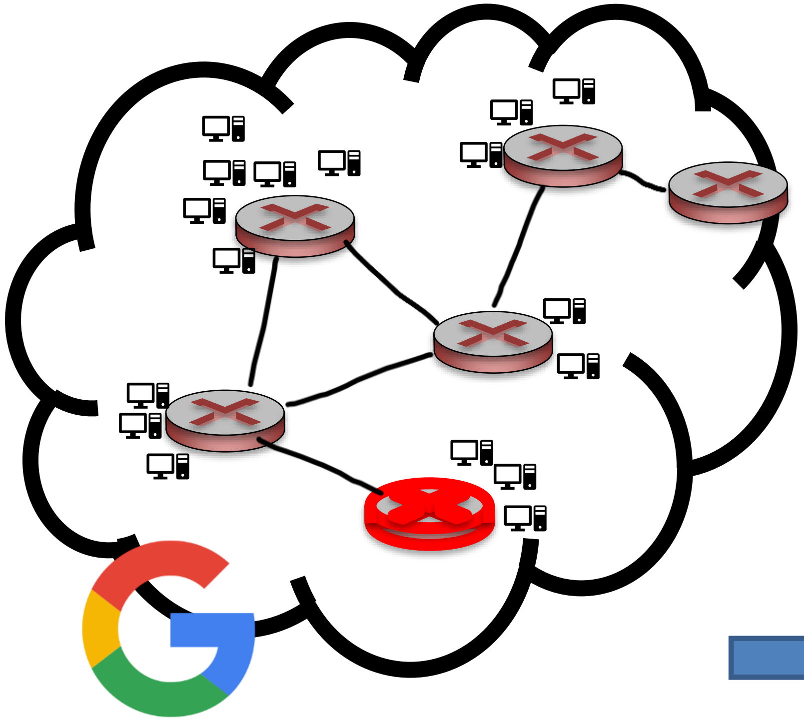
1. Each router constructors a topological map of the AS



Intra-AS Routing: OSPF

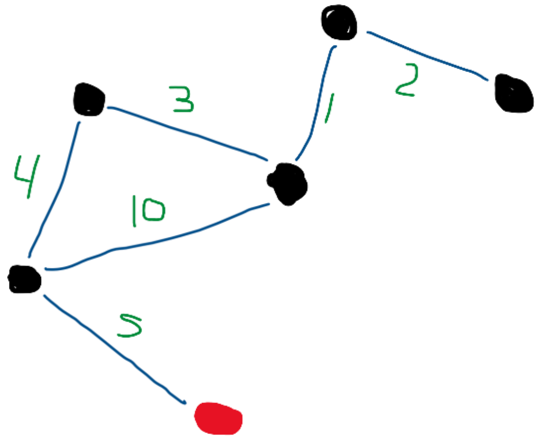
Open Shortest Path First

An **autonomous system** is a group of routers that are under the same administrative control



OSPF is a link-state protocol that uses flooding of link-state information and Dijkstra's least-cost algorithm

1. Each router constructors a topological map of the AS
2. Run Dijikstra's to determine shortest path to each subnet



(Edge costs will be set by a network administrator)

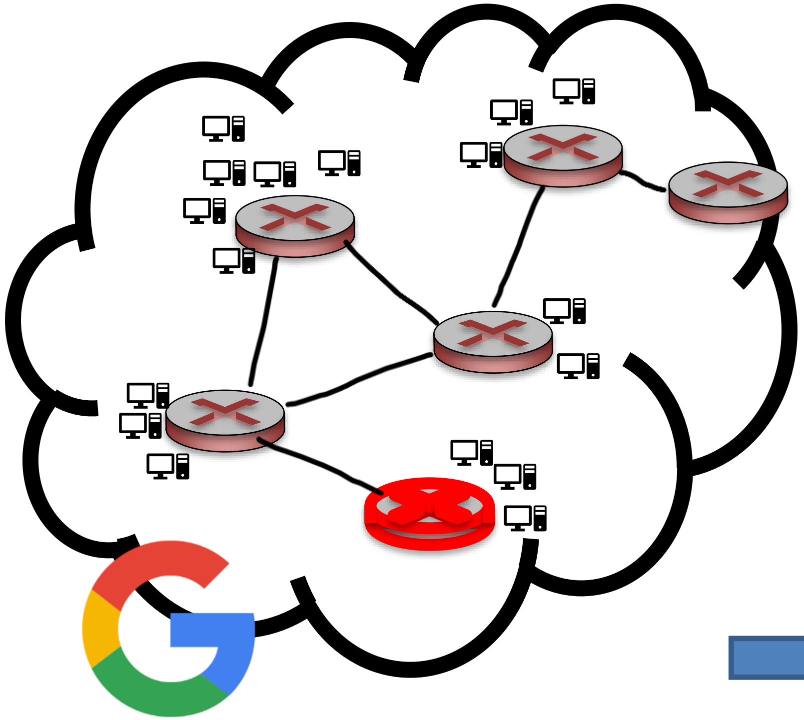
If I wanted to find the path with the shortest amount of hops, what should edge cost be?

Intra-AS Routing: OSPF

Open Shortest Path First

An **autonomous system** is a group of routers that are under the same administrative control

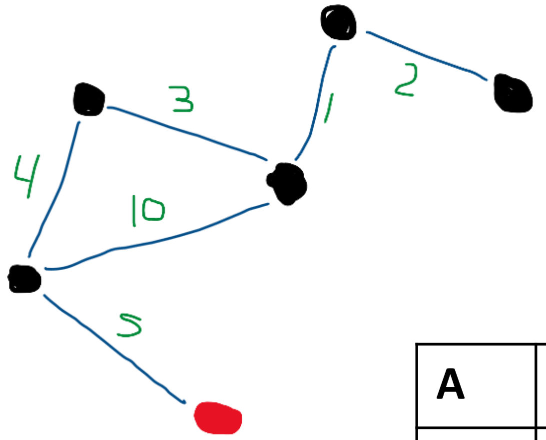
OSPF is a link-state protocol that uses flooding of link-state information and Dijkstra's least-cost algorithm



AS 1

1. Each router constructors a topological map of the AS

2. Run Dijikstra's to determine shortest path to each subnet



(Edge costs will be set by a network administrator)

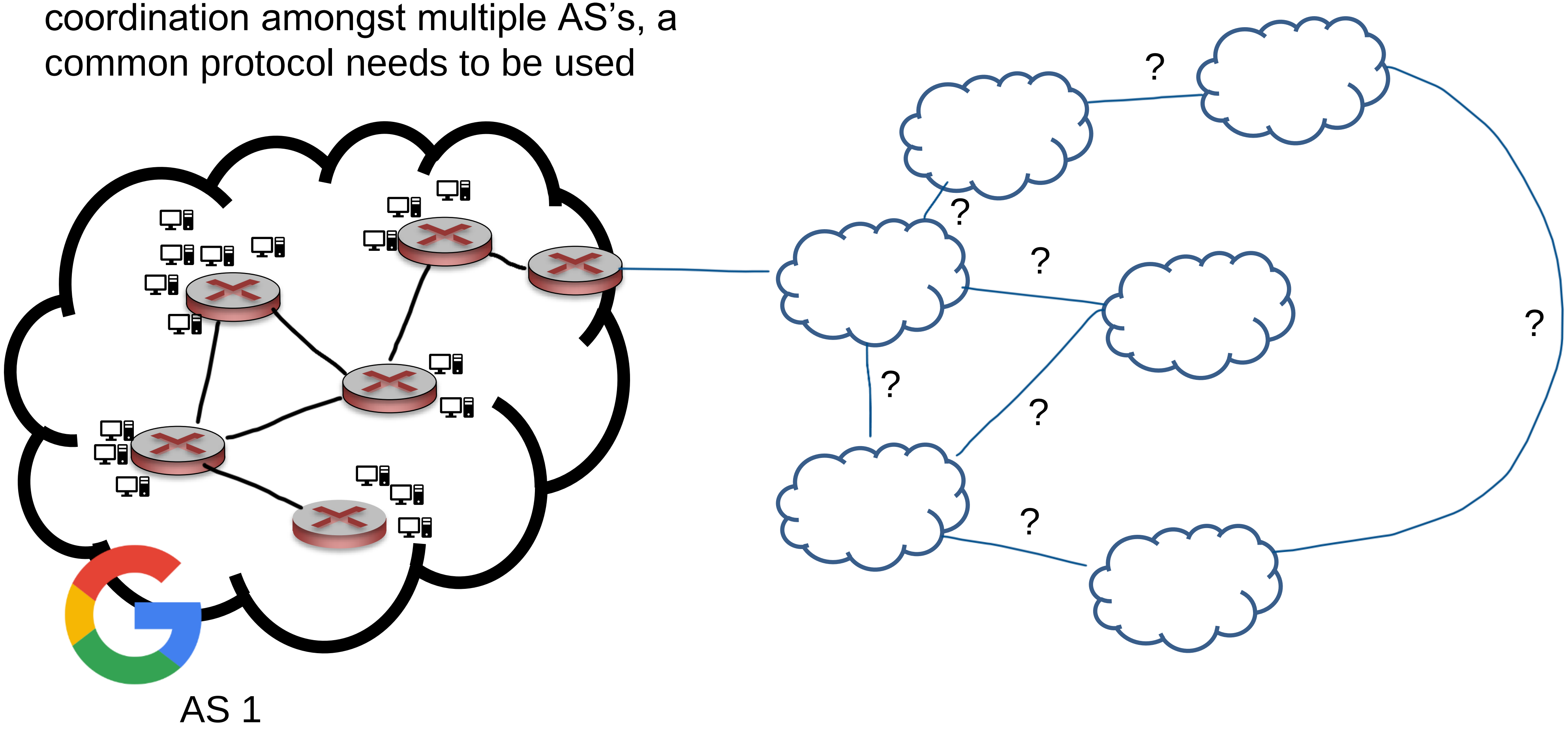
(could set all edges to be a cost of 1)

3. Fill in routing table

A	**1**
B	**2**
C	**3**
…	**…**

Routing Among the ISPs: BGP

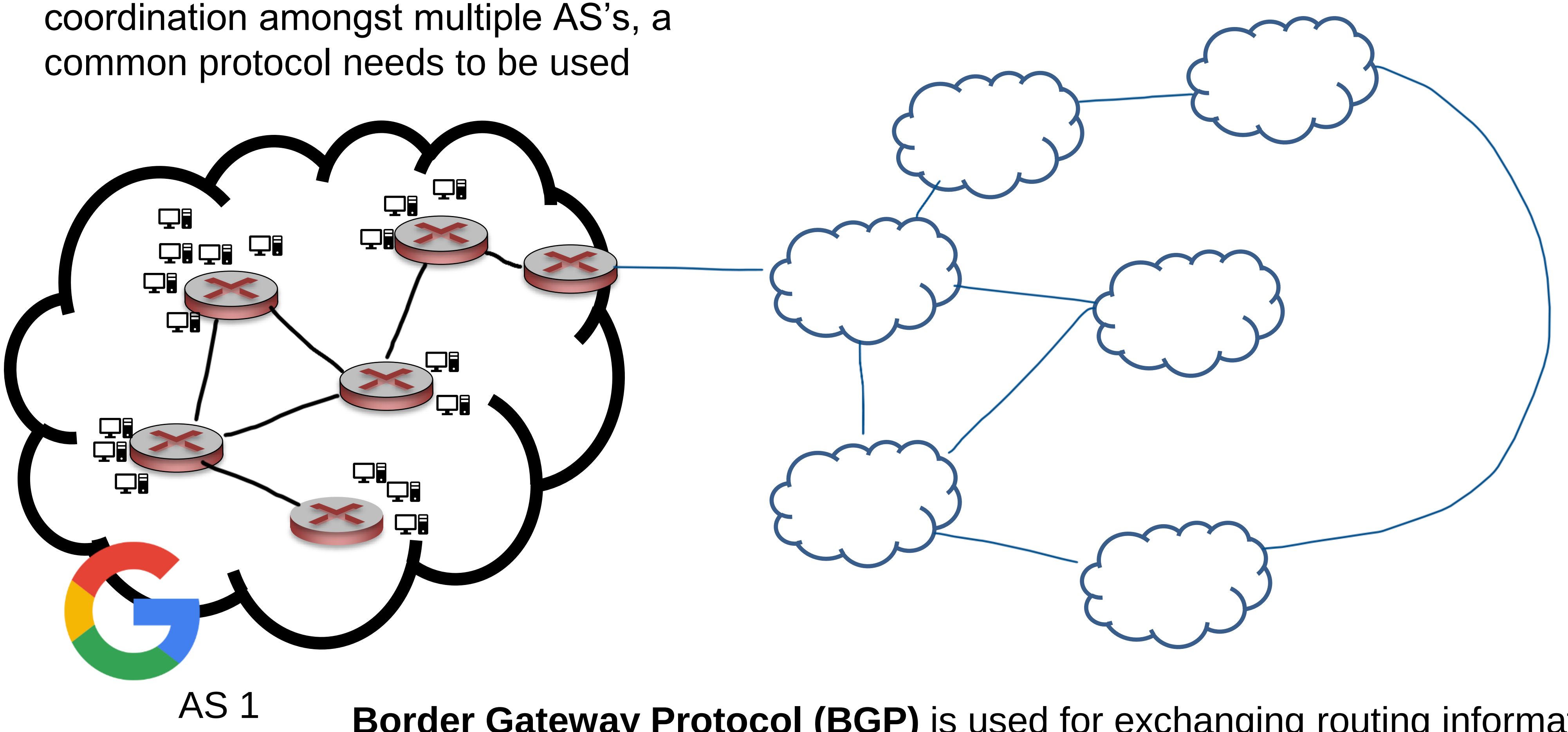
Inter-AS routing protocol involves coordination amongst multiple AS's, a common protocol needs to be used



Border Gateway Protocol (BGP) is used for exchanging routing information between AS

Routing Among the ISPs: BGP

Inter-AS routing protocol involves coordination amongst multiple AS's, a common protocol needs to be used



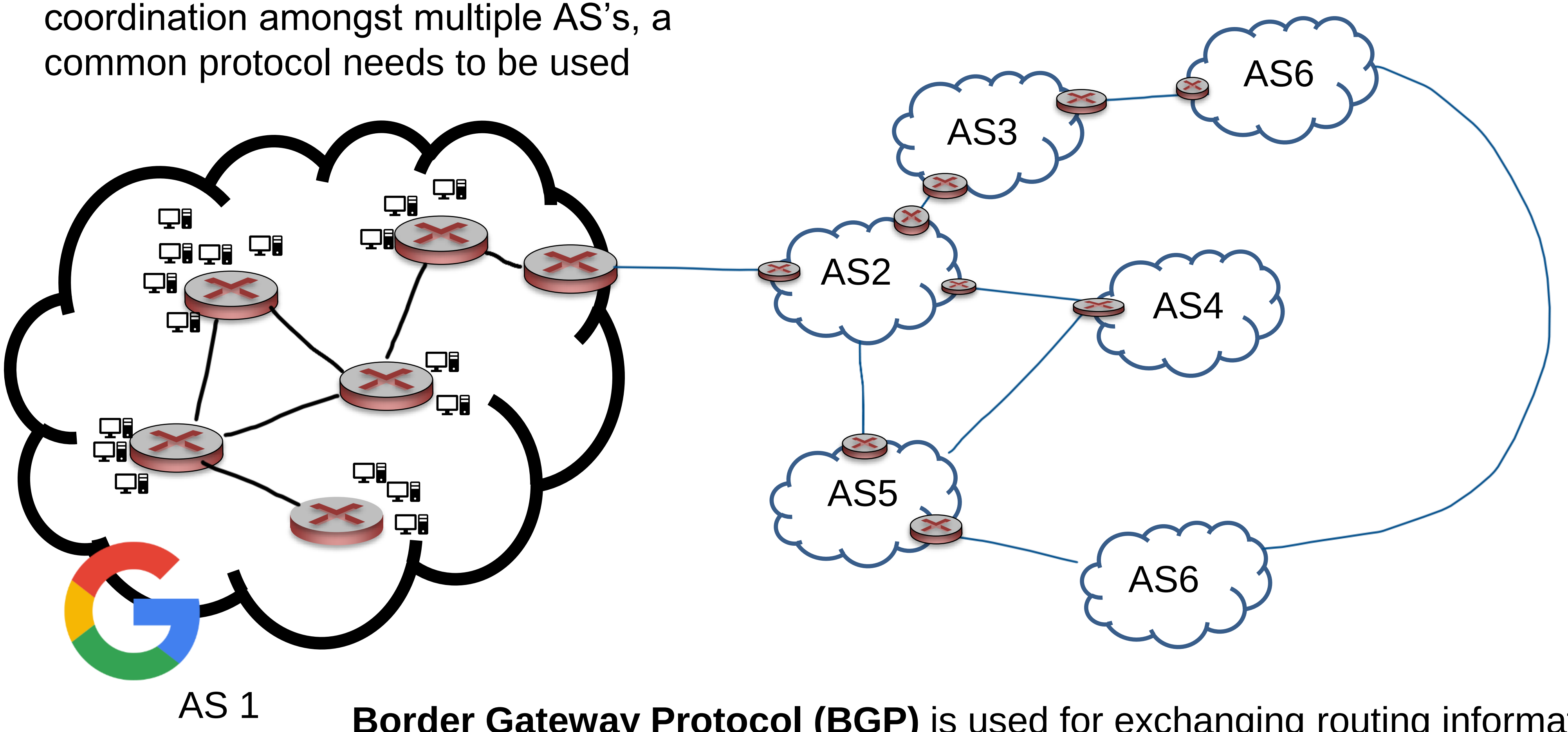
Border Gateway Protocol (BGP) is used for exchanging routing information between AS

BGP allows a router to tell other AS's that it exists and needs to be connected

Routing Among the ISPs: BGP

AS consists of **gateway routers** and **internal routers**

Inter-AS routing protocol involves coordination amongst multiple AS's, a common protocol needs to be used



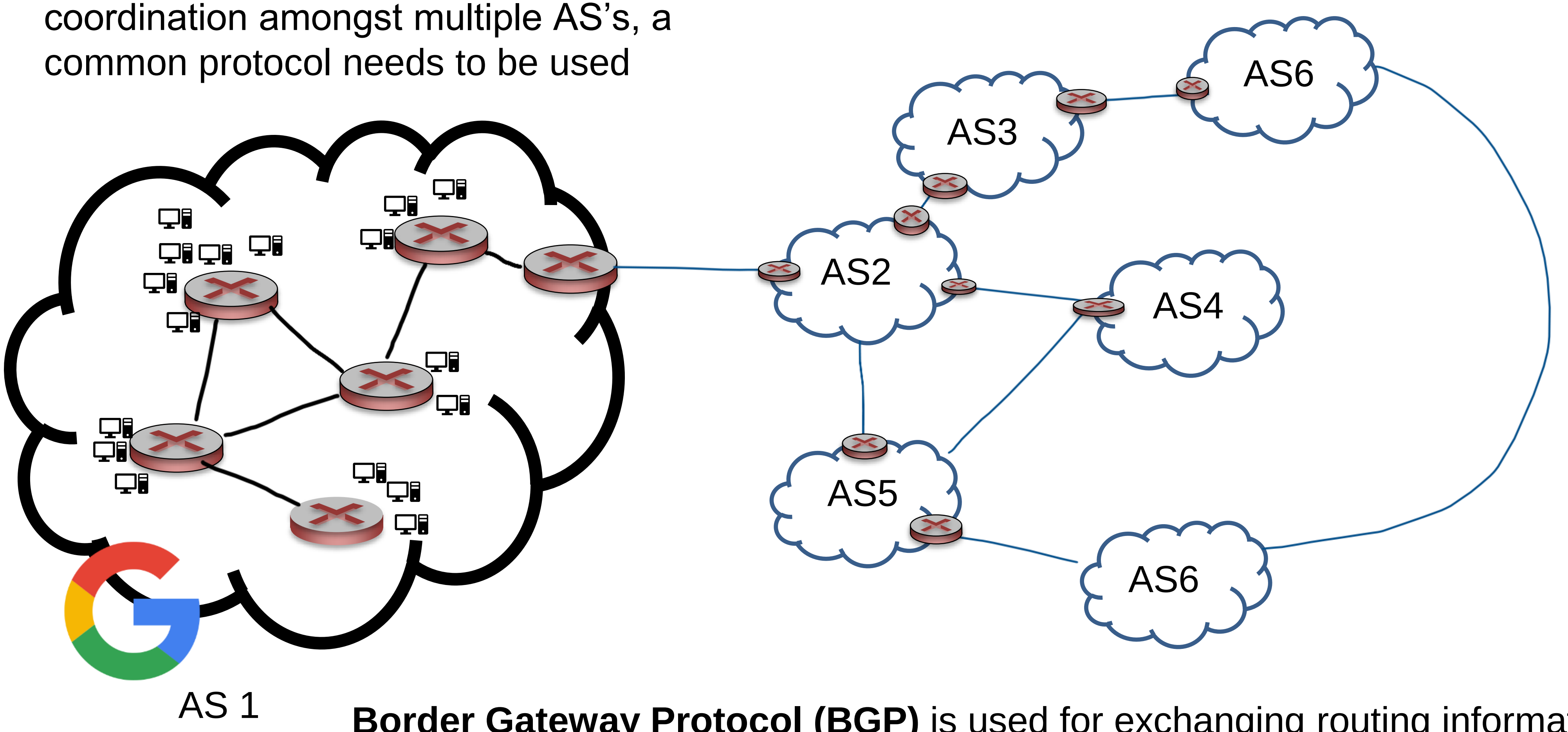
Border Gateway Protocol (BGP) is used for exchanging routing information between AS

BGP allows a router to tell other AS's that it exists and needs to be connected

Routing Among the ISPs: BGP

AS consists of **gateway routers** and **internal routers**

Inter-AS routing protocol involves coordination amongst multiple AS's, a common protocol needs to be used



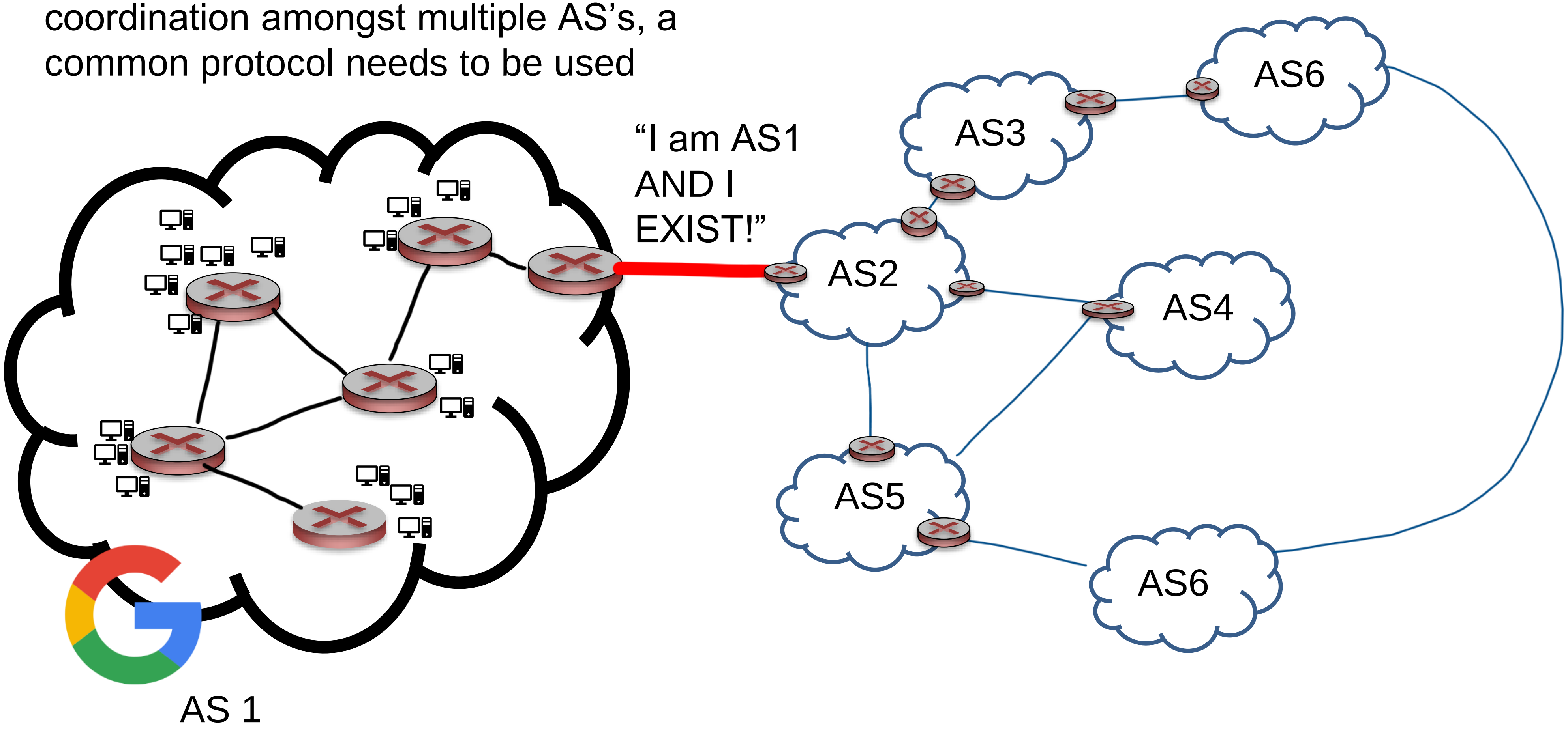
Border Gateway Protocol (BGP) is used for exchanging routing information between AS

BGP allows a router to tell other AS's that it exists and needs to be connected

Routing Among the ISPs: BGP

AS consists of **gateway routers** and **internal routers**

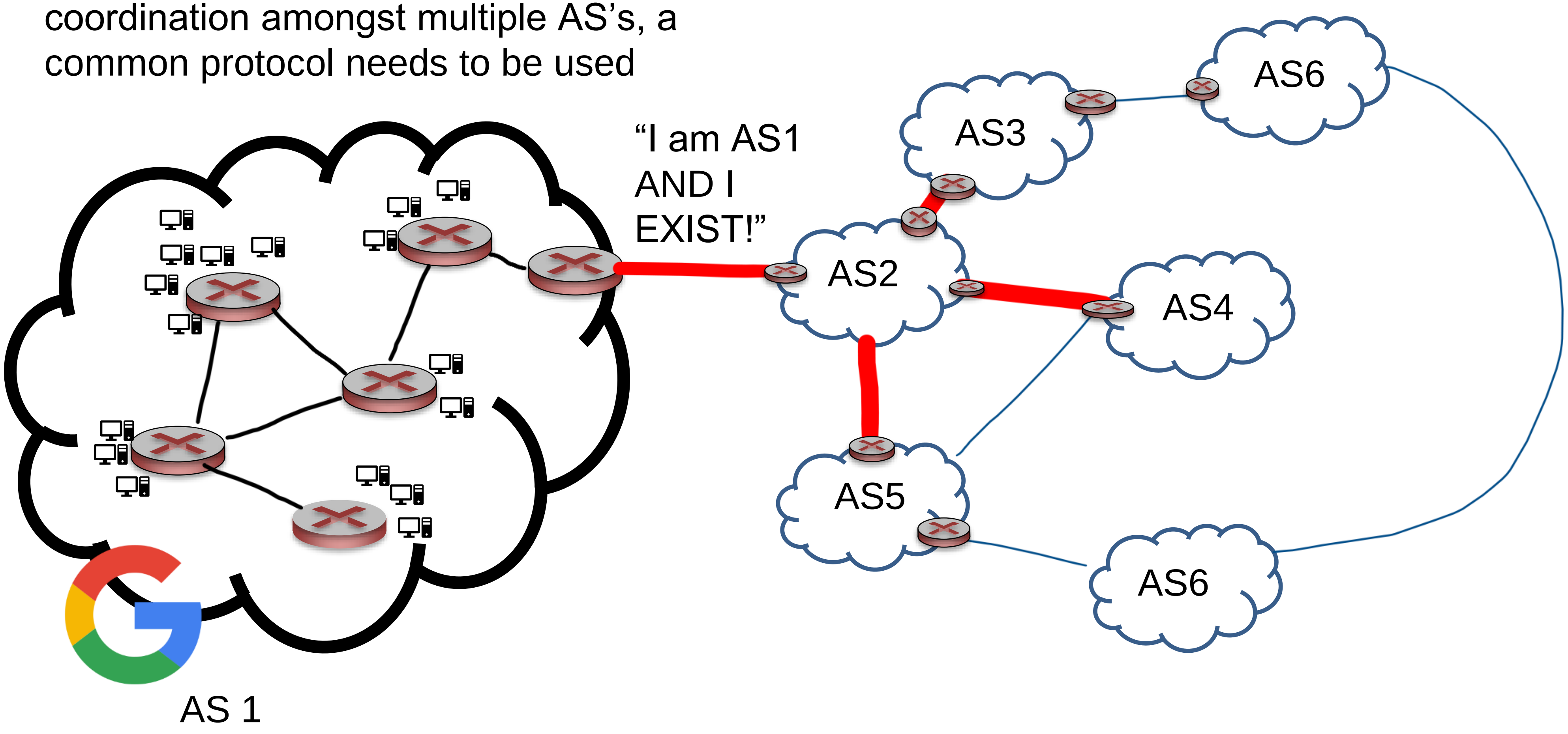
Inter-AS routing protocol involves coordination amongst multiple AS's, a common protocol needs to be used



Routing Among the ISPs: BGP

AS consists of **gateway routers** and **internal routers**

Inter-AS routing protocol involves coordination amongst multiple AS's, a common protocol needs to be used



"A1 EXISTS AND FOUND THROUGH AS2"

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- allows subnet to advertise its existence, and the destinations it can reach, to rest of Internet: *"I am here, here is who I can reach, and how"*
- BGP provides each AS a means to:
  - obtain destination network reachability info from neighboring ASes (eBGP)
  - determine routes to other networks based on reachability information and *policy*
  - propagate reachability information to all AS-internal routers (iBGP)
  - advertise (to neighboring networks) destination reachability info

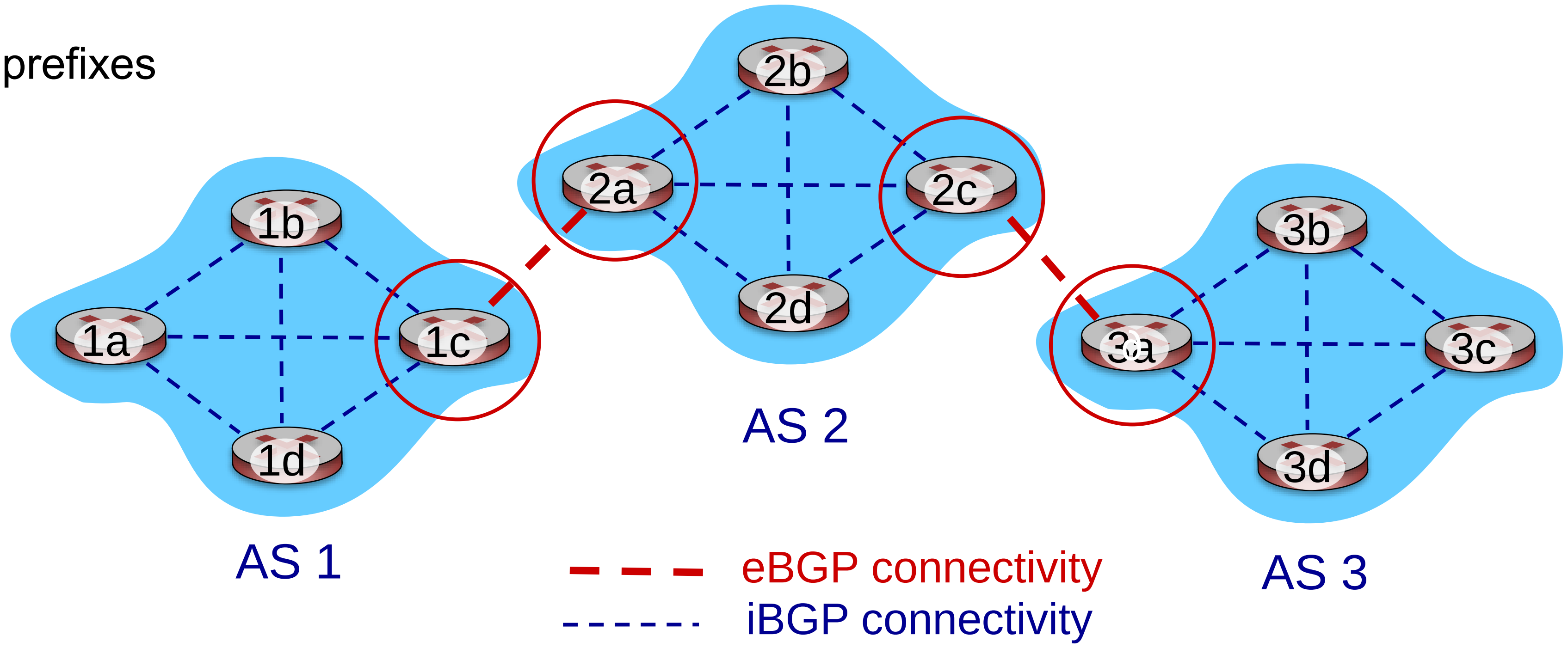
Routing Among the ISPs: BGP

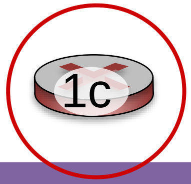
BGP is the routing protocol used for routing amongst different ISPs + AS

Two important functions

→ Obtain prefix reachability information from neighboring ASs (CIDR)

→ Determine the “best” routes to the prefixes





gateway routers run both eBGP and iBGP protools

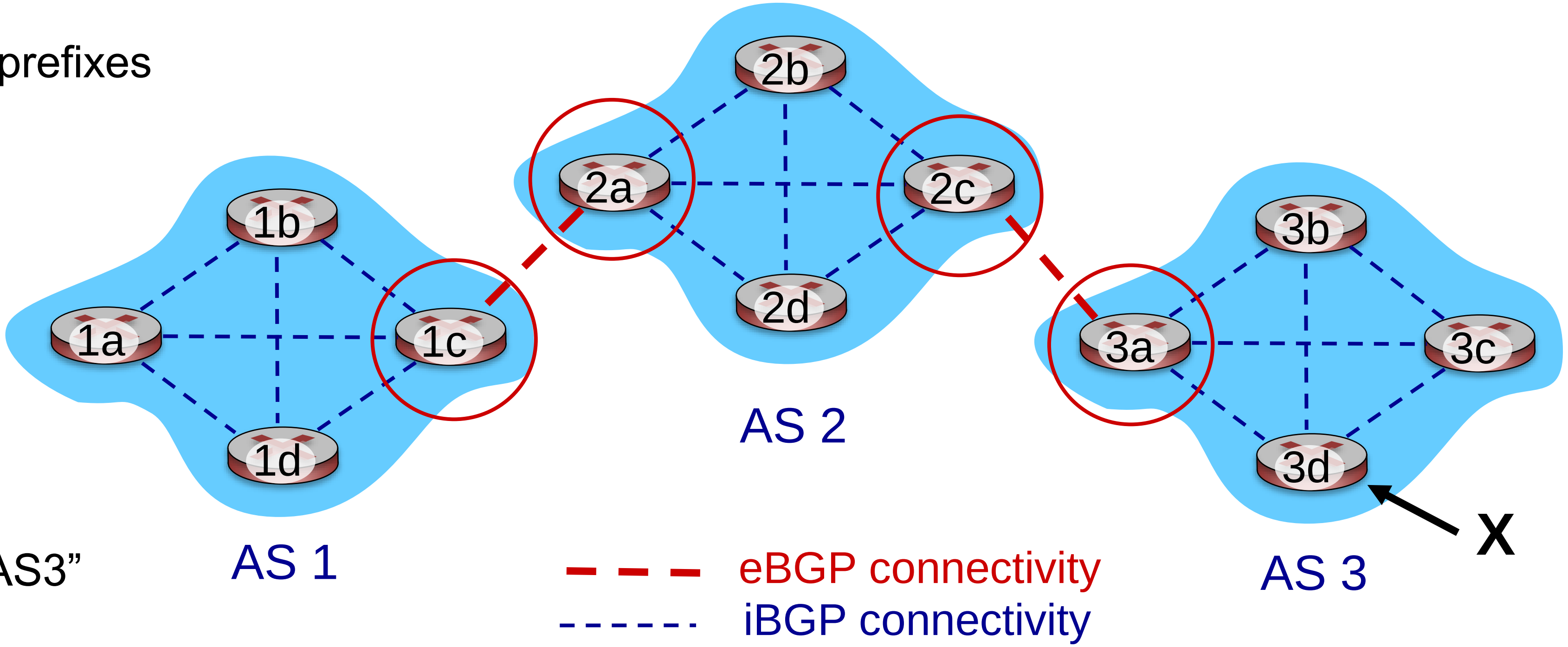
Routing Among the ISPs: BGP

BGP is the routing protocol used for
routing amongst different ISPs + AS

Two important functions
→ Obtain prefix reachability information from neighboring ASs

→ Determine the “best” routes to the prefixes

Prefix **X** connect



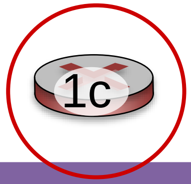
External BGP (eBGP)

3a → 2c “Hey I have X”

2a → 1c “Hey AS 3 has X and I have AS3”

Internal BGP (iBGP)

2c → 2b
2c → 2d
2c → 2a



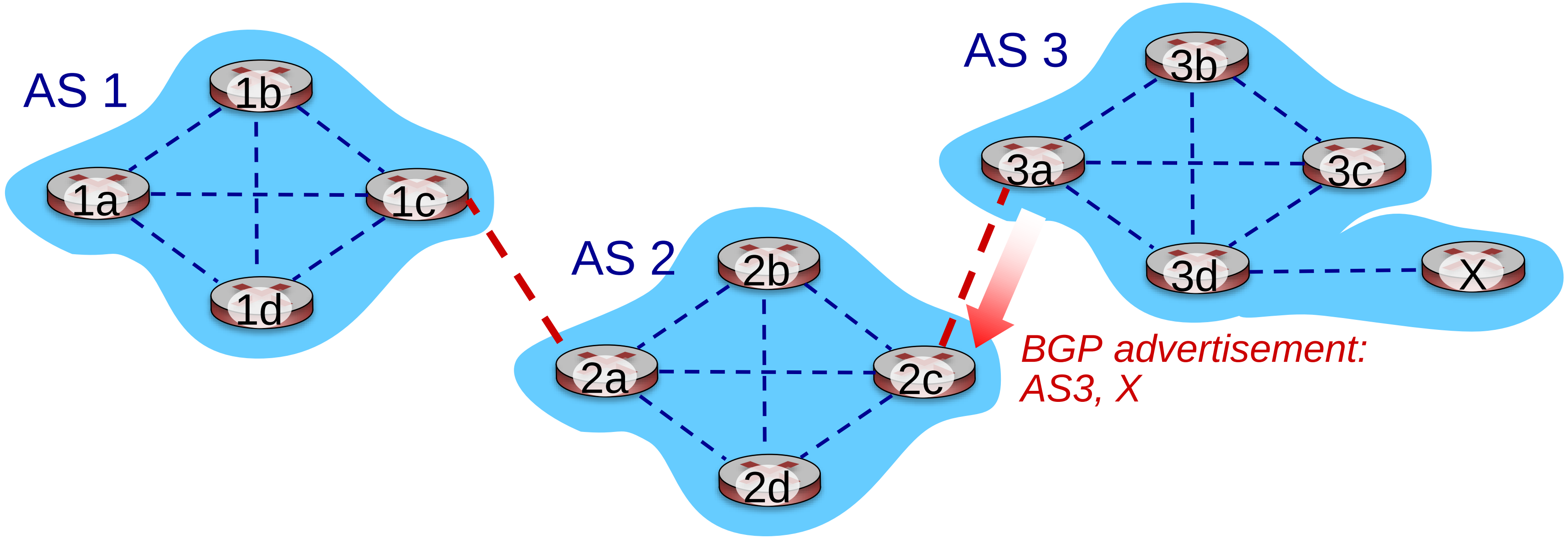
gateway routers run both eBGP and iBGP protocols

BGP basics

- BGP session: two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection:
  - advertising *paths* to different destination network prefixes (BGP is a "path vector" protocol)

when AS3 gateway router 3a advertises path AS3,X to AS2 gateway router 2c:

AS3 *promises* to AS2 it will forward datagrams towards X



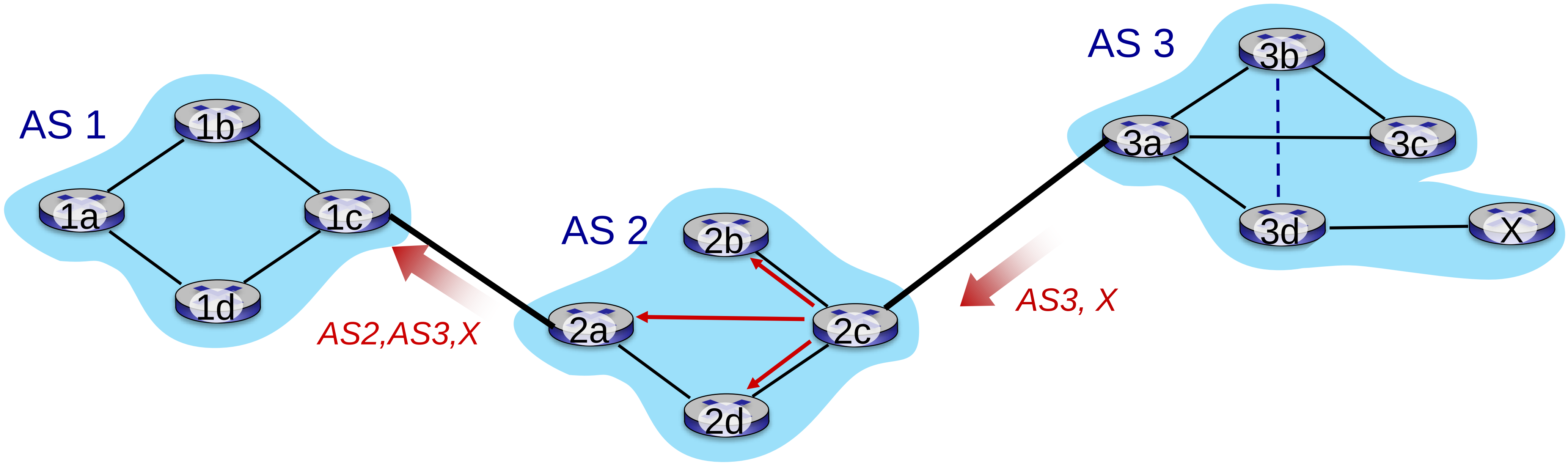
BGP protocol messages

- BGP messages exchanged between peers over TCP connection
- BGP messages [RFC 4371]:
  - OPEN: opens TCP connection to remote BGP peer and authenticates sending BGP peer
  - **UPDATE:** advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

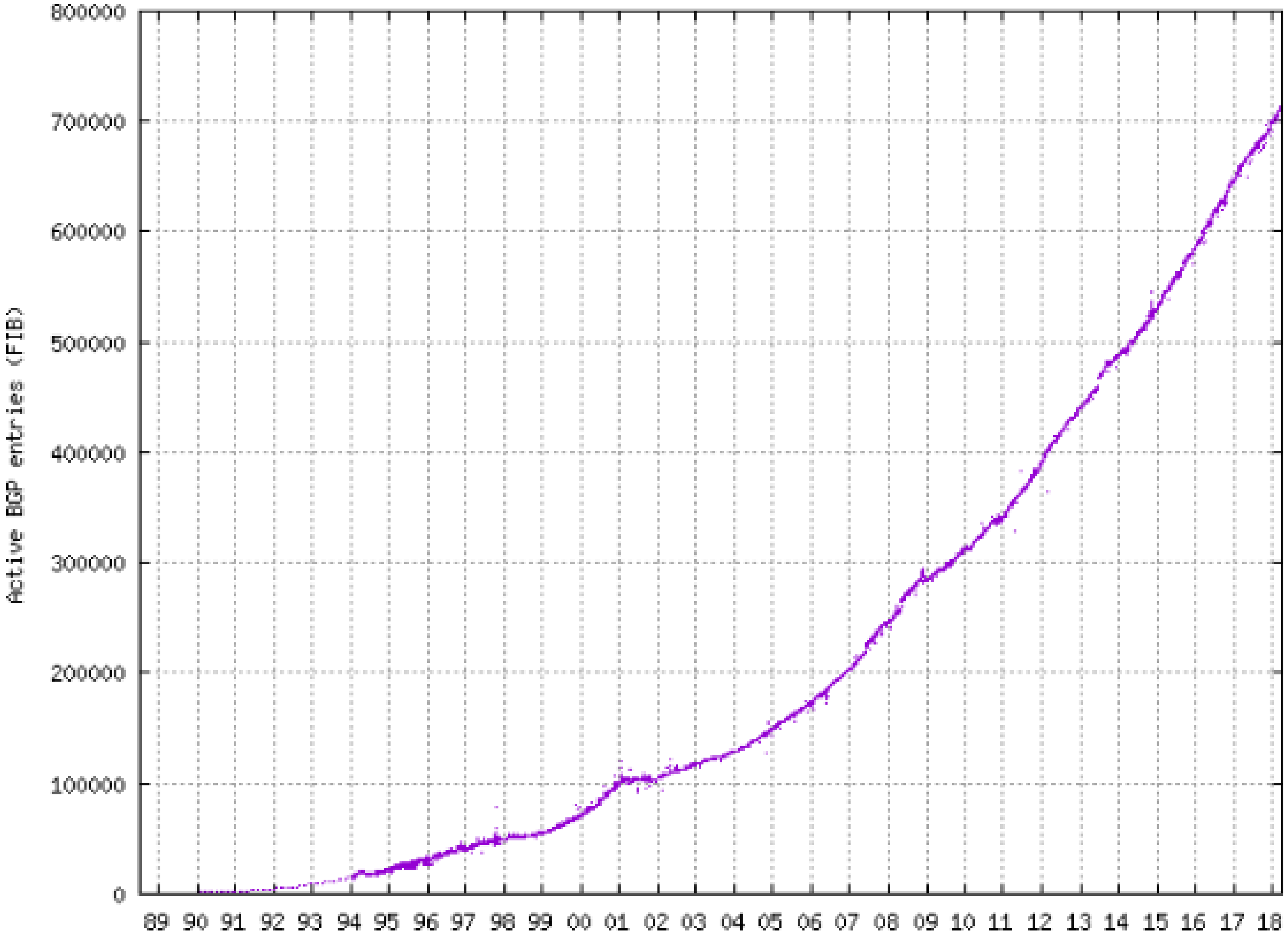
Path attributes and BGP routes

- BGP advertised route: prefix + attributes
  - prefix: destination being advertised
  - two important attributes:
    - AS-PATH: list of ASes through which prefix advertisement has passed
    - NEXT-HOP: indicates specific internal-AS router to next-hop AS
- policy-based routing:
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - AS policy also determines whether to *advertise* path to other other neighboring ASes

BGP path advertisement



- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c



800000
700000
600000
500000
400000
300000
200000
100000
0
Active BGP entries (FIB)
89 90 91 92 93 94 95 96 97 98 99 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18

ROUTE VIEWS 6447

HOME ABOUT FAQ PEERING TOOLS COLLECTORS MAP ARCHIVE PAPERS SUPPORTERS CONTACT

Collectors

RouteViews is collecting BGP Updates at the following locations

Exchanges

Host	MFG	Proto	Location
amsix.ams.routeviews.org	FRR	IPv4/6	AMS-IX Amsterdam, Netherlands
cix.atl.routeviews.org	FRR	IPv4/6	CIX-ATL Atlanta, Georgia
decix.jhb.routeviews.org	FRR	IPv4/6	DE-CIX KUL, Johor Bahru, Malaysia
iraq-ixp.bgw.routeviews.org	FRR	IPv4/6	IRAQ-IXP Baghdad, Iraq
pacwave.lax.routeviews.org	FRR	IPv4/6	Pacific Wave, Los Angeles, California
pit.scl.routeviews.org	FRR	IPv4/6	PIT Chile Santiago, Santiago, Chile
pitmx.qro.routeviews.org	FRR	IPv4/6	PIT Chile MX, Querétaro, Mexico
route-views.routeviews.org	Cisco	IPv4	U of Oregon, Eugene Oregon

https://www.routeviews.org/routeviews/index.php/collectors/

```
route-views>show ip bgp sum
BGP router identifier 128.223.51.103, local AS number 6447
BGP table version is 355532718, main routing table version 355532718
Path RPKI states: 9671162 valid, 8728431 not found, 5720 invalid
990349 network entries using 245606552 bytes of memory
18405313 path entries using 2208637560 bytes of memory
2919344/172592 BGP path/bestpath attribute entries using 723997312 bytes of memo
ry
2802466 BGP AS-PATH entries using 142729518 bytes of memory
227141 BGP community entries using 38426562 bytes of memory
2503 BGP extended community entries using 162876 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 3359560380 total bytes of memory
BGP activity 15050631/13844711 prefixes, 883710380/861987137 paths, scan interva
l 60 secs
```

Lots of network and path entries on this BGP router

The list of neighbors it interacts with

```
Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State
/PfxRcd
4.68.4.46       4         3356 15253358  590045 355538393    0    0 9w0d        9
46903
12.0.1.63       4         7018 29842482   89389 355538393    0    0 8w0d        9
47328
37.139.139.17   4        57866  271682     822 355538393    0    0 06:15:50   95
0829
64.71.137.241   4         6939 8581620   44533 355538393    0    0 4w0d       96
9900
77.39.192.30    4        20912 7062039  162429 355538393    0    0 2w3d       94
9277
89.149.178.10   4         3257 5761037    3436 355538393    0    0 1w3d       94
7649
91.218.184.60   4        49788 17612113  330474 355538393    0    0 9w0d        9
51340
94.142.247.3    4         8283 51450625  330485 355538393    0    0 9w0d        9
53214
114.31.199.16   4         4826 19029982  355588 355538393    0    0 9w0d        9
73382
132.198.255.253 4         1351 3839141   33365 355538393    0    0 3w0d       97
3404
140.192.8.16    4        20130 19891315  141651 355538393    0    0 6w3d        9
73729
144.228.241.130 4         1239  199819   40395 355538393    0    0 6w0d        3
0597
154.11.12.212   4          852 25322844  454901 355538393    0    0 9w0d        9
49731
162.250.137.254 4         4901 72911065  400400 355538393    0    0 21w6d       9
52703
```

ICMP (Internet Control Message Protocol)

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header