Dijkstra's algorithm (C++)

‘What’, ‘why’ and ‘how’

#csspree Online

Dijkstra's algorithm

Dijkstra's algorithm

It is a path-finding algorithm that:

Dijkstra's algorithm

It is a path-finding algorithm that:

1. Visits nodes distance-wise. That is, from the currently discovered nodes, the one with the shortest distance will be 'visited' first.

Dijkstra's algorithm

It is a path-finding algorithm that:

1. Visits nodes distance-wise. That is, from the currently discovered nodes, the one with the shortest distance will be 'visited' first.

2. Finds shortest path to every node given that the edge weight is non-negative for the entire graph.

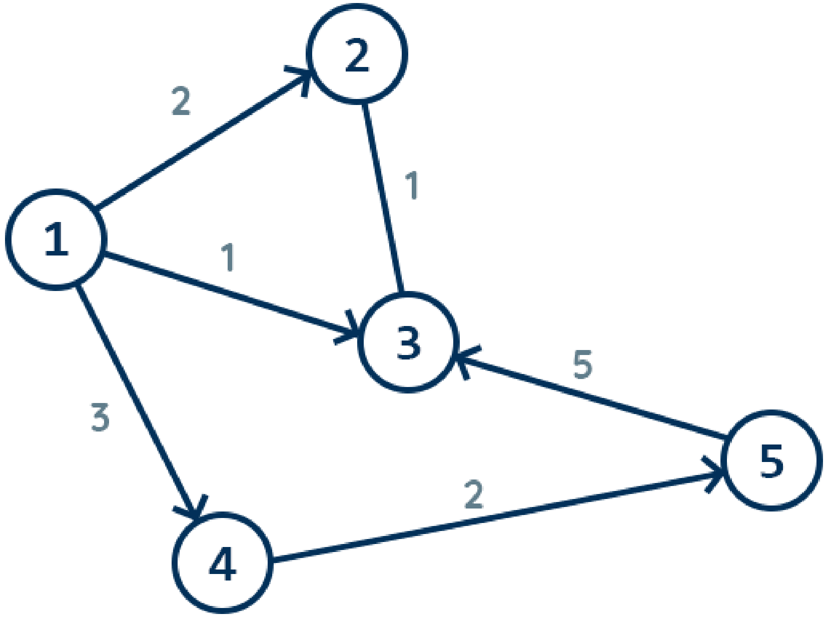
Assumption of Dijkstra

Assumption of Dijkstra

There is no negative edge cost present in graph.

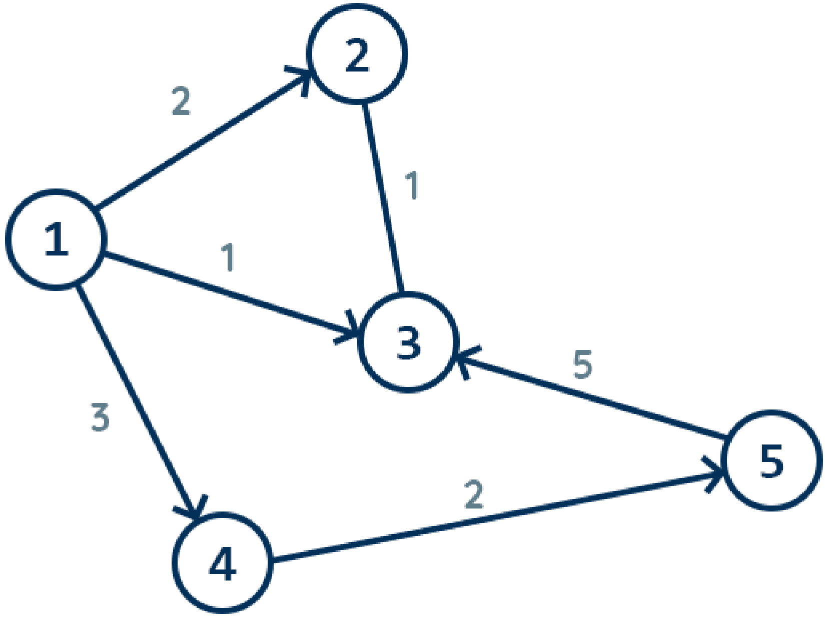
Assumption of Dijkstra

There is no negative edge cost present in graph.



Assumption of Dijkstra

There is no negative edge cost present in graph.

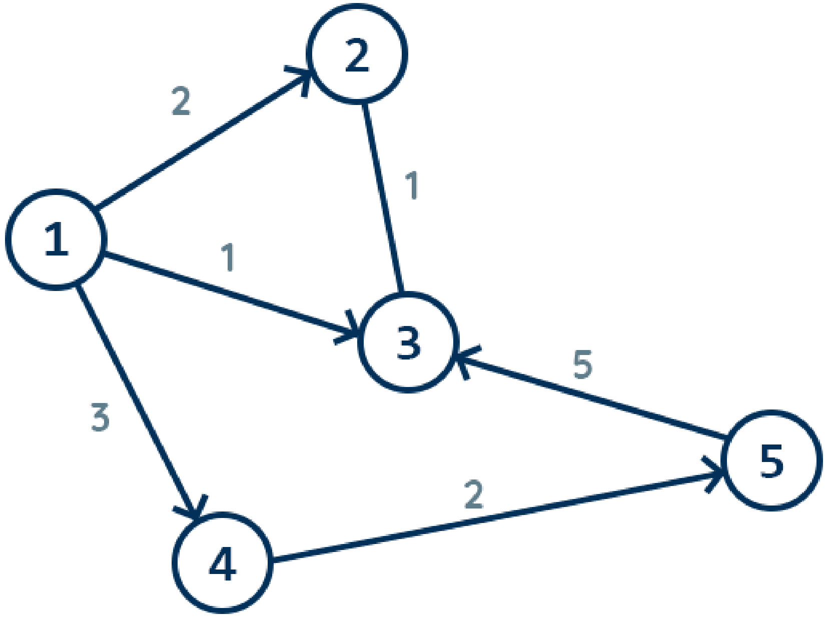


No negative cost is present

Valid ✓

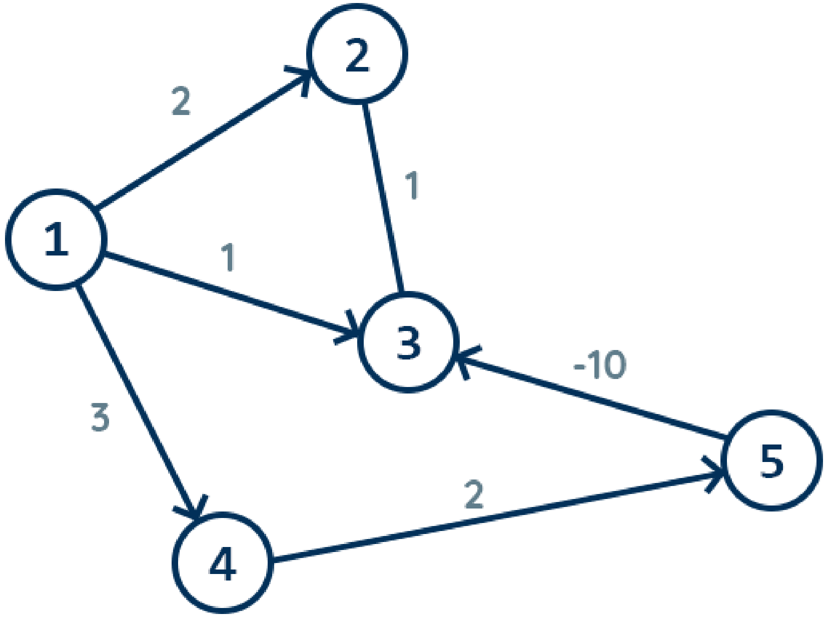
Assumption of Dijkstra

There is no negative edge cost present in graph.



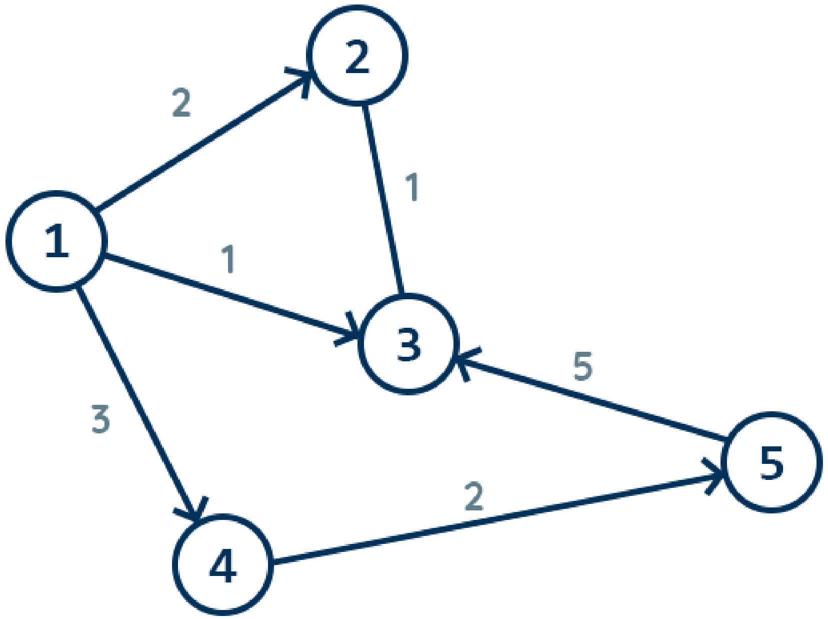
No negative cost is present

Valid ✓



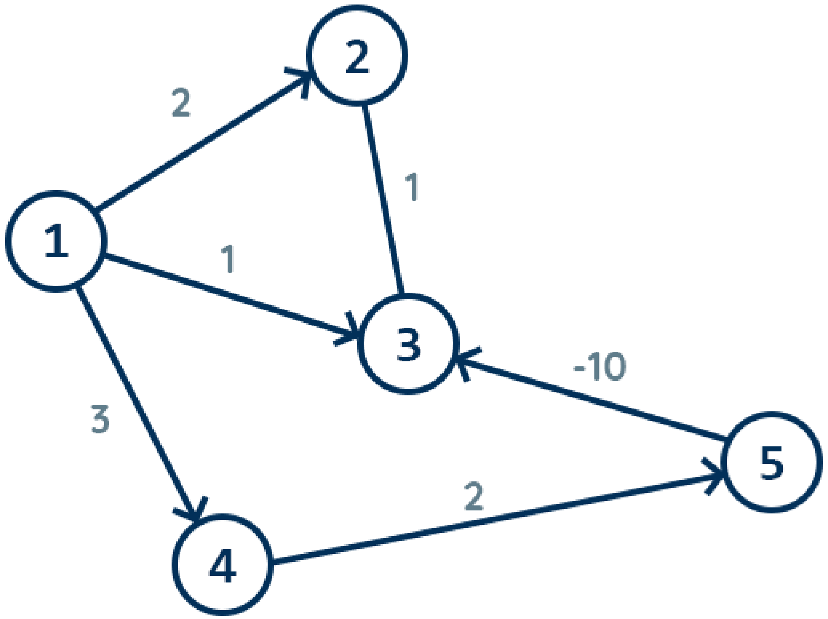
Assumption of Dijkstra

There is no negative edge cost present in graph.



No negative cost is present

Valid ✓

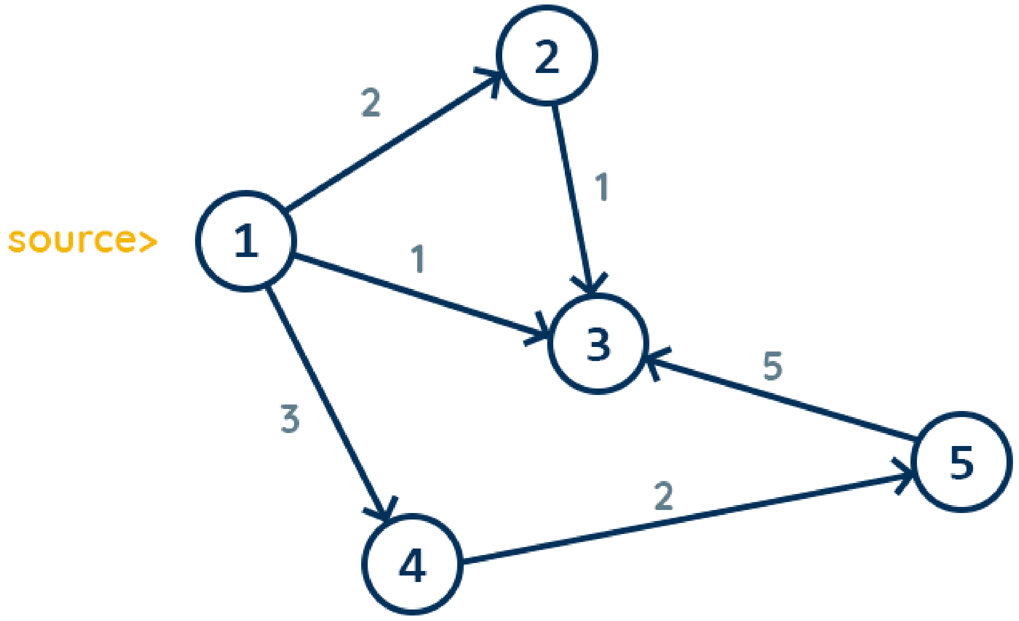


Negative cost is present

Invalid ×

Working Procedure of Dijkstra

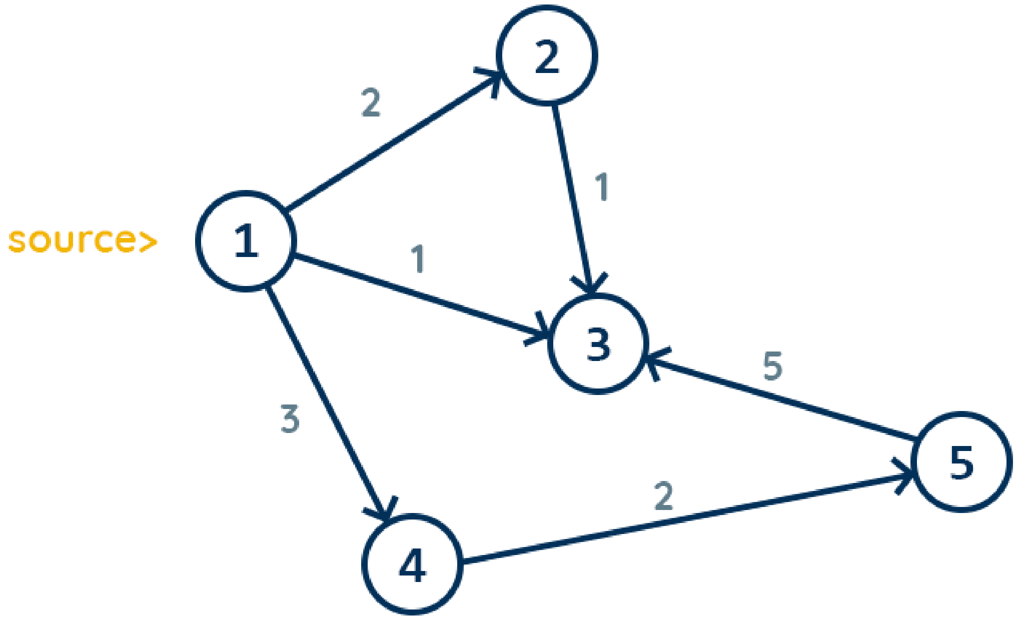
Working Procedure of Dijkstra



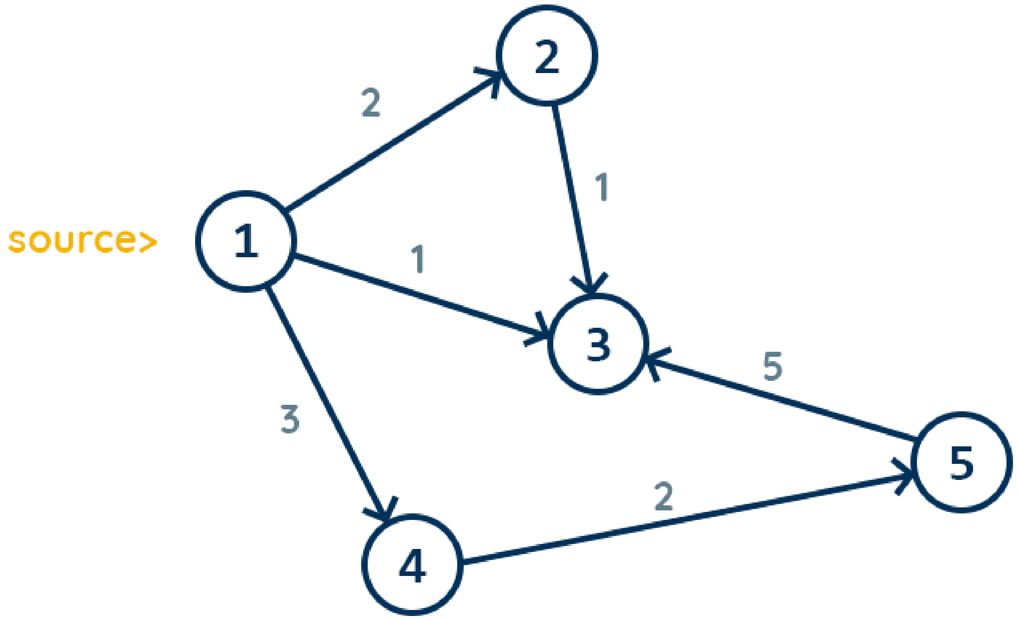
Working Procedure of Dijkstra

Min Priority Queue

Shortest Path



Working Procedure of Dijkstra

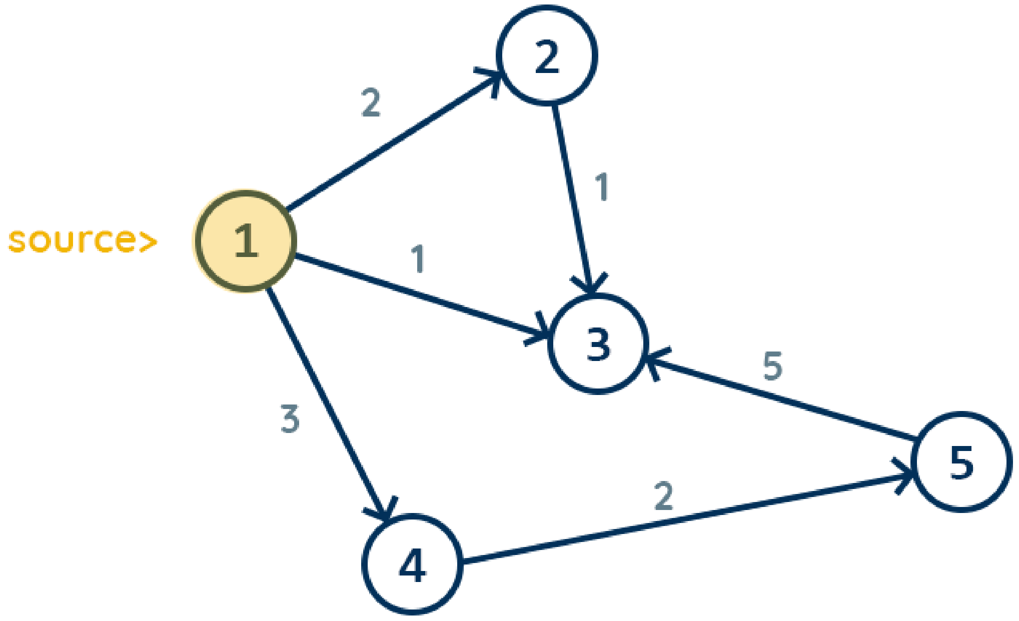


Min Priority Queue

push {node, distance} pair here

Shortest Path

Working Procedure of Dijkstra



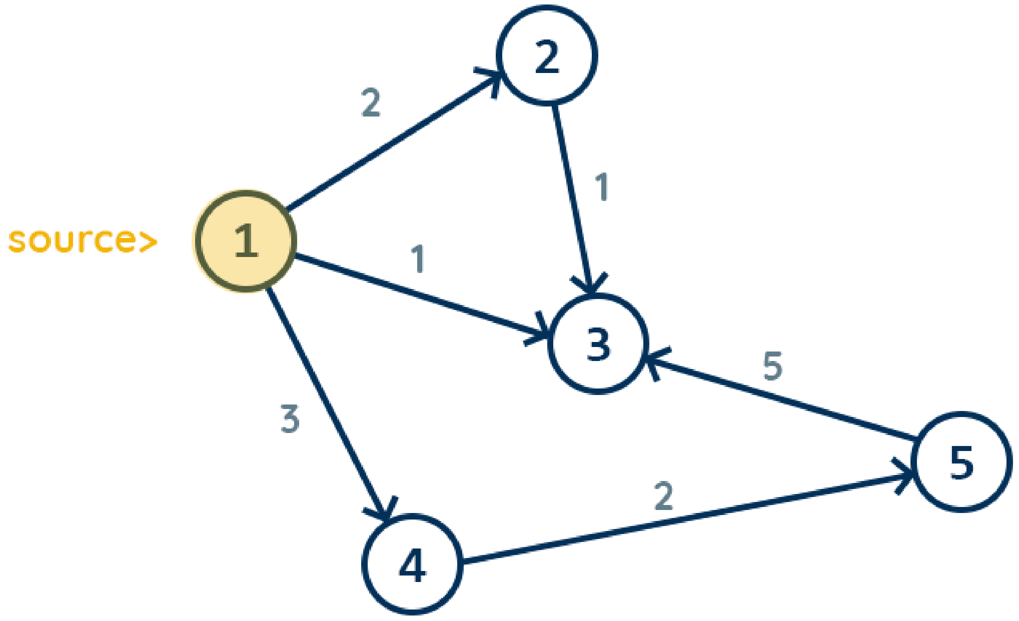
Min Priority Queue

push {node, distance} pair here

{1, 0}

Shortest Path

Working Procedure of Dijkstra



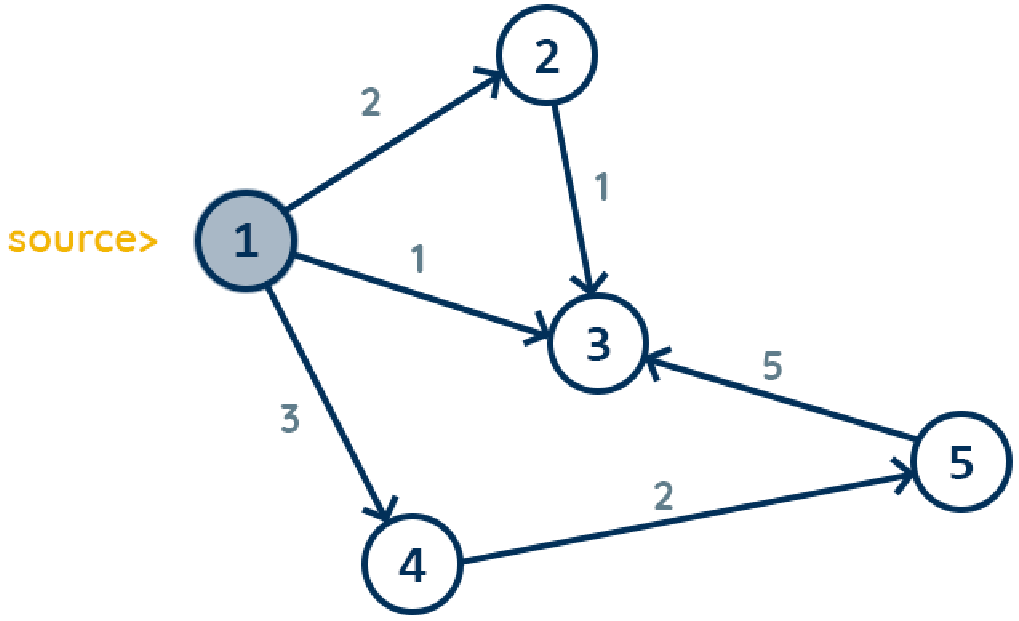
Min Priority Queue

push {node, distance} pair here

{1, 0} //popped

Shortest Path

Working Procedure of Dijkstra



// marked as visited
after popping

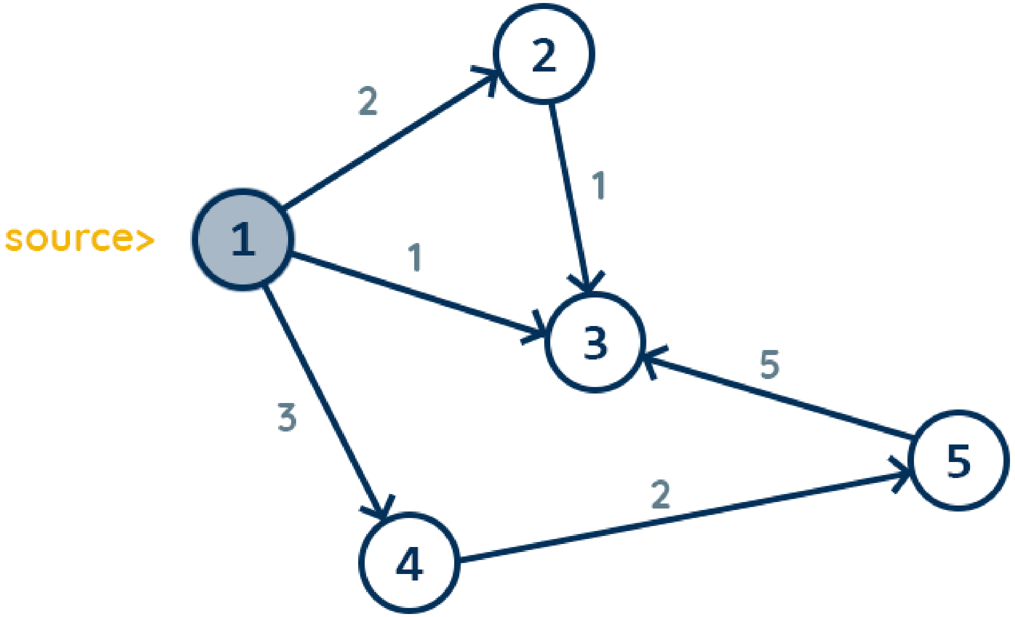
Min Priority Queue

push {node, distance} pair here

{1, 0} //popped

Shortest Path

Working Procedure of Dijkstra



Min Priority Queue

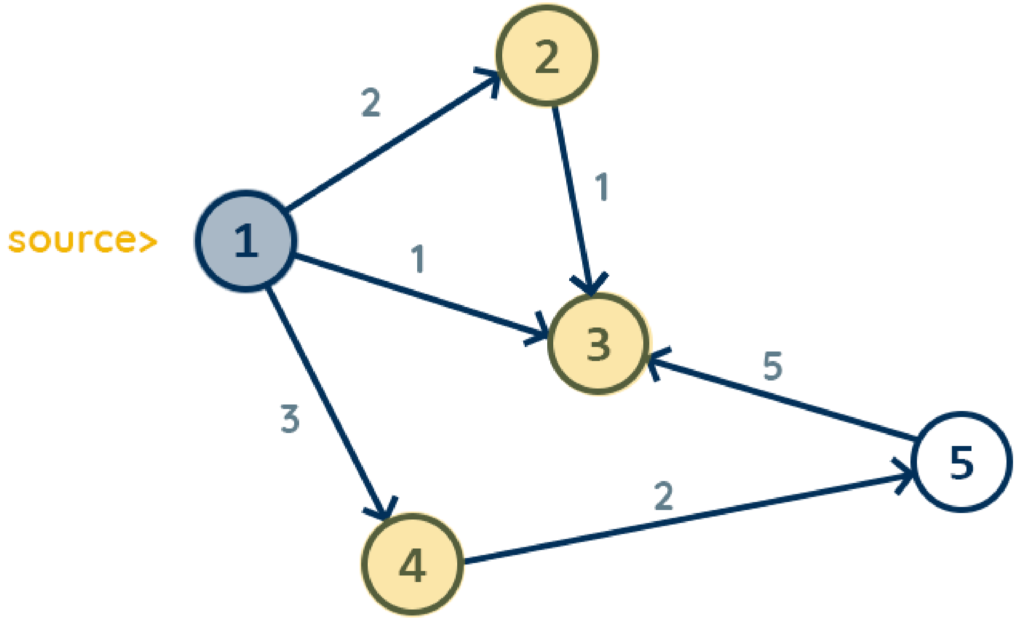
push {node, distance} pair here

{1, 0} //popped

Shortest Path

1 : 0

Working Procedure of Dijkstra



Min Priority Queue

push {node, distance} pair here

{2, 2}

{3, 1}

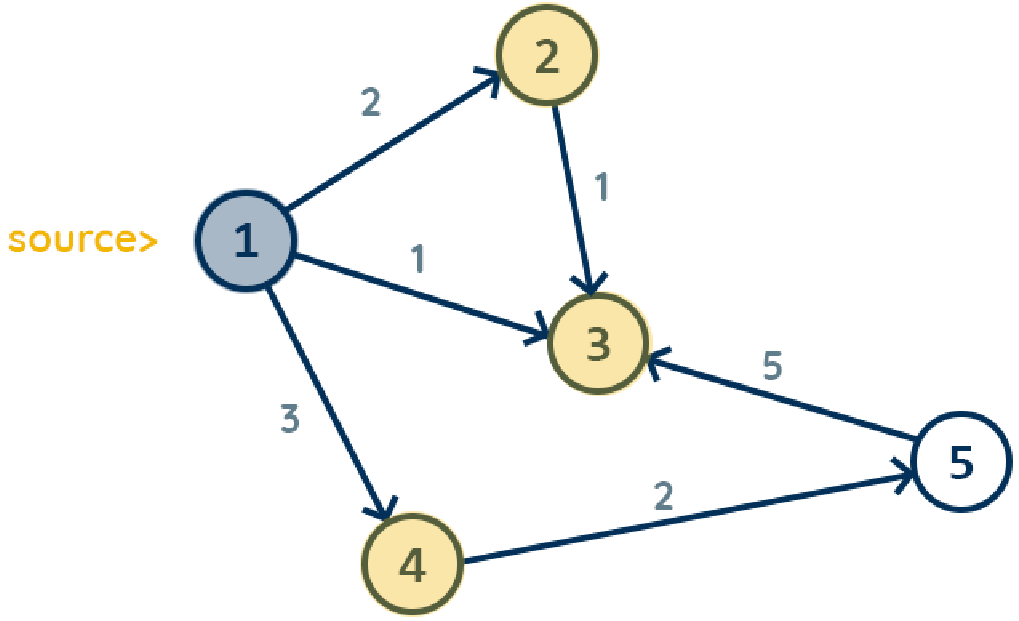
{4, 3}

Shortest Path

1 : 0

// pushed the neighbor node(s) into queue with path cost

Working Procedure of Dijkstra



Min Priority Queue

push {node, distance} pair here

{2, 2}

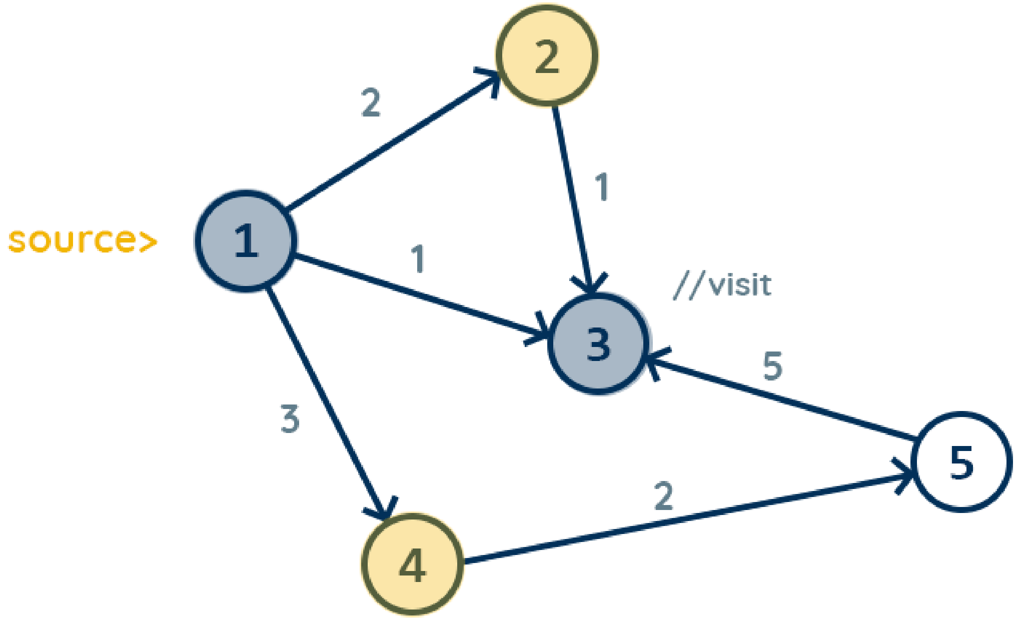
{3, 1} //popped

{4, 3}

Shortest Path

1 : 0

Working Procedure of Dijkstra



Min Priority Queue

push {node, distance} pair here

{2, 2}

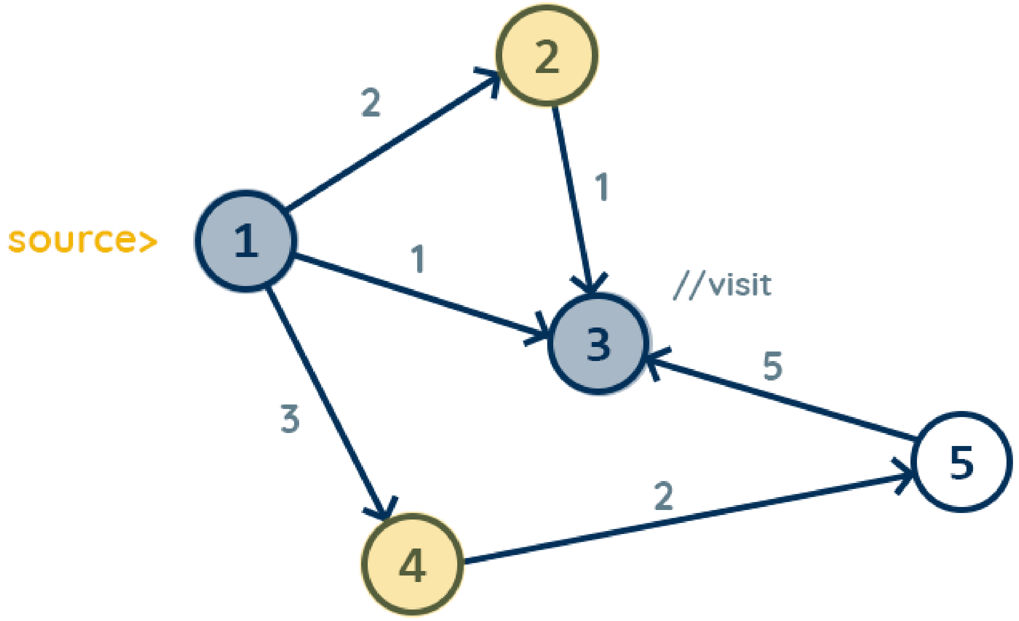
{3, 1} //popped

{4, 3}

Shortest Path

1	:	0

Working Procedure of Dijkstra



Min Priority Queue

push {node, distance} pair here

{2, 2}

{3, 1} //popped

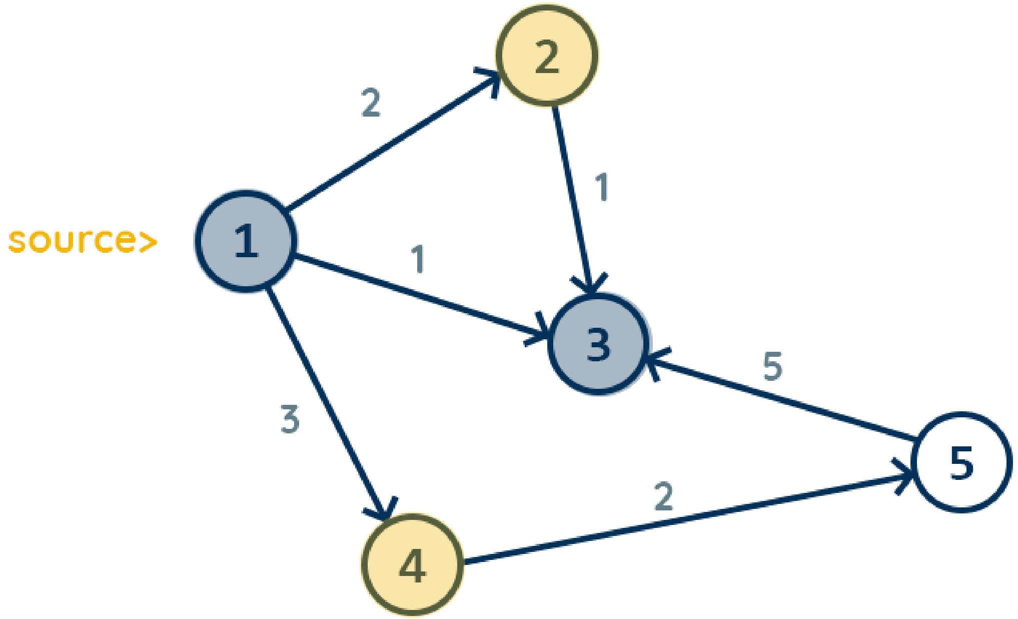
{4, 3}

Shortest Path

1	:	0
3	:	1

//set distance

Working Procedure of Dijkstra



Min Priority Queue

push {node, distance} pair here

{2, 2}

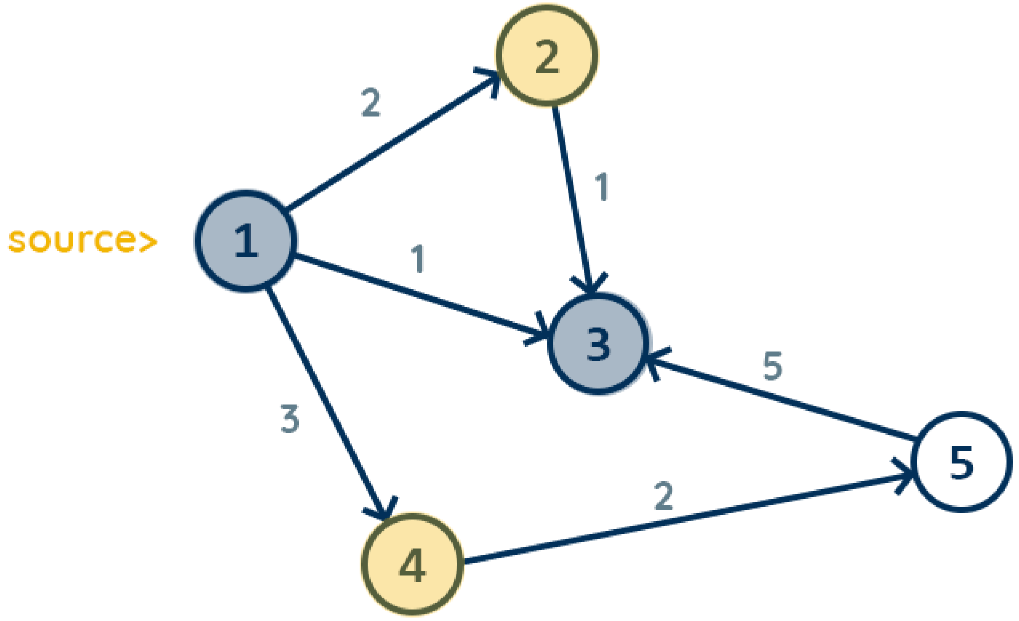
{4, 3}

Shortest Path

1 : 0

3 : 1

Working Procedure of Dijkstra



Min Priority Queue

push {node, distance} pair here

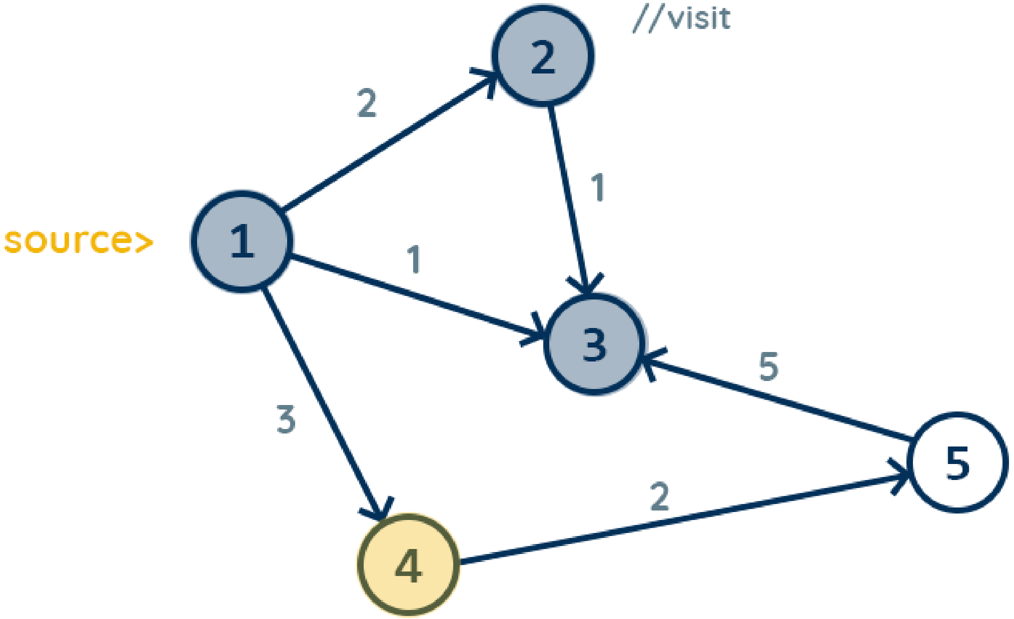
{2, 2} //popped

{4, 3}

Shortest Path

1	:	0
3	:	1

Working Procedure of Dijkstra



Min Priority Queue

push {node, distance} pair here

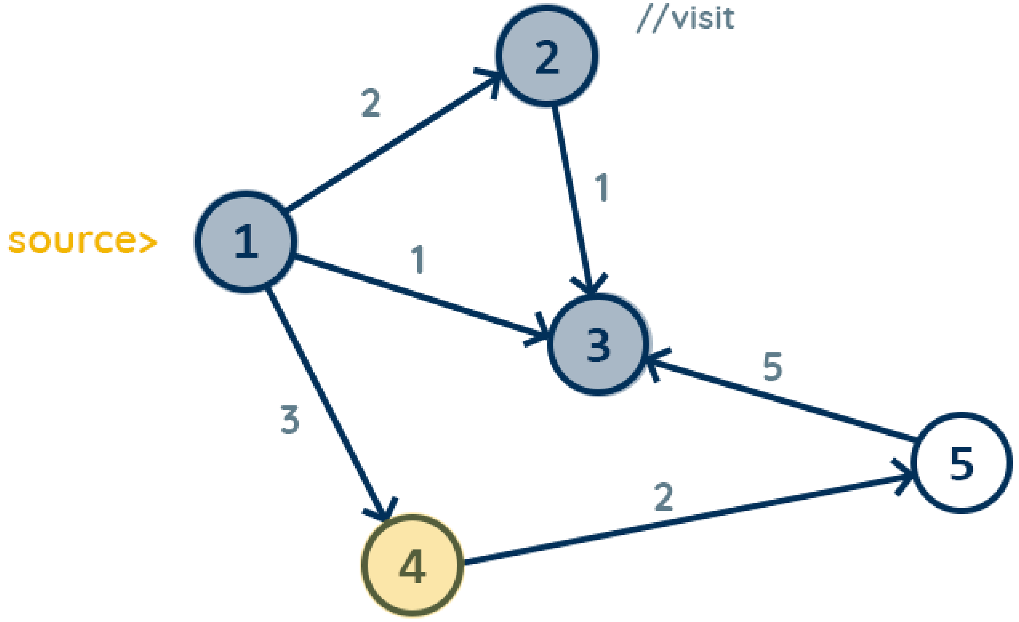
{2, 2} //popped

{4, 3}

Shortest Path

1	:	0
3	:	1

Working Procedure of Dijkstra



Min Priority Queue

push {node, distance} pair here

{2, 2} //popped

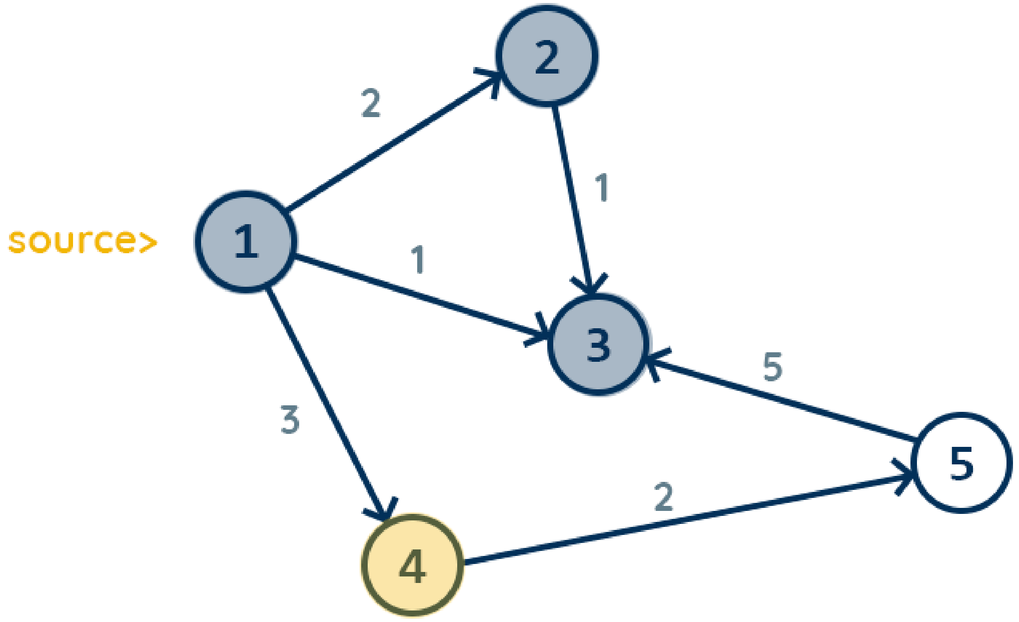
{4, 3}

Shortest Path

1	:	0
3	:	1
2	:	2

//set distance

Working Procedure of Dijkstra



Min Priority Queue

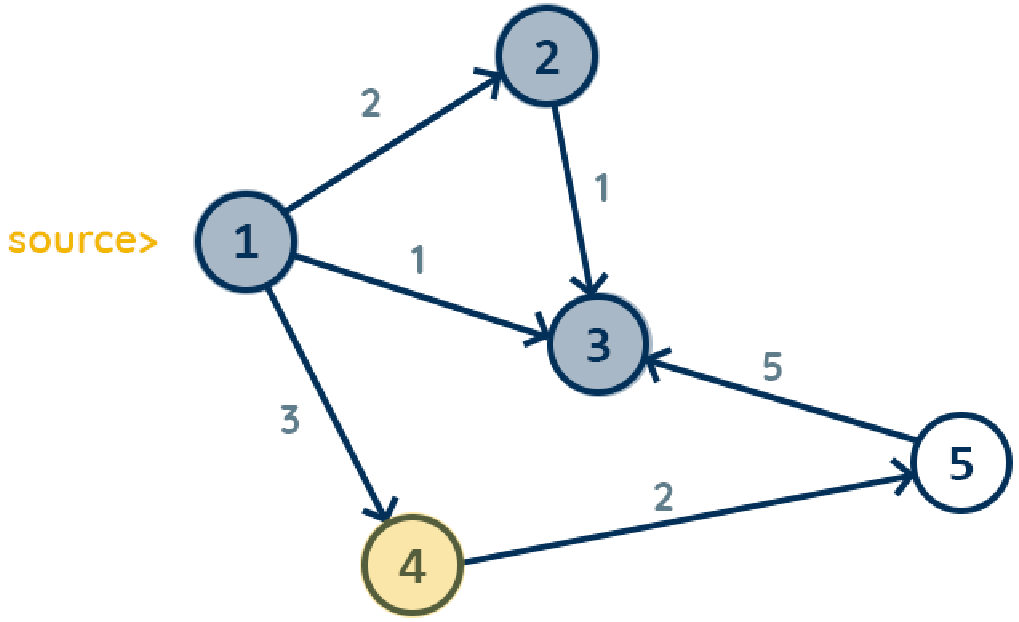
push {node, distance} pair here

{4, 3}

Shortest Path

1	:	0
3	:	1
2	:	2

Working Procedure of Dijkstra



Min Priority Queue

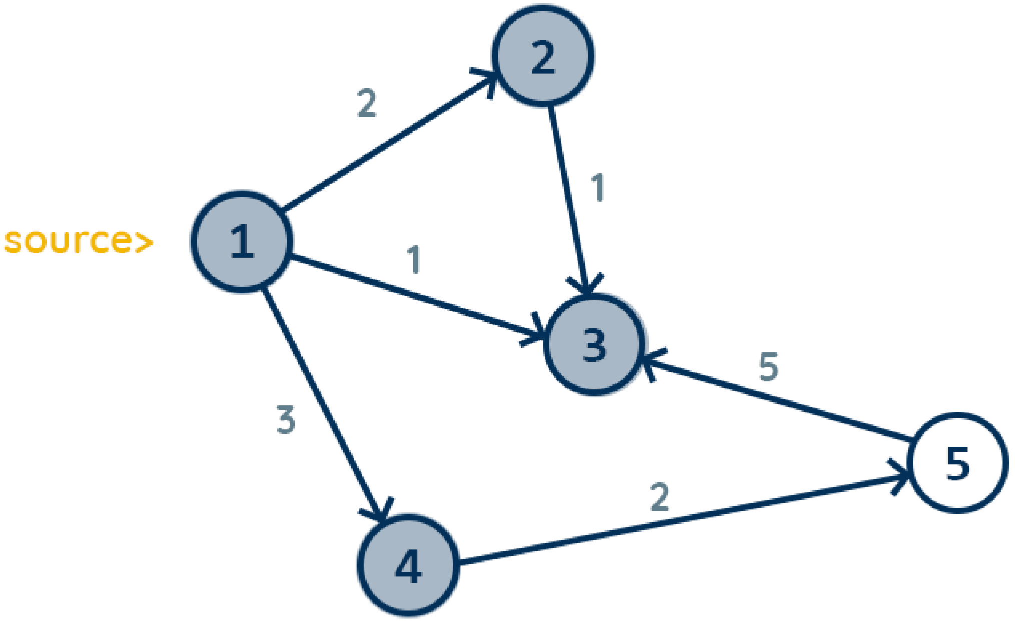
push {node, distance} pair here

{4, 3} //popped

Shortest Path

1	:	0
3	:	1
2	:	2

Working Procedure of Dijkstra



//visit

Min Priority Queue

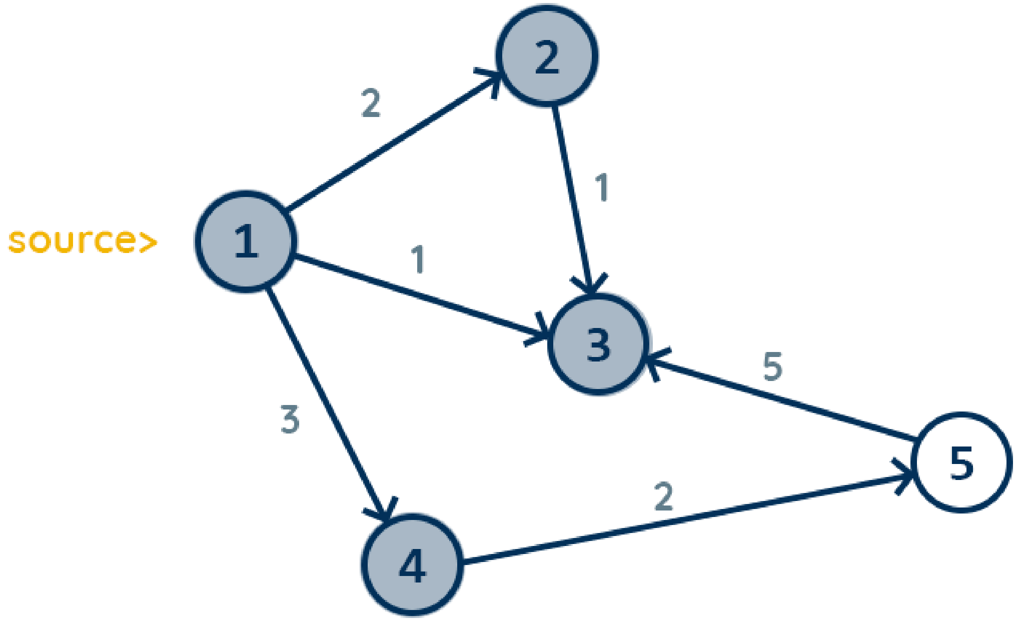
push {node, distance} pair here

{4, 3} //popped

Shortest Path

1	:	0
3	:	1
2	:	2

Working Procedure of Dijkstra



//visit

Min Priority Queue

push {node, distance} pair here

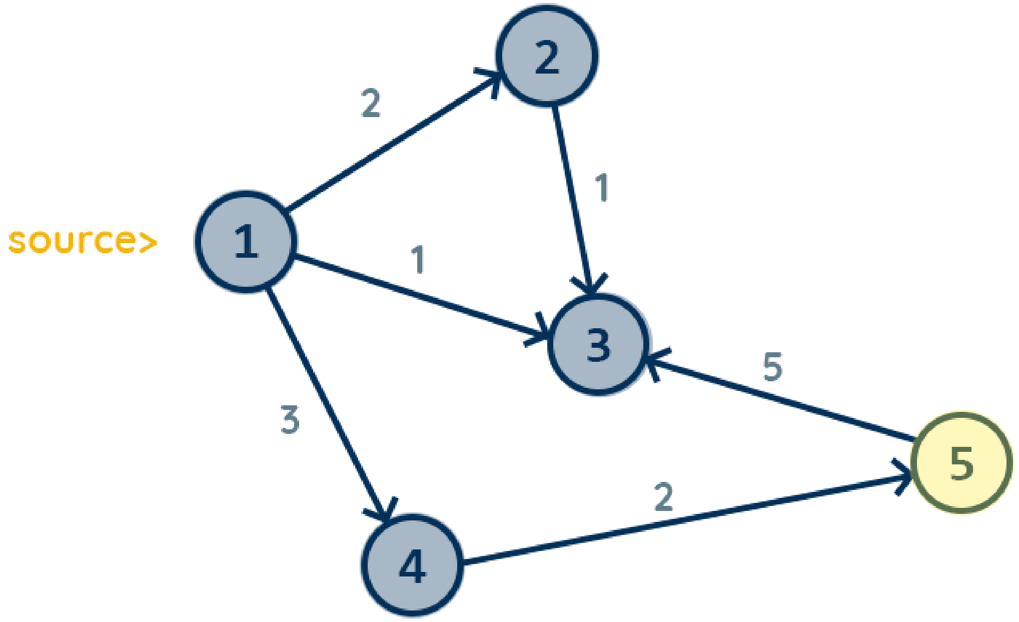
{4, 3} //popped

Shortest Path

1	:	0
3	:	1
2	:	2
4	:	3

//set distance

Working Procedure of Dijkstra



// pushed the neighbor node(s)
into queue with path cost

Min Priority Queue

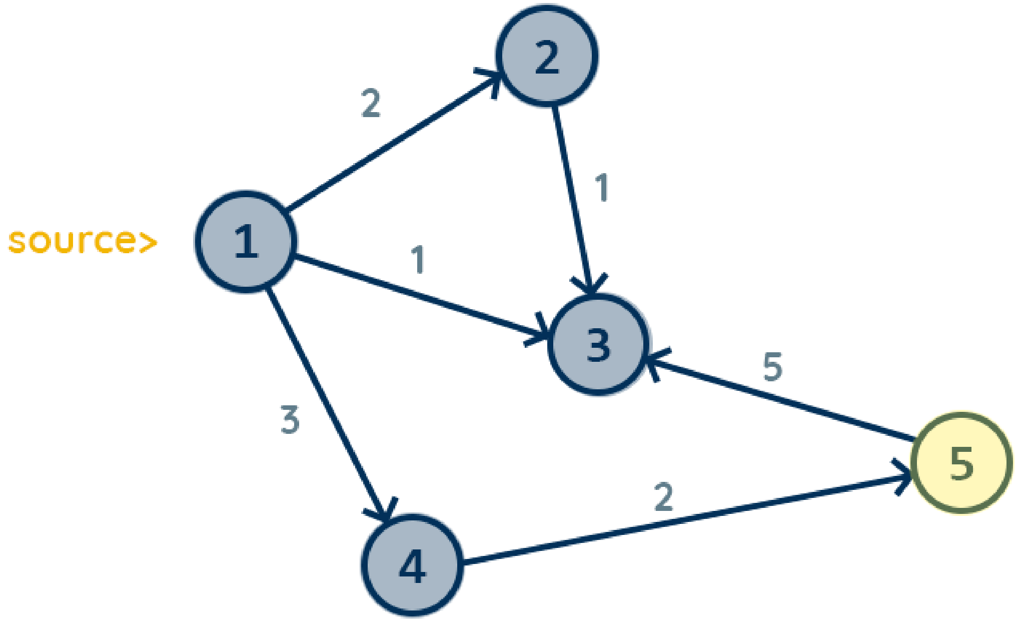
push {node, distance} pair here

{5, 5}

Shortest Path

1	:	0
3	:	1
2	:	2
4	:	3

Working Procedure of Dijkstra



Min Priority Queue

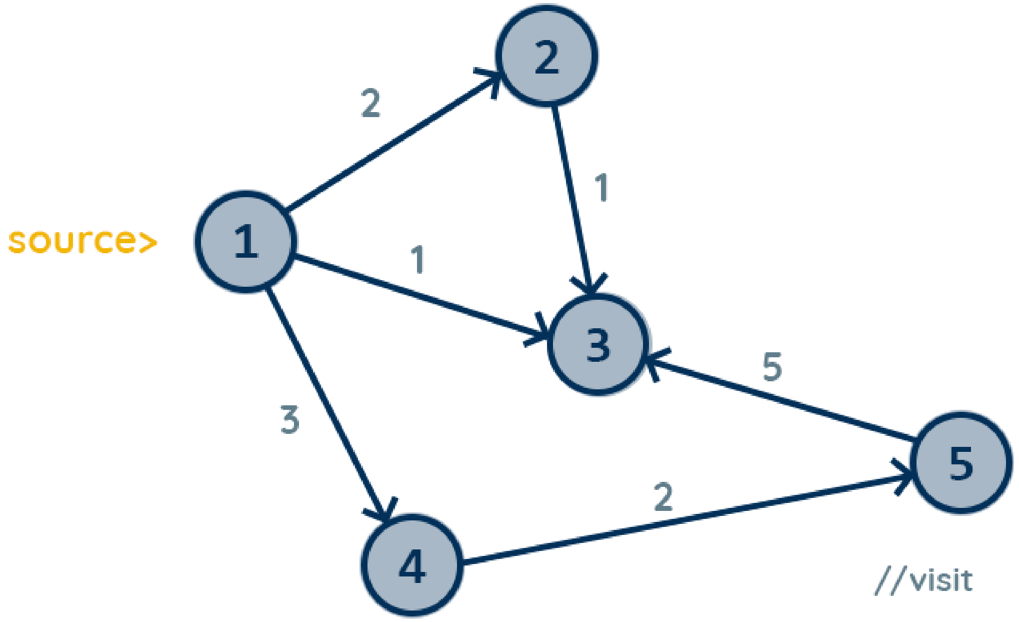
push {node, distance} pair here

{5, 5} //popped

Shortest Path

1	:	0
3	:	1
2	:	2
4	:	3

Working Procedure of Dijkstra



Min Priority Queue

push {node, distance} pair here

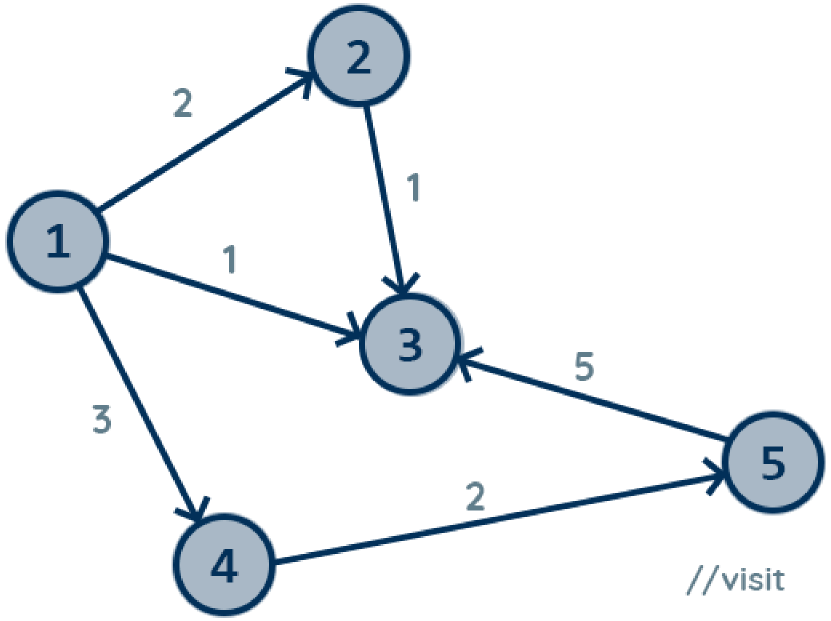
{5, 5} //popped

Shortest Path

1	:	0
3	:	1
2	:	2
4	:	3

Working Procedure of Dijkstra

source>



//visit

Min Priority Queue

push {node, distance} pair here

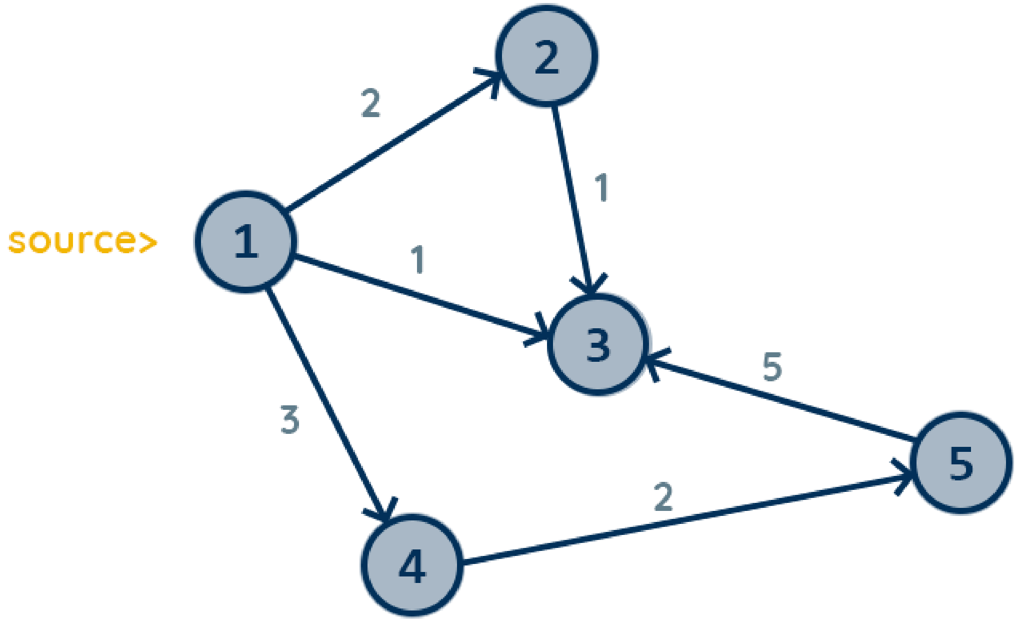
{5, 5} //popped

Shortest Path

1	:	0
3	:	1
2	:	2
4	:	3
5	:	5

//set distance

Working Procedure of Dijkstra



Min Priority Queue

push {node, distance} pair here

Shortest Path

1	:	0
3	:	1
2	:	2
4	:	3
5	:	5

answer

Dijkstra's Assumption on Shortest Path

Dijkstra’s Assumption on Shortest Path

Adding an edge can never make a path shorter

Dijkstra’s Assumption on Shortest Path

Adding an edge can never make a path shorter

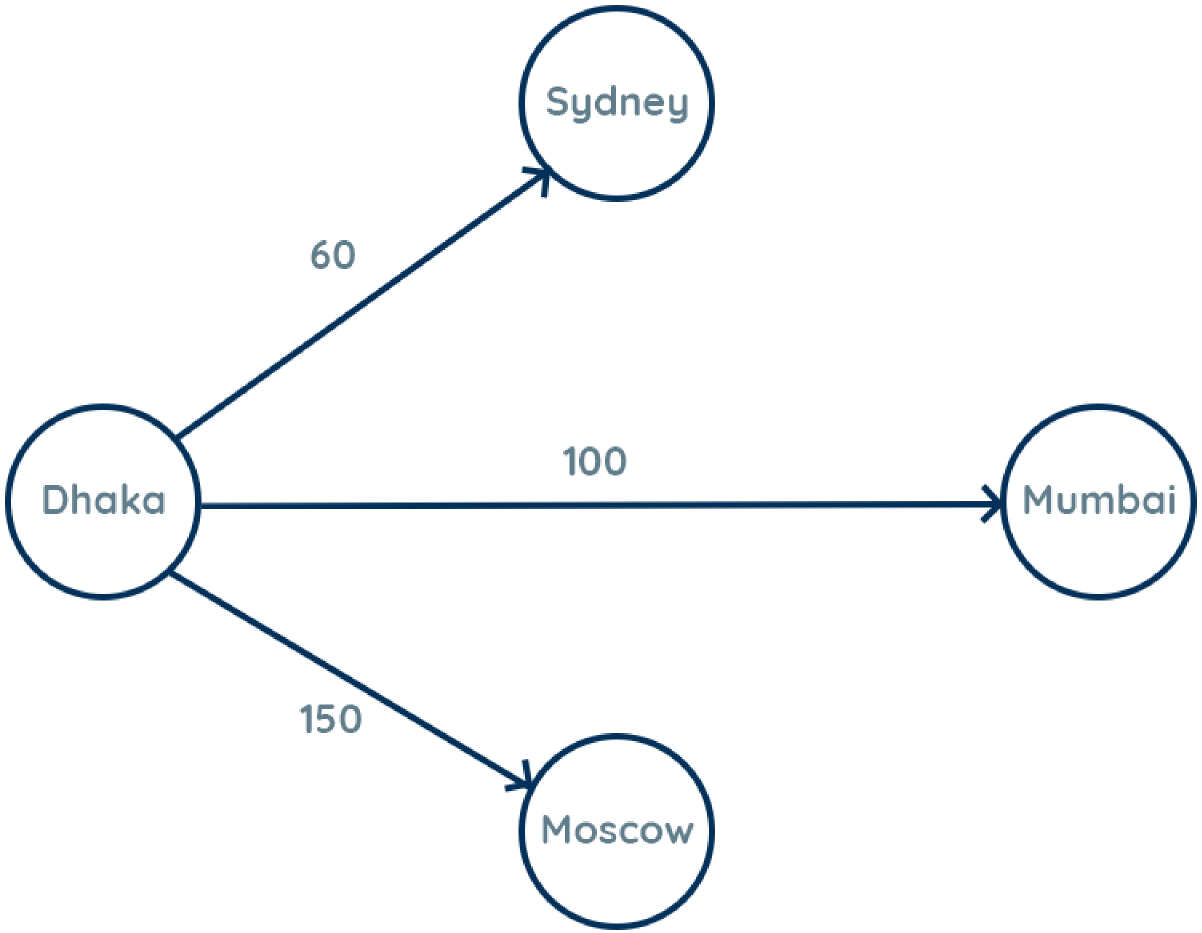
Dijkstra's Assumption on Shortest Path

Adding an edge can never make a path shorter

Dhaka
Mumbai

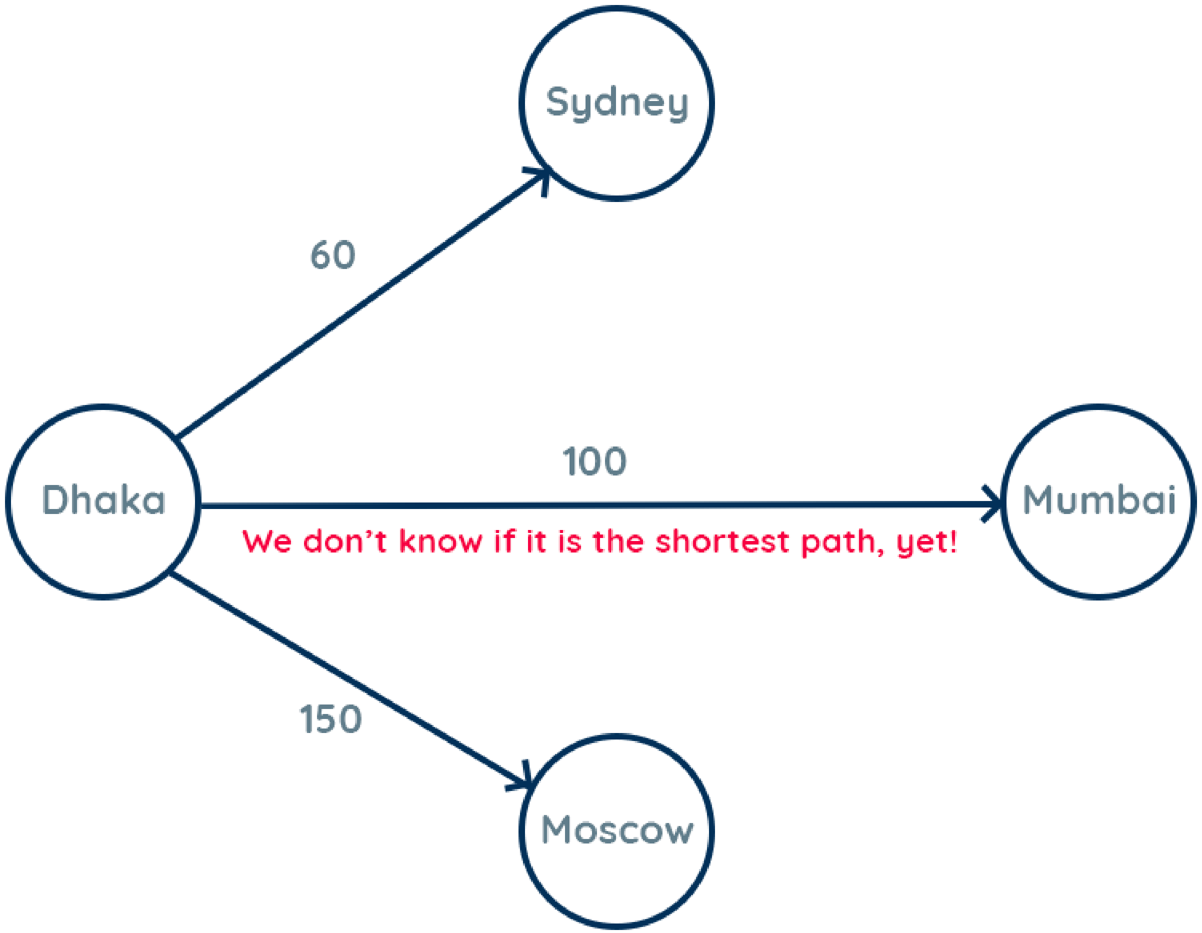
Dijkstra’s Assumption on Shortest Path

Adding an edge can never make a path shorter



Dijkstra’s Assumption on Shortest Path

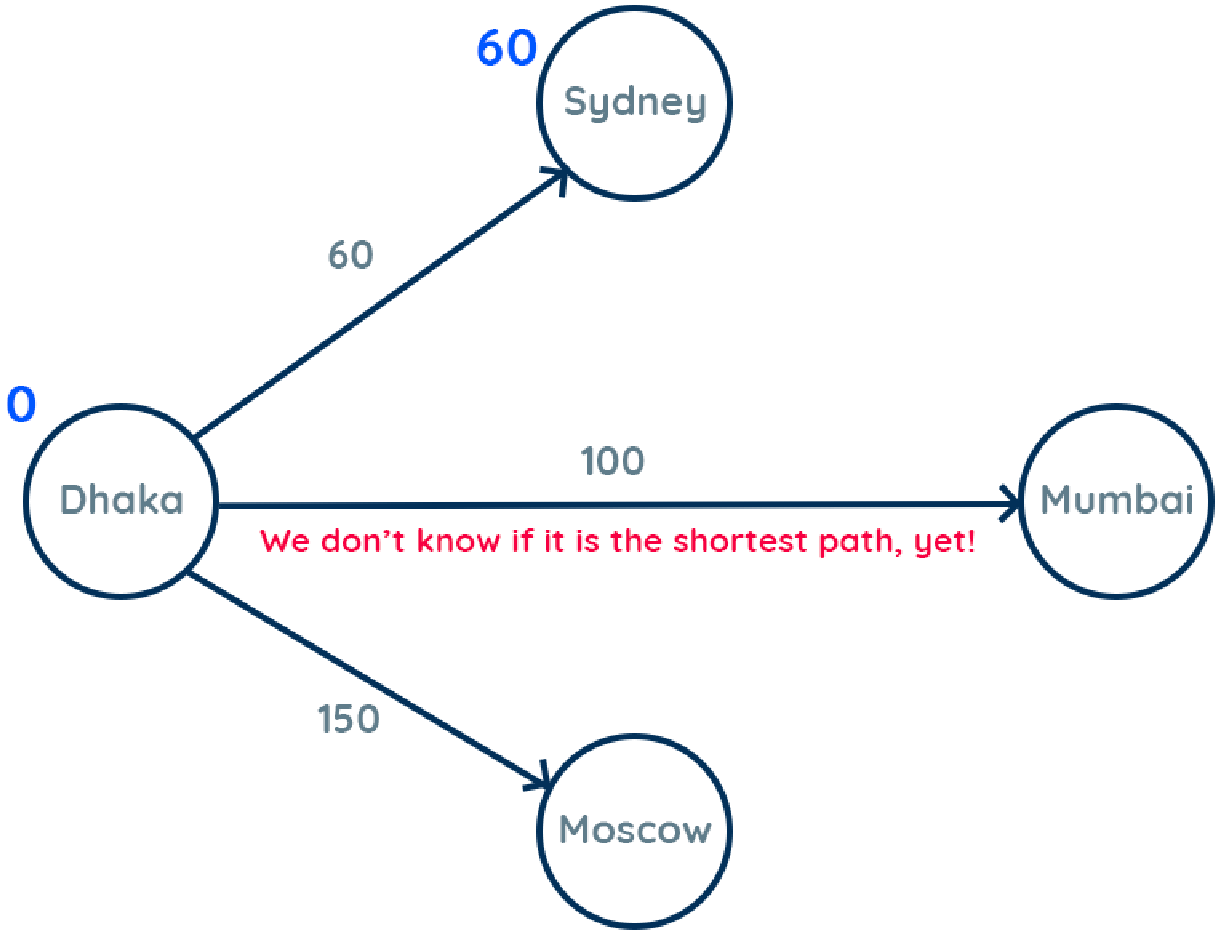
Adding an edge can never make a path shorter



We don’t know if it is the shortest path, yet!

Dijkstra's Assumption on Shortest Path

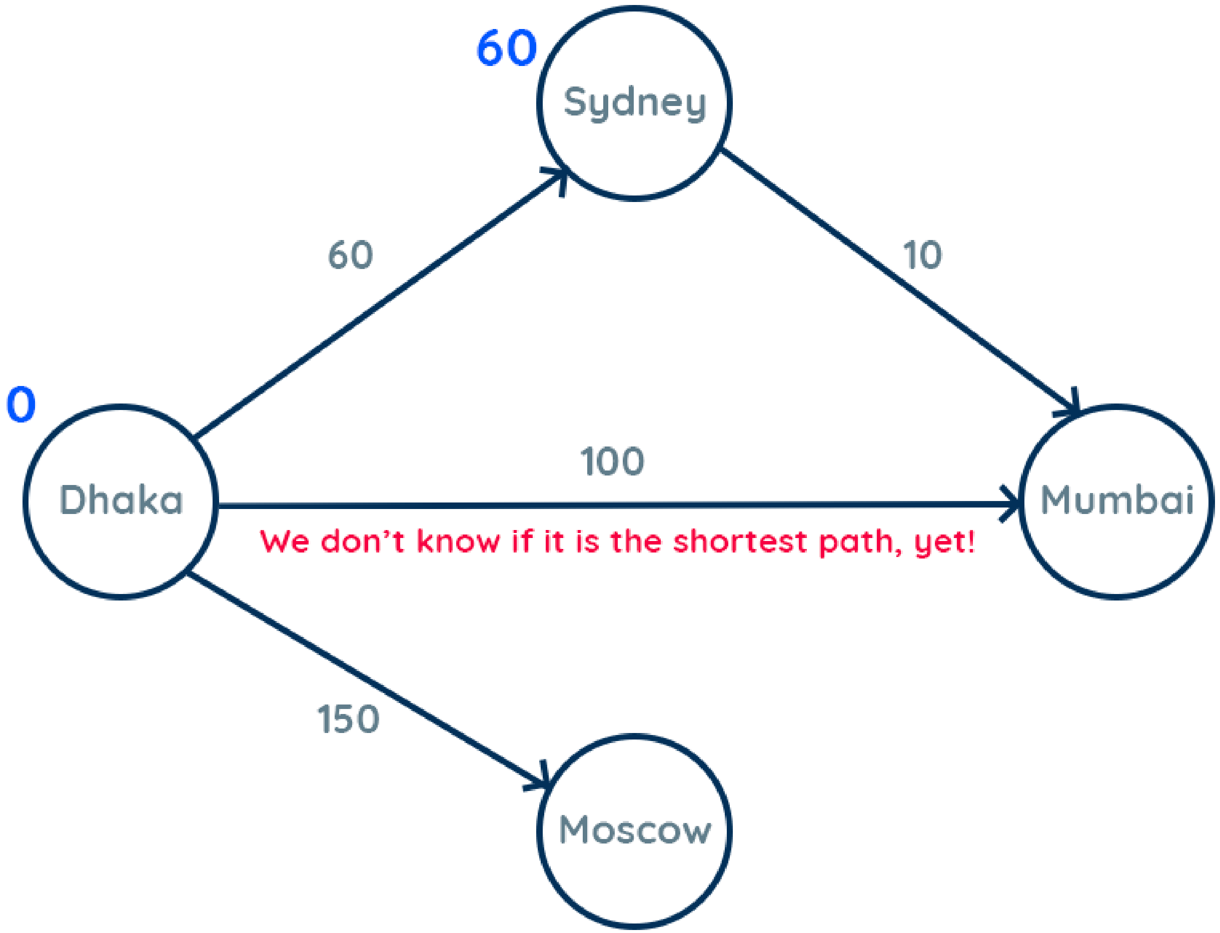
Adding an edge can never make a path shorter



We don't know if it is the shortest path, yet!

Dijkstra’s Assumption on Shortest Path

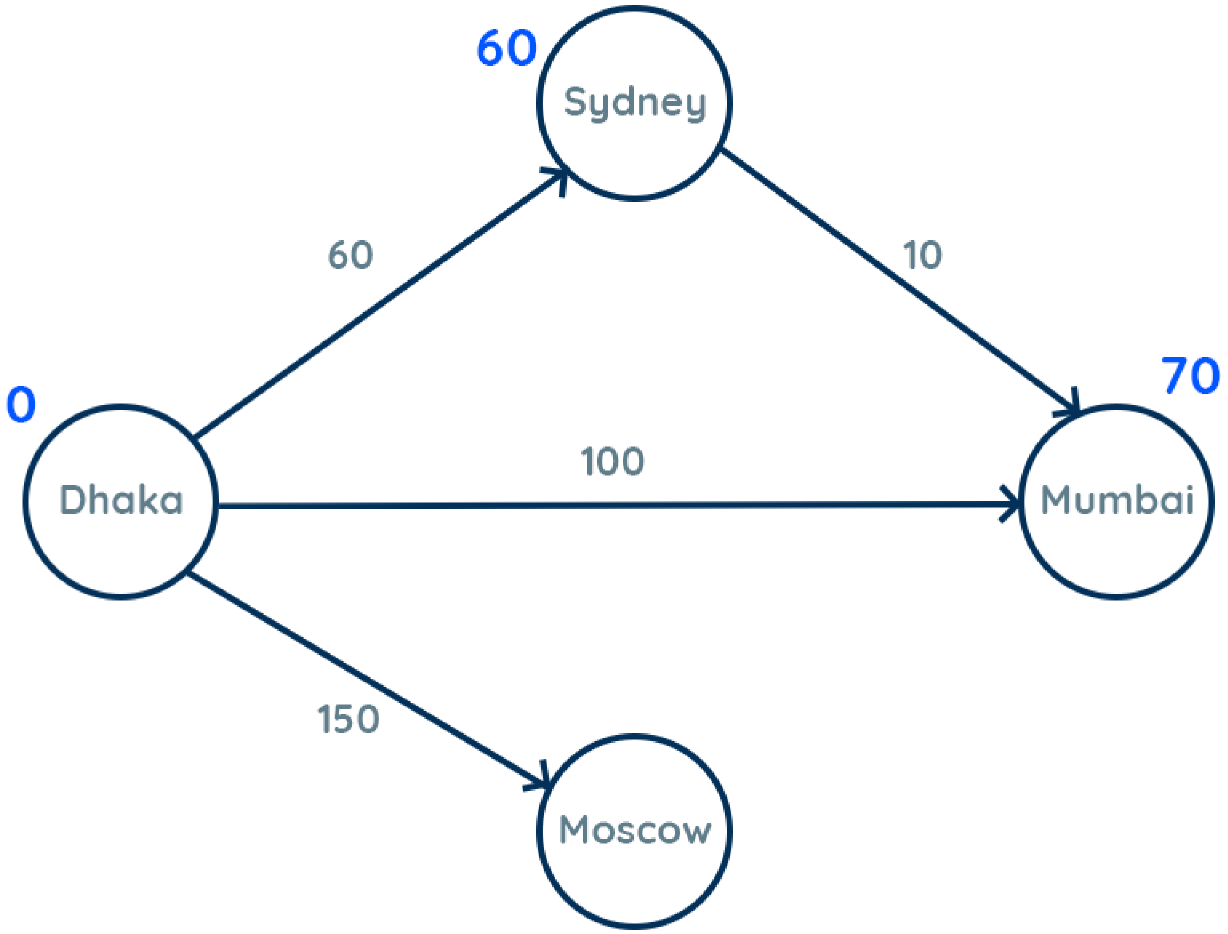
Adding an edge can never make a path shorter



We don’t know if it is the shortest path, yet!

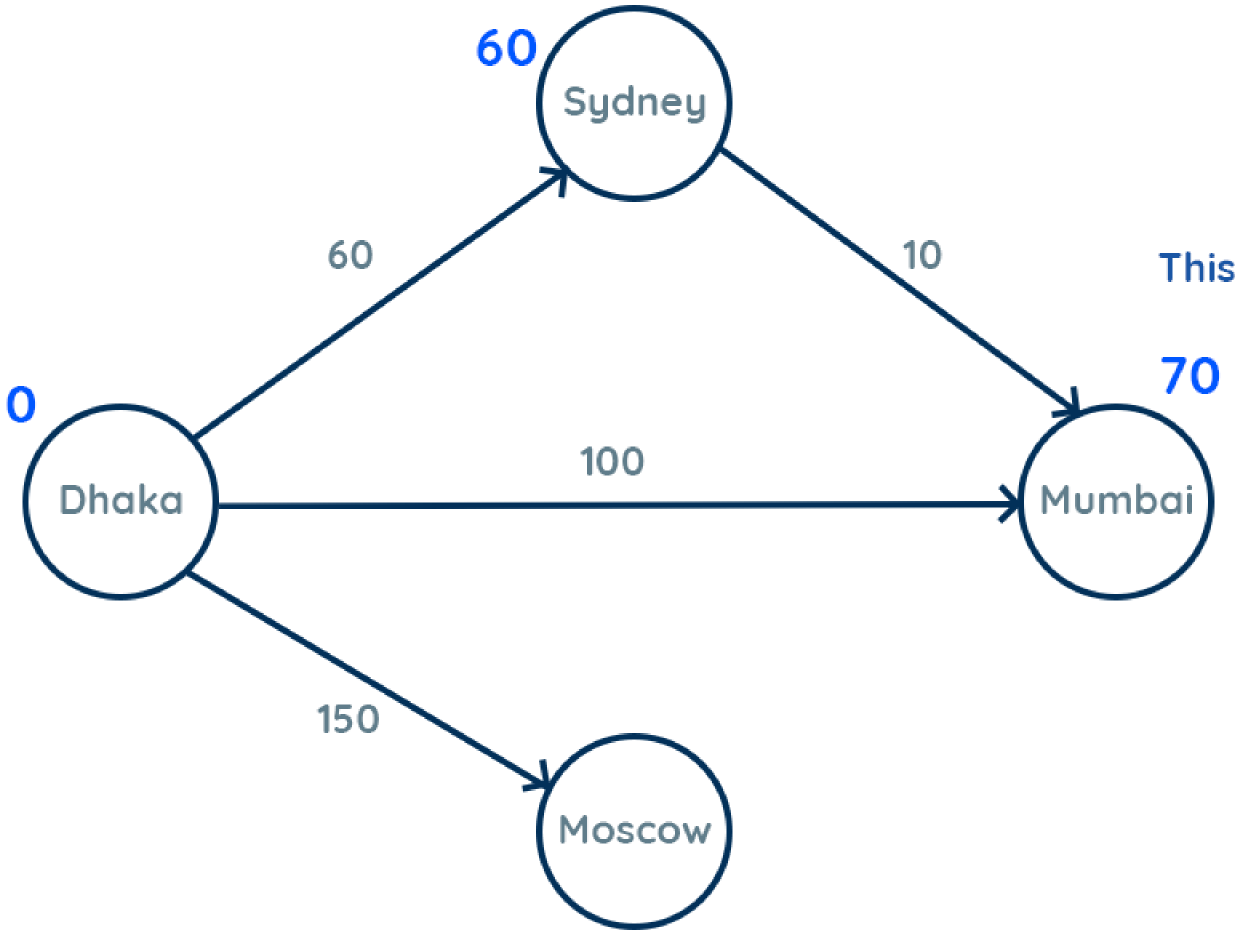
Dijkstra’s Assumption on Shortest Path

Adding an edge can never make a path shorter



Dijkstra’s Assumption on Shortest Path

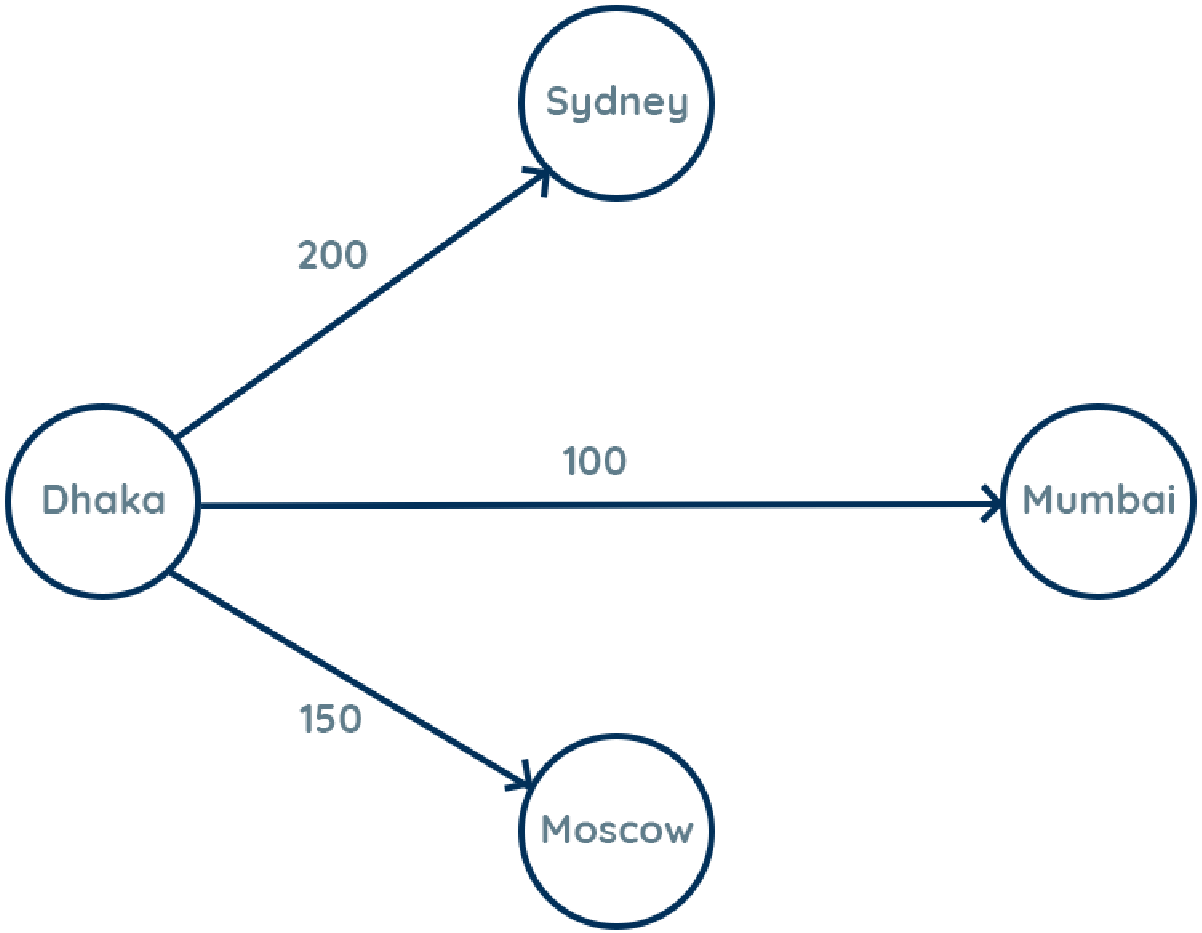
Adding an edge can never make a path shorter



This is the shortest cost!

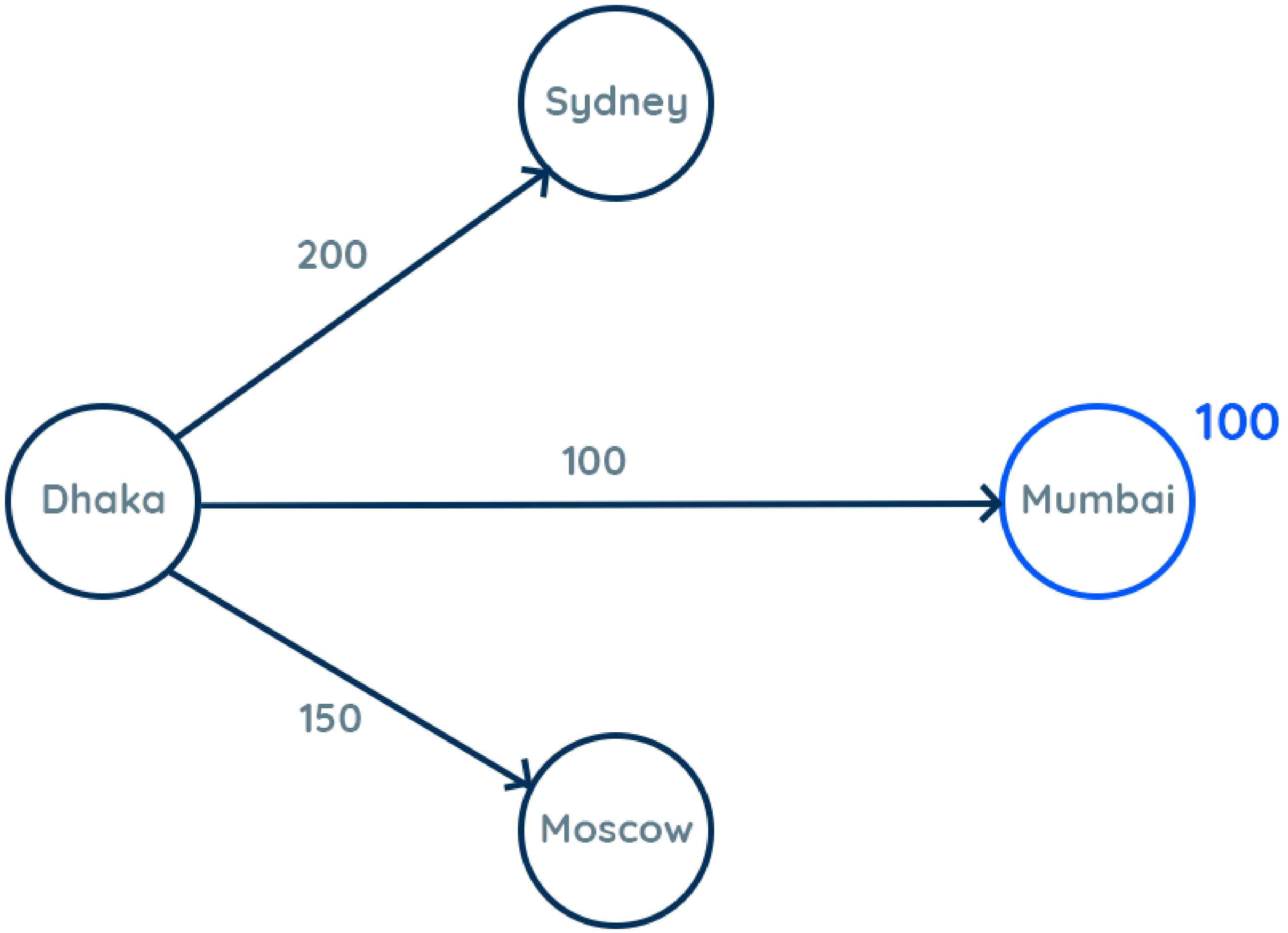
Dijkstra's Assumption on Shortest Path

Adding an edge can never make a path shorter



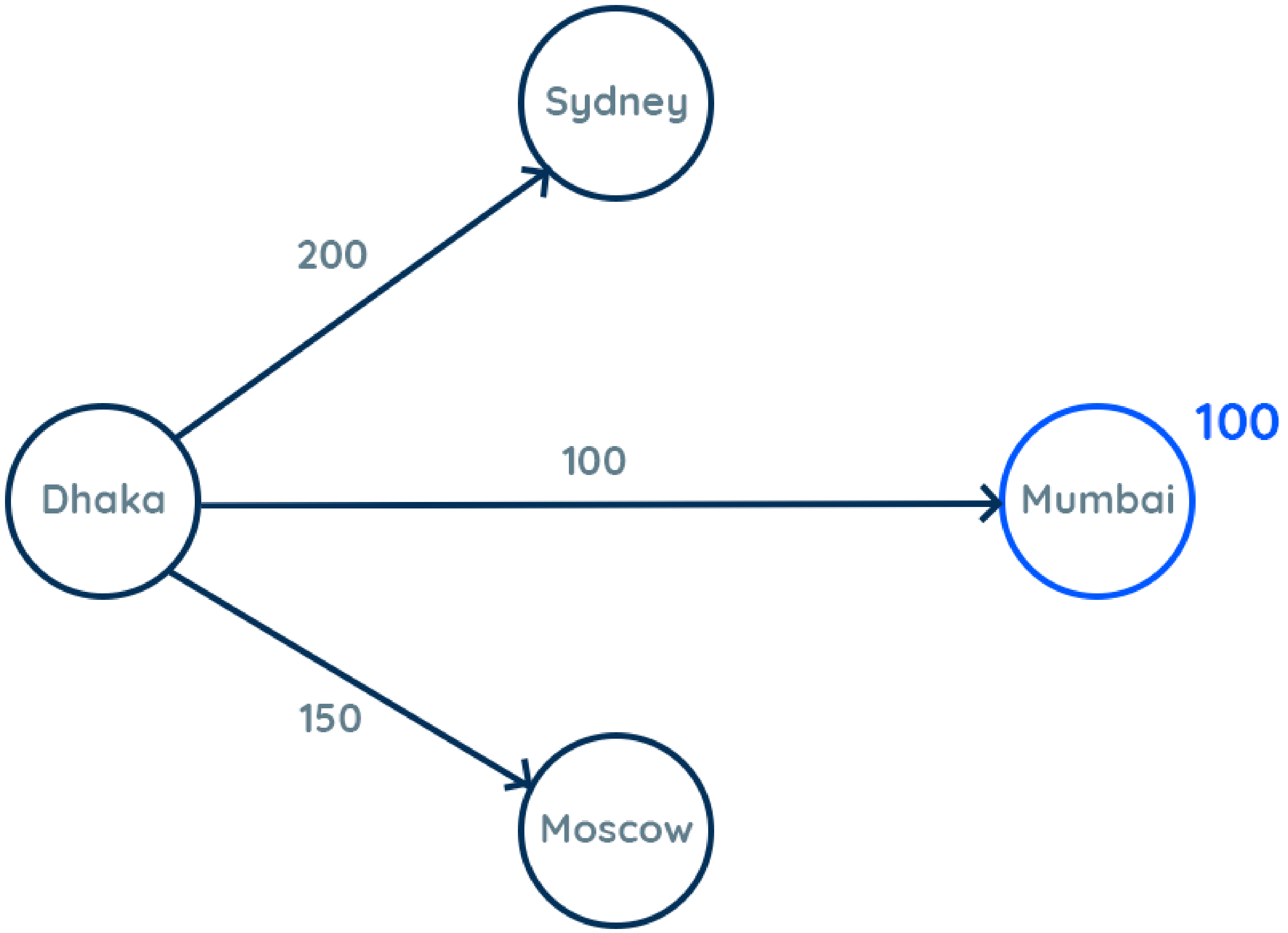
Dijkstra’s Assumption on Shortest Path

Adding an edge can never make a path shorter



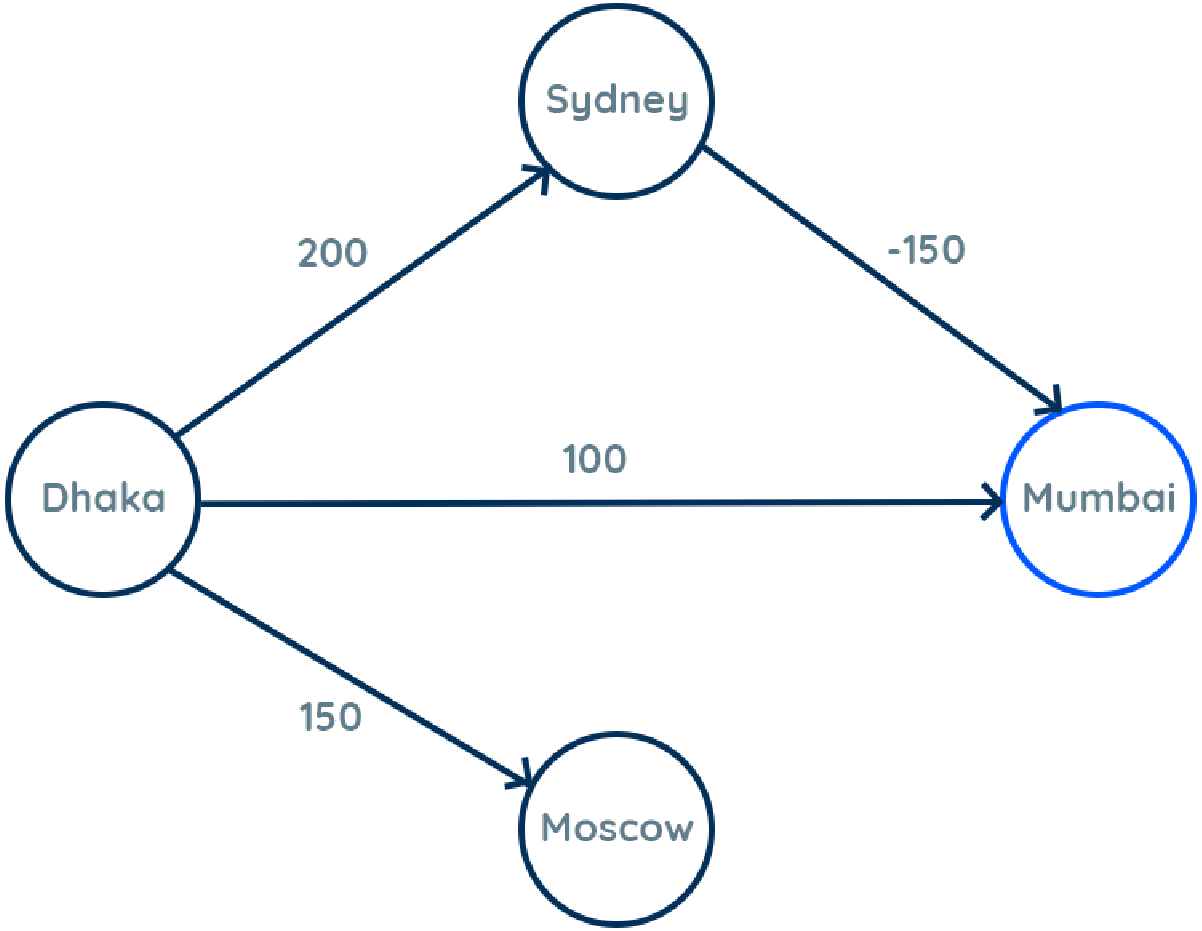
Problem With Negative Edge Cost

Dijkstra’s assumption fails here



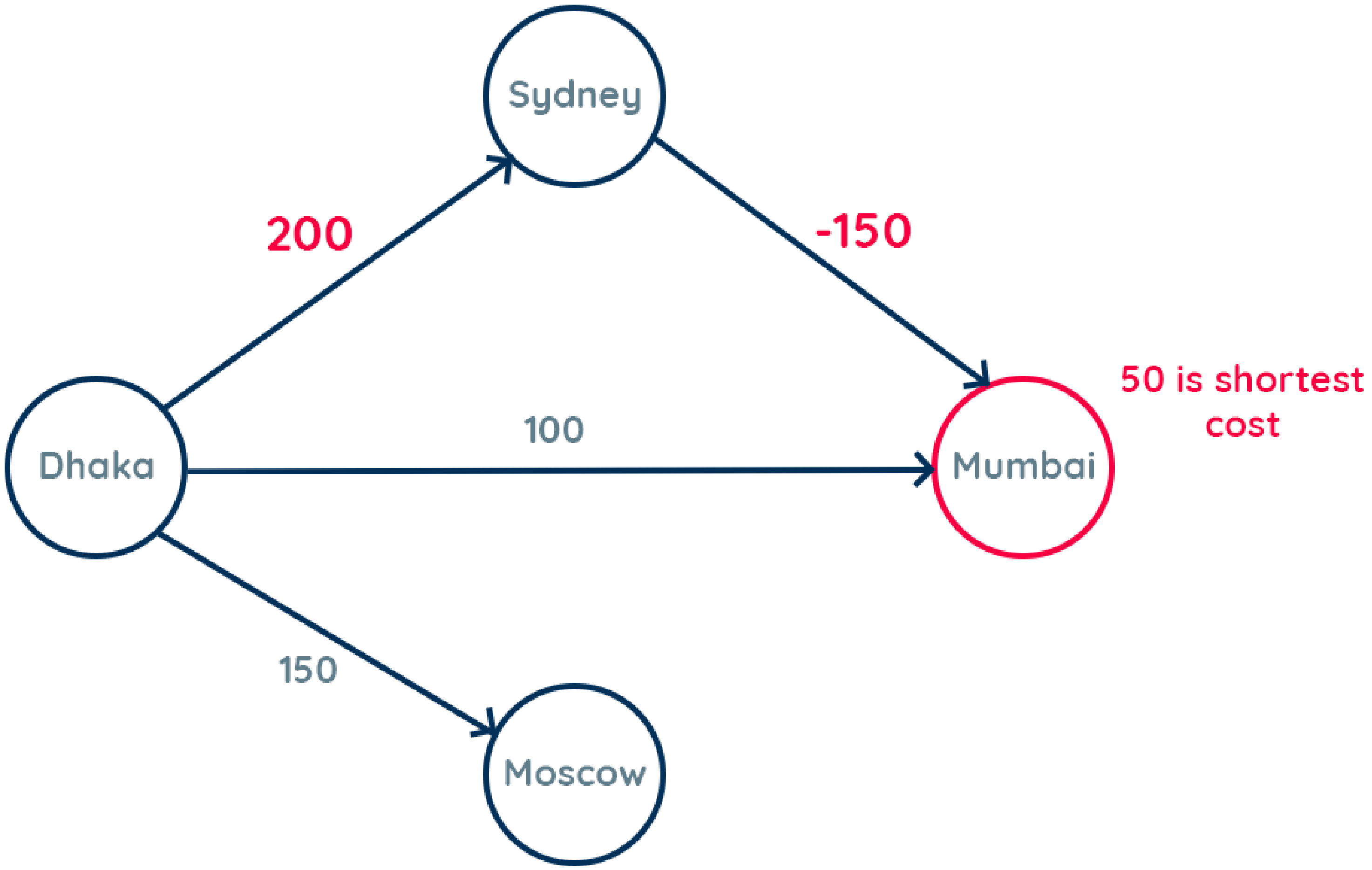
Problem With Negative Edge Cost

Dijkstra’s assumption fails here



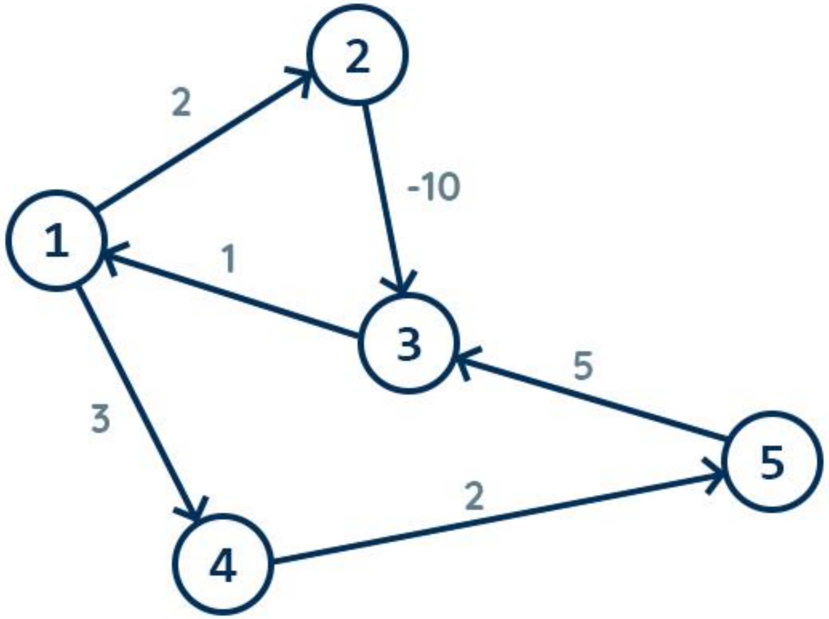
Problem With Negative Edge Cost

Dijkstra's assumption fails here



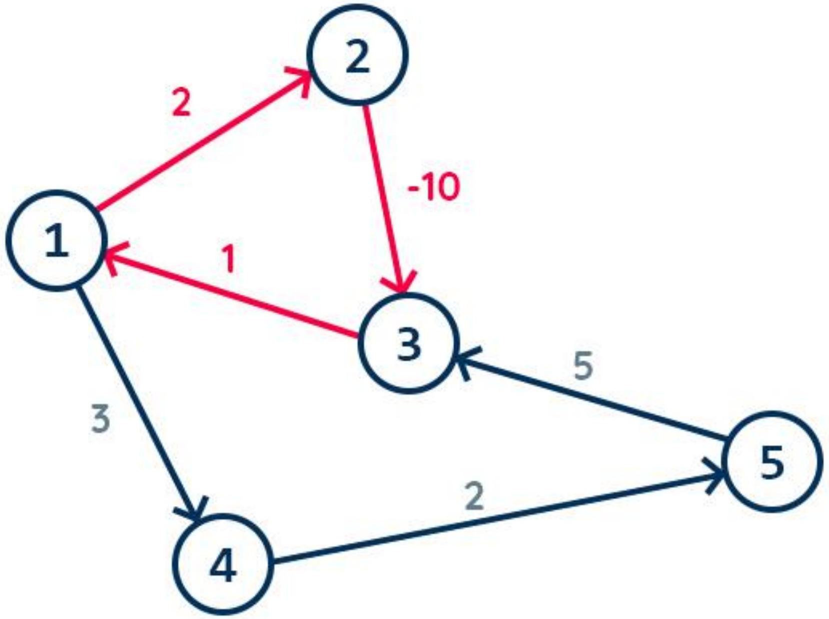
Negative Cycle

Negative Cycle



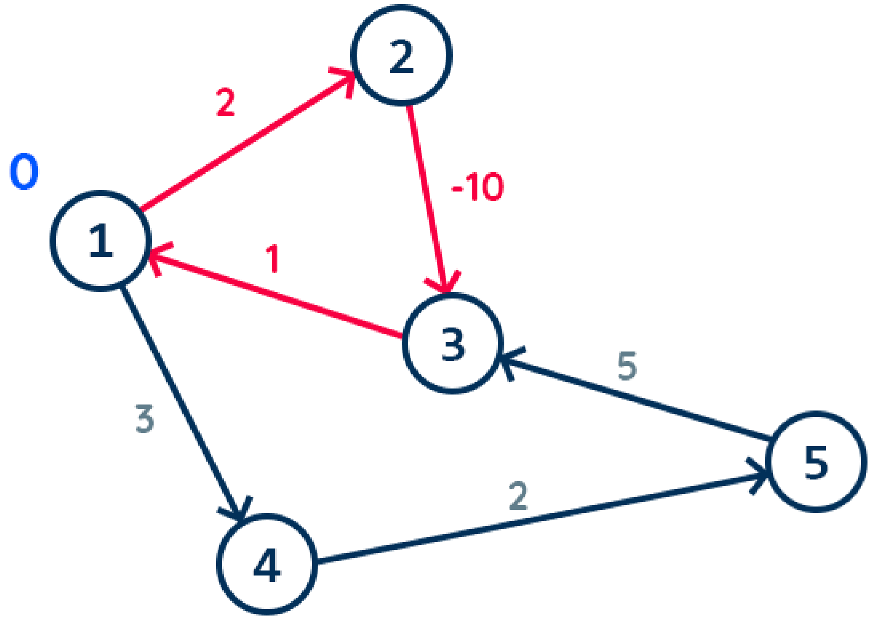
Negative Cycle

Negative cycle exists!



Negative Cycle

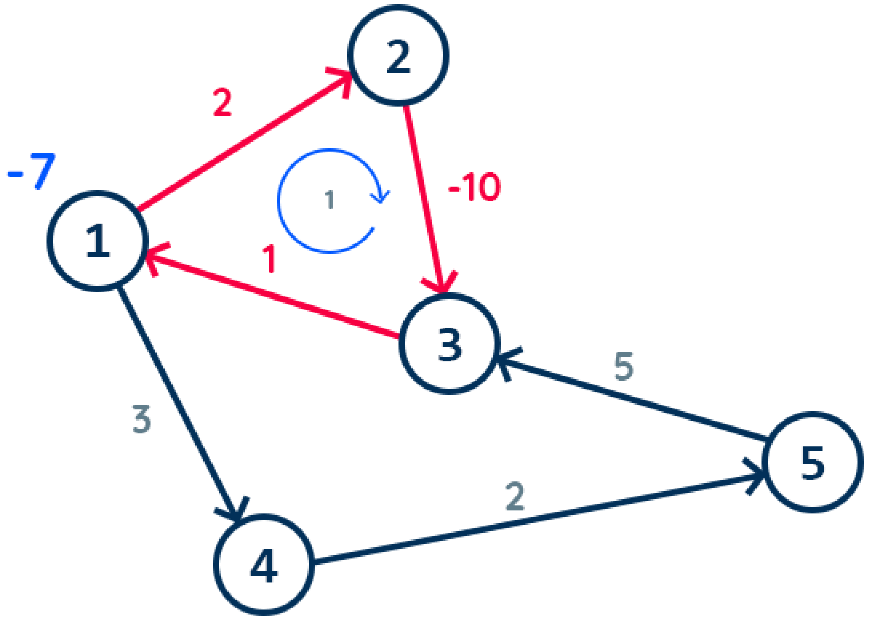
Negative cycle exists!



// Let's consider node 1 as source

Negative Cycle

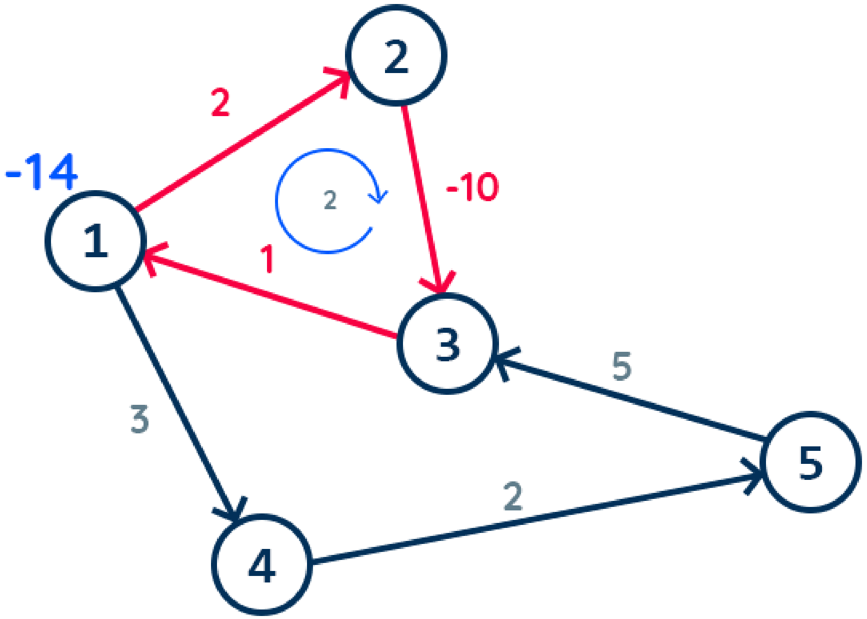
Negative cycle exists!



// Let’s consider node 1 as source

Negative Cycle

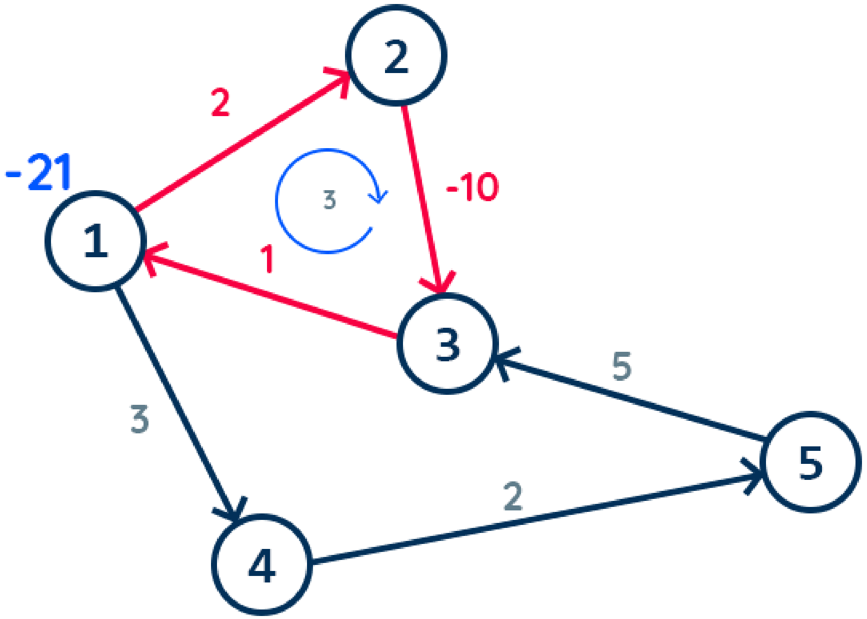
Negative cycle exists!



// Let's consider node 1 as source

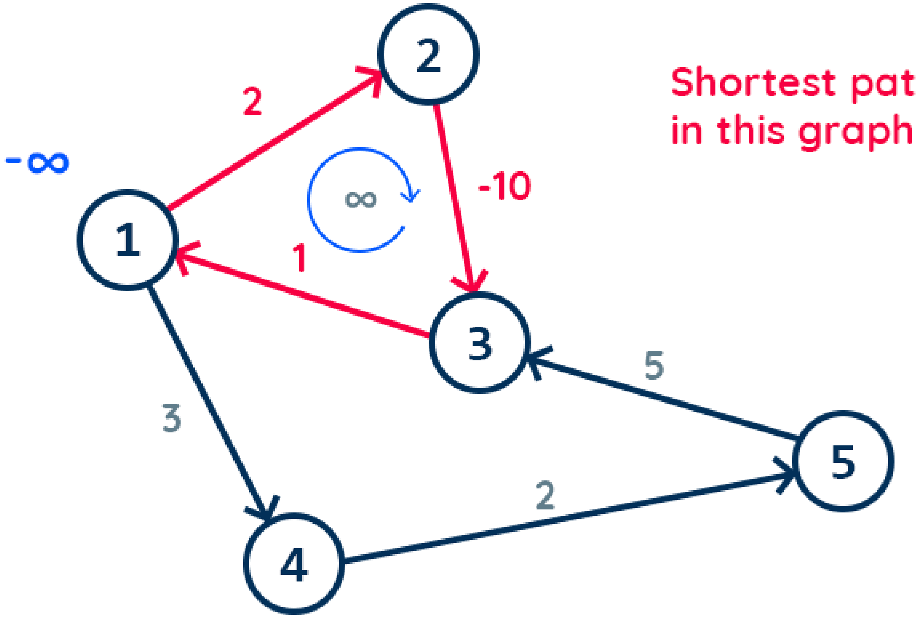
Negative Cycle

Negative cycle exists!



// Let's consider node 1 as source

Negative Cycle

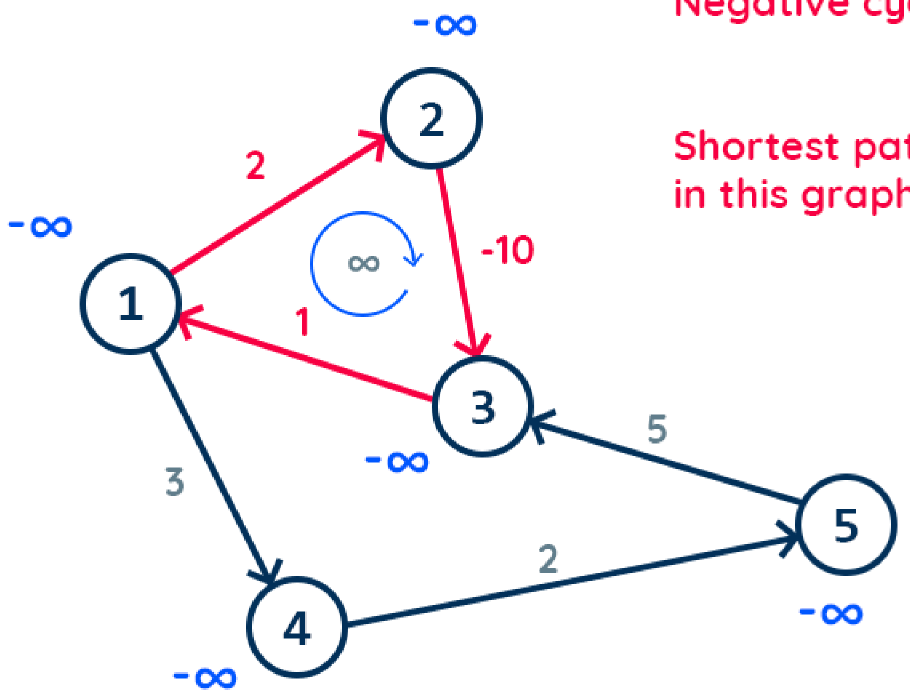


Negative cycle exists!

Shortest path is undefined in this graph.

// Let's consider node 1 as source

Negative Cycle



Negative cycle exists!

Shortest path is undefined in this graph.

// Let's consider node 1 as source