Breadth First Search (C++)

And finding Shortest Path with it

#csspree Online

Breadth First search

Breadth First search

Breadth First Search is a graph searching algorithm that:

Breadth First search

Breadth First Search is a graph searching algorithm that:

1. Visits nodes level-wise. That is all level n nodes will be explored before it moves on to level n+1 nodes.

Breadth First search

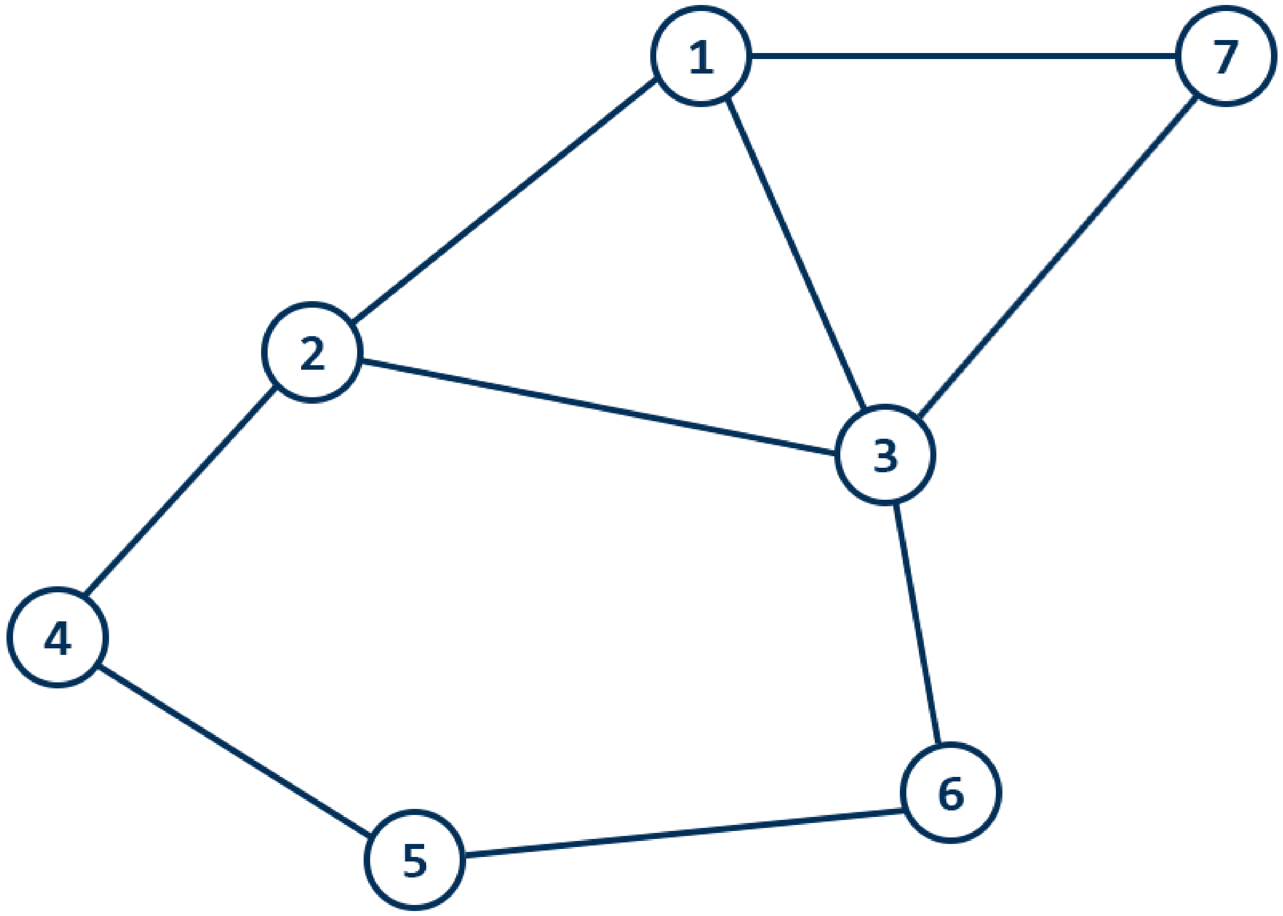
Breadth First Search is a graph searching algorithm that:

1. Visits nodes level-wise. That is all level n nodes will be explored before it moves on to level n+1 nodes.

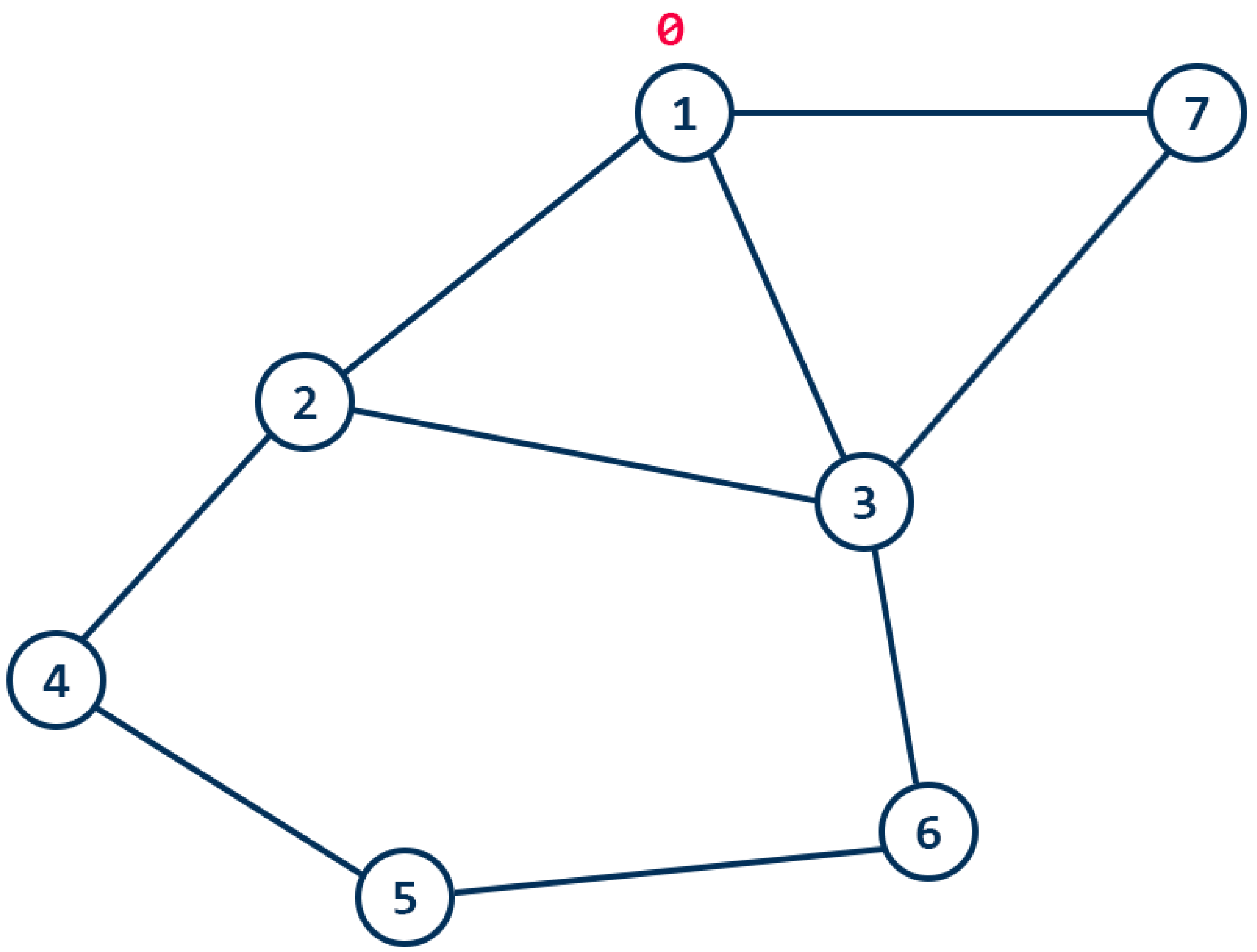
2. Finds shortest path to every nodes given that the edge weight is constant for entire graph.

Levels in a Graph:

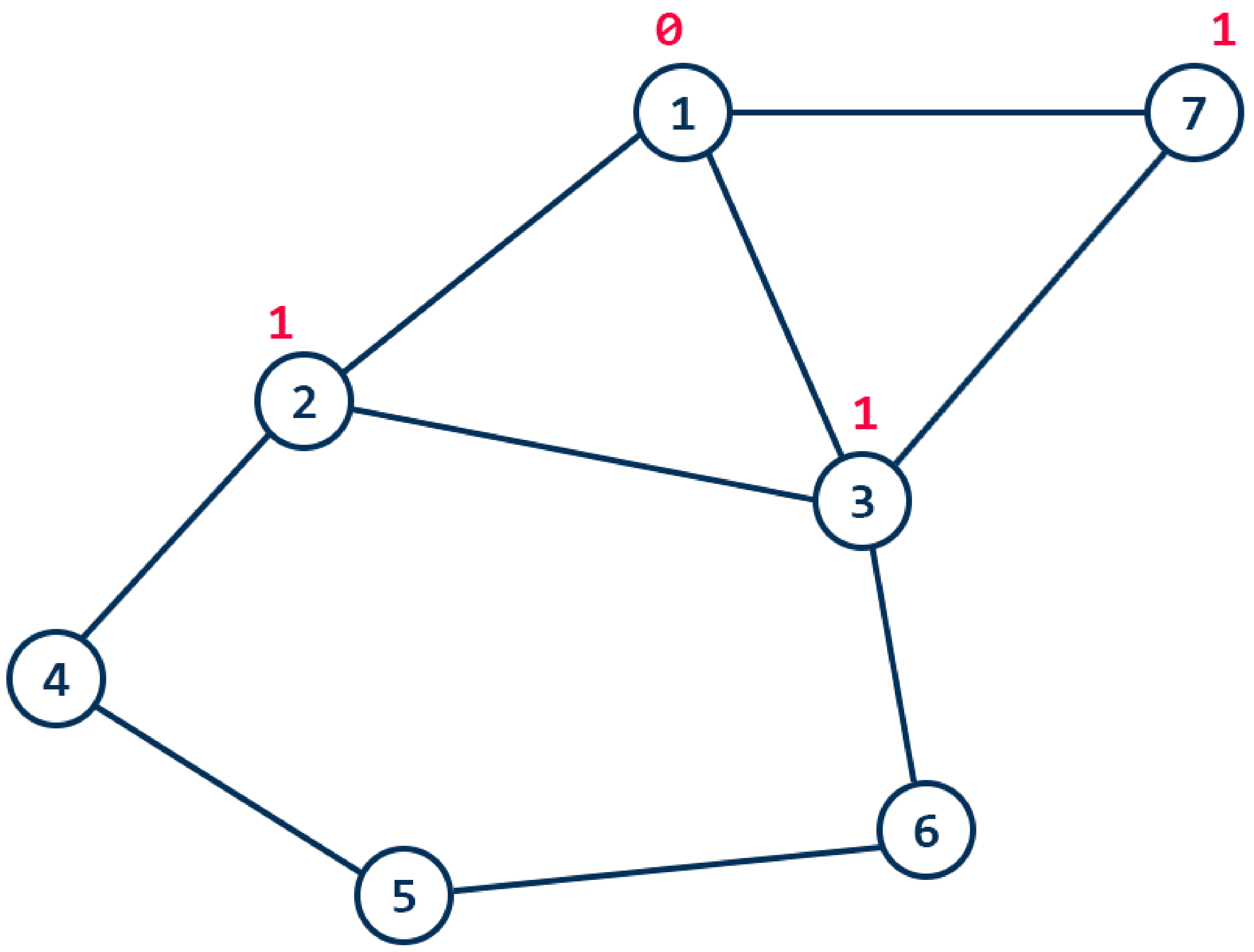
Levels in a Graph:



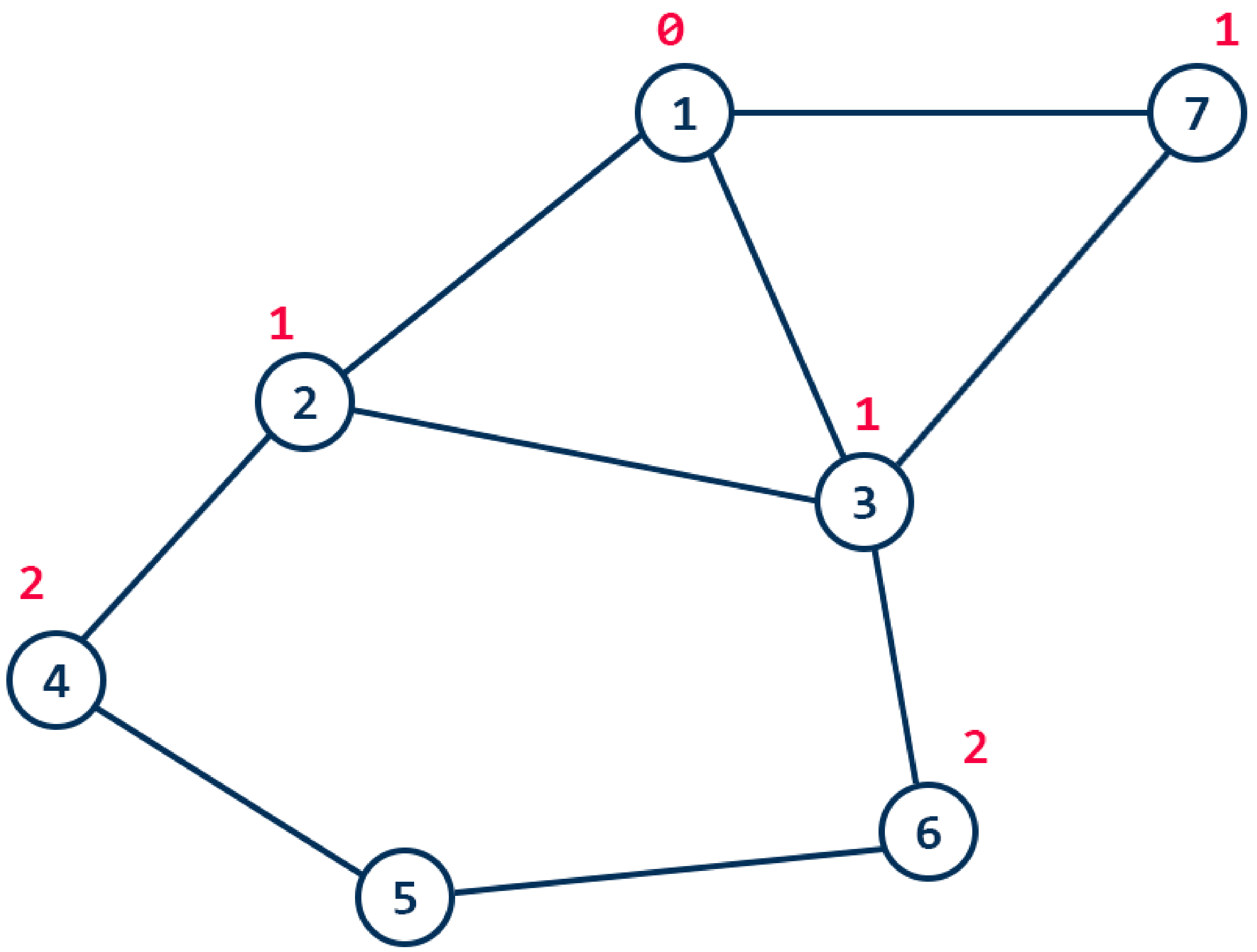
Levels in a Graph:



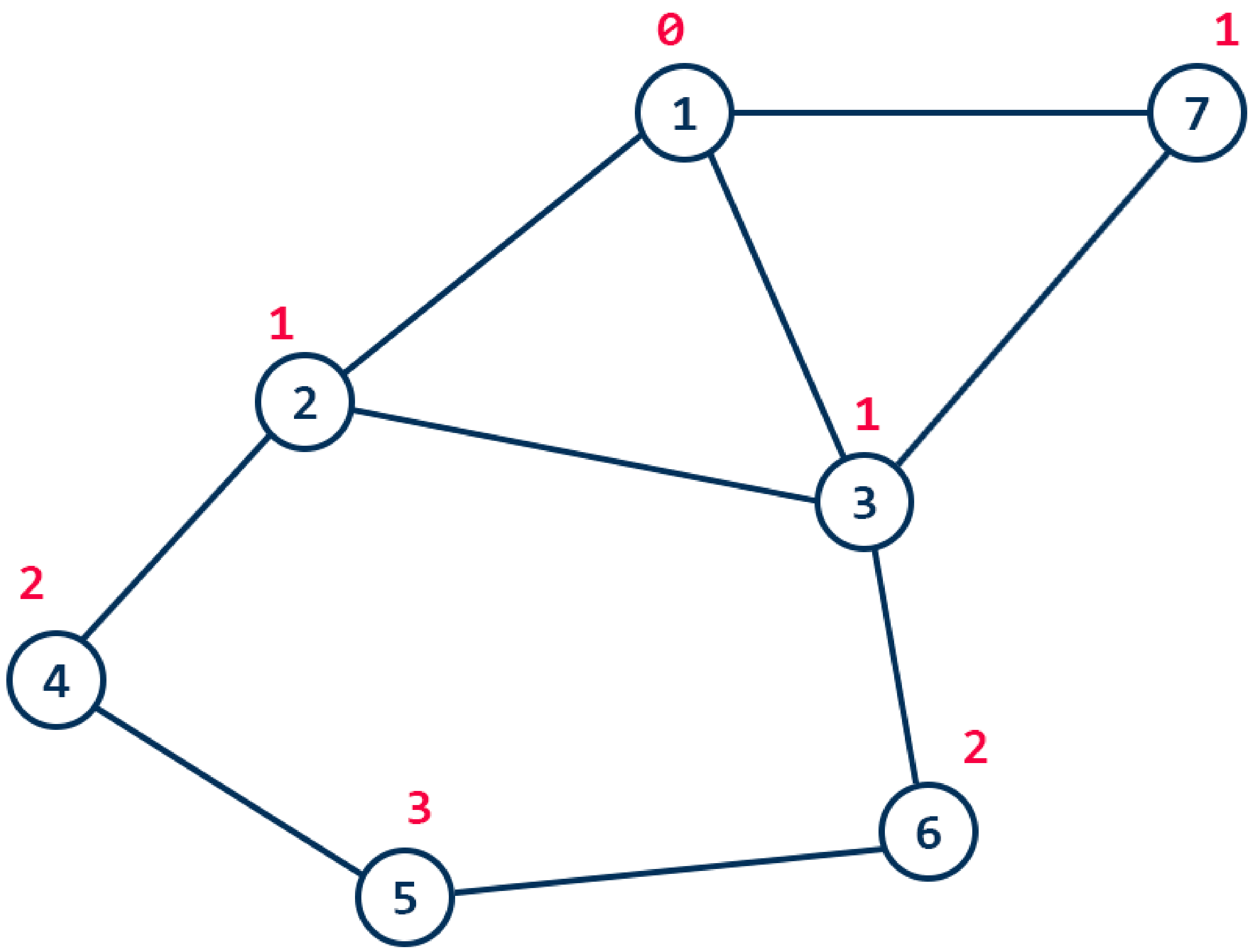
Levels in a Graph:



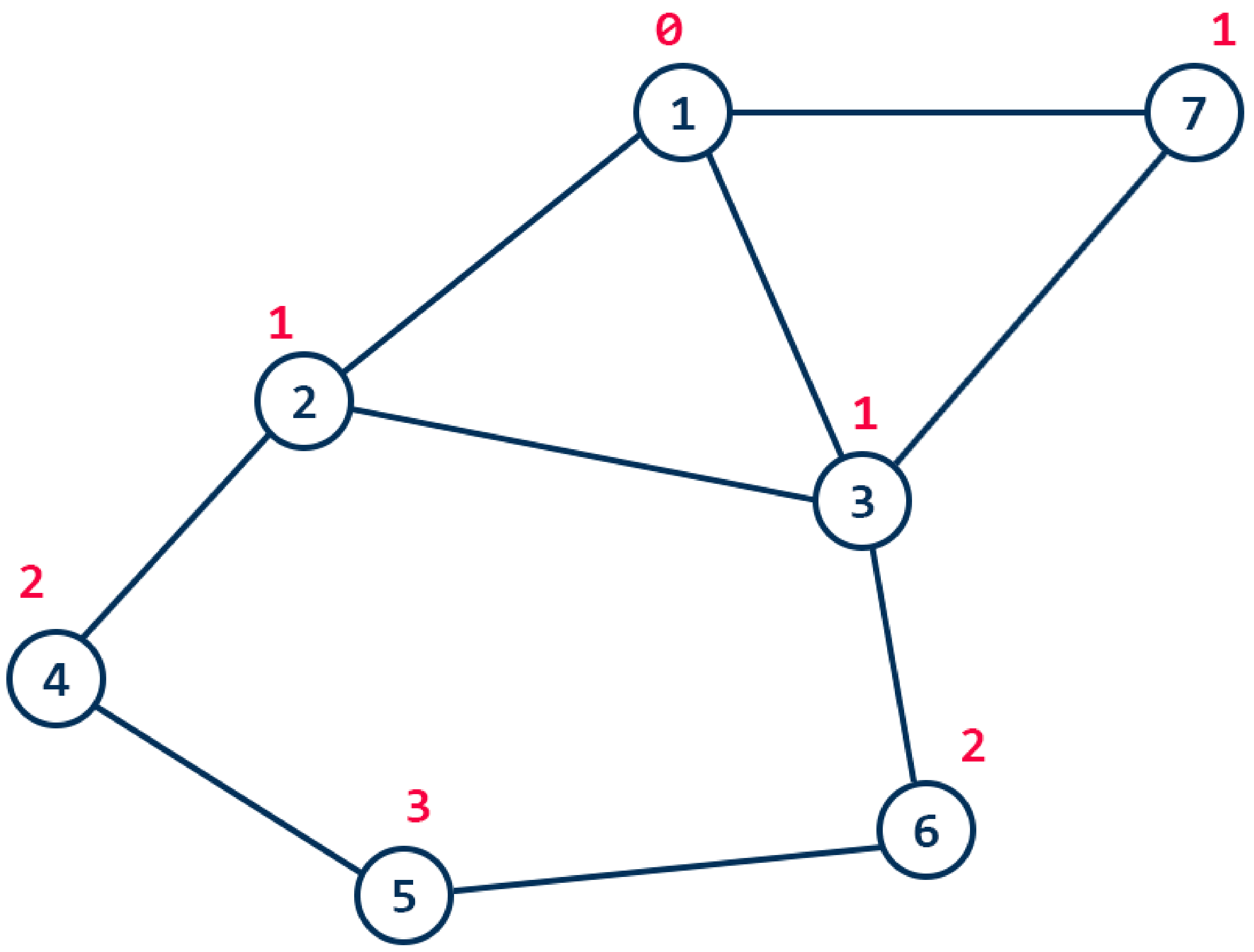
Levels in a Graph:



Levels in a Graph:

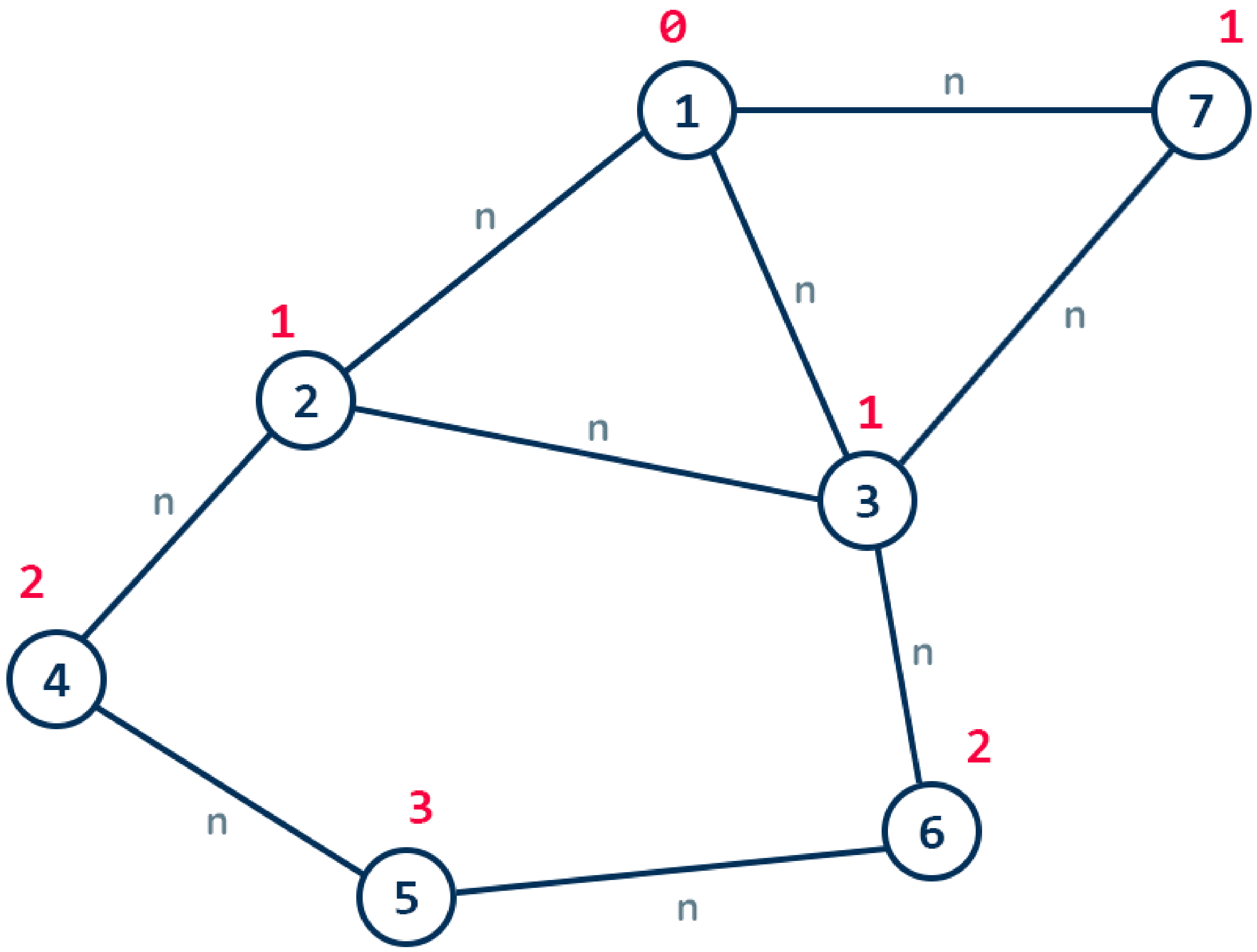


Shortest Path Finding:



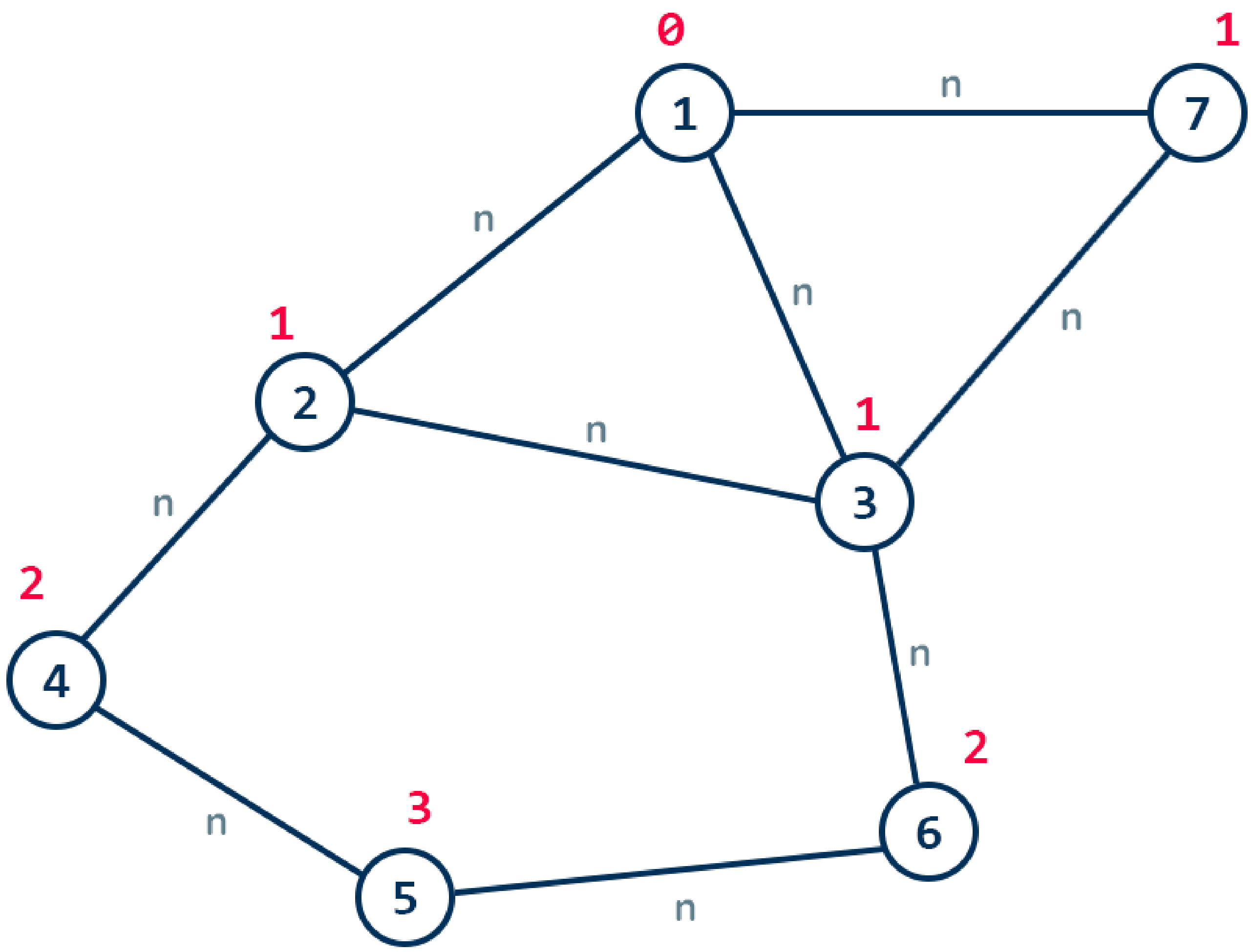
Shortest Path Finding:

cost of all
edges are
the same



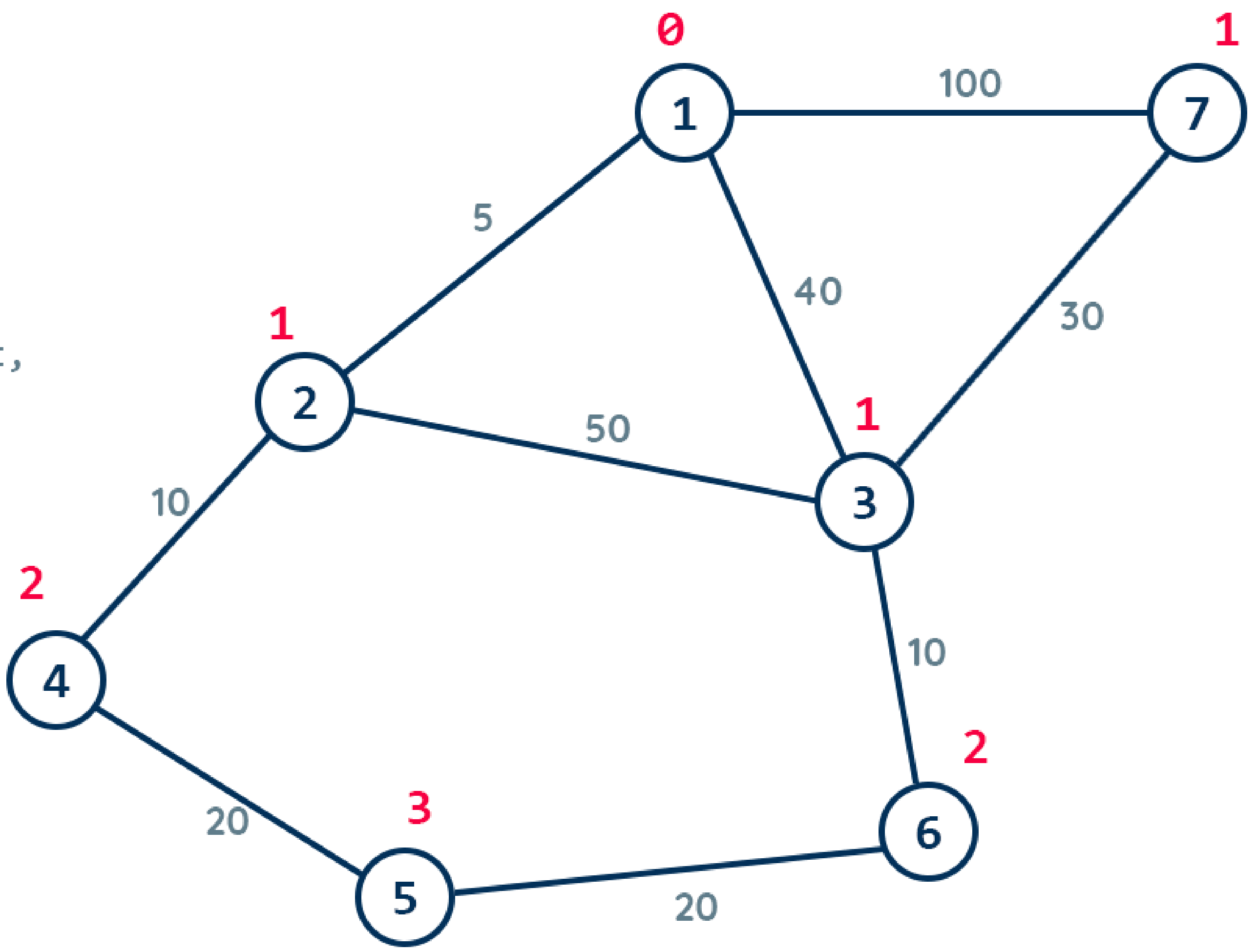
Shortest Path Finding:

From 1, 7 is closer than 5.



cost of all
edges are
the same

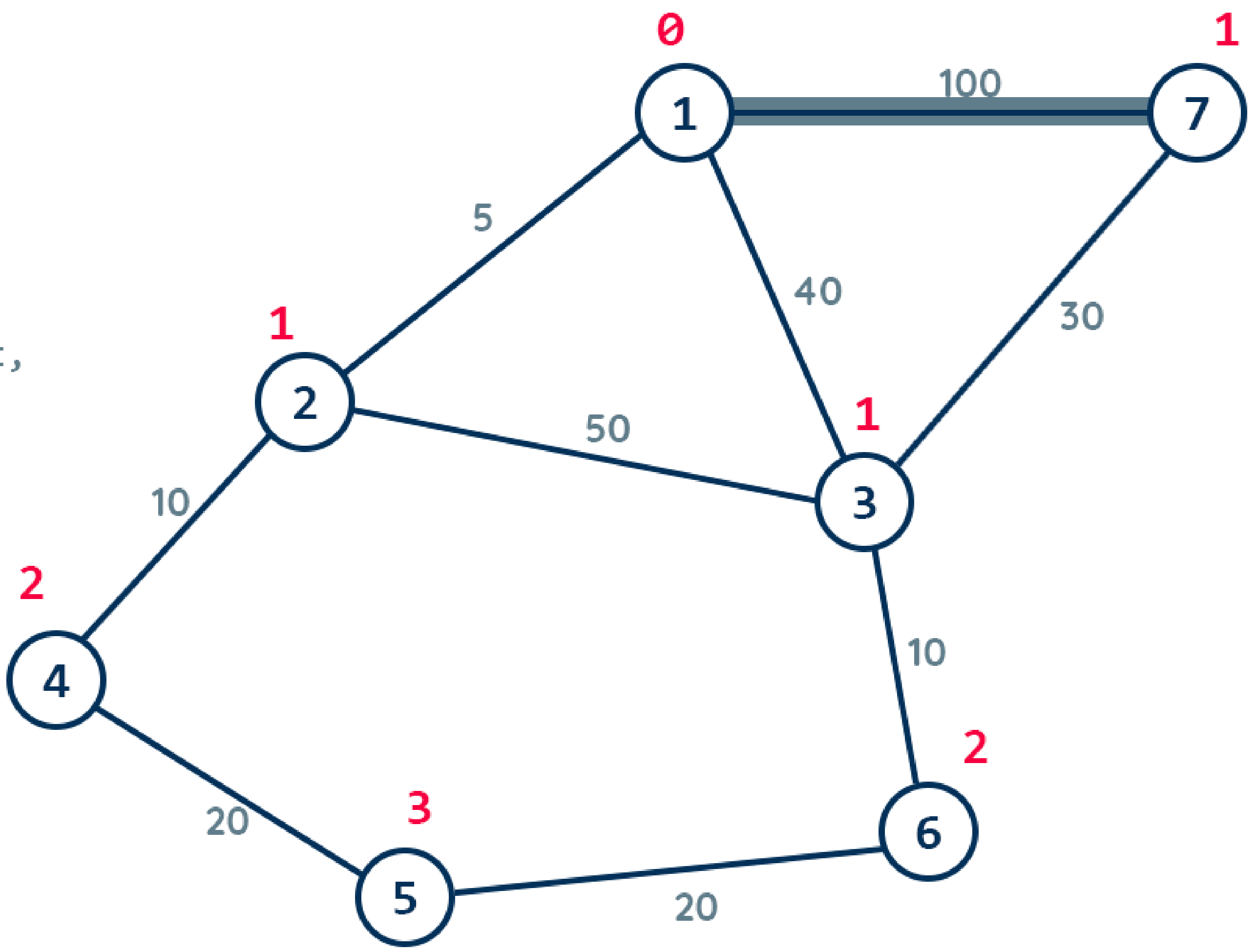
Shortest Path Finding:



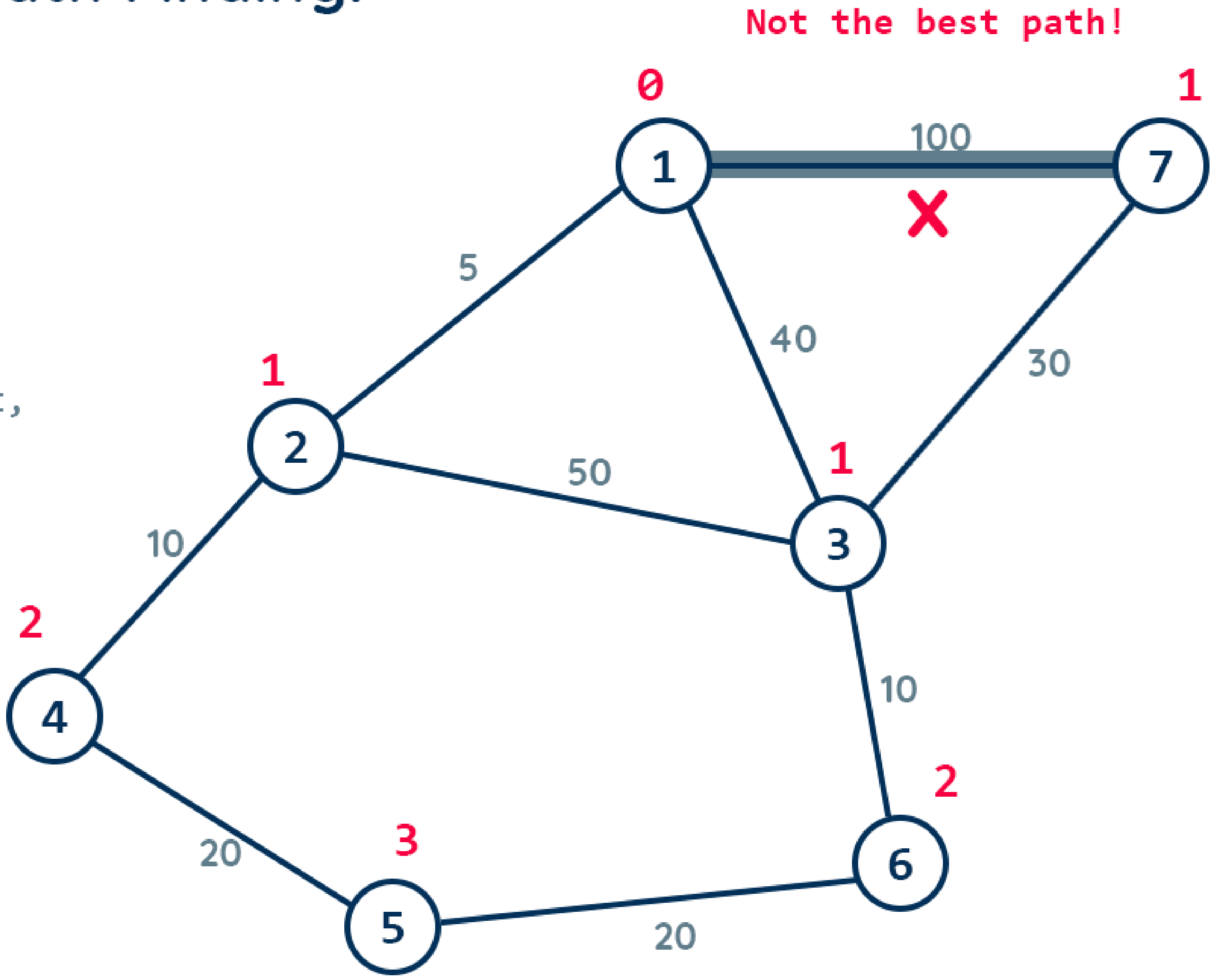
```
if variable
cost is present,
BFS will not
find shortest
path
```

Shortest Path Finding:

```
if variable
cost is present,
BFS will not
find shortest
path
```

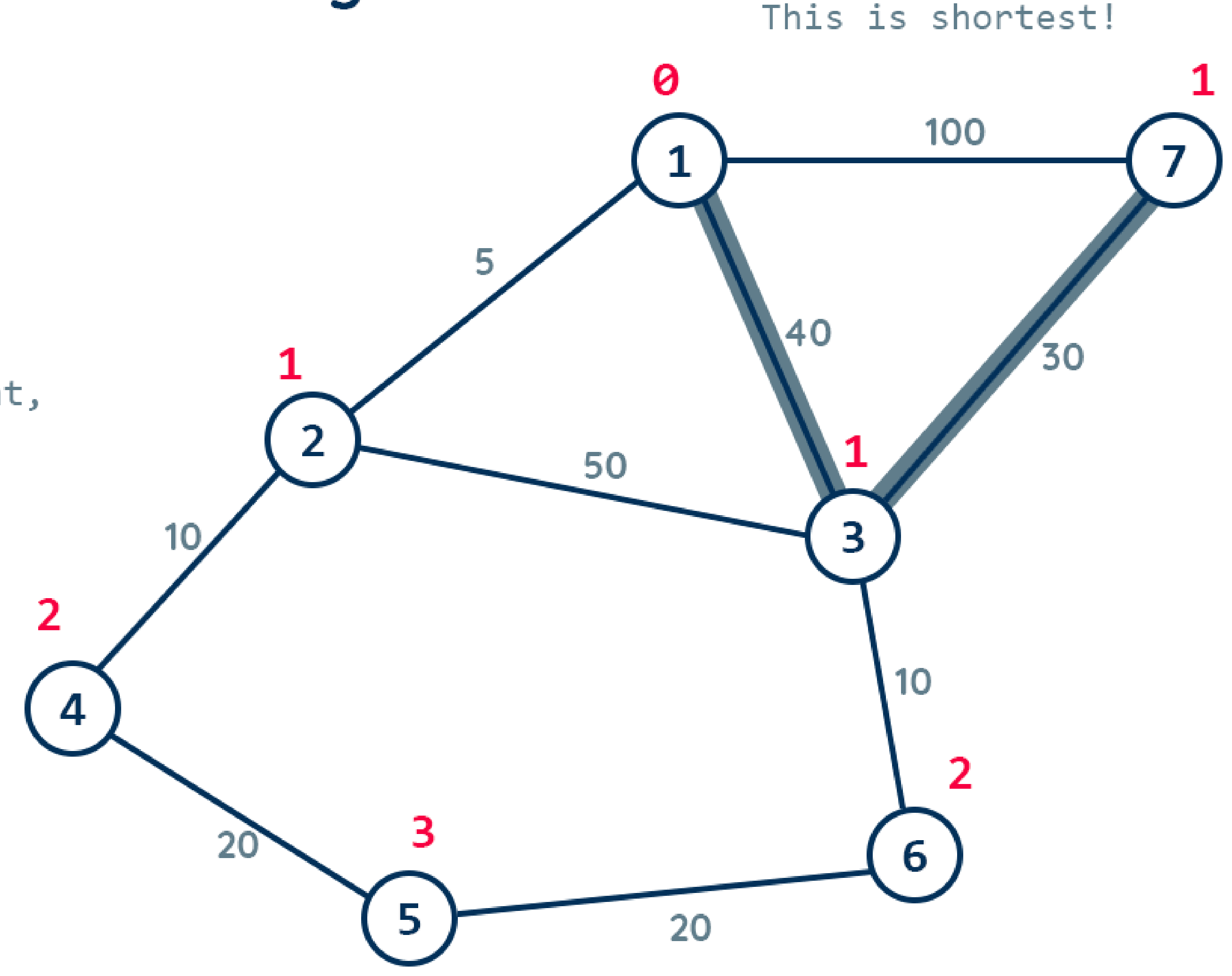


Shortest Path Finding:



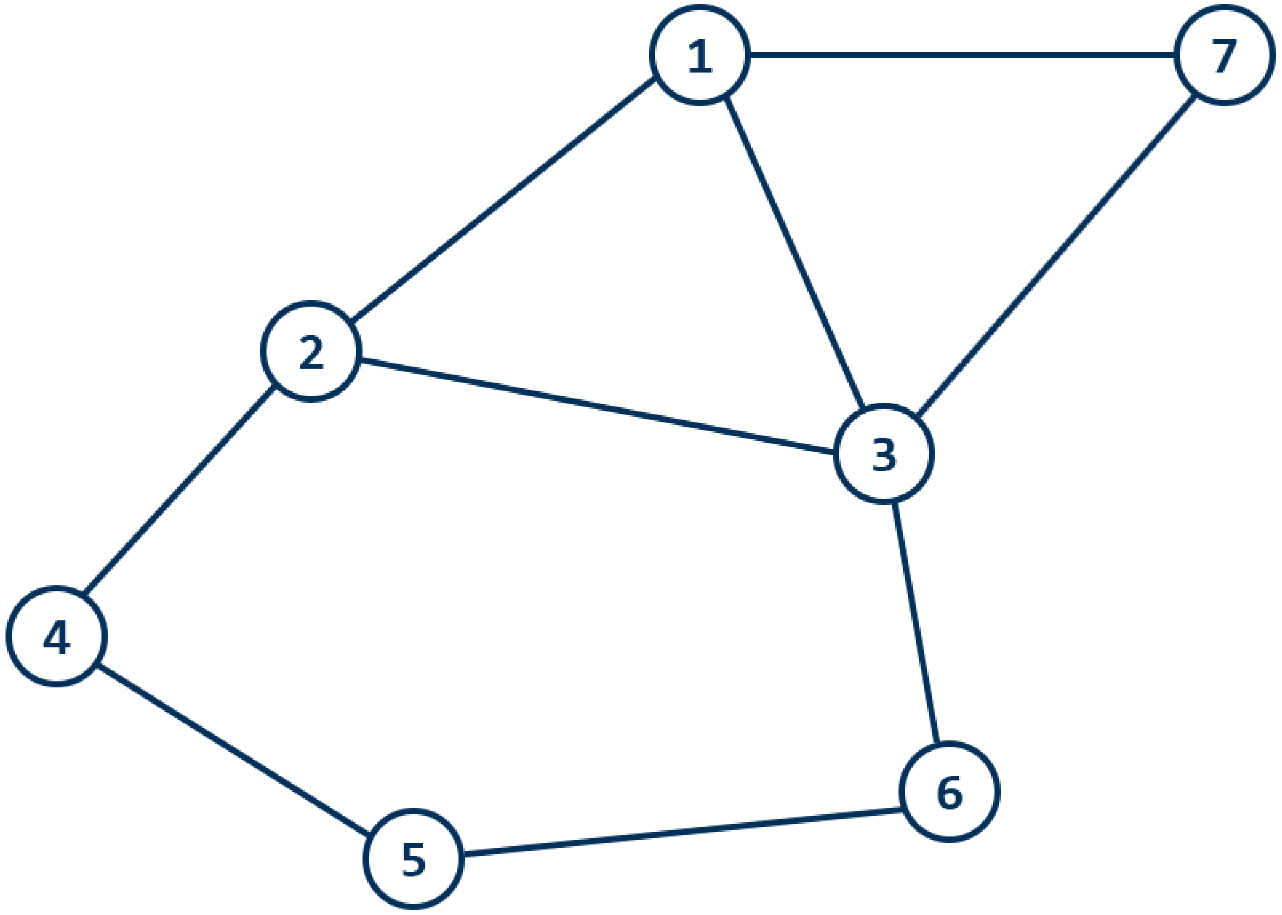
if variable
cost is present,
BFS will not
find shortest
path

Shortest Path Finding:

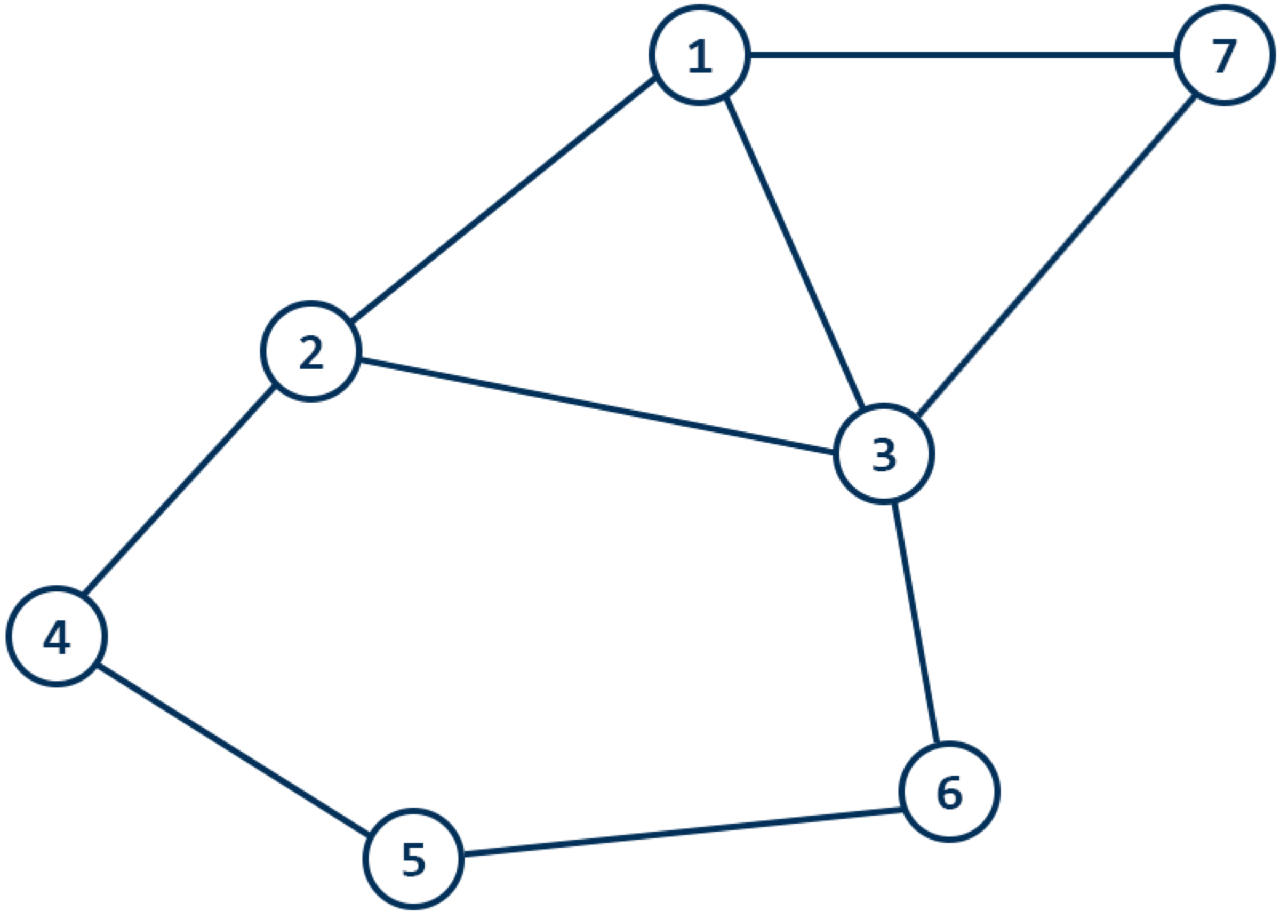


if variable cost is present, BFS will not find shortest path

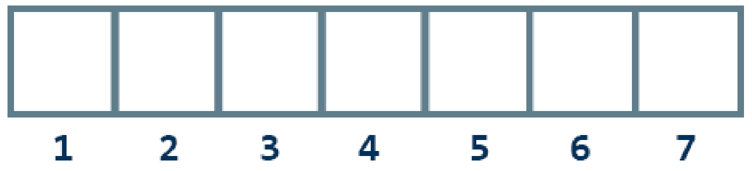
BFS Simulation



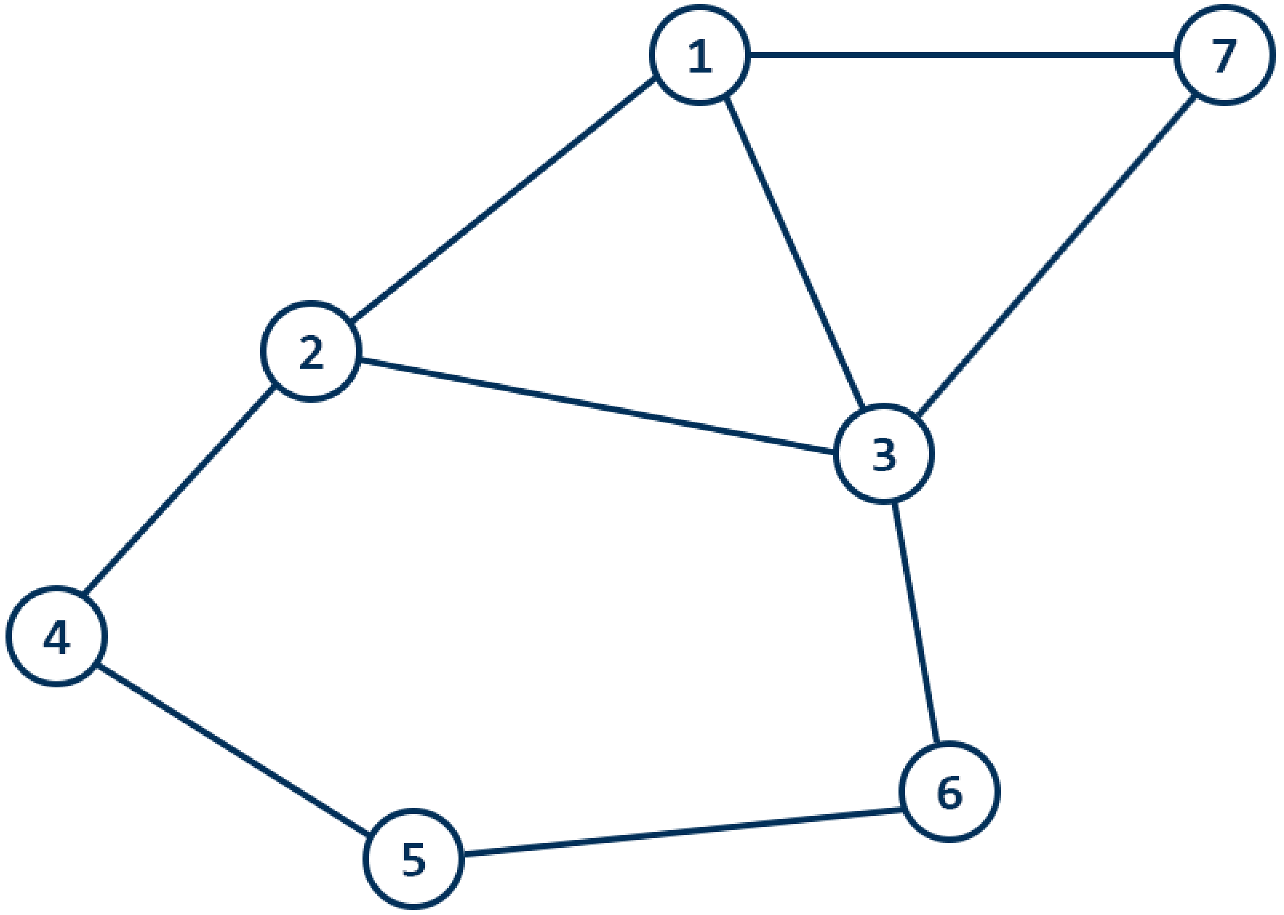
BFS Simulation



```
bool visited[MX];
```



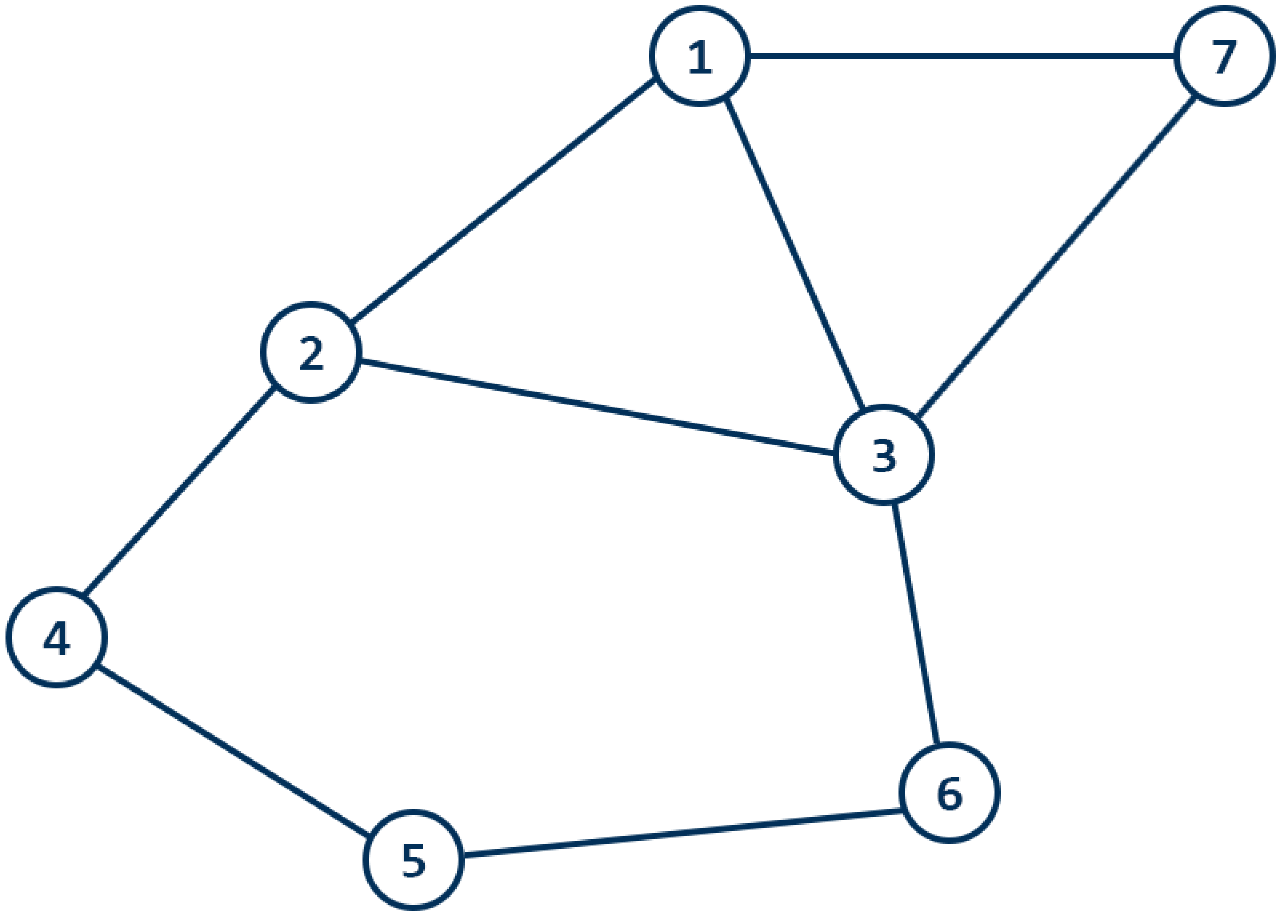
BFS Simulation



```
bool visited[MX];
```

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

BFS Simulation



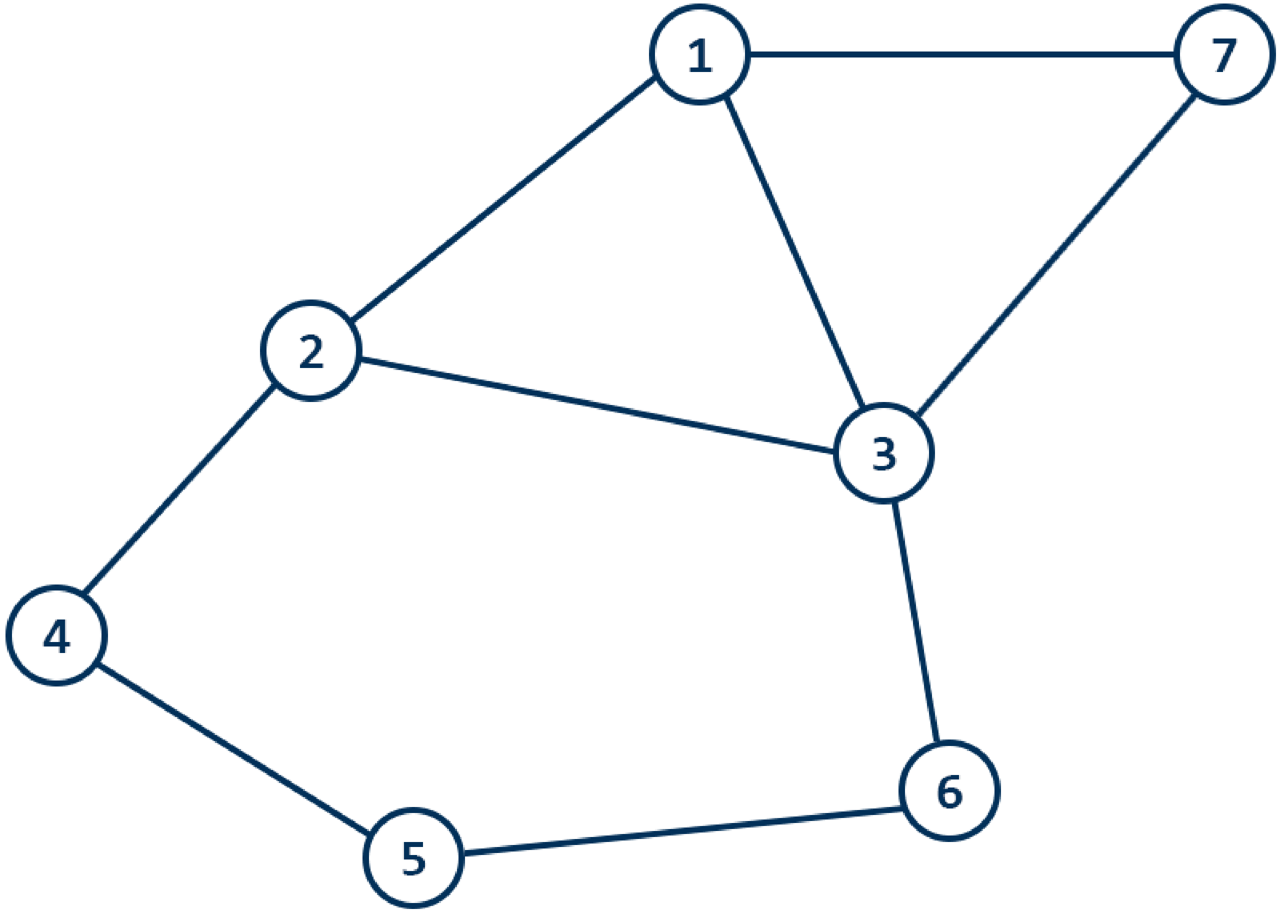
```
int distance[MX];
```

1	2	3	4	5	6	7

```
bool visited[MX];
```

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

BFS Simulation



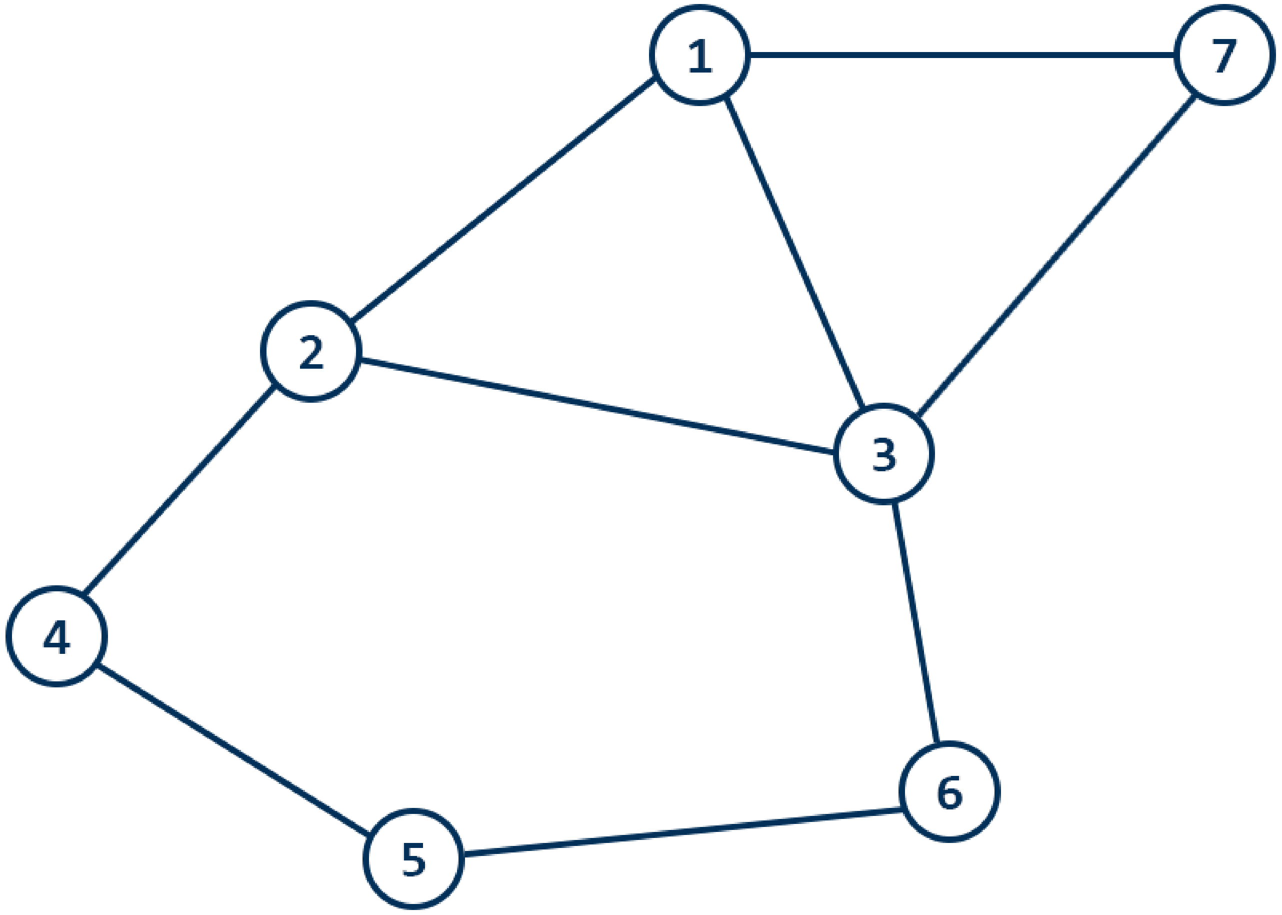
```
int distance[MX];
```

inf	inf	inf	inf	inf	inf	inf
1	2	3	4	5	6	7

```
bool visited[MX];
```

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

BFS Simulation



```
int distance[MX];
```

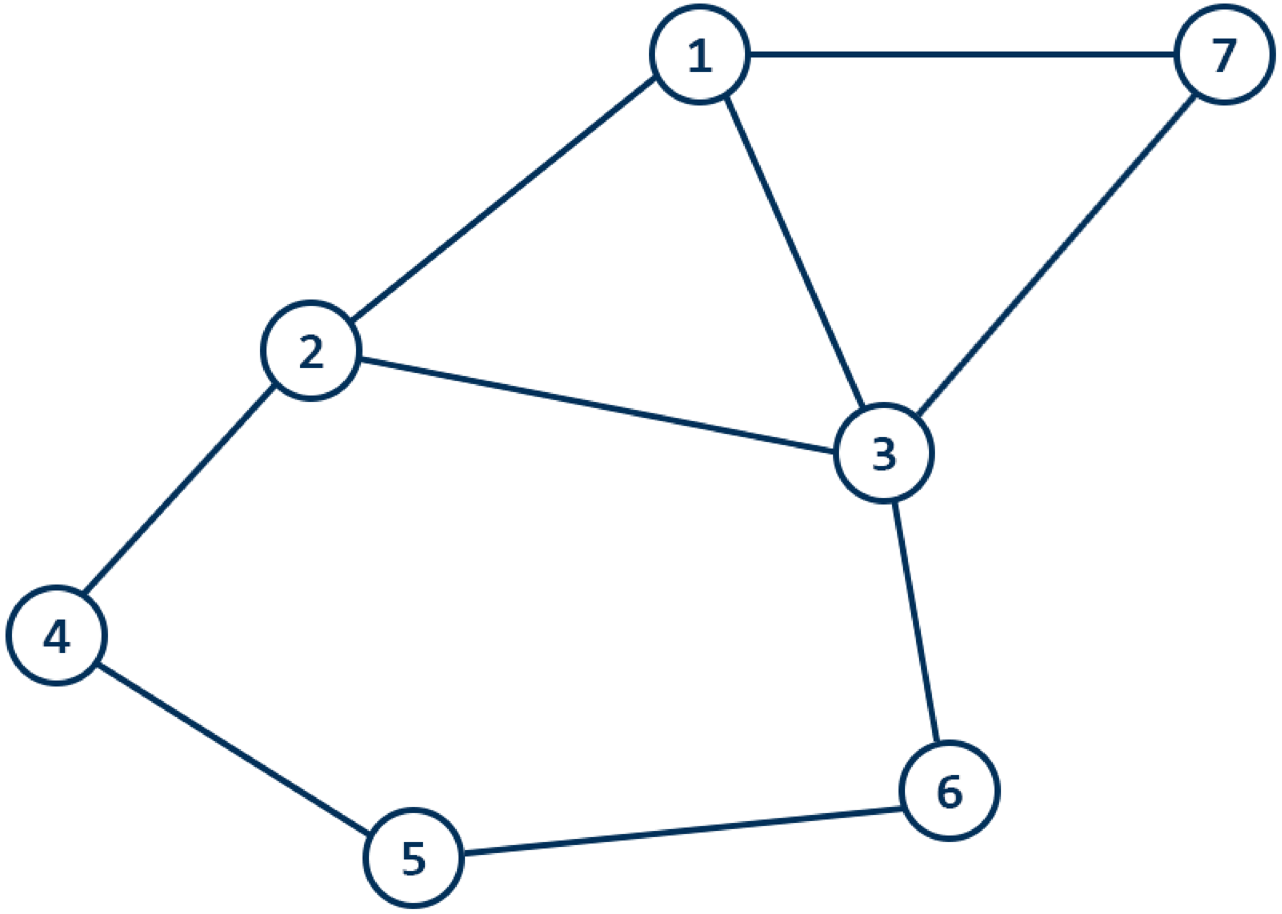
inf	inf	inf	inf	inf	inf	inf
1	2	3	4	5	6	7

```
bool visited[MX];
```

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

```
queue < int > Q;
```

BFS Simulation



```
int distance[MX];
```

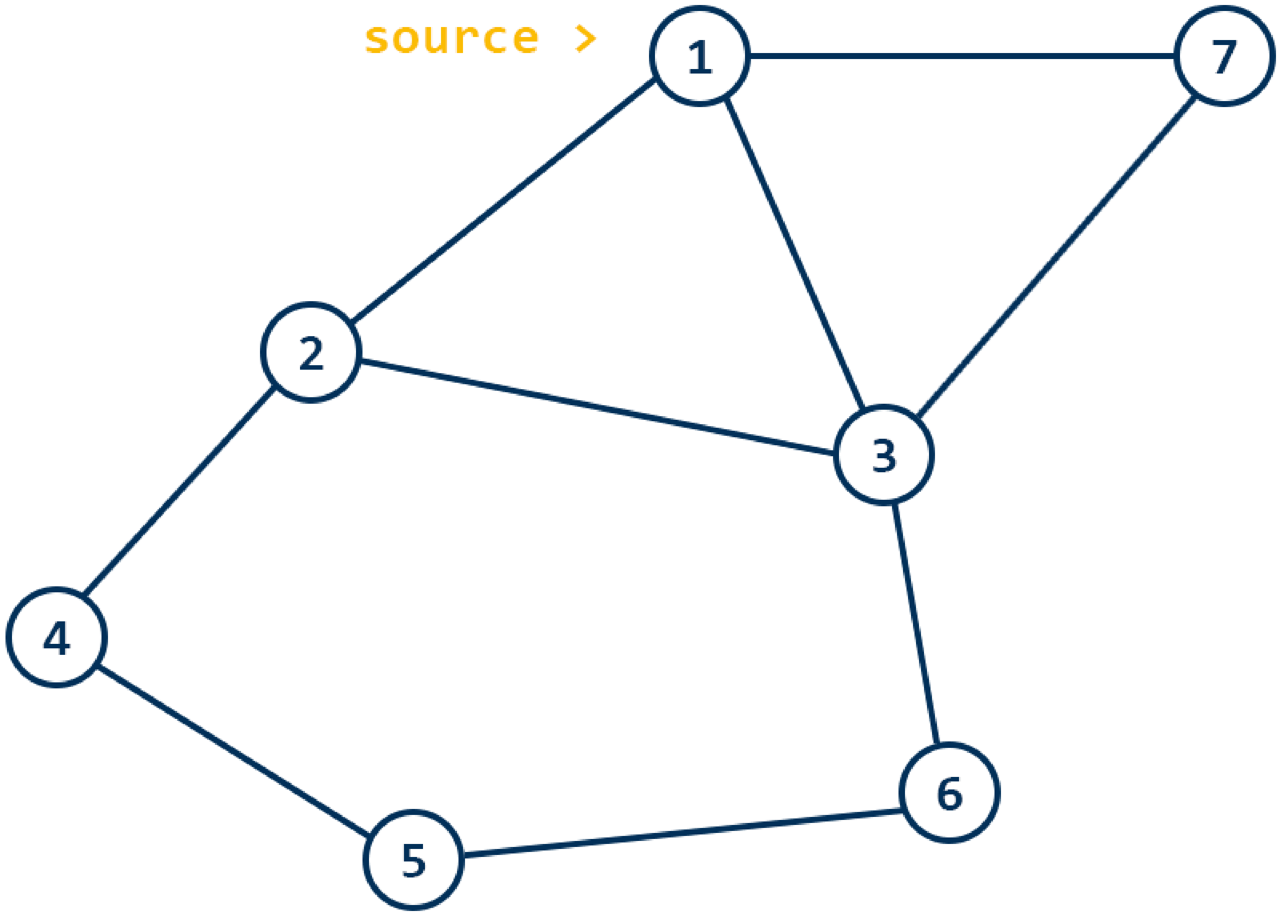
1	2	3	4	5	6	7
inf	inf	inf	inf	inf	inf	inf

```
bool visited[MX];
```

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

```
queue < int > Q;
```


BFS Simulation



```
int distance[MX];
```

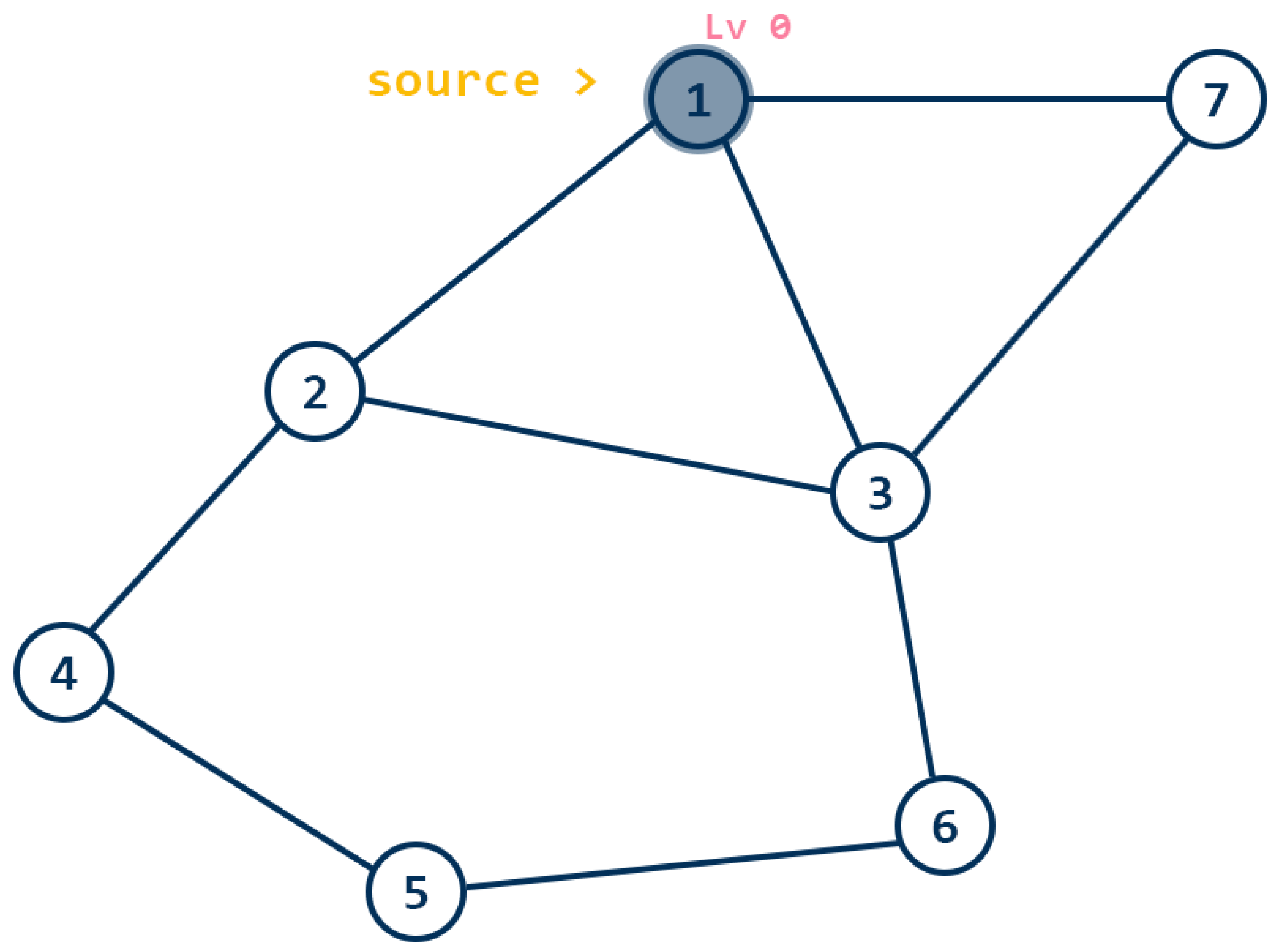
1	2	3	4	5	6	7
inf	inf	inf	inf	inf	inf	inf

```
bool visited[MX];
```

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

```
queue < int > Q;
```


BFS Simulation



```
int distance[MX];
```

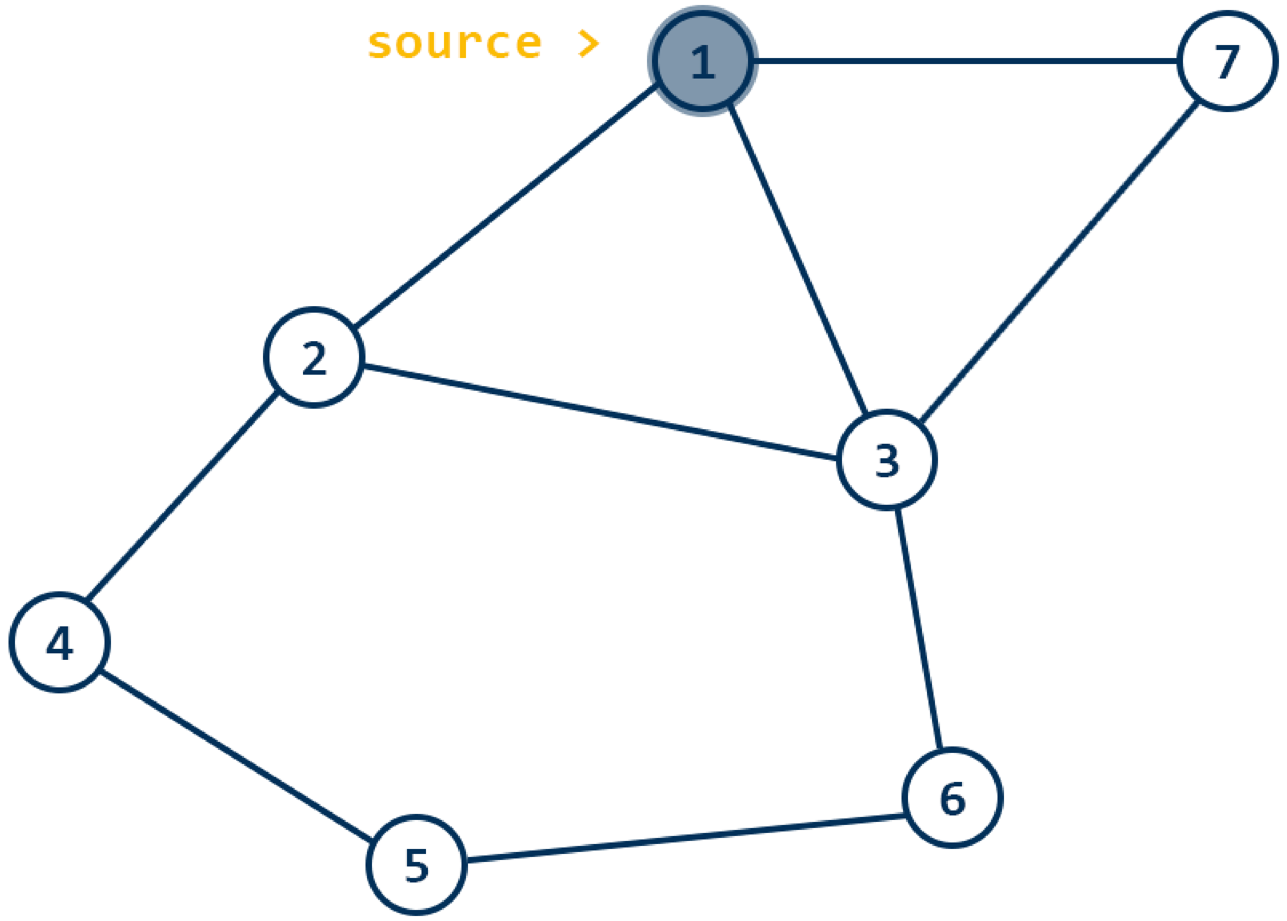
1	2	3	4	5	6	7
0	inf	inf	inf	inf	inf	inf

```
bool visited[MX];
```

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

```
queue < int > Q;
```


BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

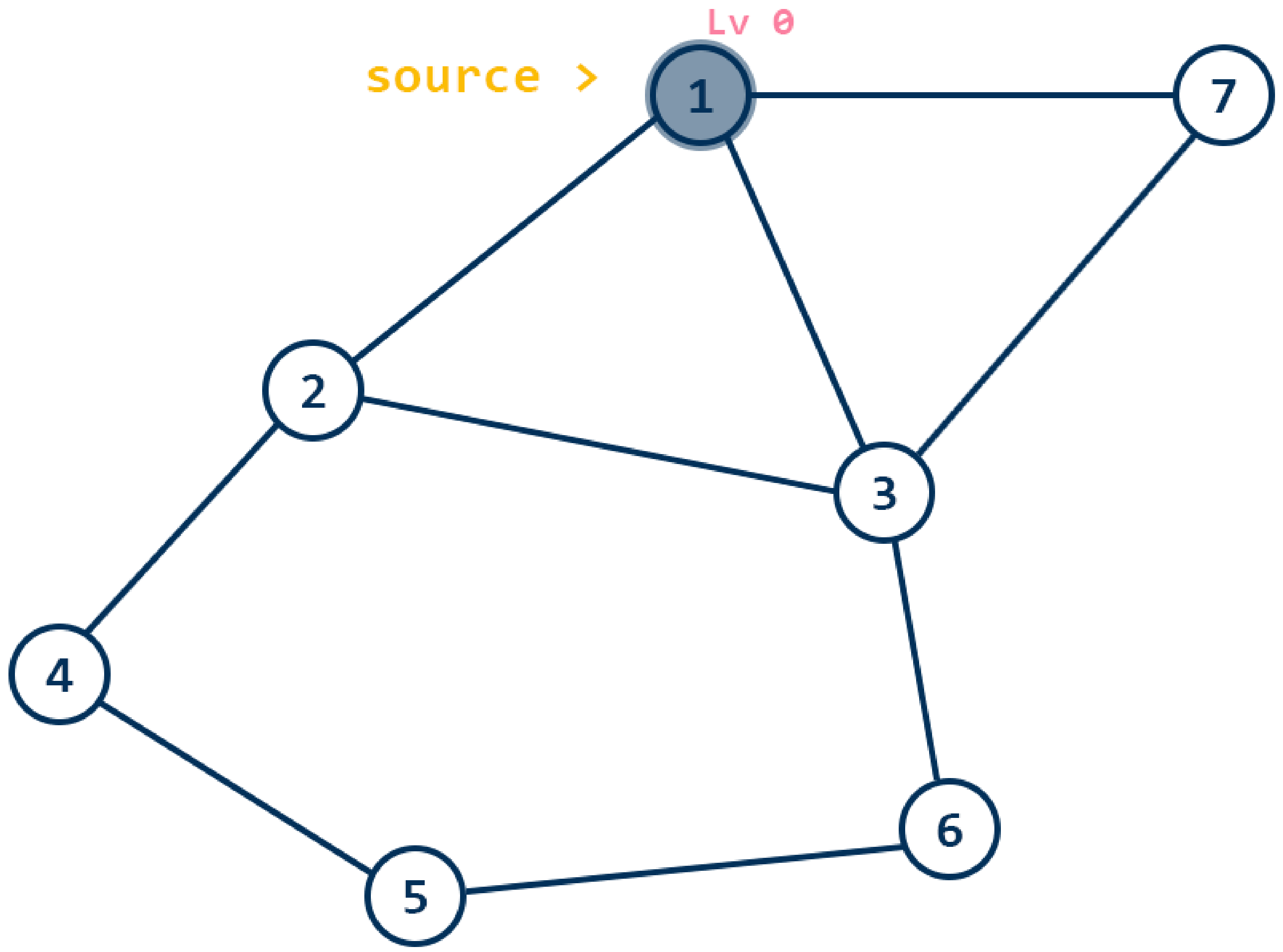
1	0	0	0	0	0	0
1	2	3	4	5	6	7

```
queue < int > Q;
```

1						

```
// pushed (1)
```

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

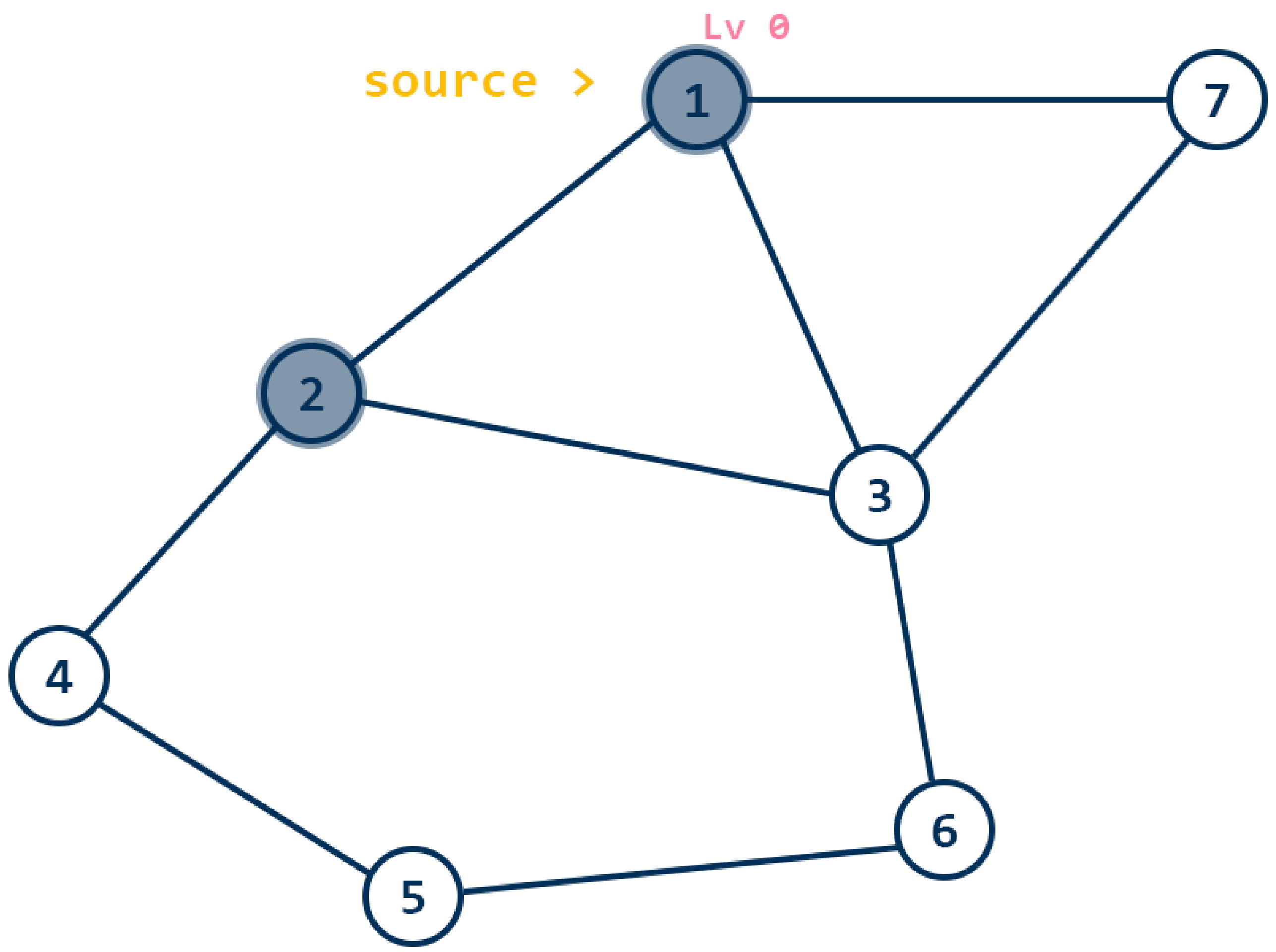
1	2	3	4	5	6	7
1	0	0	0	0	0	0

```
queue < int > Q;
```


```
// popped front (1)
```

1

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

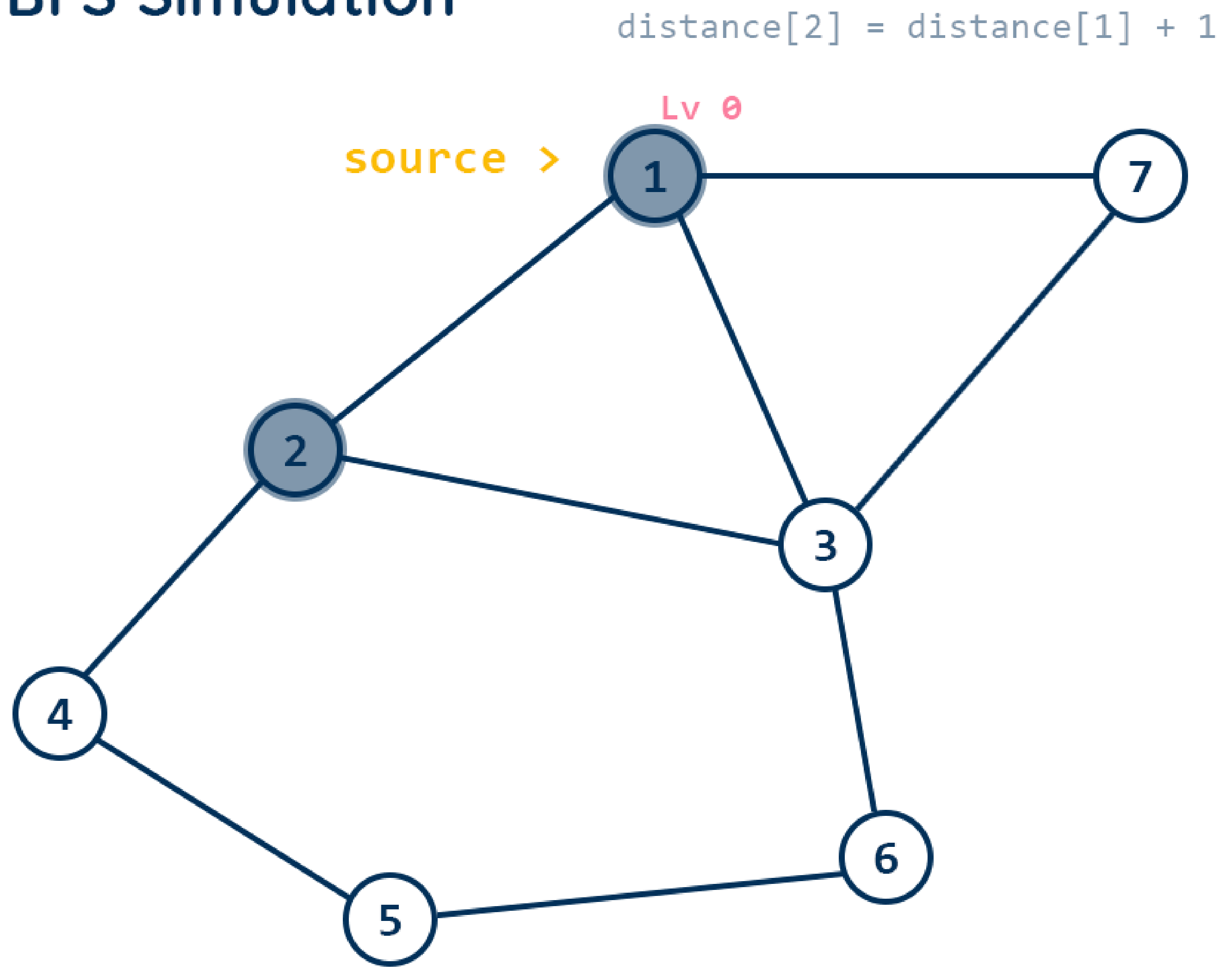
1	2	3	4	5	6	7
1	1	0	0	0	0	0

```
queue < int > Q;
```


```
// popped front (1)
```

1

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

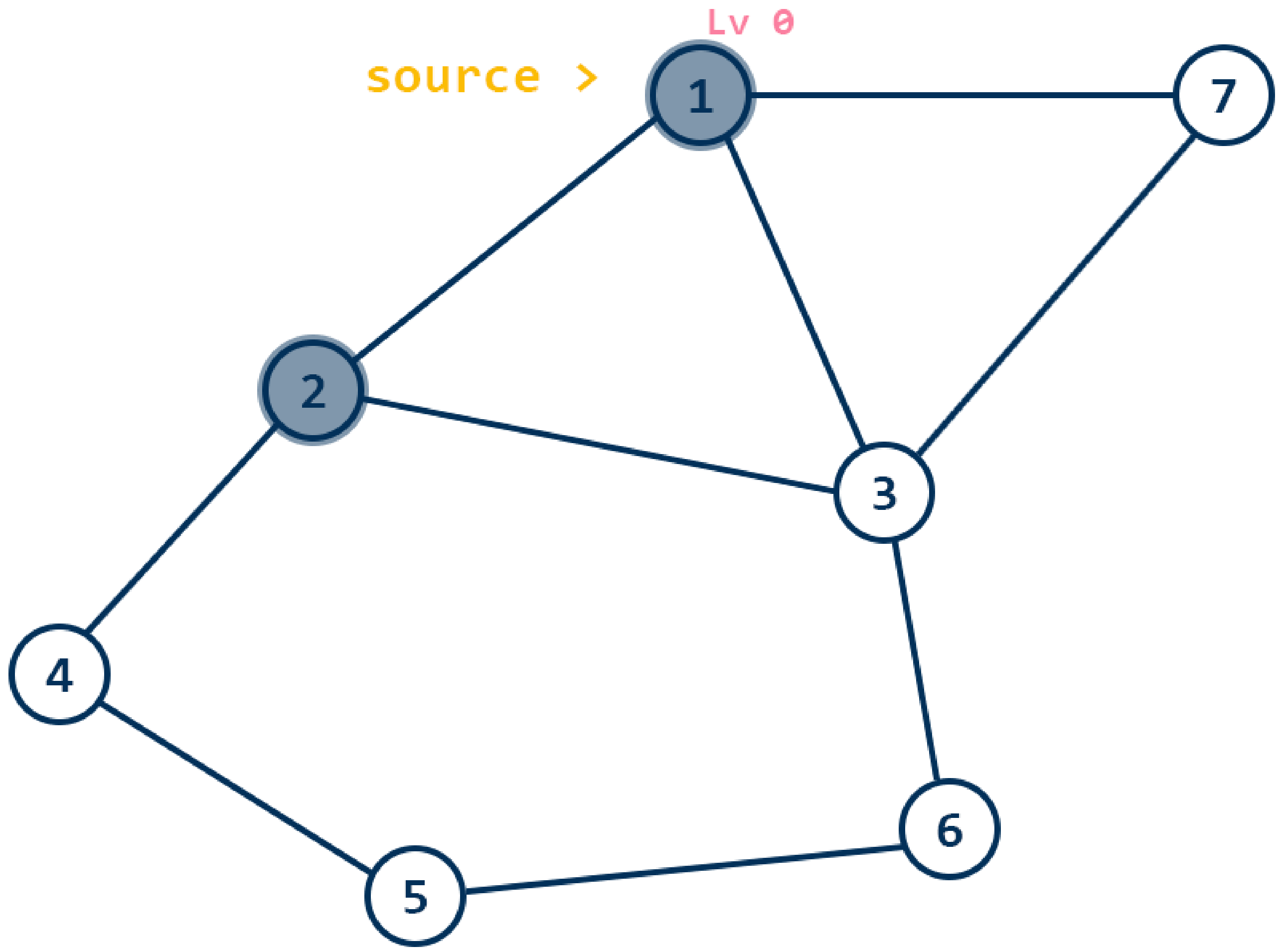
1	1	0	0	0	0	0
1	2	3	4	5	6	7

```
queue < int > Q;
```


```
// popped front (1)
```

1

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

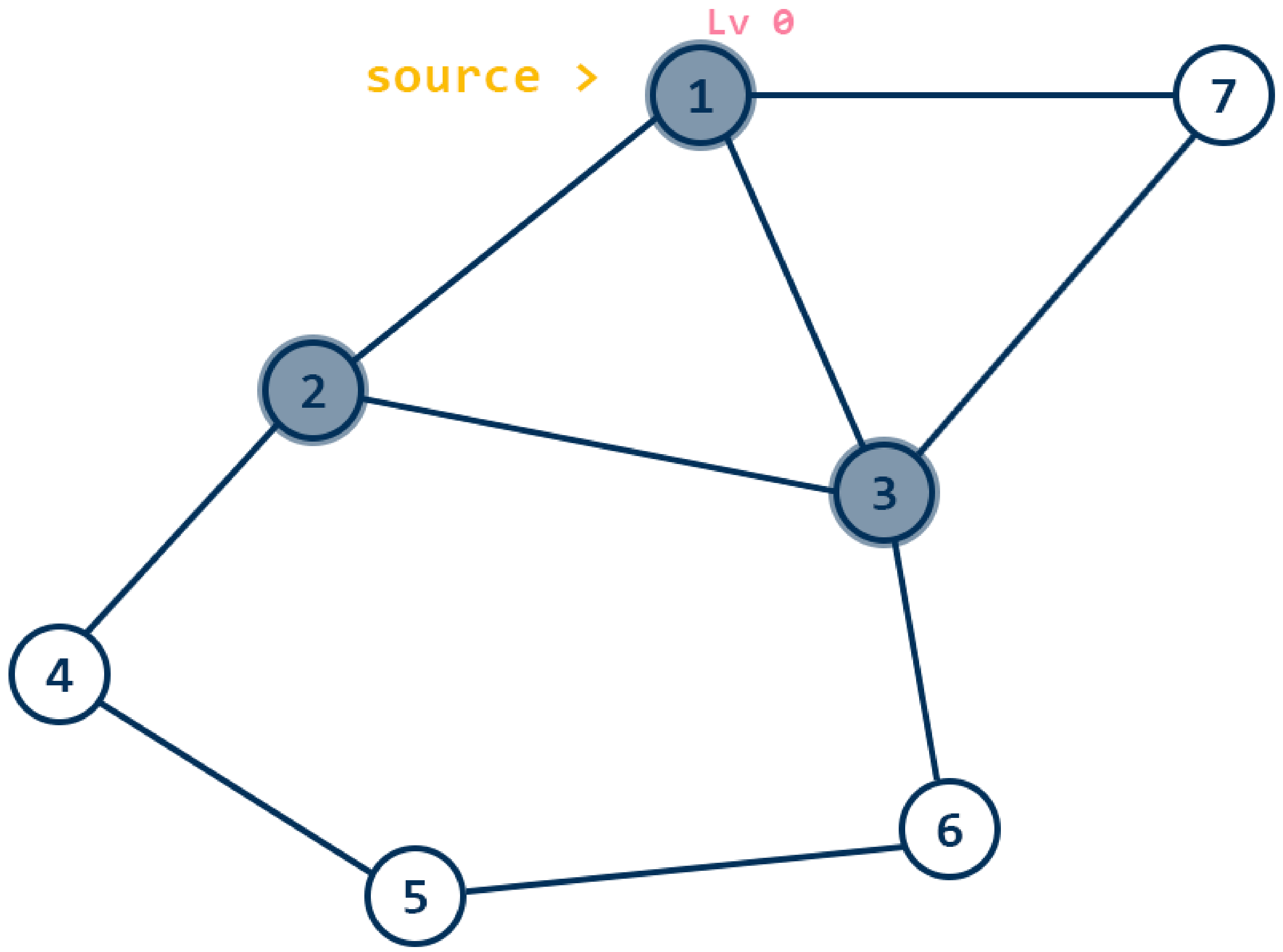
```
queue < int > Q;
```

2						

```
// pushed (2)
```

1

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

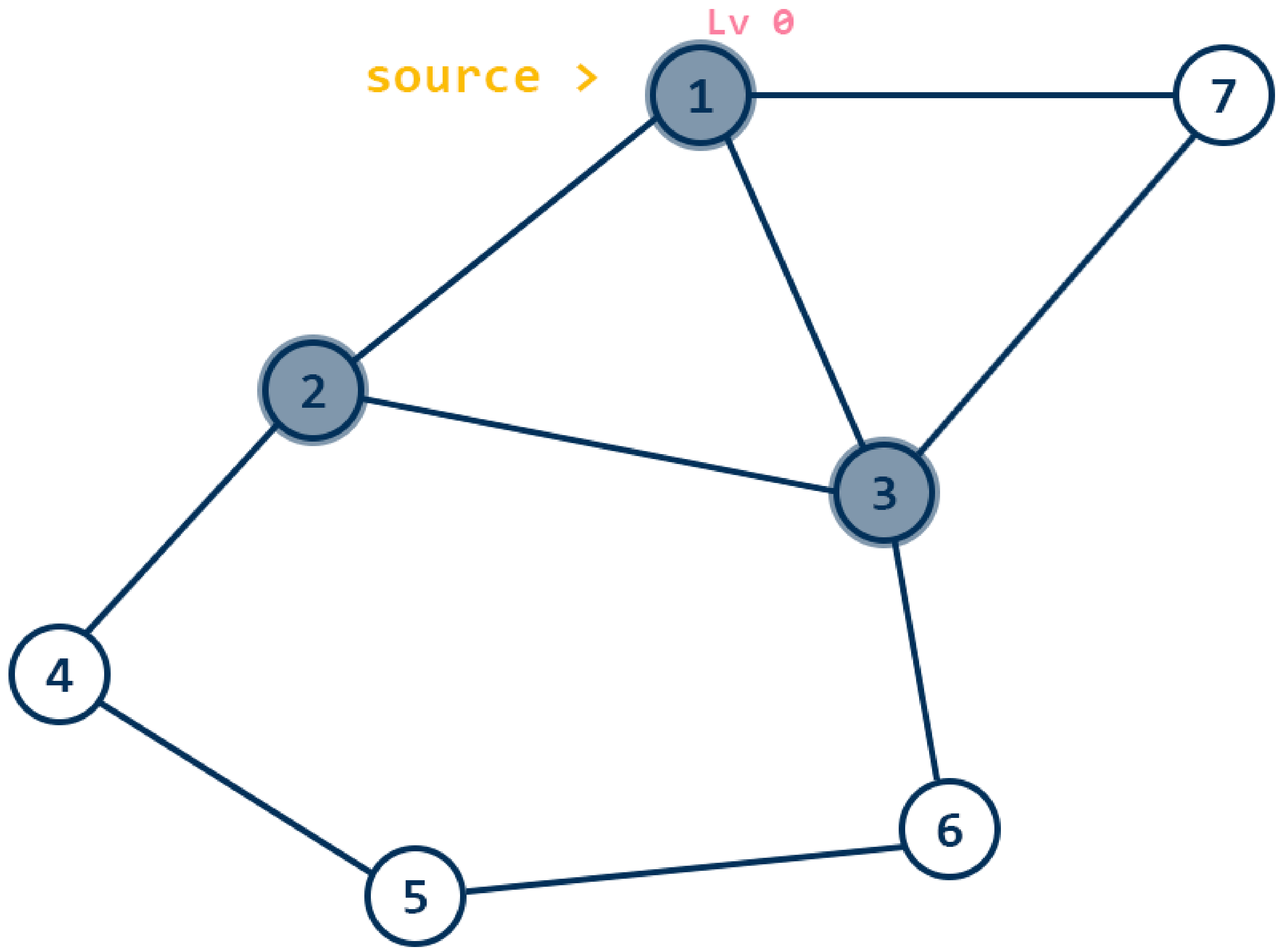
1	1	1	0	0	0	0
1	2	3	4	5	6	7

```
queue < int > Q;
```

2						

1

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

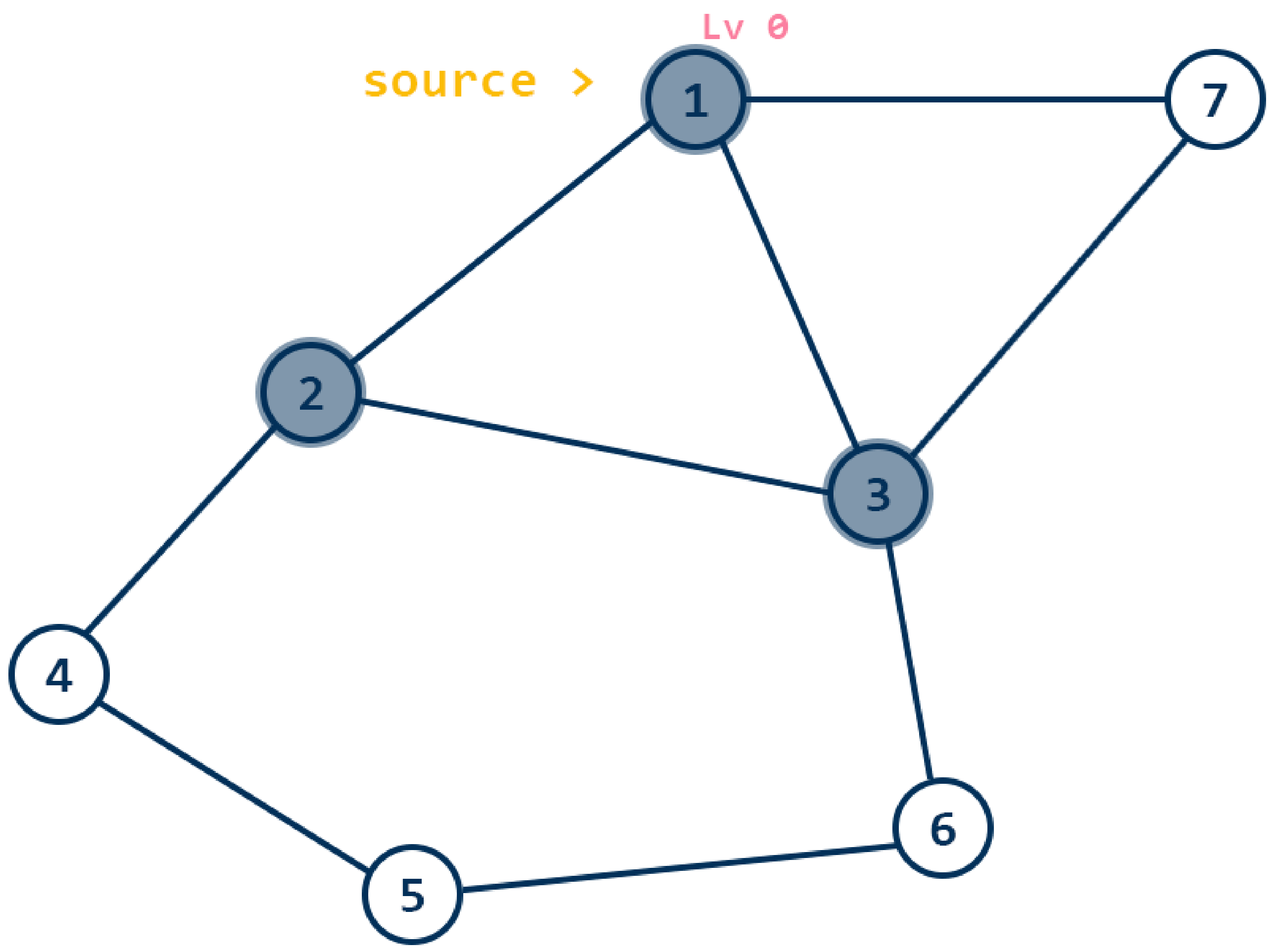
1	1	1	0	0	0	0
1	2	3	4	5	6	7

```
queue < int > Q;
```

2						

1

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

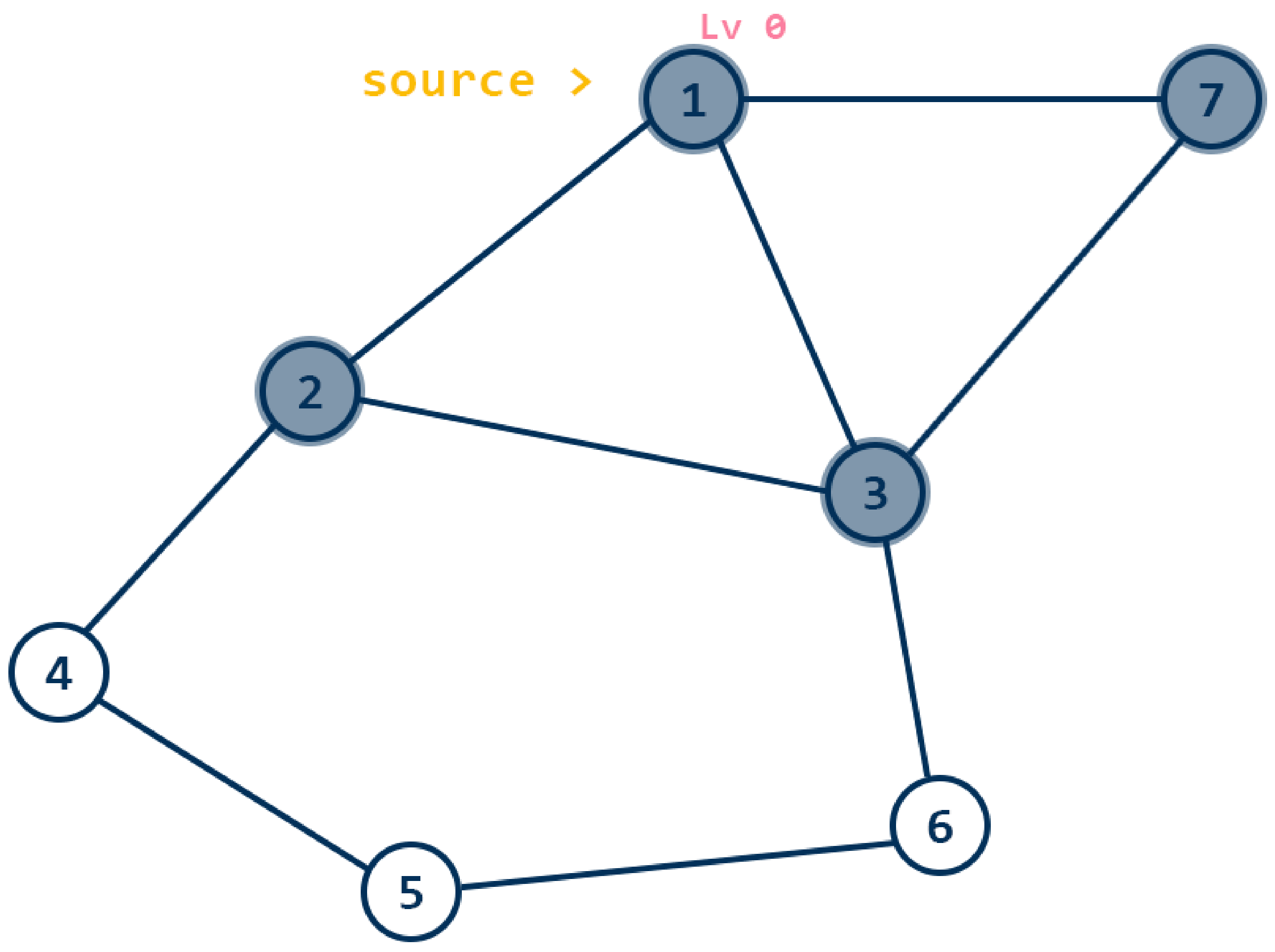
```
queue < int > Q;
```

2	3					

```
// pushed (3)
```

1

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

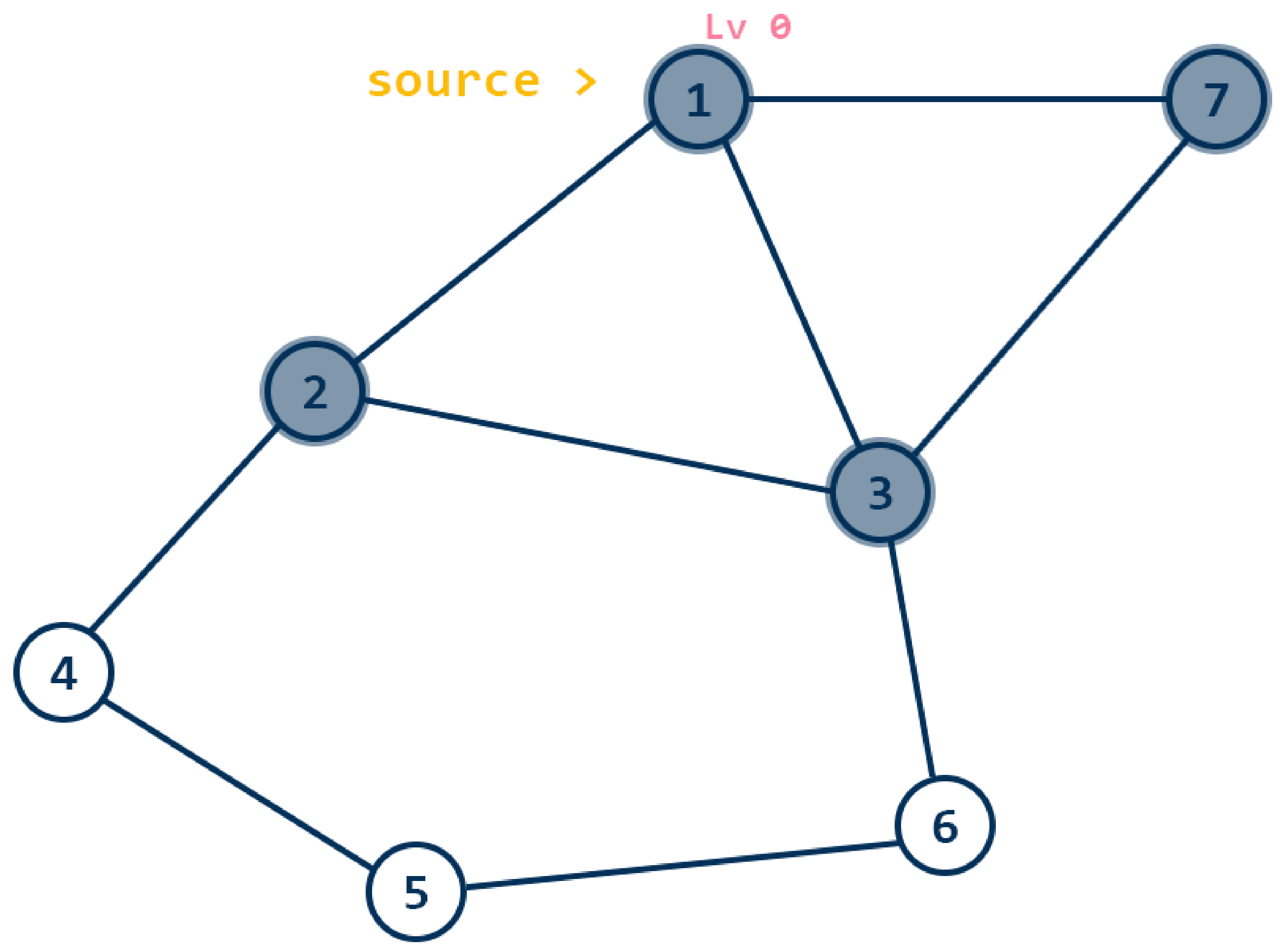
1	1	1	0	0	0	1
1	2	3	4	5	6	7

```
queue < int > Q;
```

2	3					

1

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

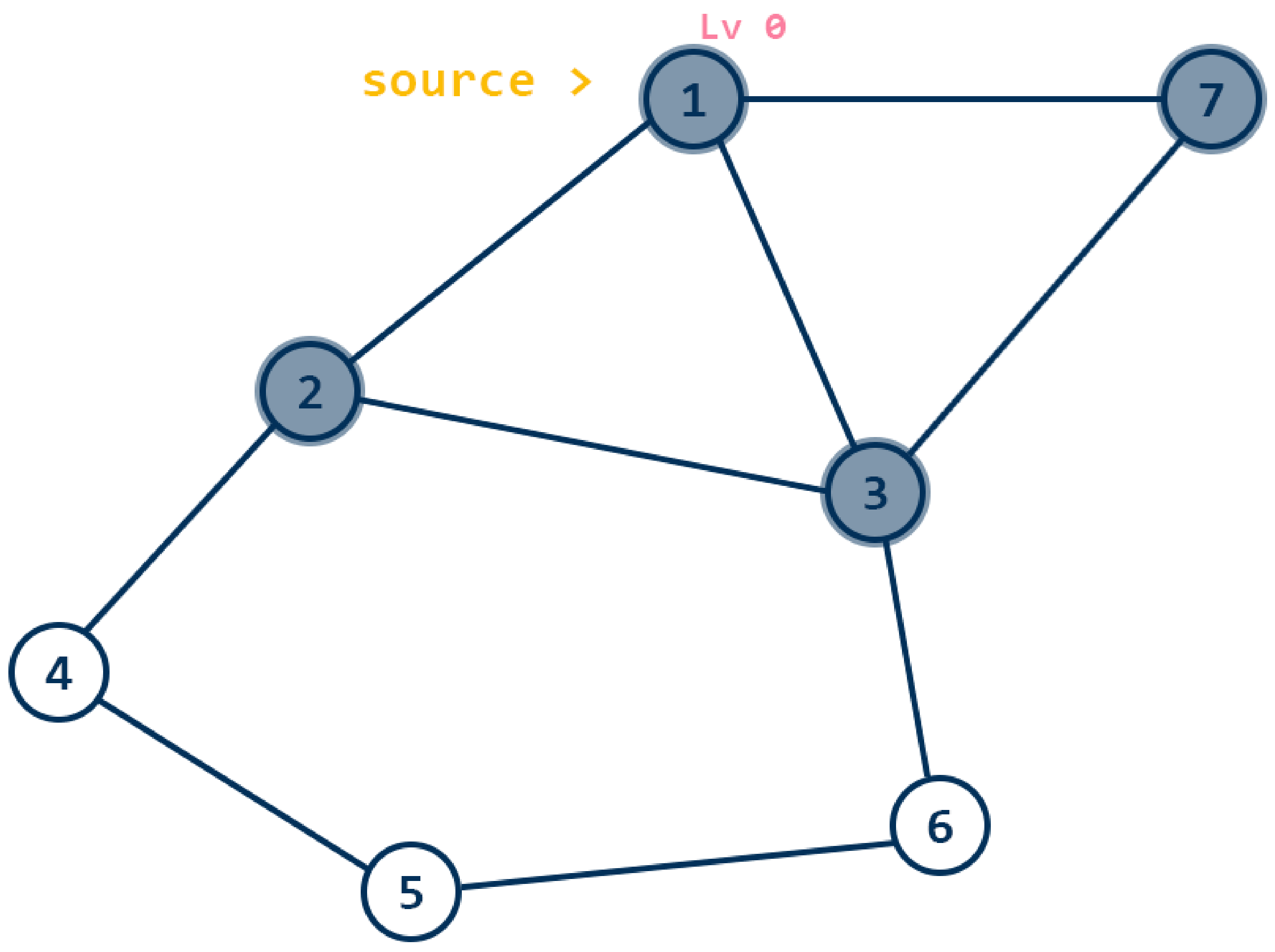
1	1	1	0	0	0	1
1	2	3	4	5	6	7

```
queue < int > Q;
```

2	3					

1

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

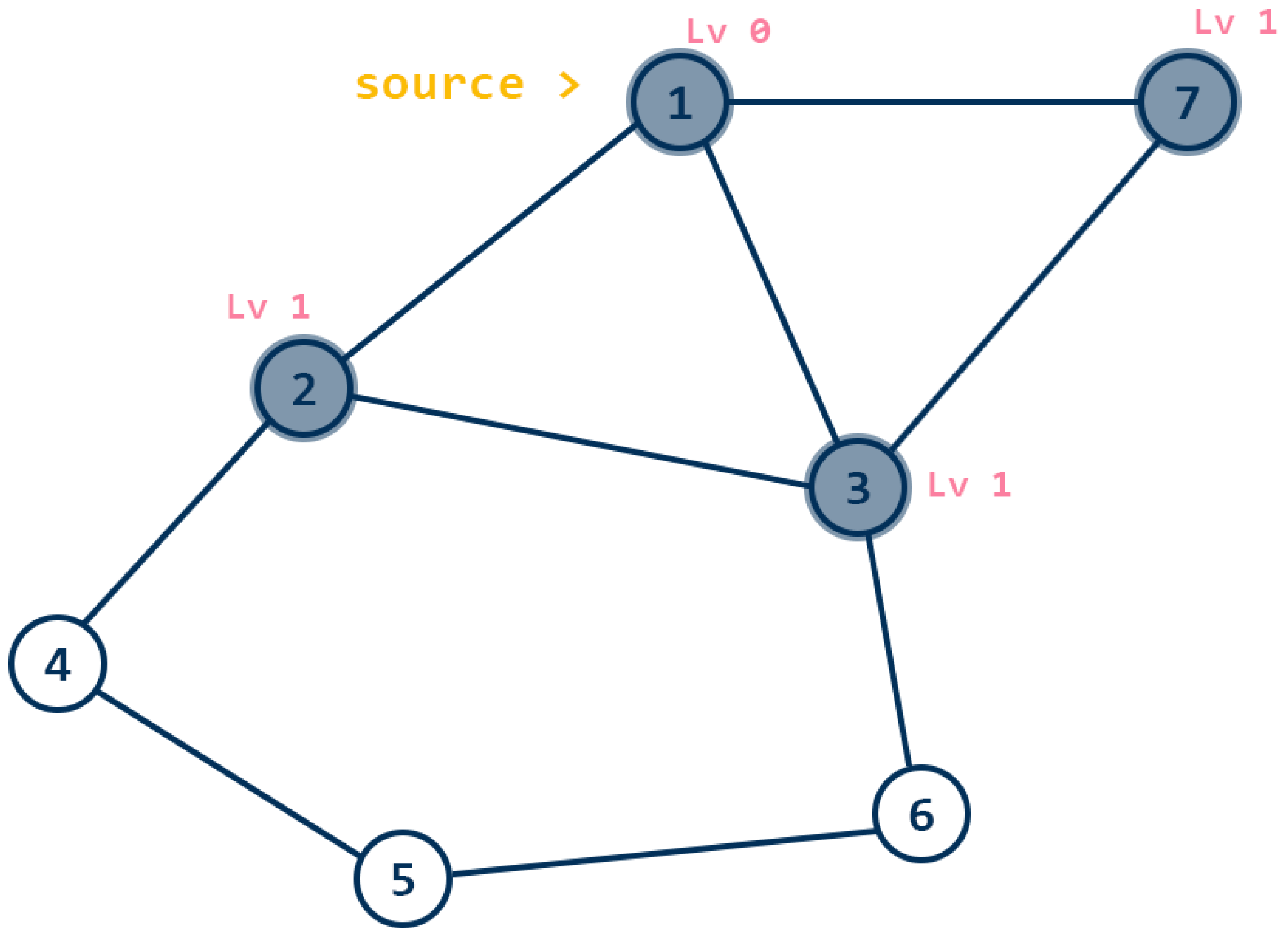
```
queue < int > Q;
```

2	3	7				

```
// pushed (7)
```

1

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

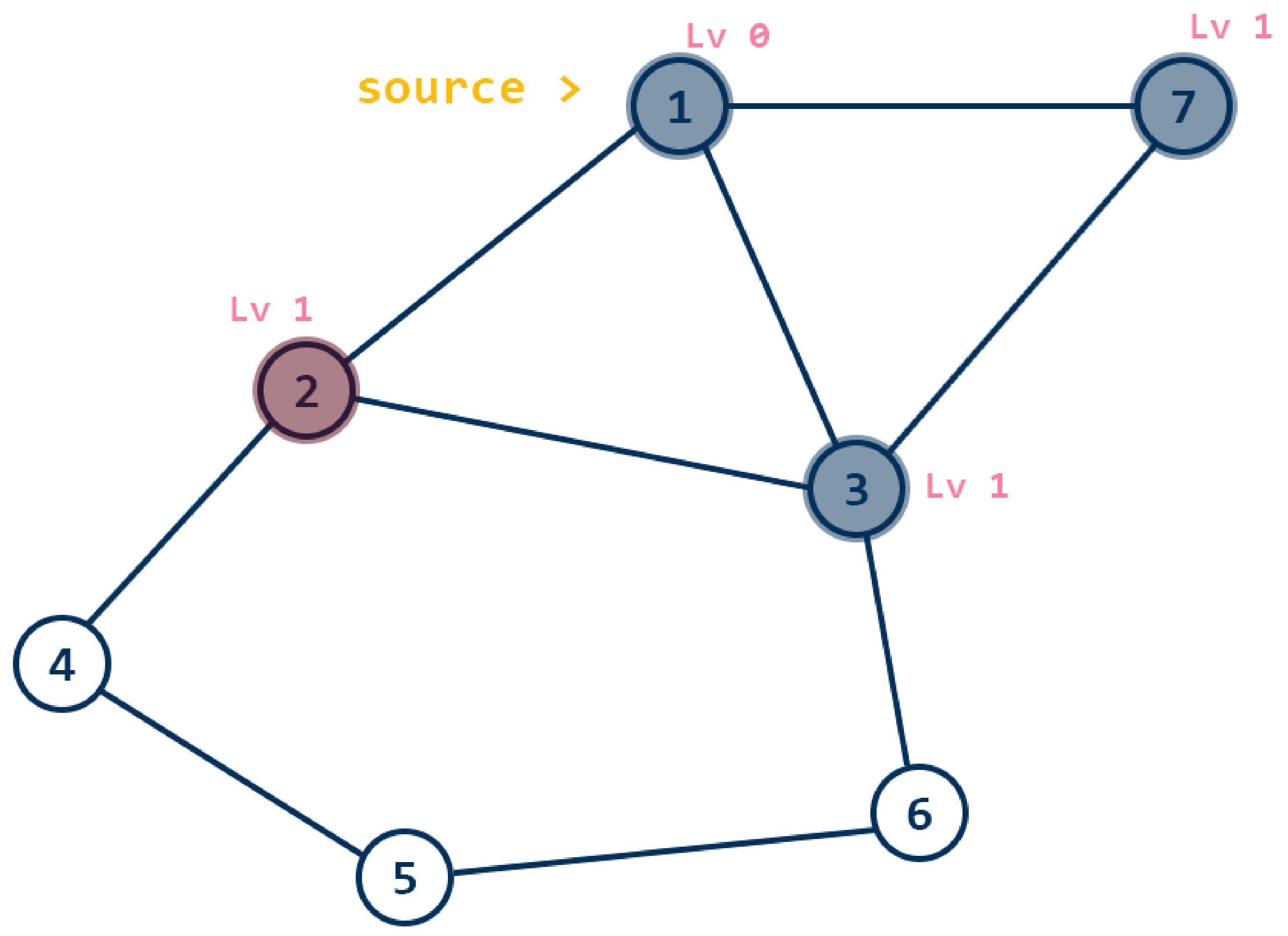
```
queue < int > Q;
```

2	3	7				

```
// pushed (7)
```

1

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

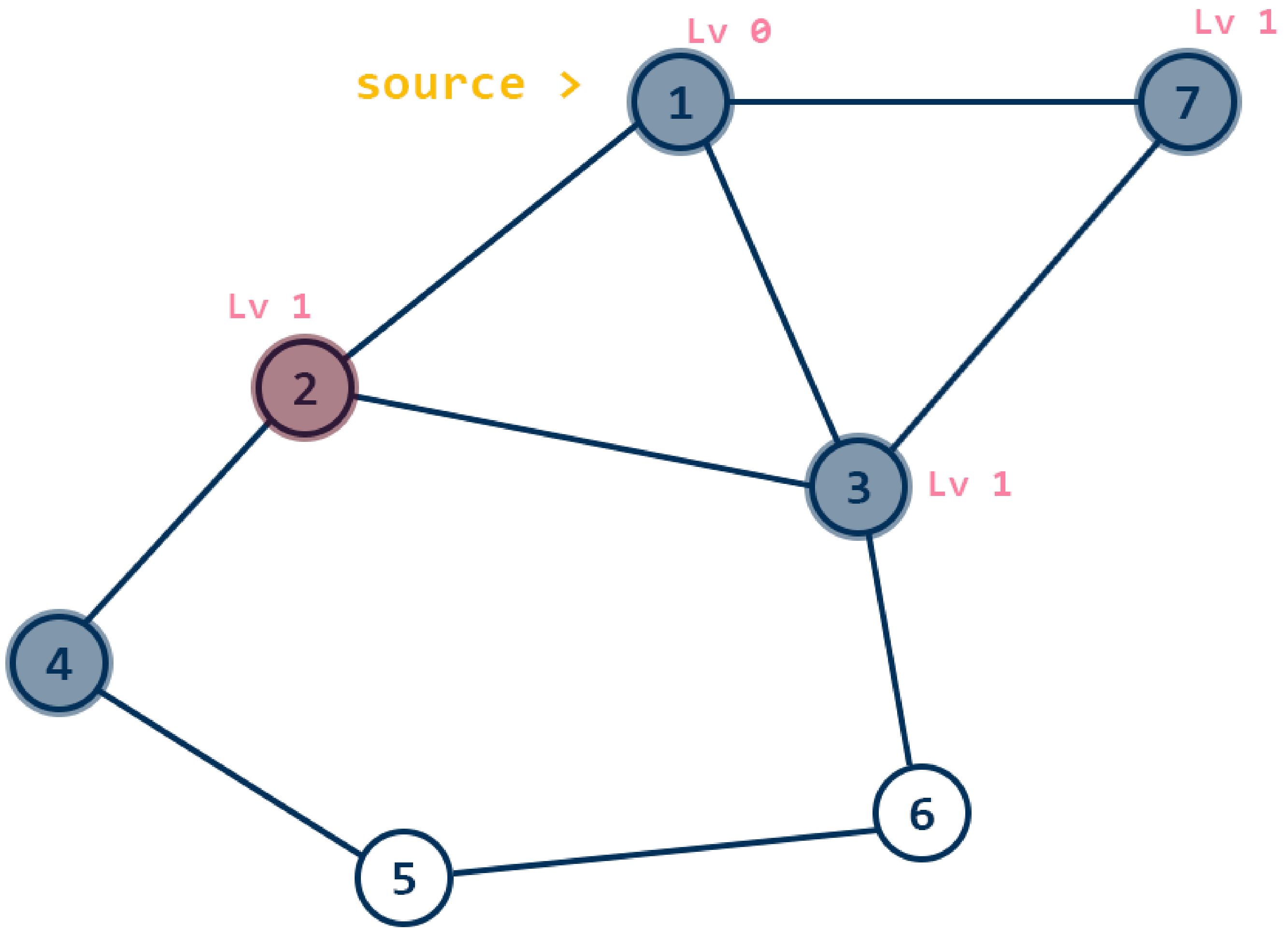
```
queue < int > Q;
```

3	7					

```
// popped front (2)
```

2

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

```
queue < int > Q;
```

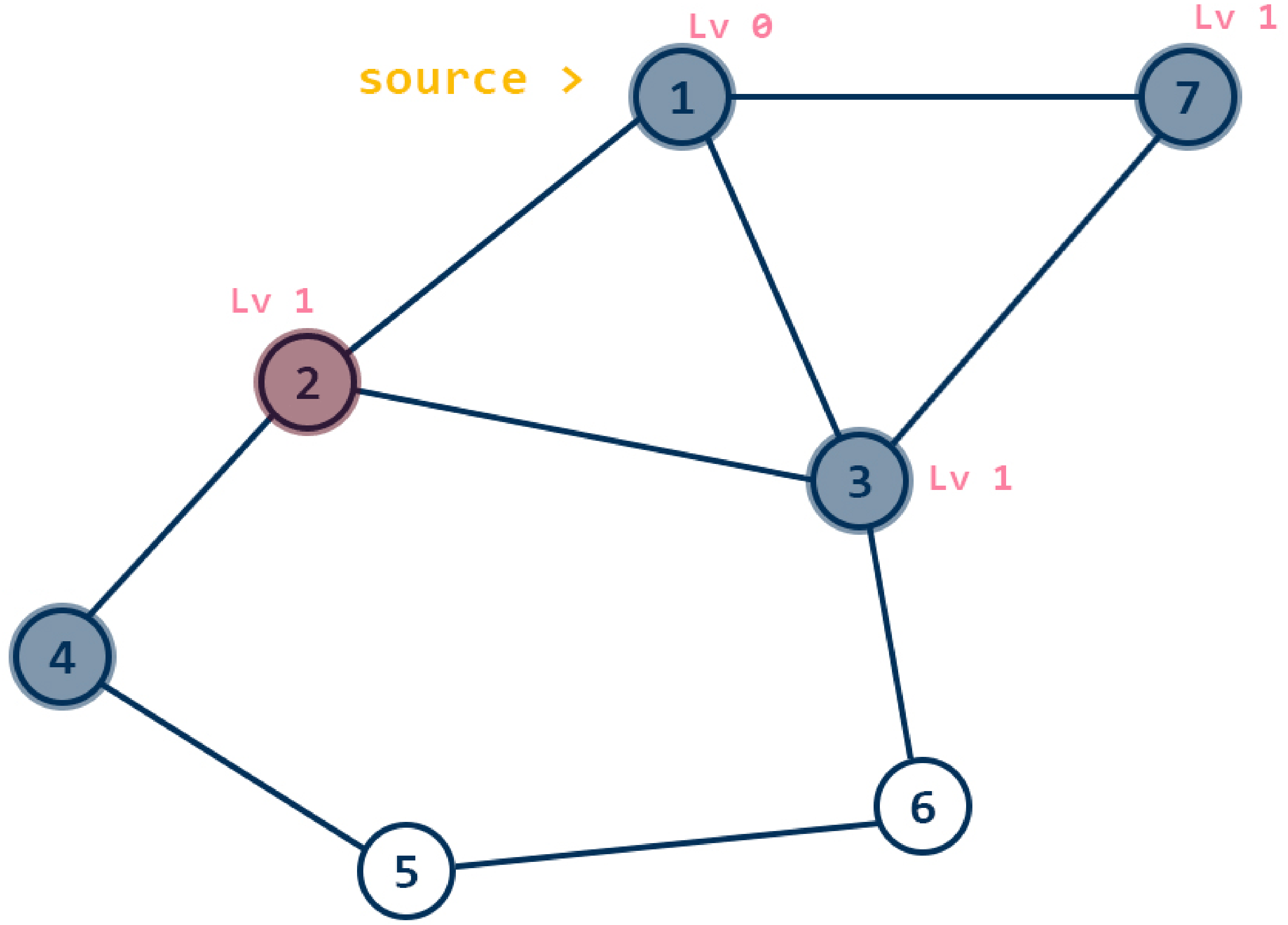
3	7

```
// popped front (2)
```

2

#csspree Online

BFS Simulation



distance[next] = distance[current] + edgeCost

```
int distance[MX];
```

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

```
bool visited[MX];
```

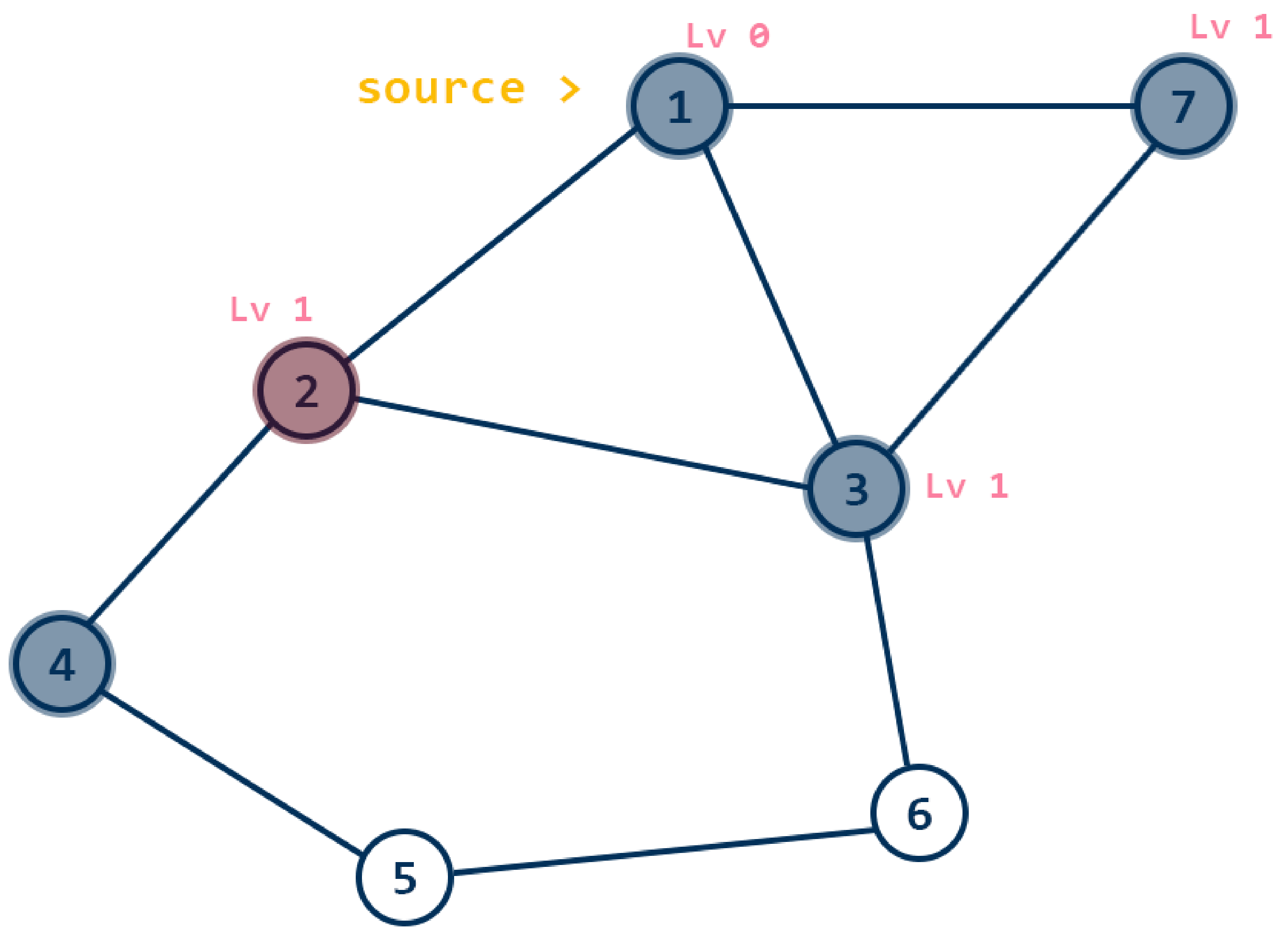
1	1	1	1	0	0	1
1	2	3	4	5	6	7

```
queue < int > Q;
```

3	7					

2

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

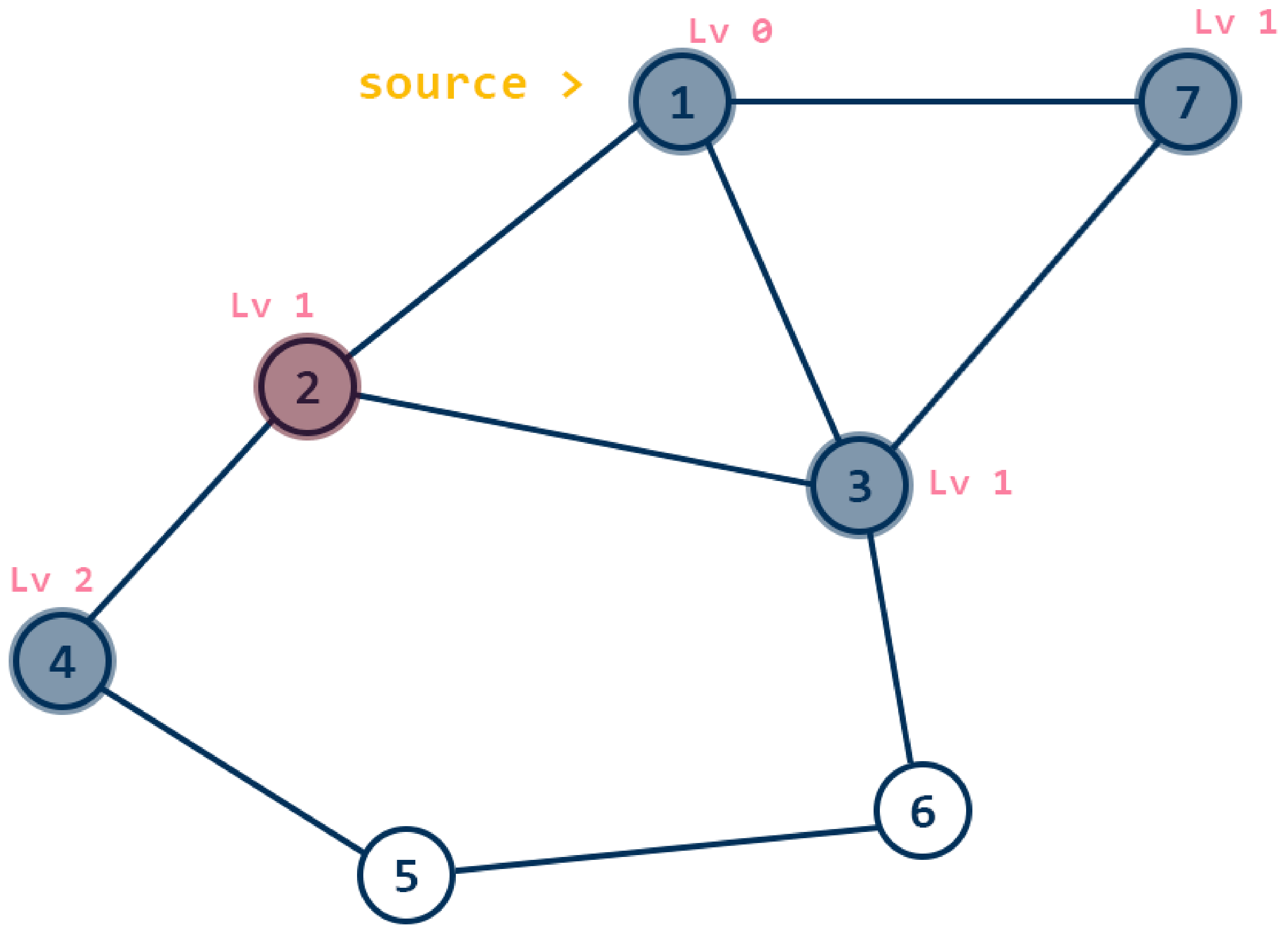
```
queue < int > Q;
```

3	7	4				

```
// pushed (4)
```

2

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

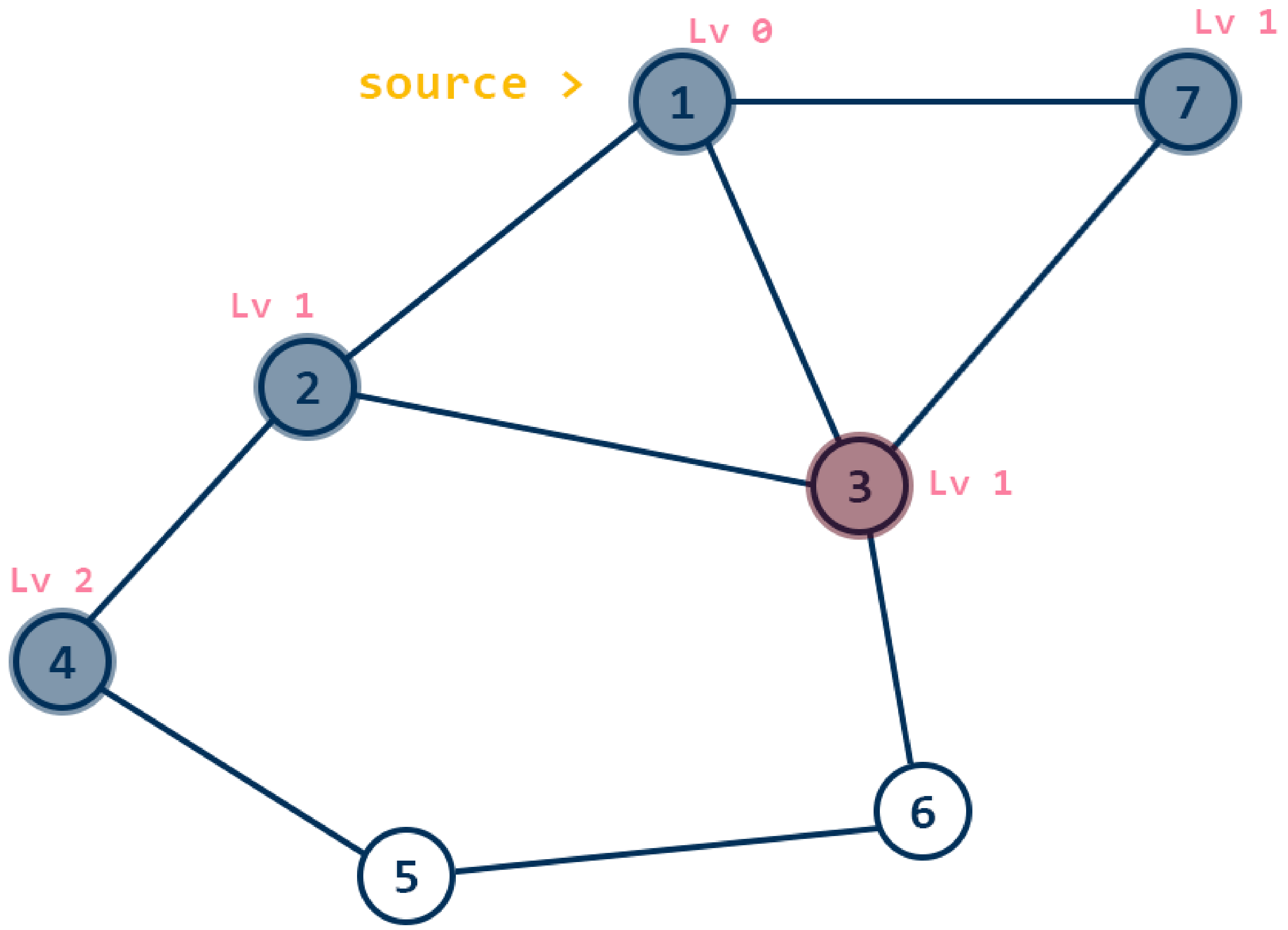
1	2	3	4	5	6	7
1	1	1	1	0	0	1

```
queue < int > Q;
```

3	7	4				

2

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

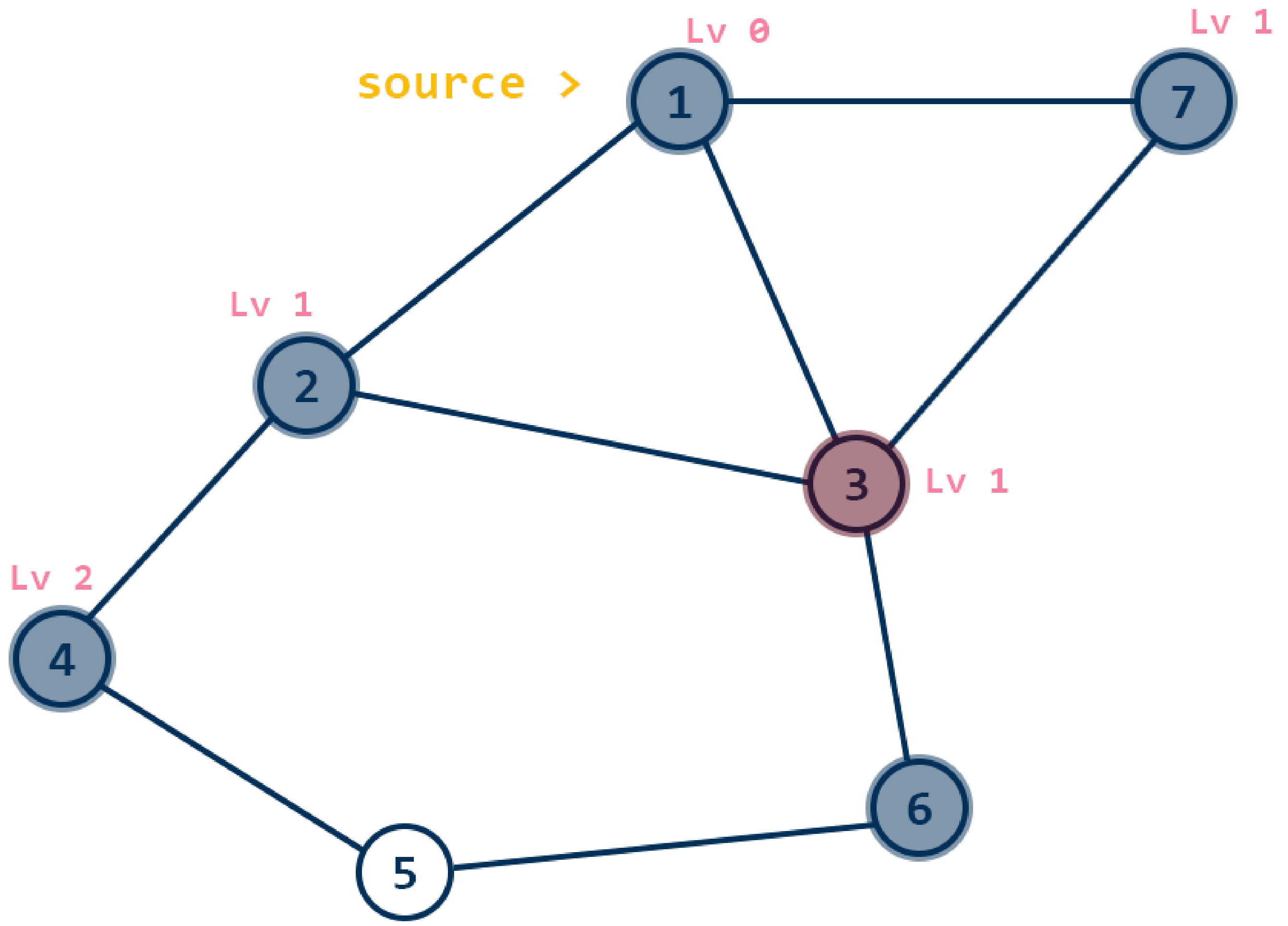
```
queue < int > Q;
```

7	4					

```
// popped front (3)
```

3

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

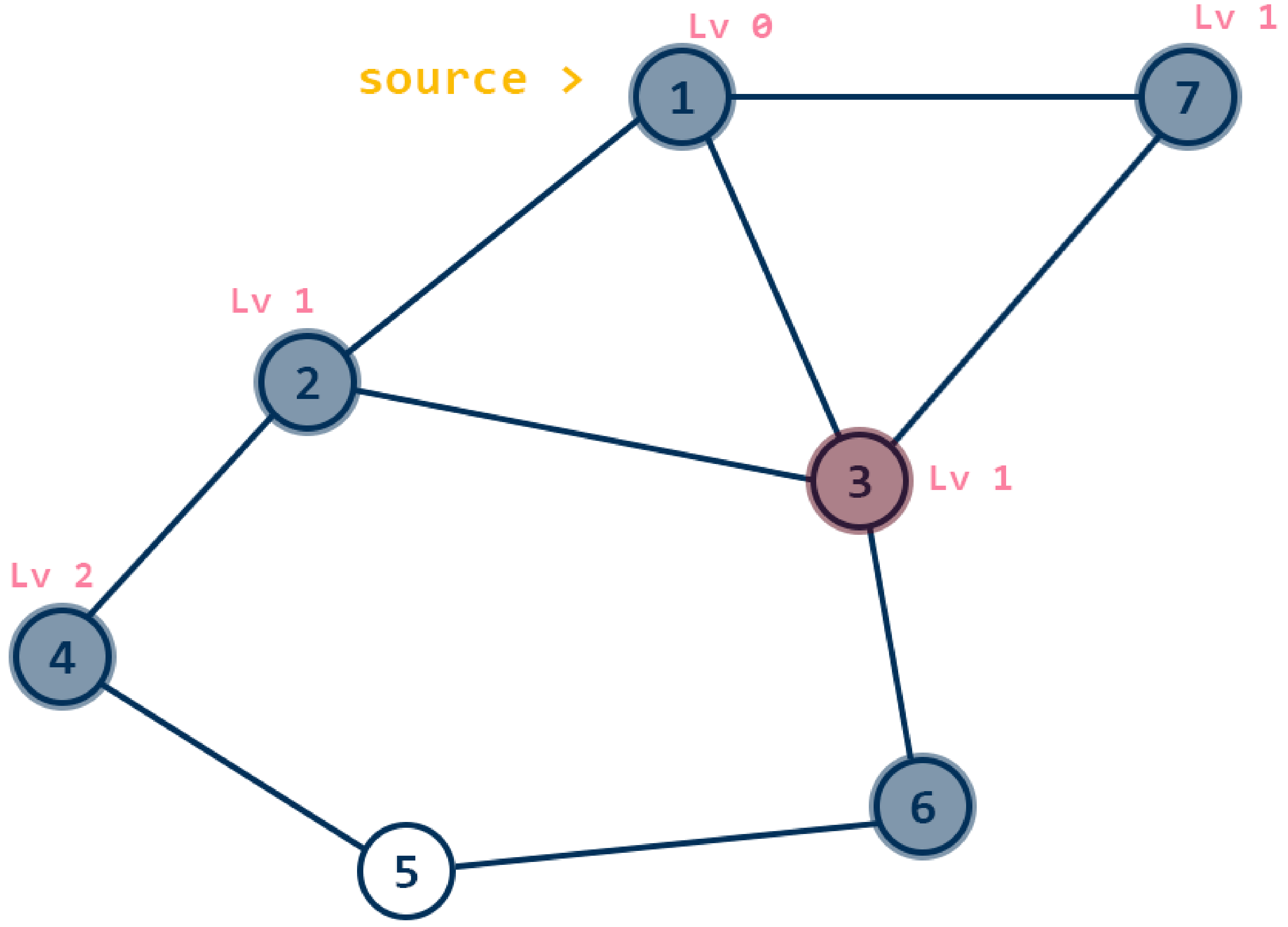
1	1	1	1	0	1	1
1	2	3	4	5	6	7

```
queue < int > Q;
```

7	4					

3

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

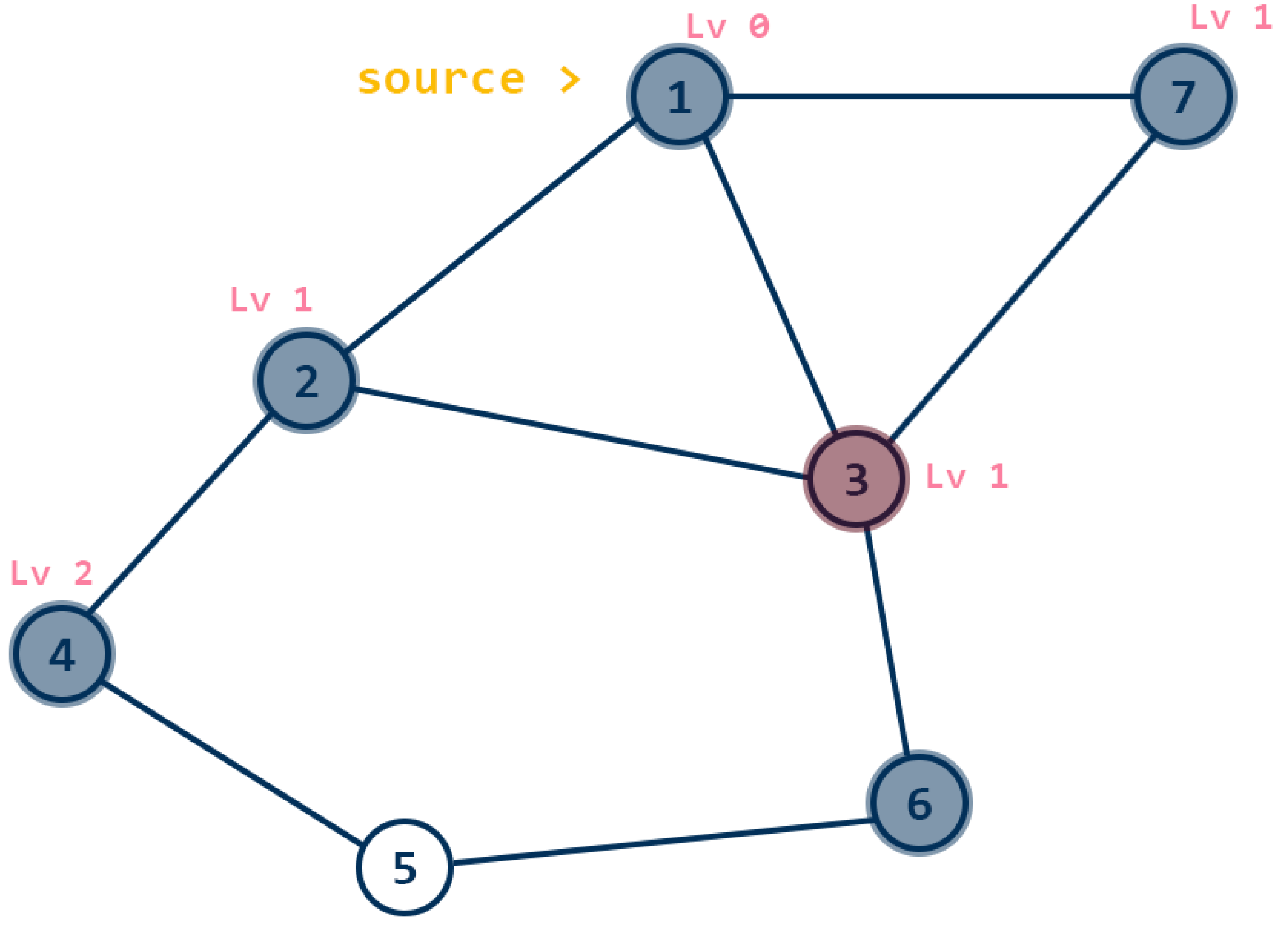
1	2	3	4	5	6	7
1	1	1	1	0	1	1

```
queue < int > Q;
```

7	4					

3

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

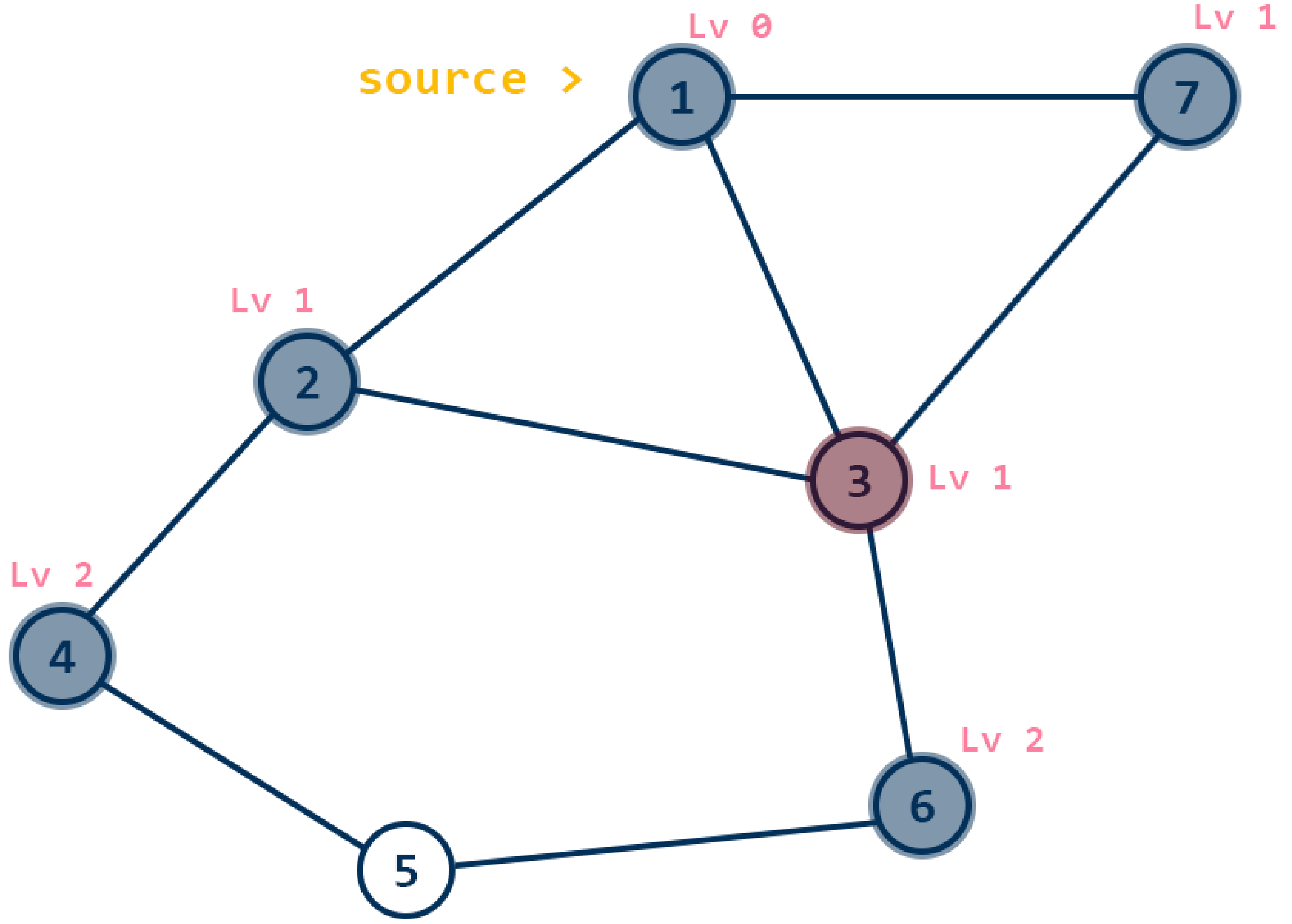
```
queue < int > Q;
```

7	4	6				

```
// pushed (6)
```

3

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

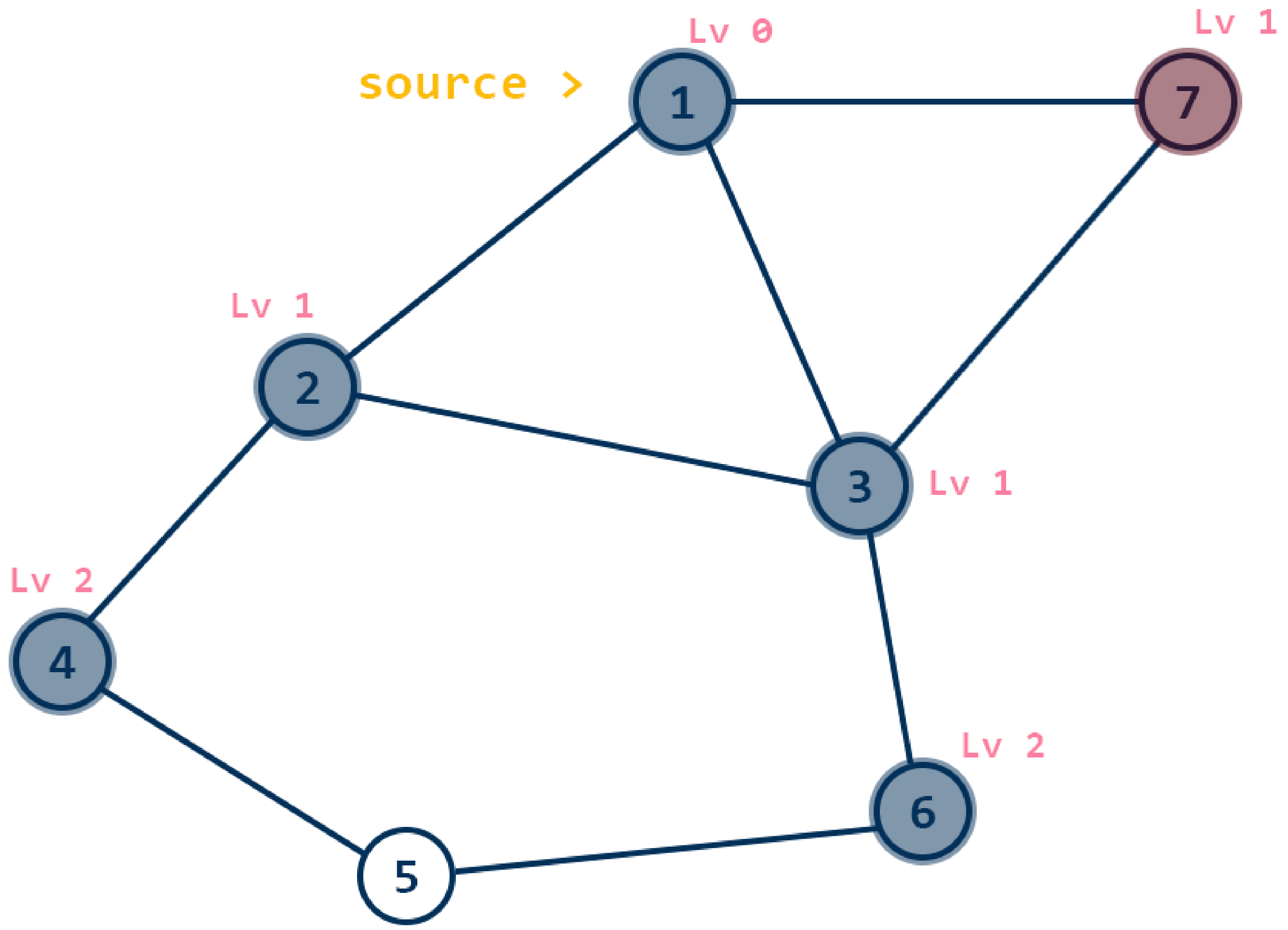
```
queue < int > Q;
```

7	4	6				

```
// pushed (6)
```

3

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

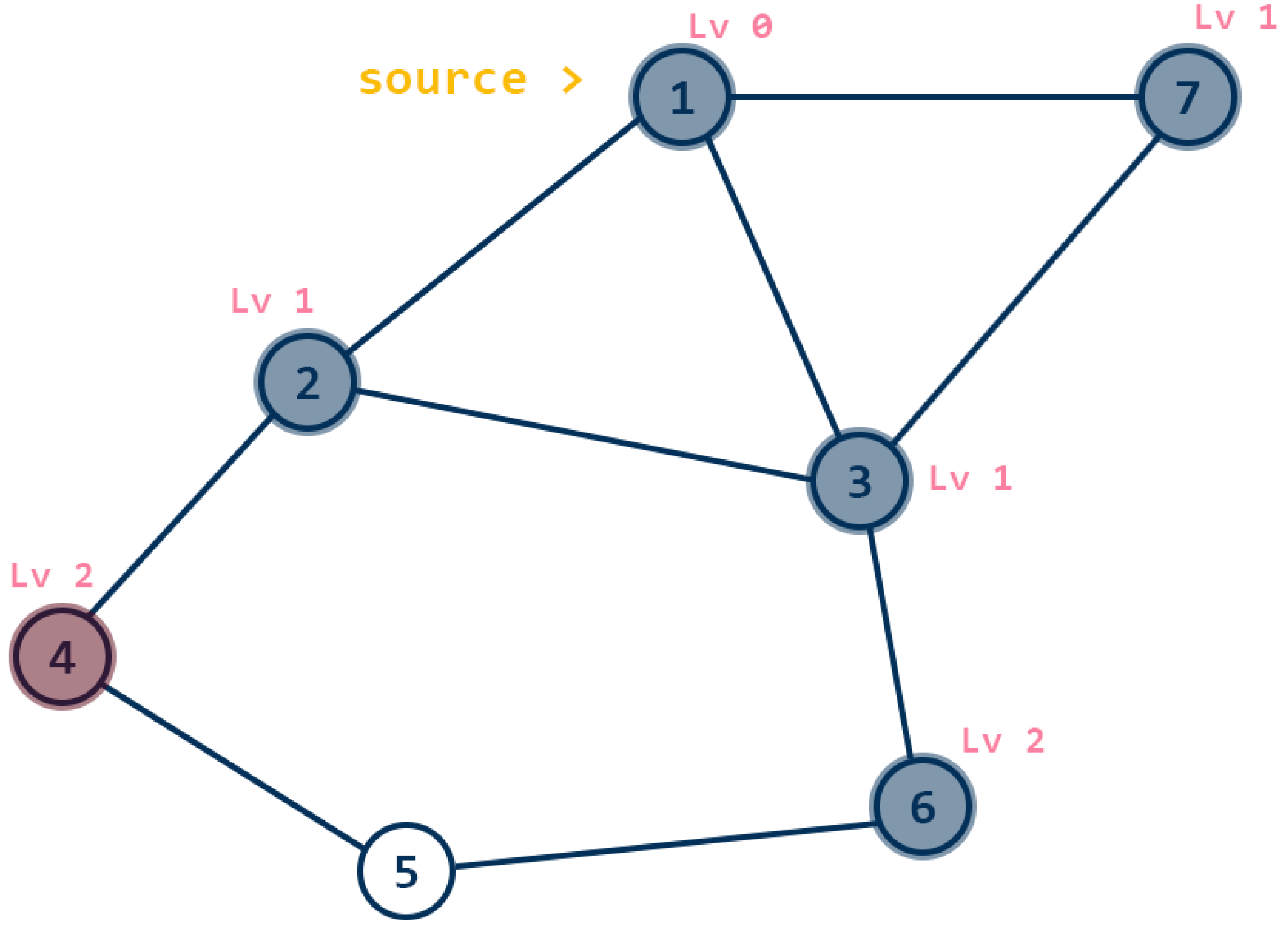
```
queue < int > Q;
```

4	6					

```
// popped front (7)
```

7

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

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

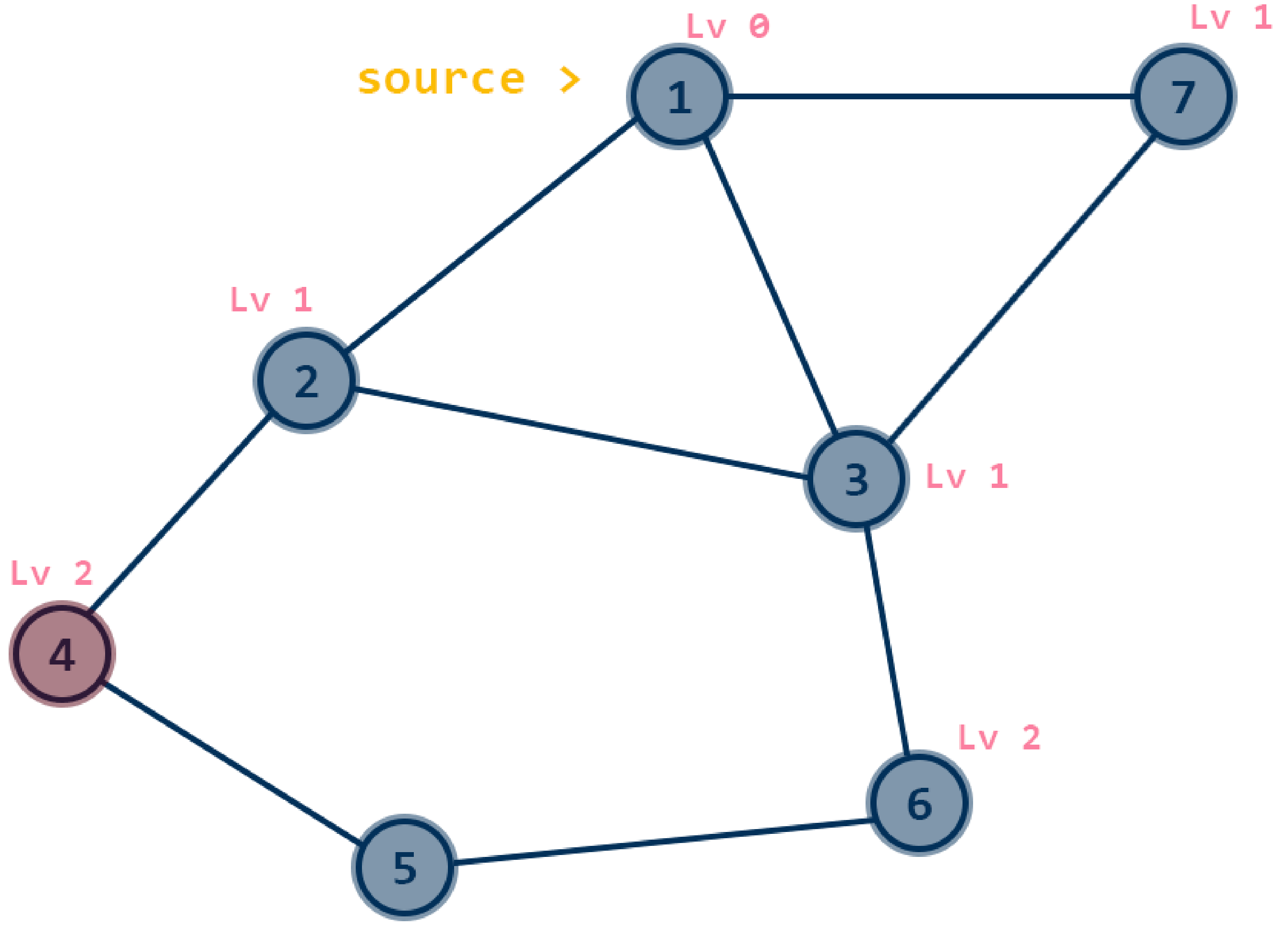
```
queue < int > Q;
```

6						

```
// popped front (4)
```

4

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

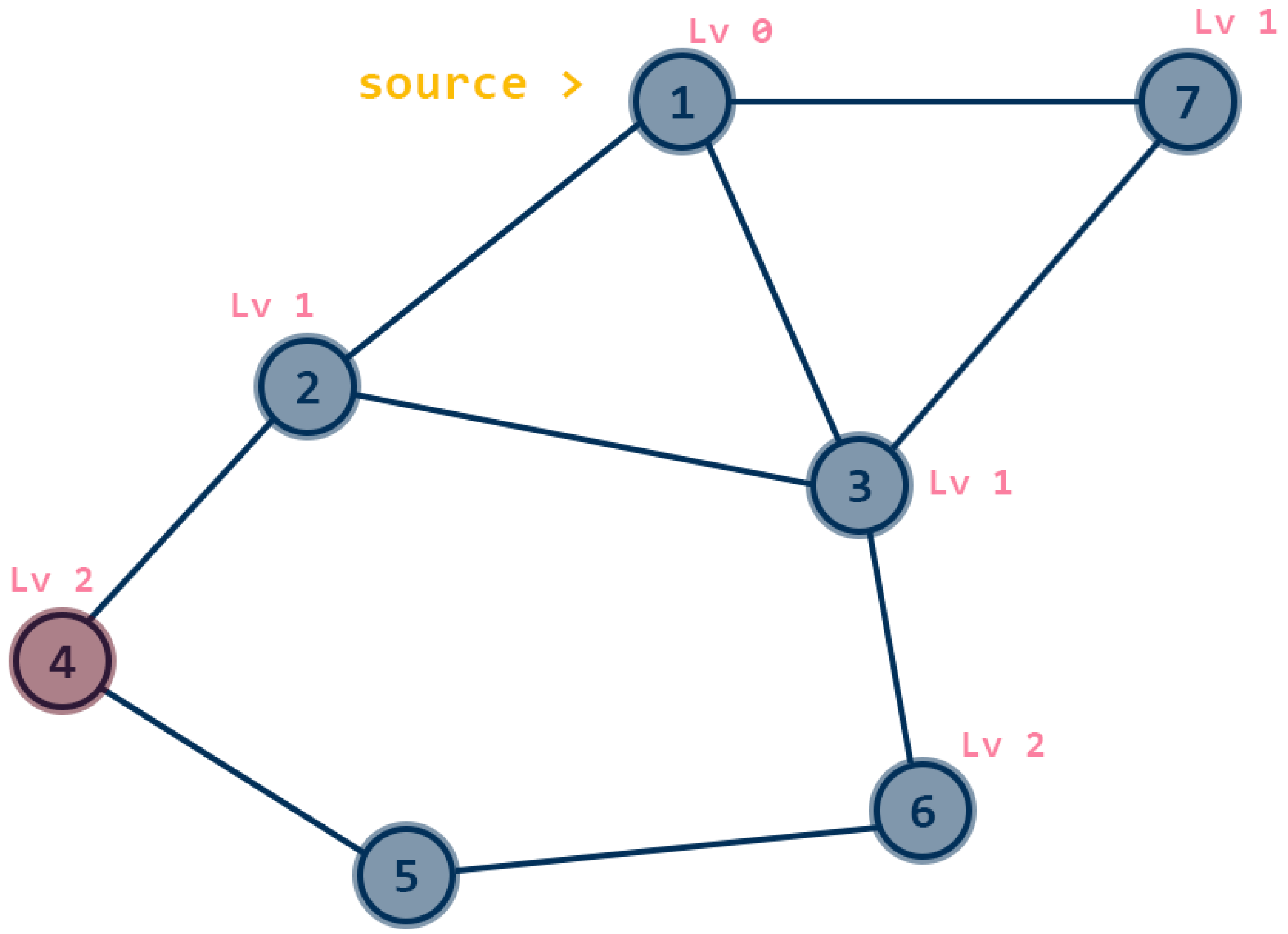
1	1	1	1	1	1	1
1	2	3	4	5	6	7

```
queue < int > Q;
```

6						

4

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

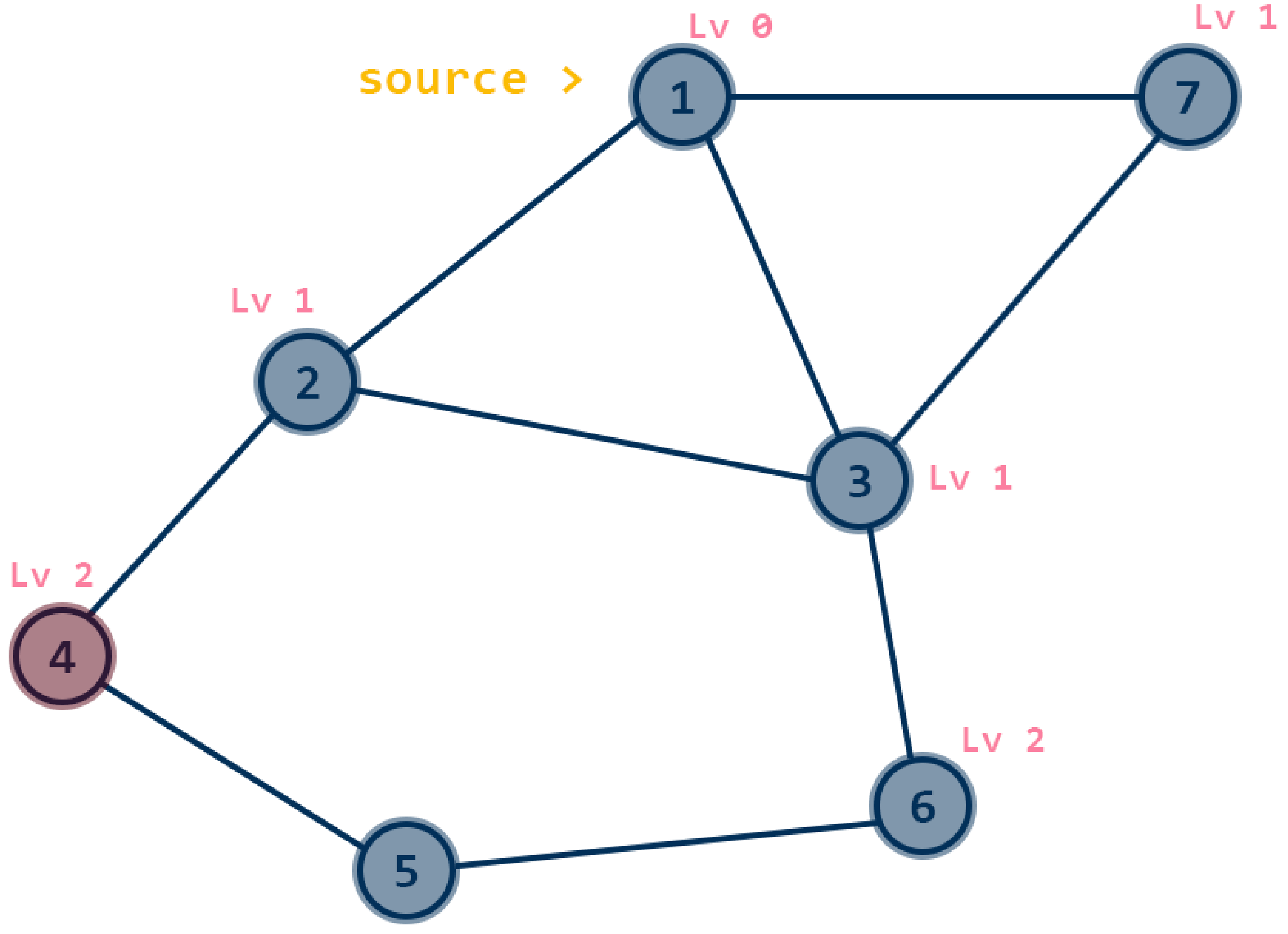
1	1	1	1	1	1	1
1	2	3	4	5	6	7

```
queue < int > Q;
```

6						

4

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

1	2	3	4	5	6	7
1	1	1	1	1	1	1

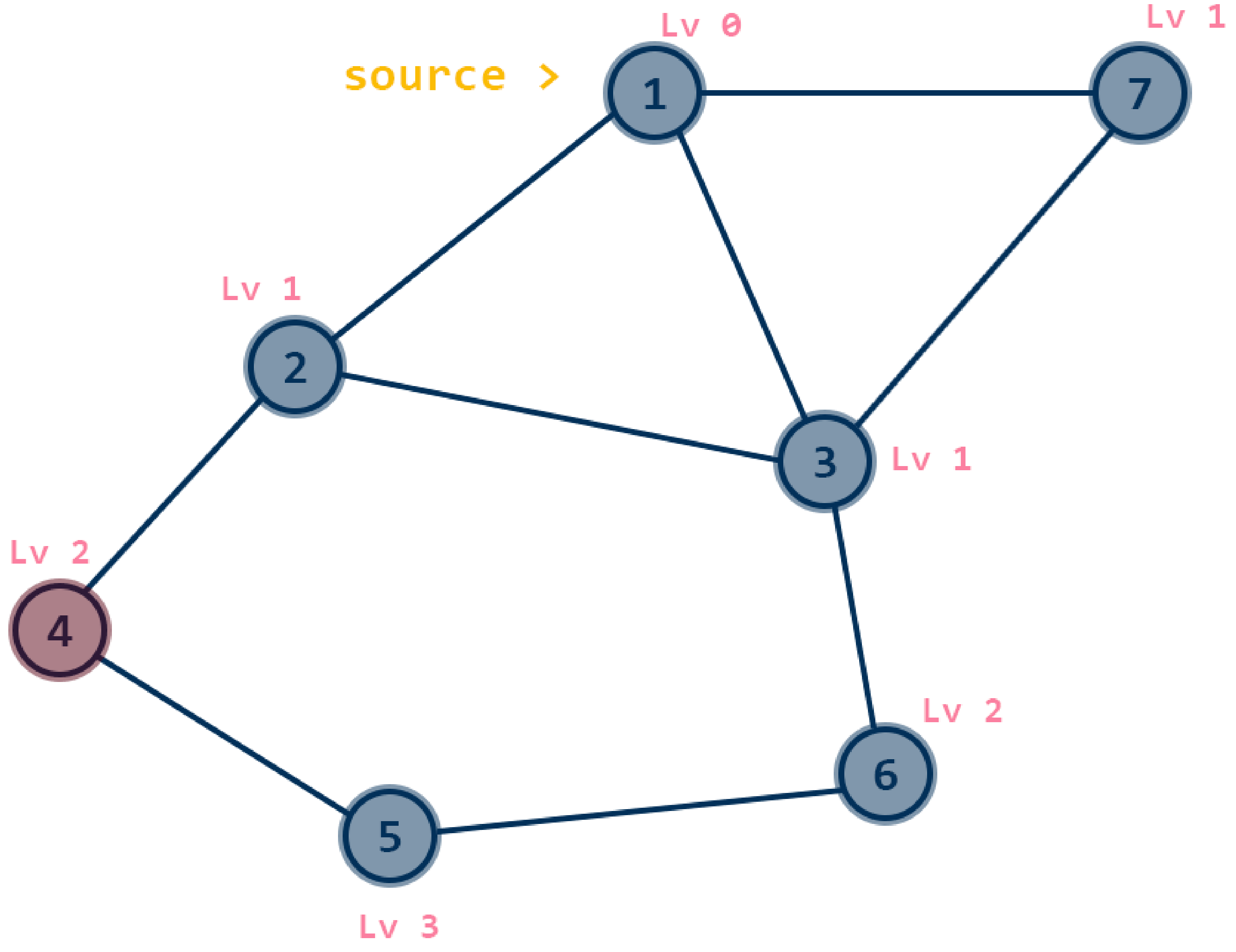
```
queue < int > Q;
```

6	5					

```
// pushed (5)
```

4

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

1	1	1	1	1	1	1
1	2	3	4	5	6	7

```
queue < int > Q;
```

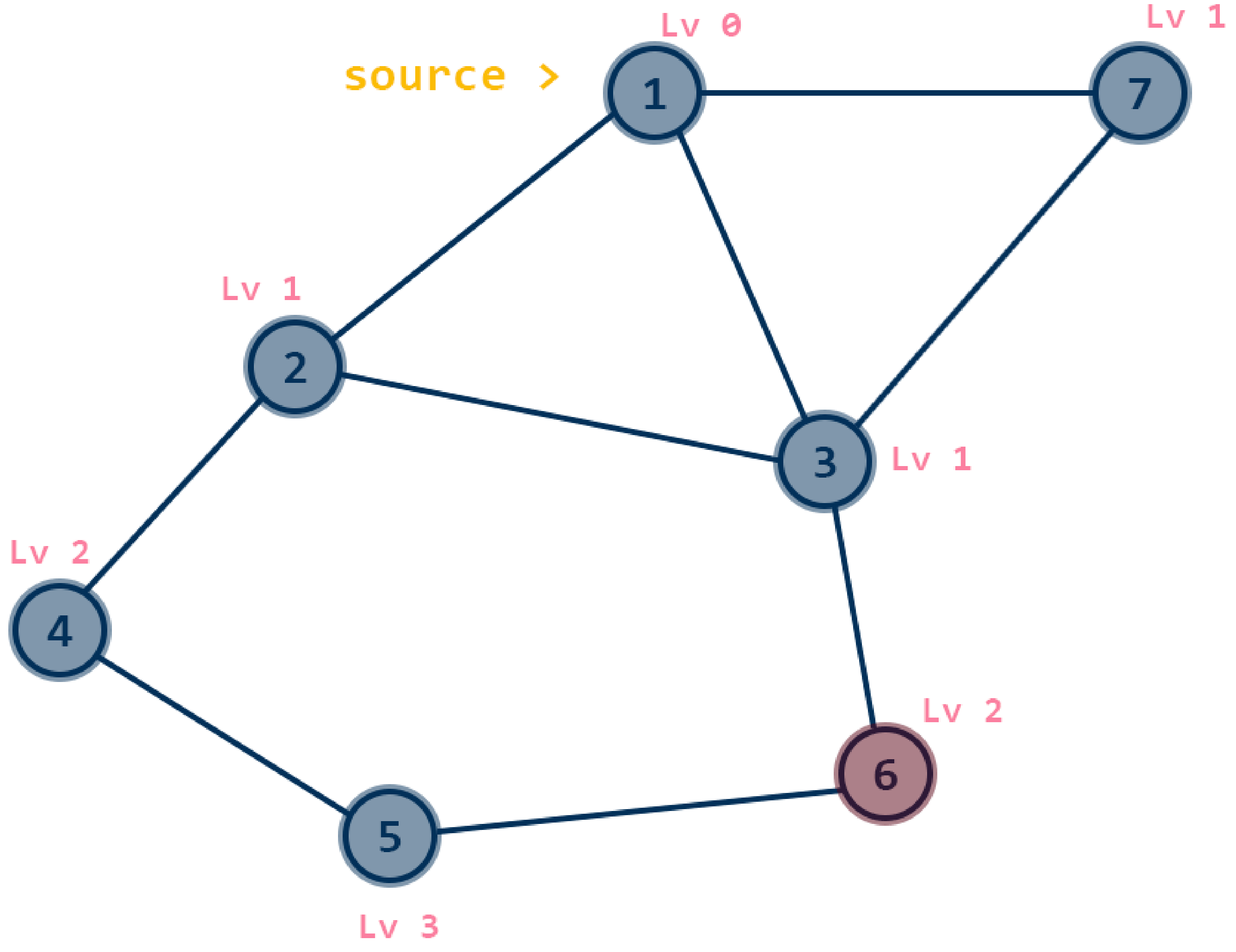
6	5					

```
// pushed (5)
```

4

#csspree Online

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

1	1	1	1	1	1	1
1	2	3	4	5	6	7

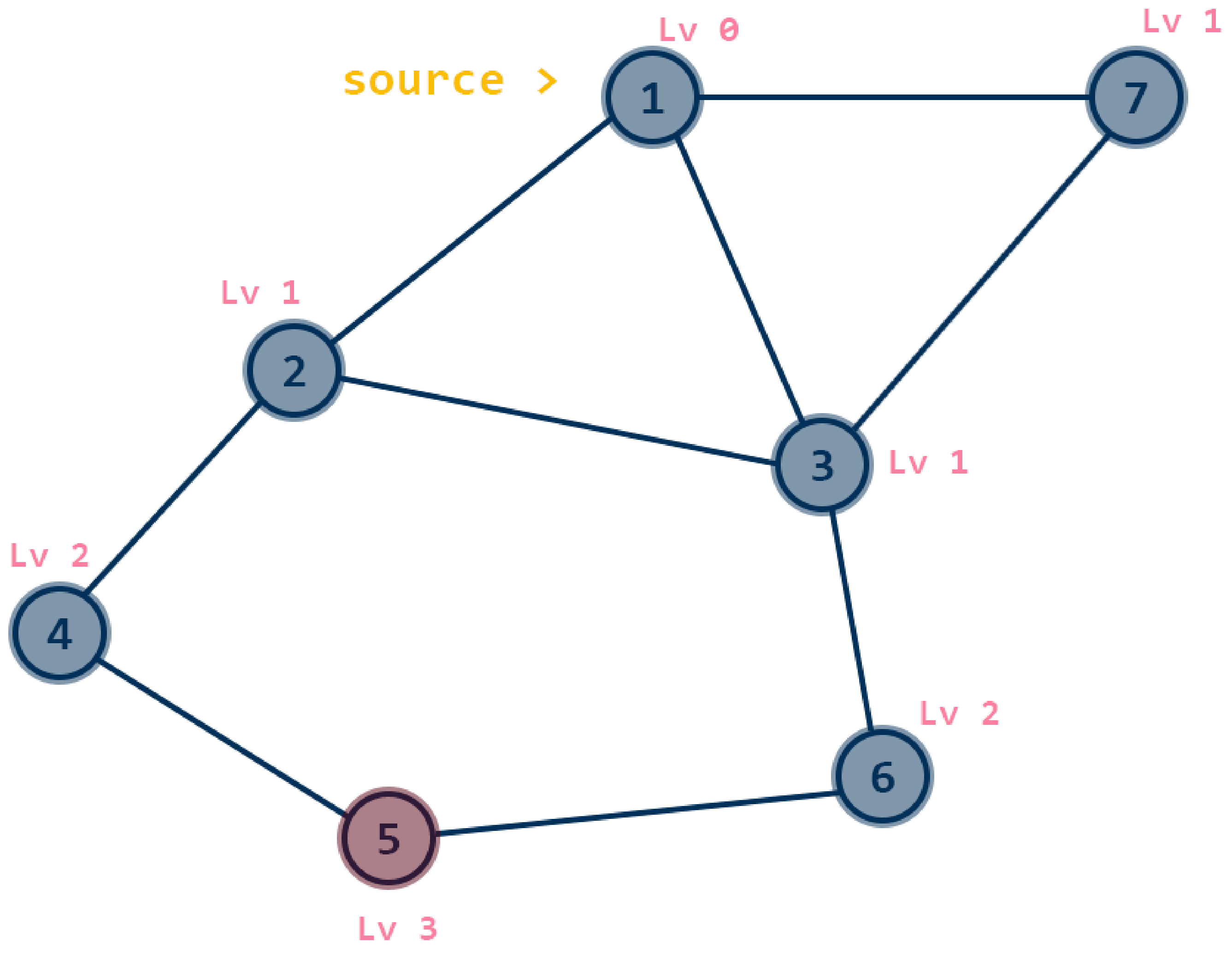
```
queue < int > Q;
```

5						

```
// popped front (6)
```

6

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

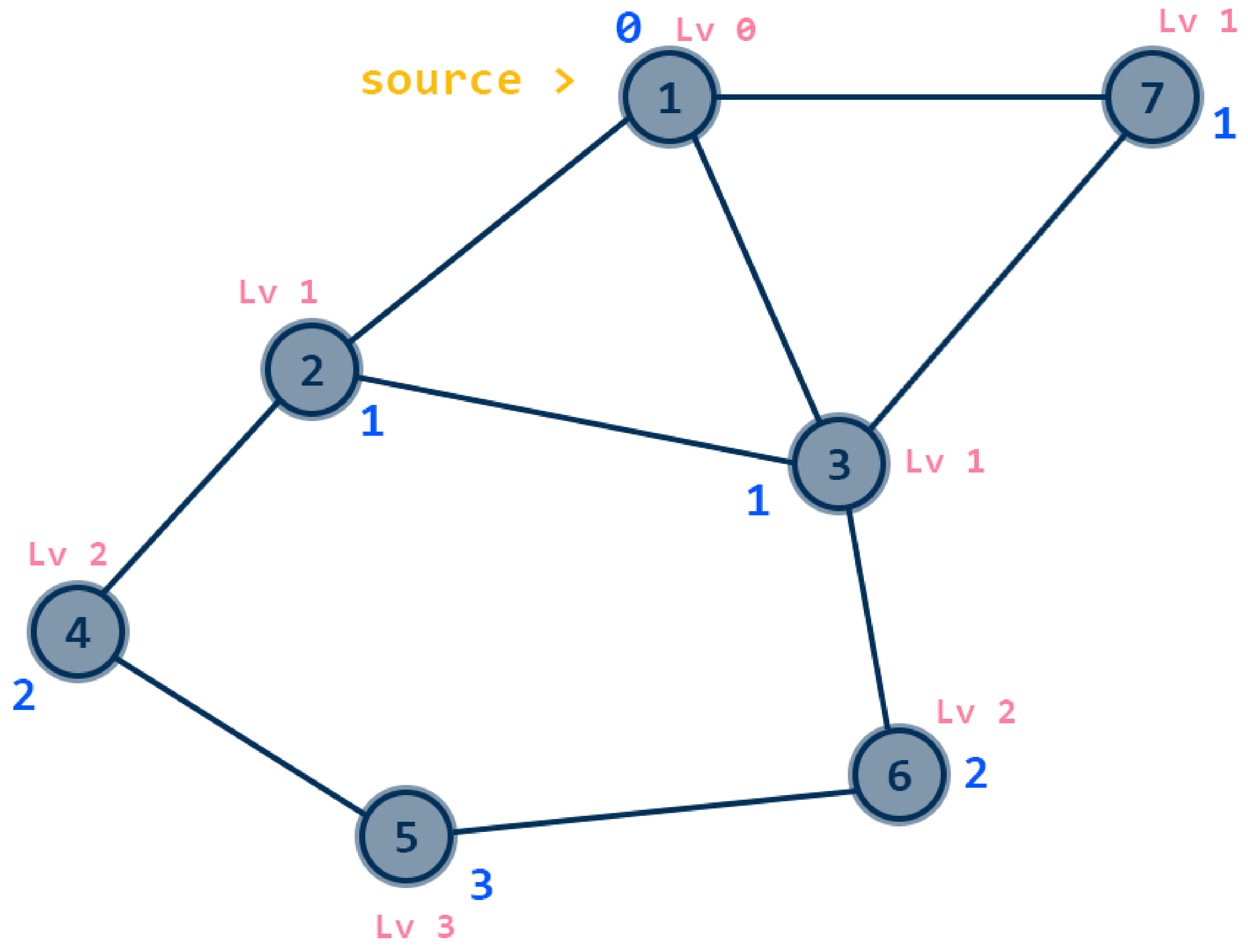
1	1	1	1	1	1	1
1	2	3	4	5	6	7

```
queue < int > Q;
```


```
// popped front (5)
```

5

BFS Simulation



```
int distance[MX];
```

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

```
bool visited[MX];
```

1	1	1	1	1	1	1
1	2	3	4	5	6	7

```
queue < int > Q;
```
