# LINK STATE ROUTING PROTOCOL

Using dijkstra 's algorithm

Program in Java & explanation

Howard
00000099772
Lecturer – Fransiscus ati Halim

STEP 1:

## WHAT IS THE LINK STATE PROTOCOL?

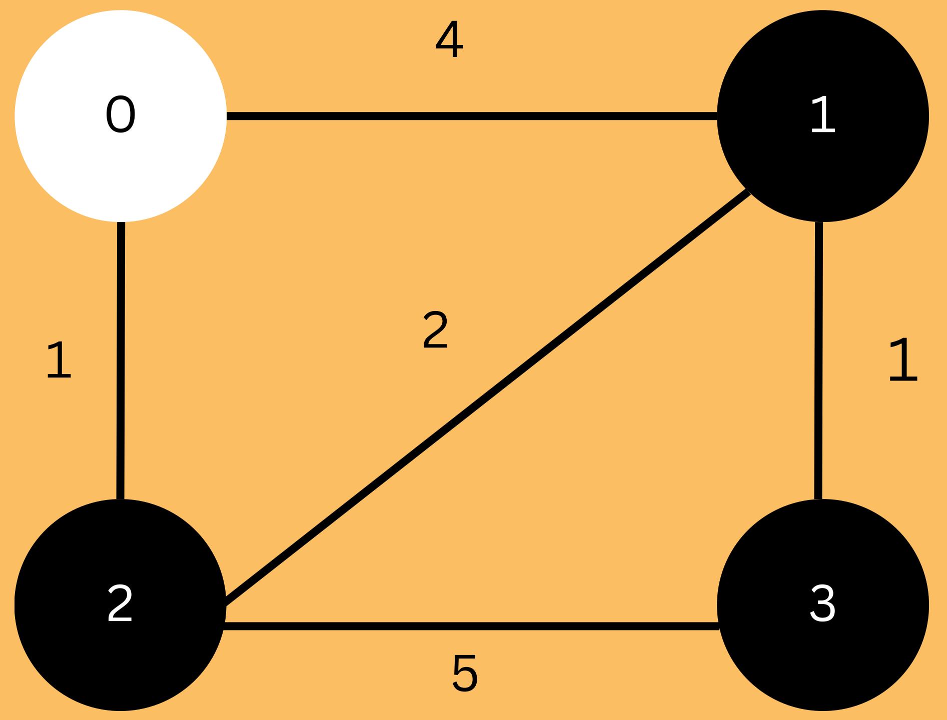
STEP 1:

## WHAT IS THE LINK STATE PROTOCOL?

A link-state protocol is a type of routing protocol used in computer networks that builds a complete map of the network topology to determine the best path for data packets.

# JUMLAH NETWORK : 4

## DARI NETWORK 0



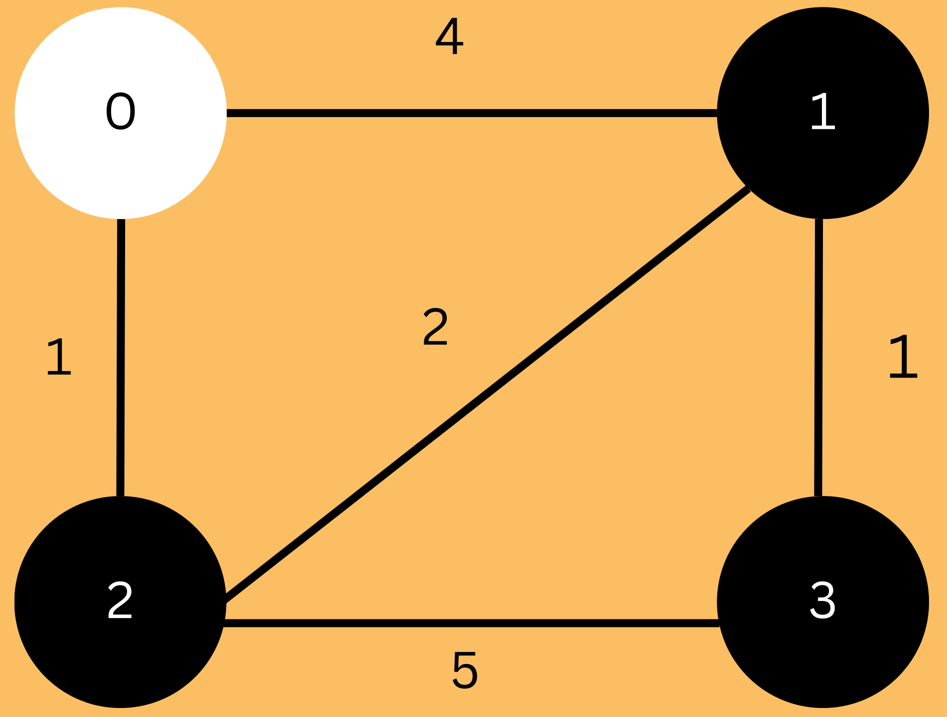
**distances : [inf, inf, inf, inf ]**

**distances[0] : 0**

# JUMLAH NETWORK : 4

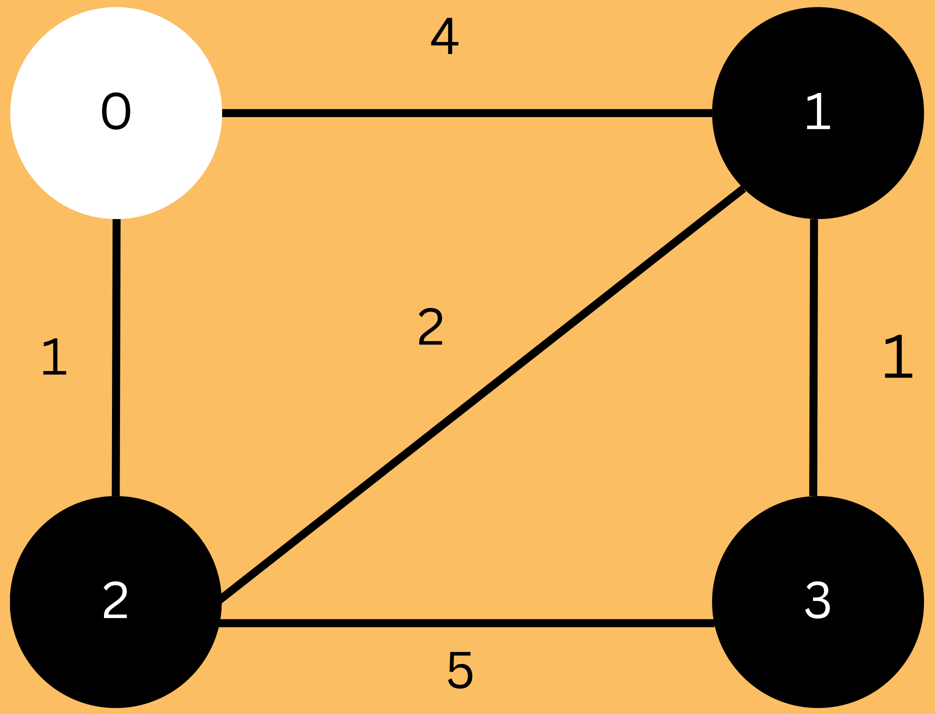
## DARI NETWORK 0

distances : [0, inf, inf, inf ]



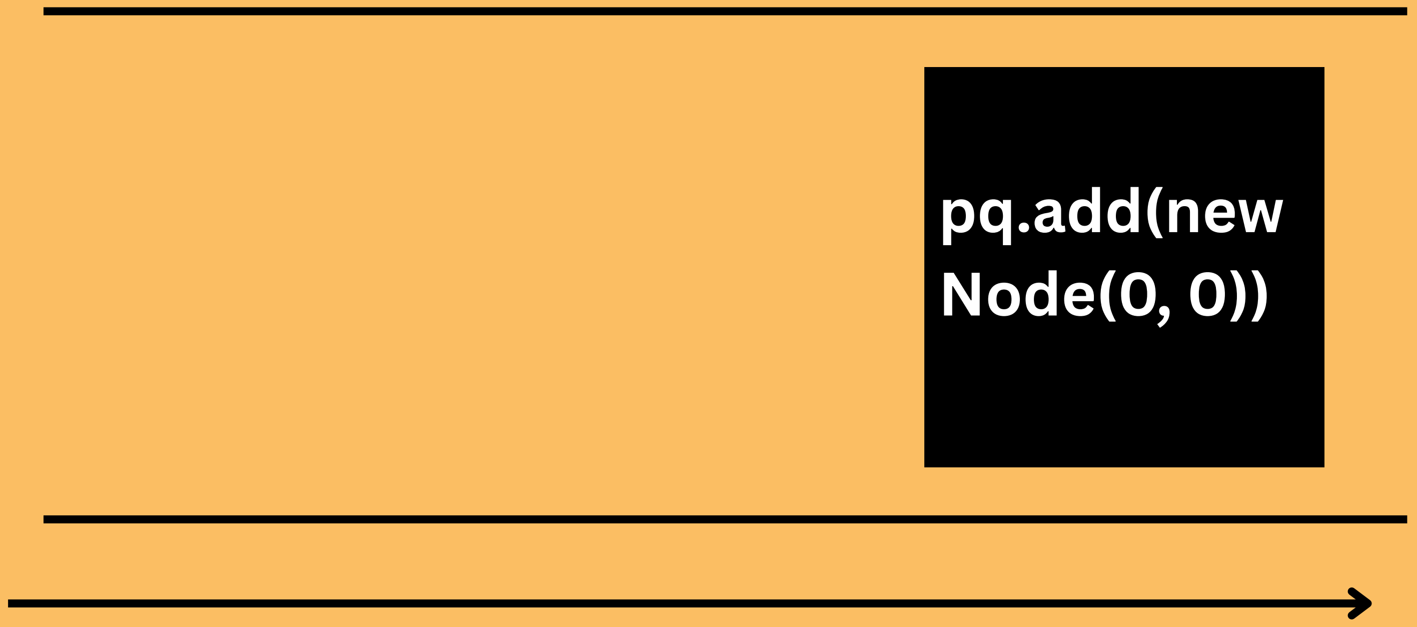
# JUMLAH NETWORK : 4

## DARI NETWORK 0



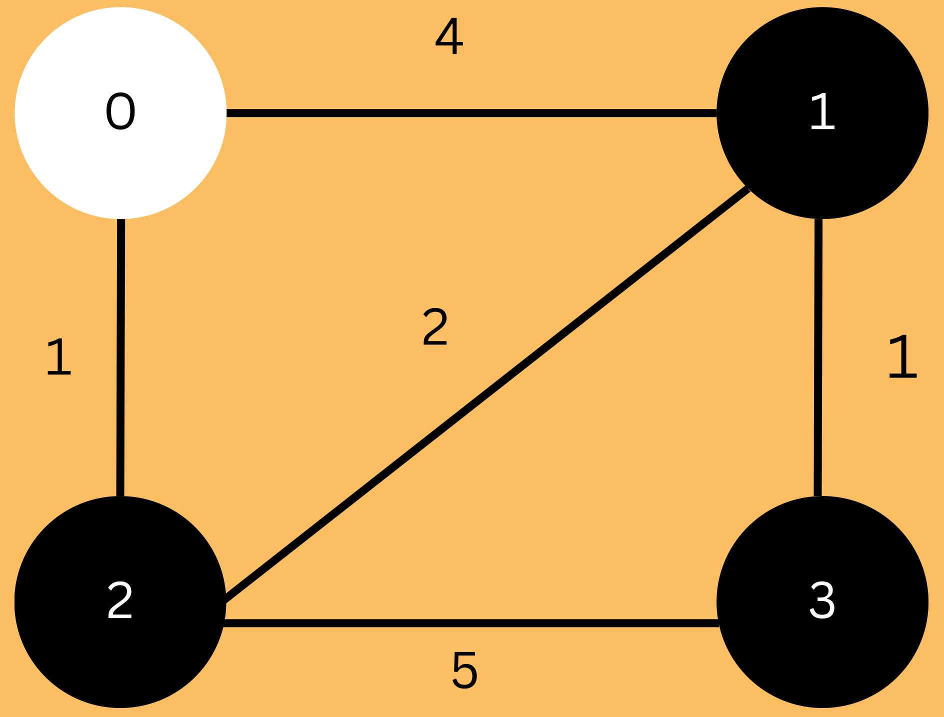
distances : [0, inf, inf, inf ]

pq.add(new Node(0, 0))



# JUMLAH NETWORK : 4

## DARI NETWORK 0

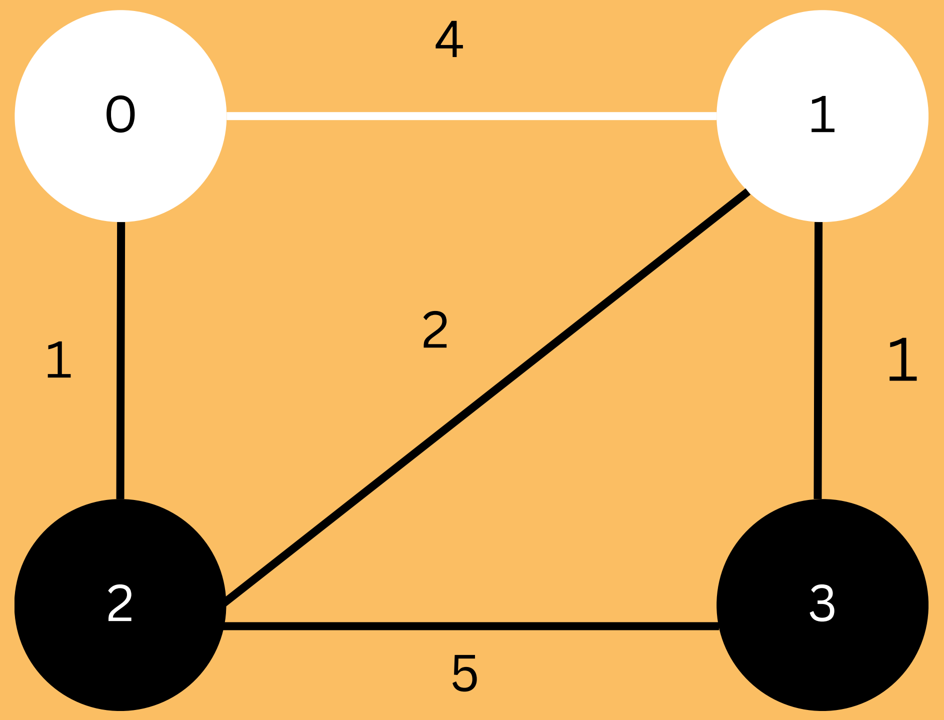


distances : [0, inf, inf, inf ]

pq.poll()

# JUMLAH NETWORK : 4

## DARI NETWORK 0



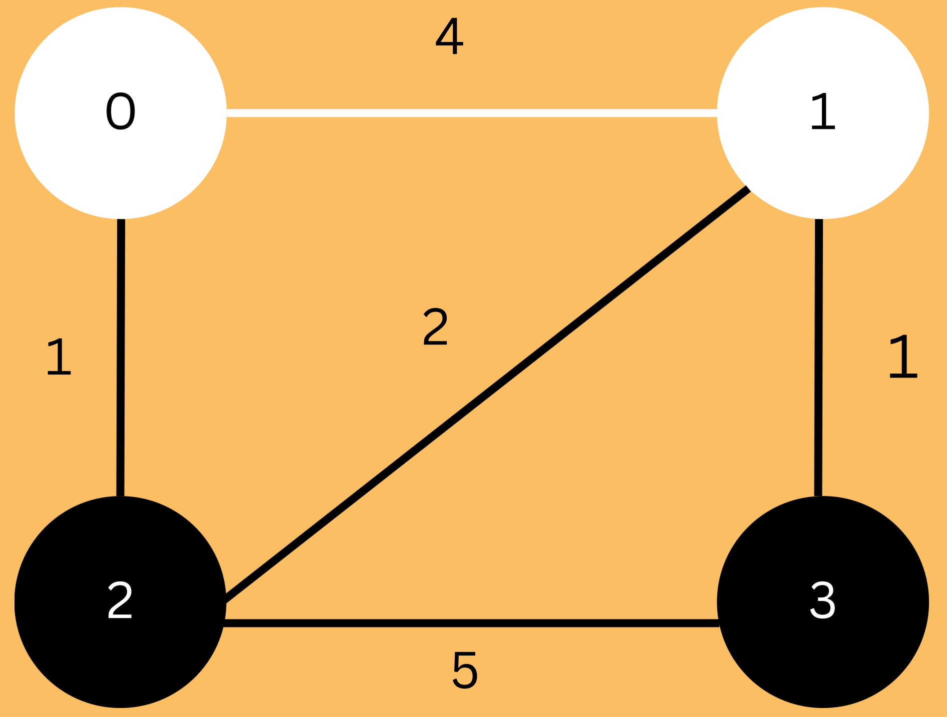
**distances : [0, inf, inf, inf ]**

**u = 0**
**v = 1** (edge.dest)
**weight = 4** (edge.weight)
**distances[0** (u) **] = 0**

# JUMLAH NETWORK : 4

## DARI NETWORK 0

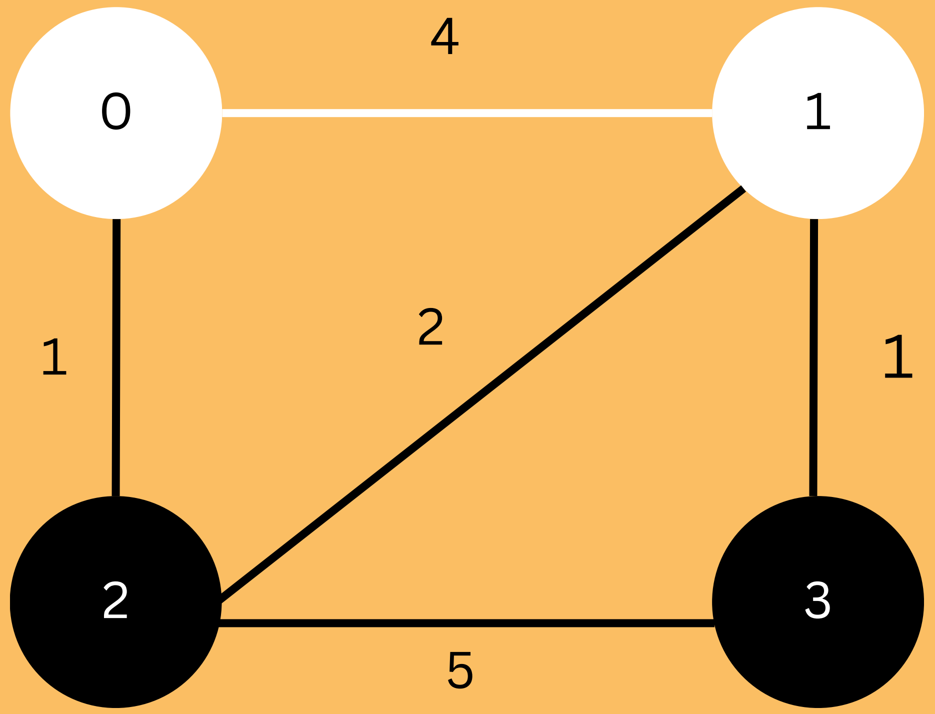
**distances : [0, inf, inf, inf ]**



**u = 0**
**v = 1** (edge.dest)
**weight = 4** (edge.weight)
**distances[0** (u) **] = 0**
**0 + 4 < inf (distances[1** (v) **]) ? True**

# JUMLAH NETWORK : 4

## DARI NETWORK 0



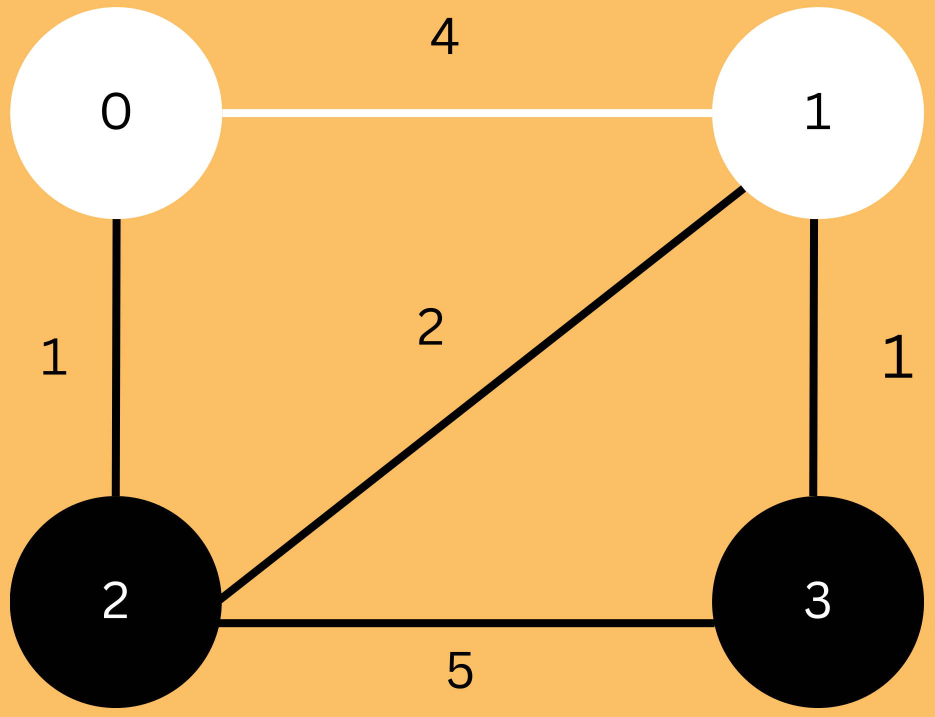
**distances : [0, inf, inf, inf ]**

**u = 0**
**v = 1** (edge.dest)
**weight = 4** (edge.weight)
**distances[0** (u) **] = 0**
**0 + 4 < inf (distances[1** (v) **]) ? True**

**distances[1 (v) ] = 0 + 4 = 4**

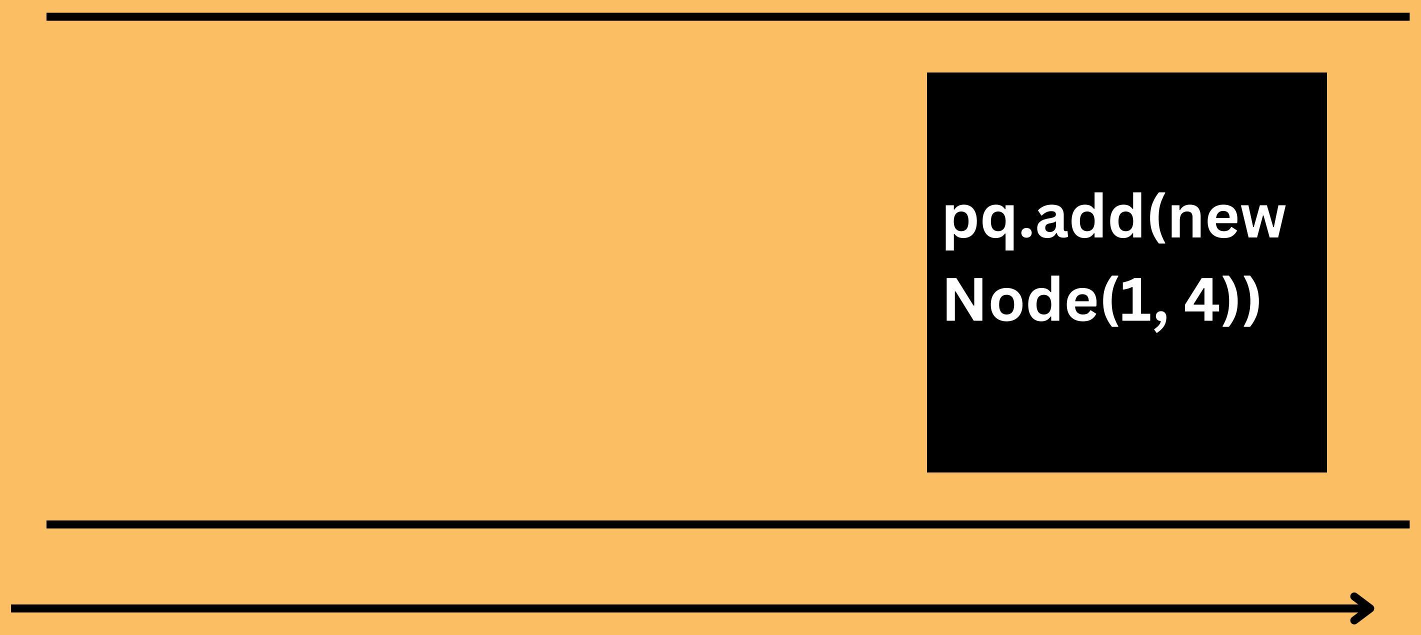
# JUMLAH NETWORK : 4

## DARI NETWORK 0



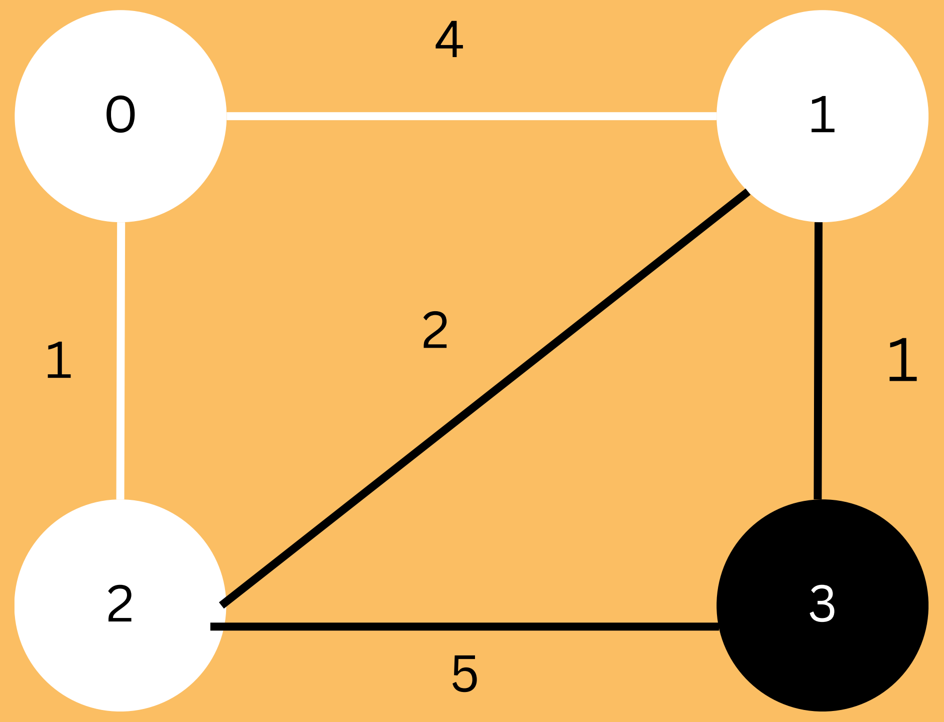
**distances : [0, inf, inf, inf ]**

**u = 0**
**v = 1** (edge.dest)
**weight = 4** (edge.weight)
**distances[0** (u) **] = 0**
**0 + 4 < inf (distances[1** (v) **]) ? True**
**distances[1 (v) ] = 0 + 4 = 4**
**pq.add(new Node(1, 4))**
**distances : [0, 4, inf, inf ]**



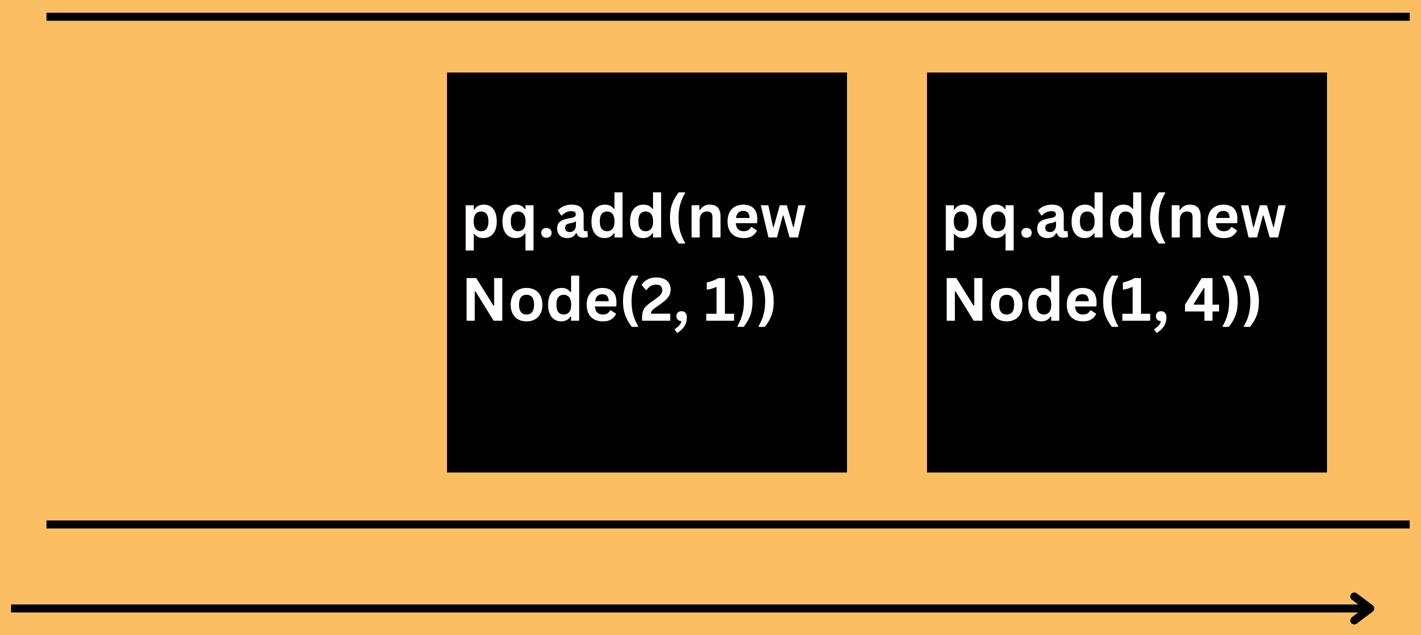
# JUMLAH NETWORK : 4

## DARI NETWORK 0



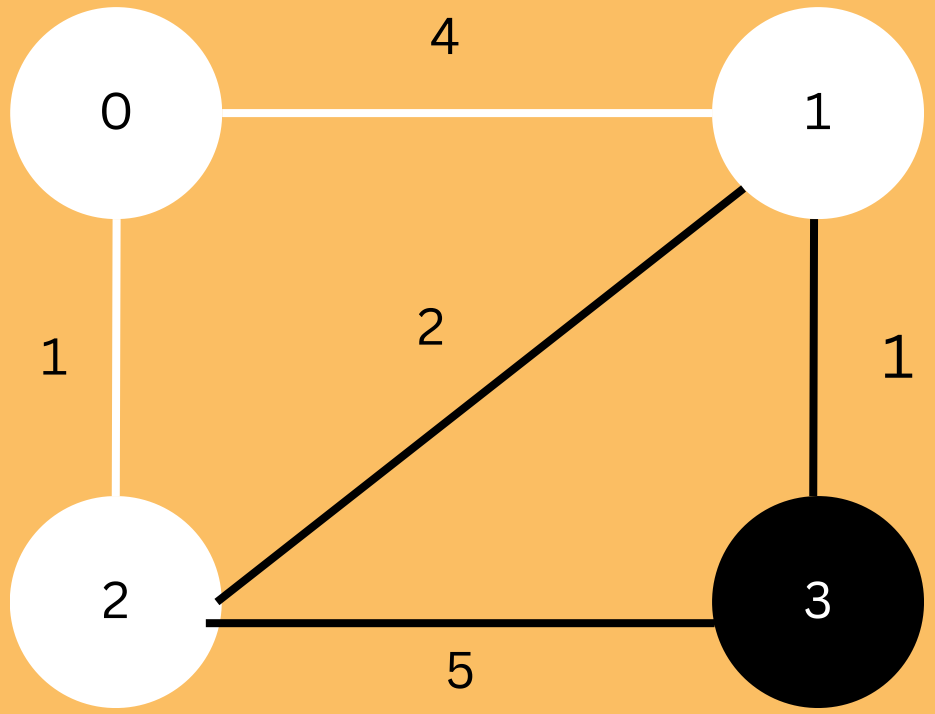
**distances : [0, 4, inf, inf ]**

**u = 0**
**v = 2** (edge.dest)
**weight = 1** (edge.weight)
**distances[0** (u) **] = 0**
**0 + 1 < inf (distances[2** (v) **]) ? True**
**distances[2 (v) ] = 0 + 1 = 1**
**pq.add(new Node(2, 1))**
**distances : [0, 4, 1, inf ]**



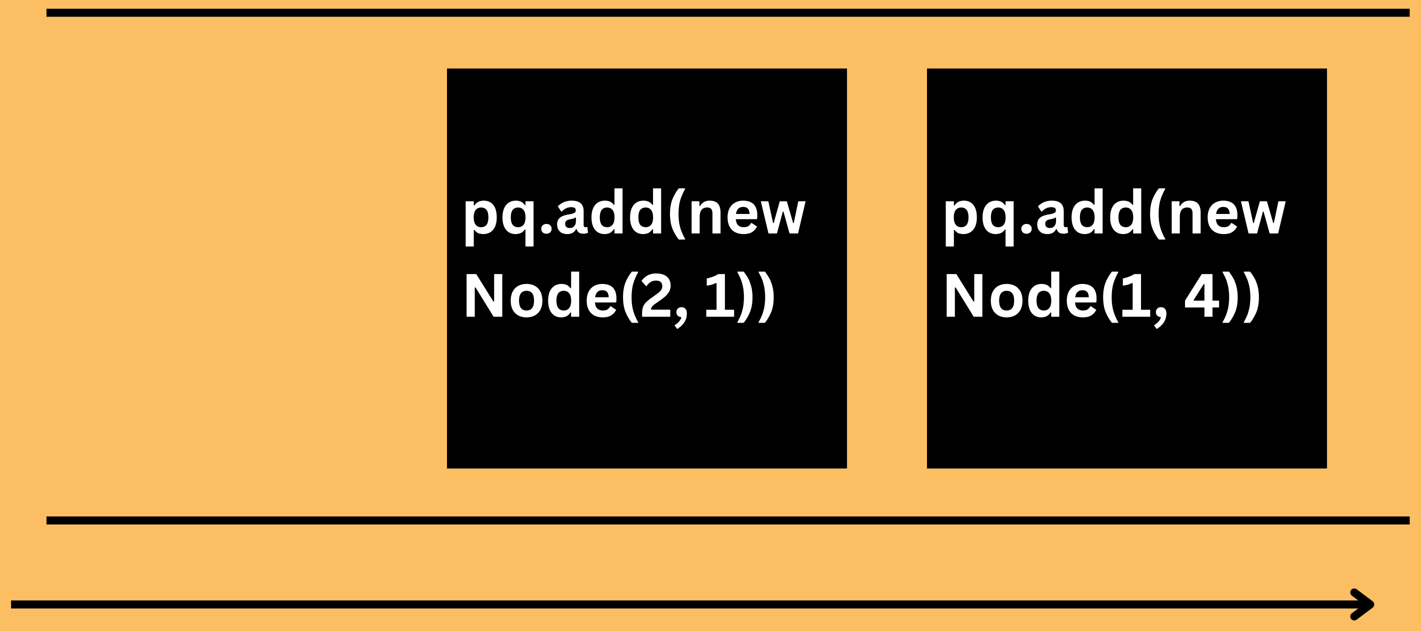
# JUMLAH NETWORK : 4

## DARI NETWORK 0



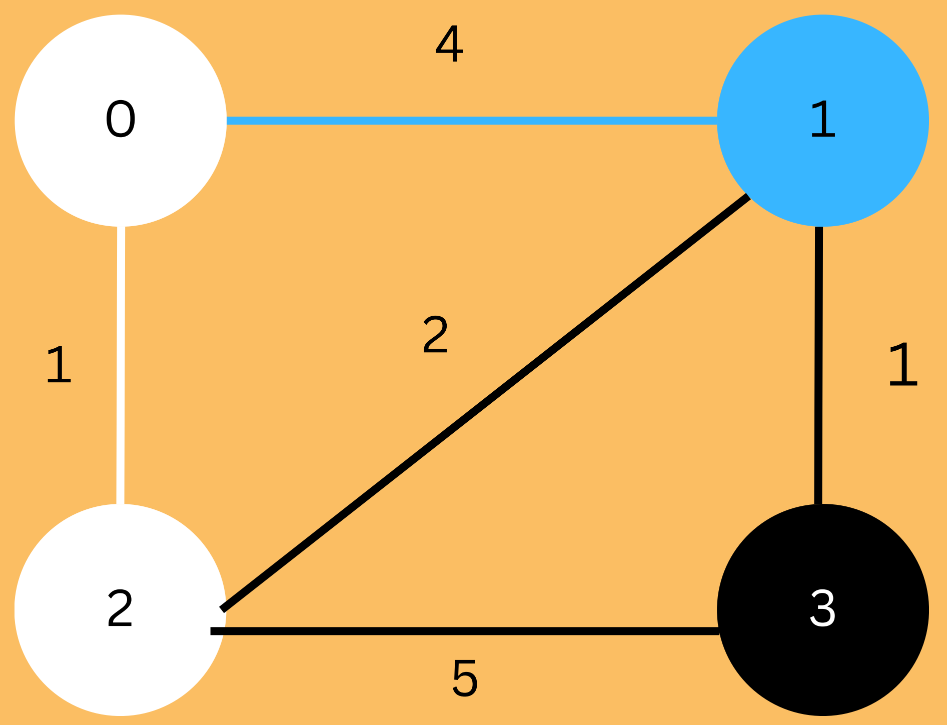
distances : [0, 4, 1, inf ]

pq.poll()



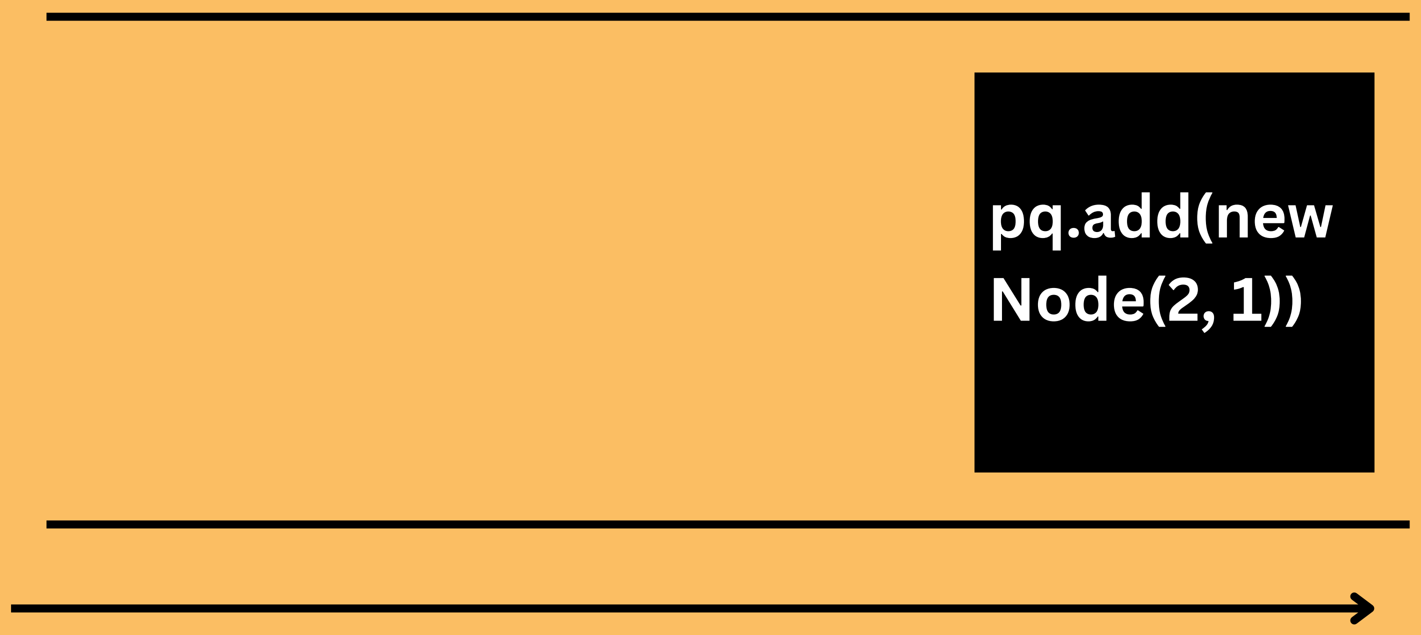
# JUMLAH NETWORK : 4

## DARI NETWORK 1



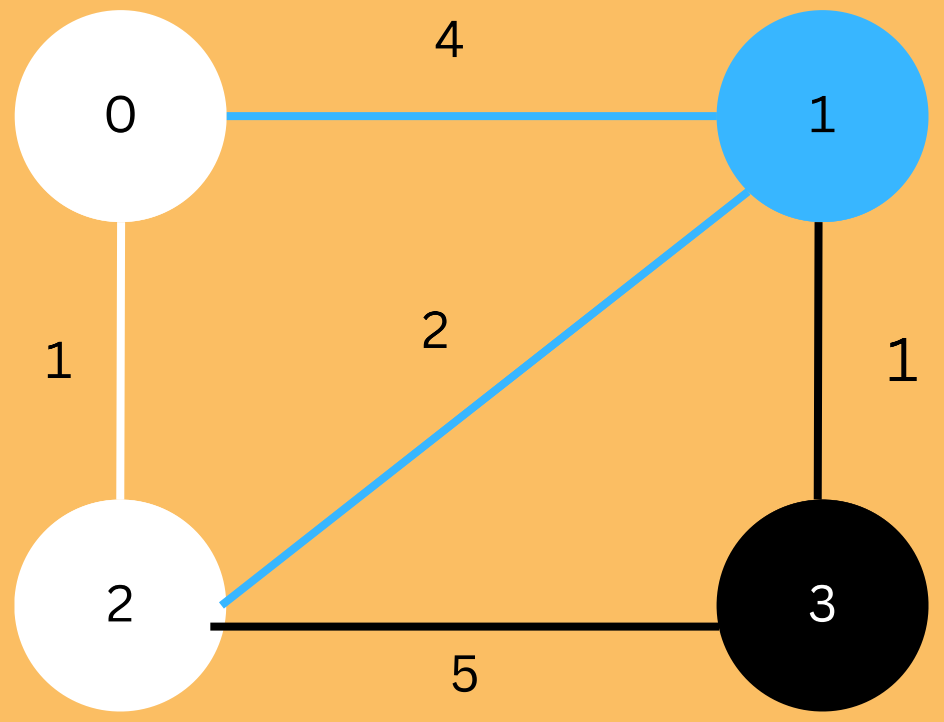
**distances : [0, 4, 1, inf ]**

**u = 1**
**v = 0** (edge.dest)
**weight = 4** (edge.weight)
**distances[1** (u) **] = 4**
**4 + 4 < 0 (distances[0** (v) **]) ? False**



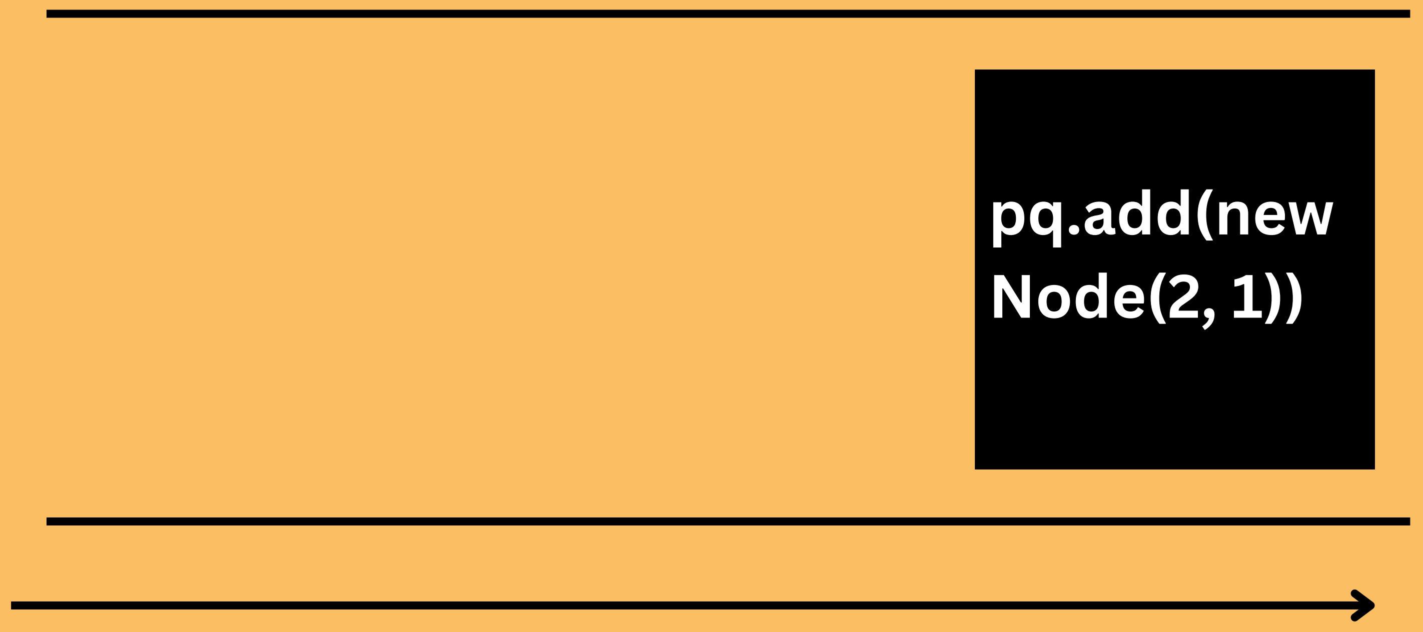
# JUMLAH NETWORK : 4

## DARI NETWORK 1



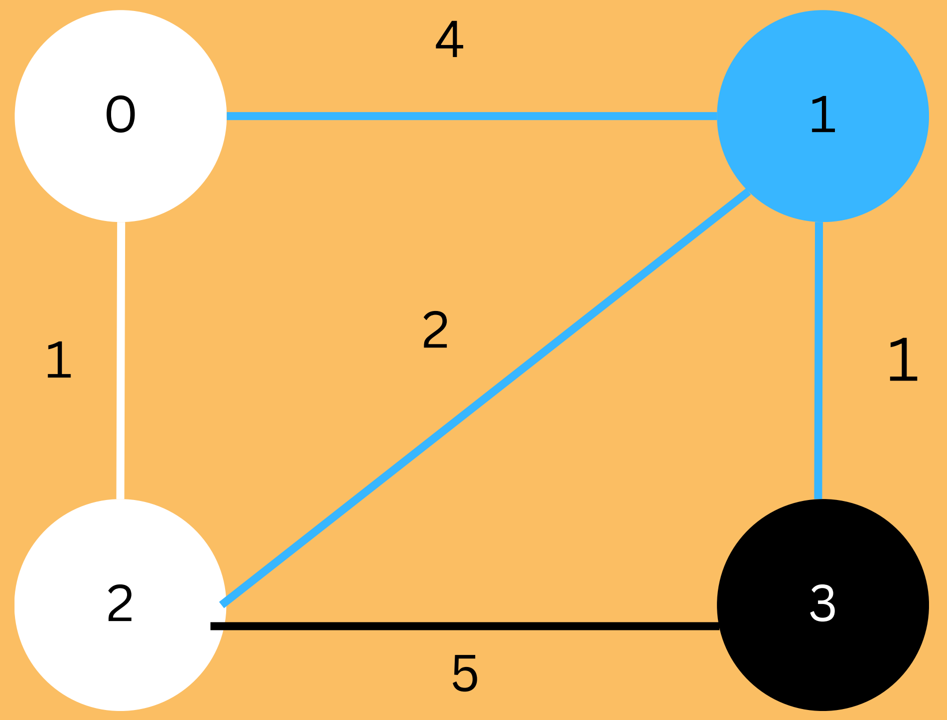
**distances : [0, 4, 1, inf ]**

**u = 1**
**v = 2** (edge.dest)
**weight = 2** (edge.weight)
**distances[1** (u) **] = 4**
**4 + 2 < 1 (distances[2** (v) **]) ? False**



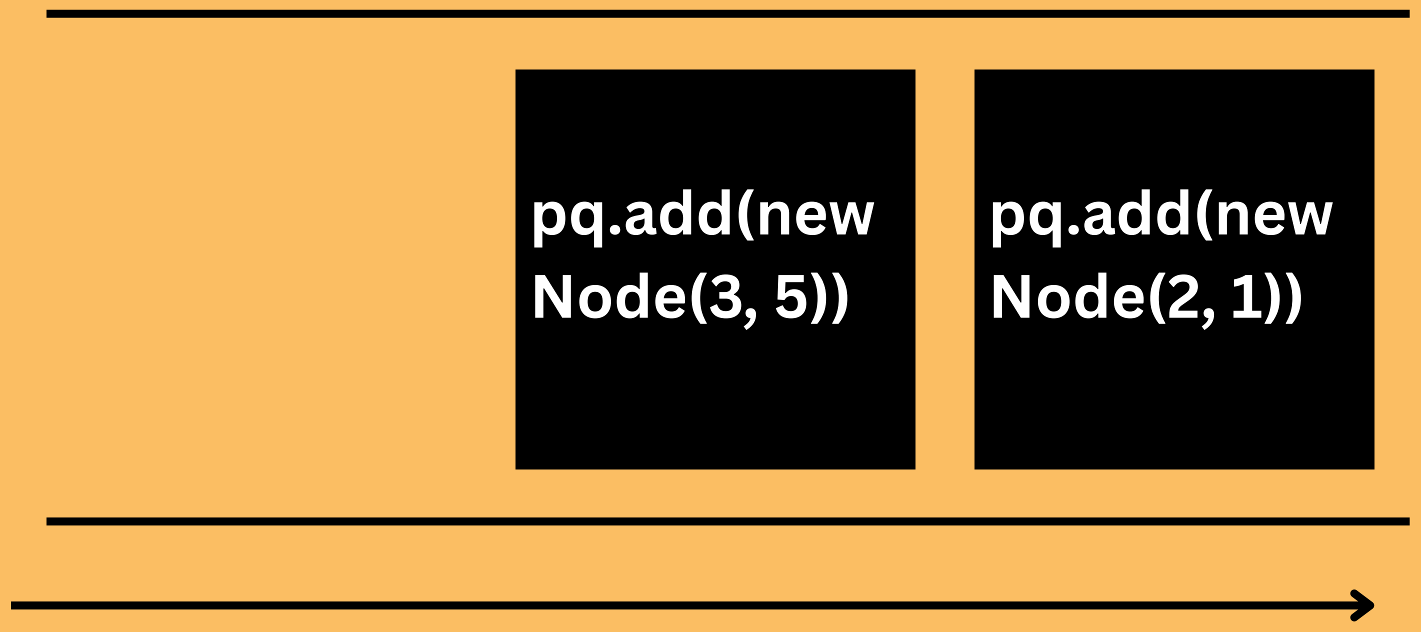
# JUMLAH NETWORK : 4

## DARI NETWORK 1



**distances : [0, 4, 1, inf ]**

**u = 1**
**v = 3** (edge.dest)
**weight = 1** (edge.weight)
**distances[1** (u) **] = 4**
**4 + 1 < inf (distances[3** (v) **]) ? True**
**distances[3 (v) ] = 4 + 1 = 5**
**pq.add(new Node(3, 5))**
**distances : [0, 4, 1, 5 ]**

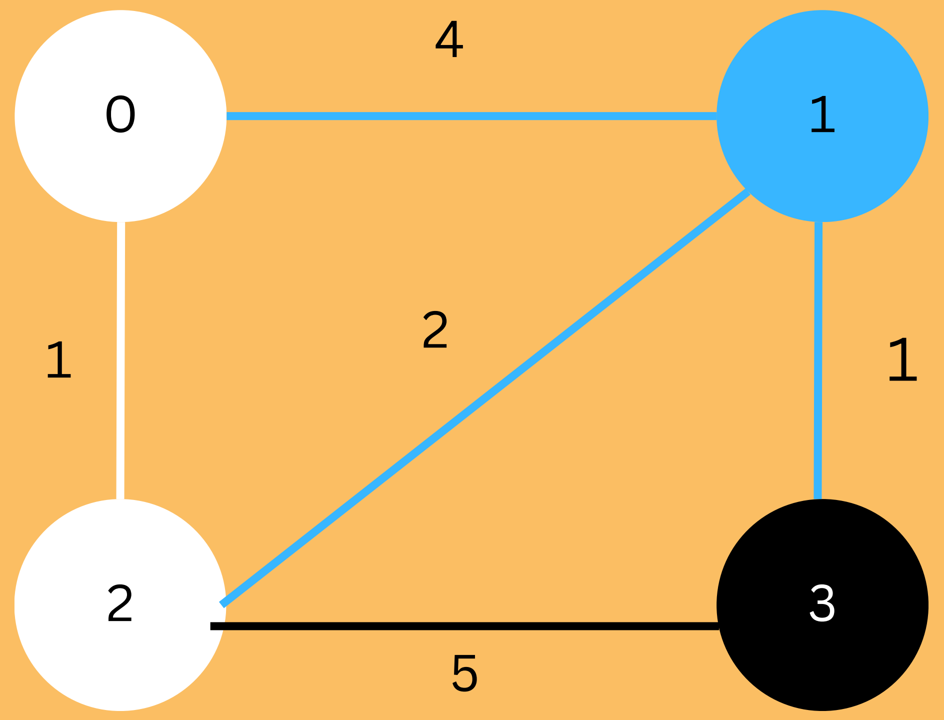


```
pq.add(new Node(3, 5))
```

```
pq.add(new Node(2, 1))
```

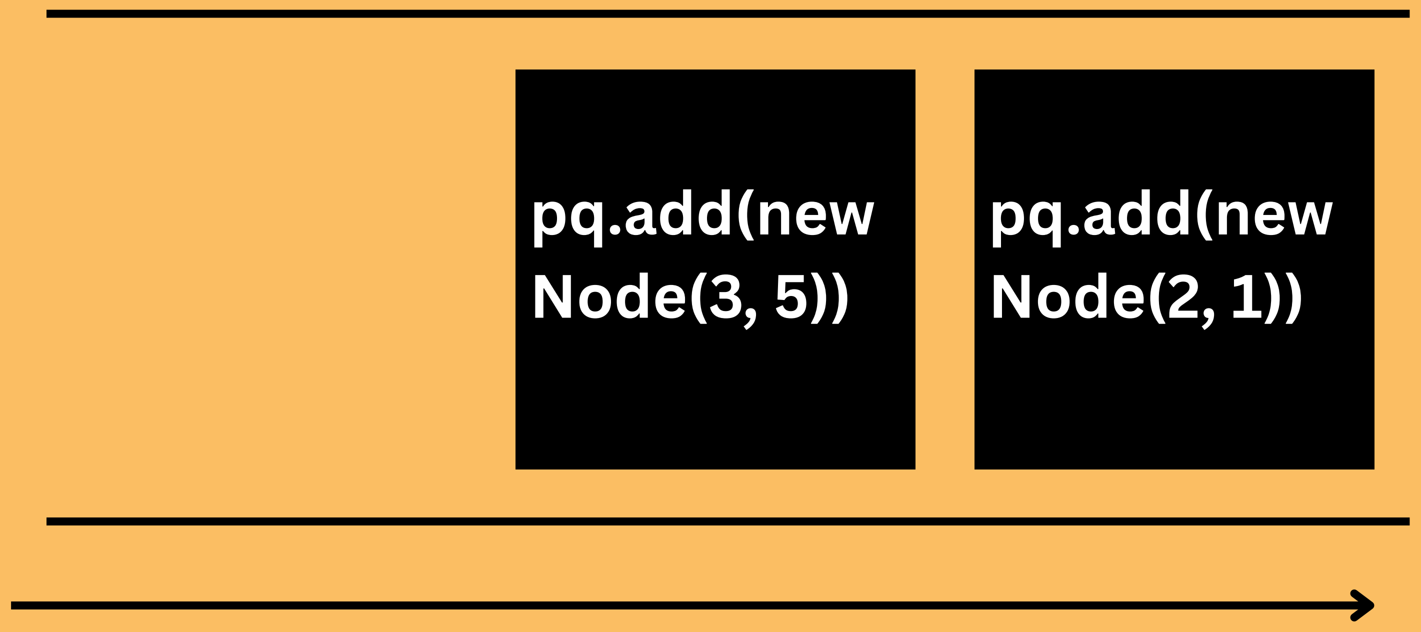
# JUMLAH NETWORK : 4

## DARI NETWORK 1



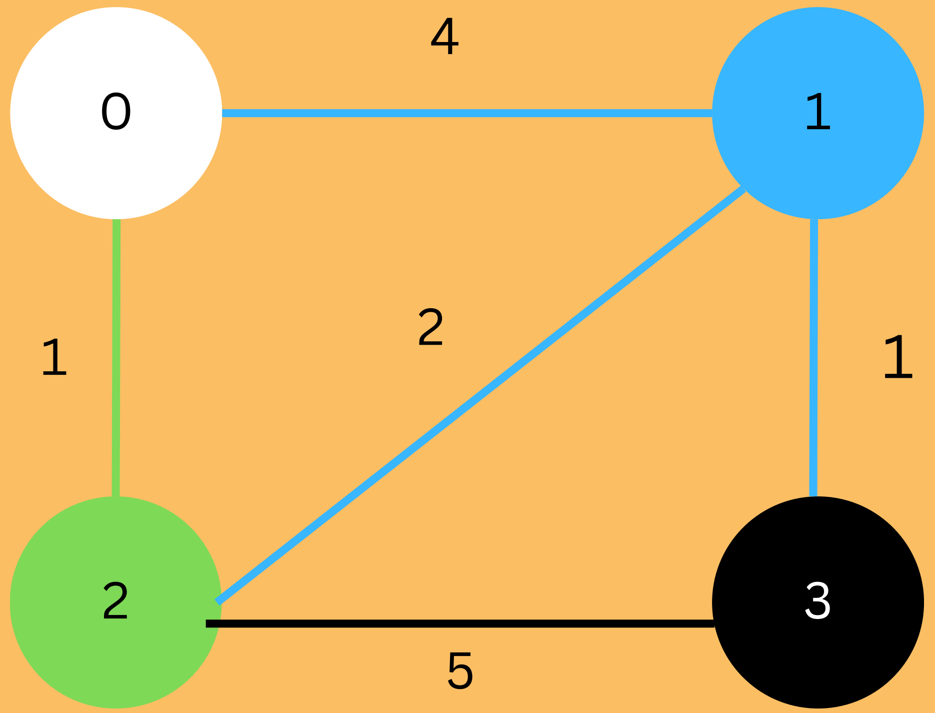
distances : [0, 4, 1, 5 ]

pq.poll()



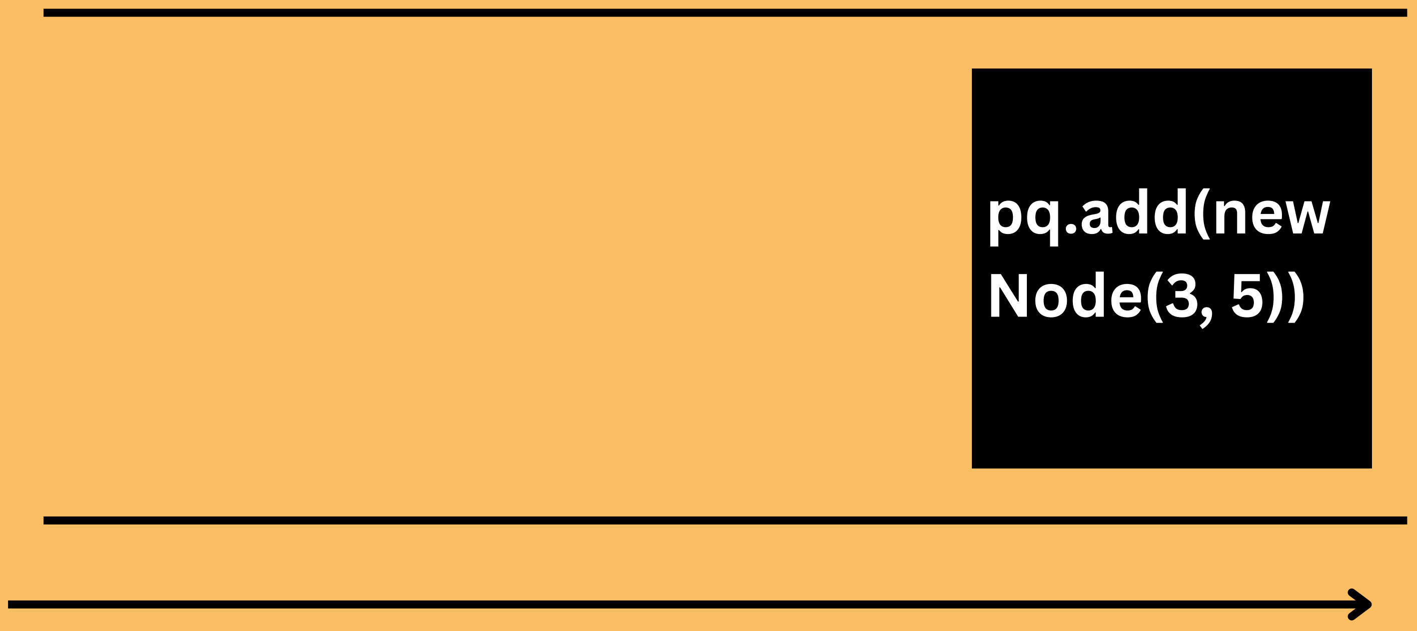
# JUMLAH NETWORK : 4

## DARI NETWORK 2



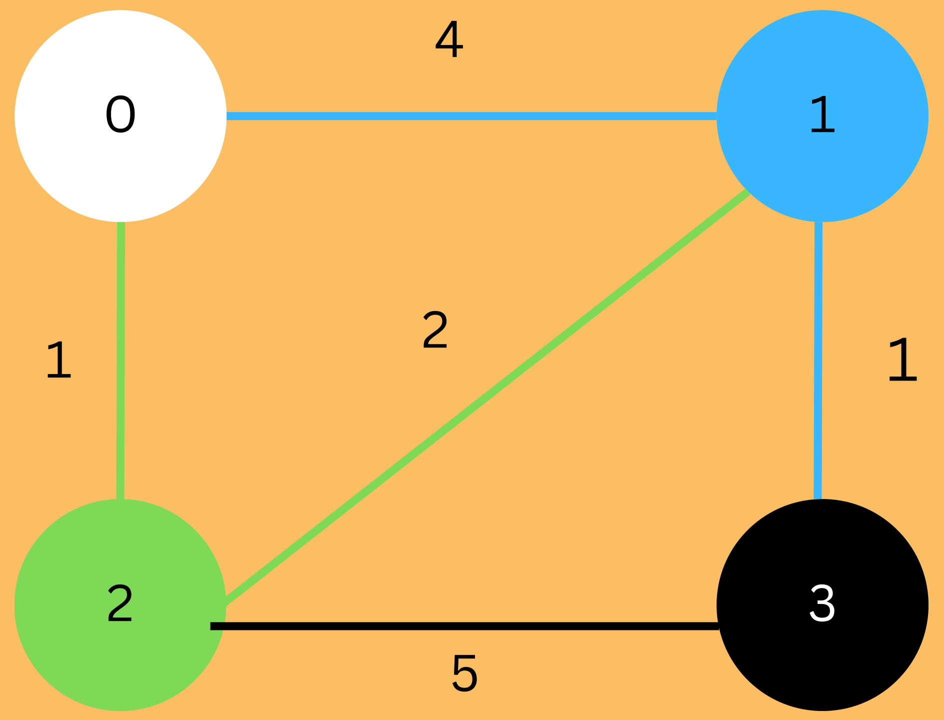
**distances : [0, 4, 1, 5 ]**

**u = 2**
**v = 0** (edge.dest)
**weight = 1** (edge.weight)
**distances[2** (u) **] = 1**
**1 + 1 < 0 (distances[0** (v) **]) ? False**



# JUMLAH NETWORK : 4

## DARI NETWORK 2

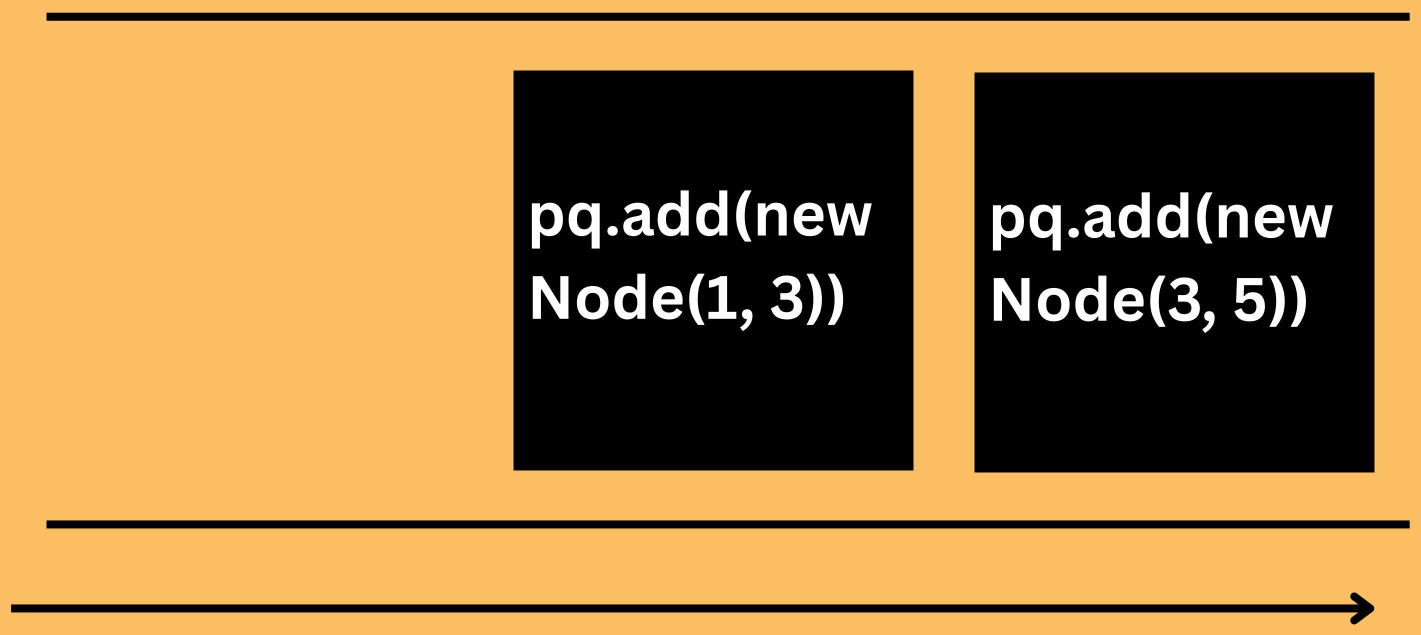


**distances : [0, 4, 1, 5 ]**

**u = 2**
**v = 1** (edge.dest)
**weight = 2** (edge.weight)
**distances[2** (u) **] = 1**
**1 + 2 < 4 (distances[1** (v) **]) ? True**
**distances[1 (v) ] = 1 + 2 = 3**

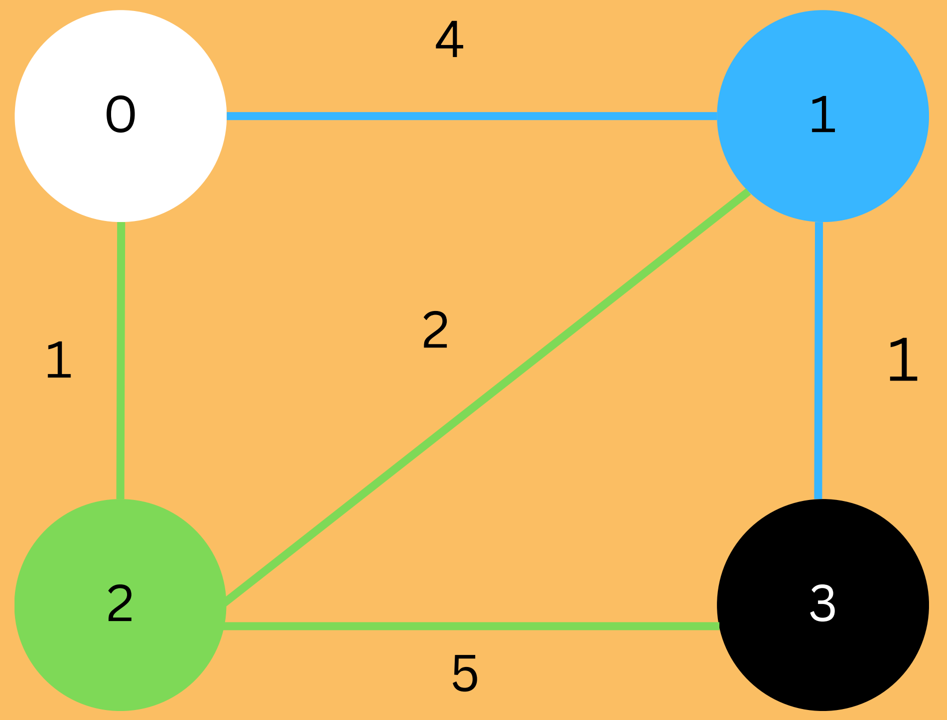
**pq.add(new Node(1, 3))**

**distances : [0, 3, 1, 5 ]**


pq.add(new Node(1, 3))
pq.add(new Node(3, 5))

# JUMLAH NETWORK : 4

## DARI NETWORK 2

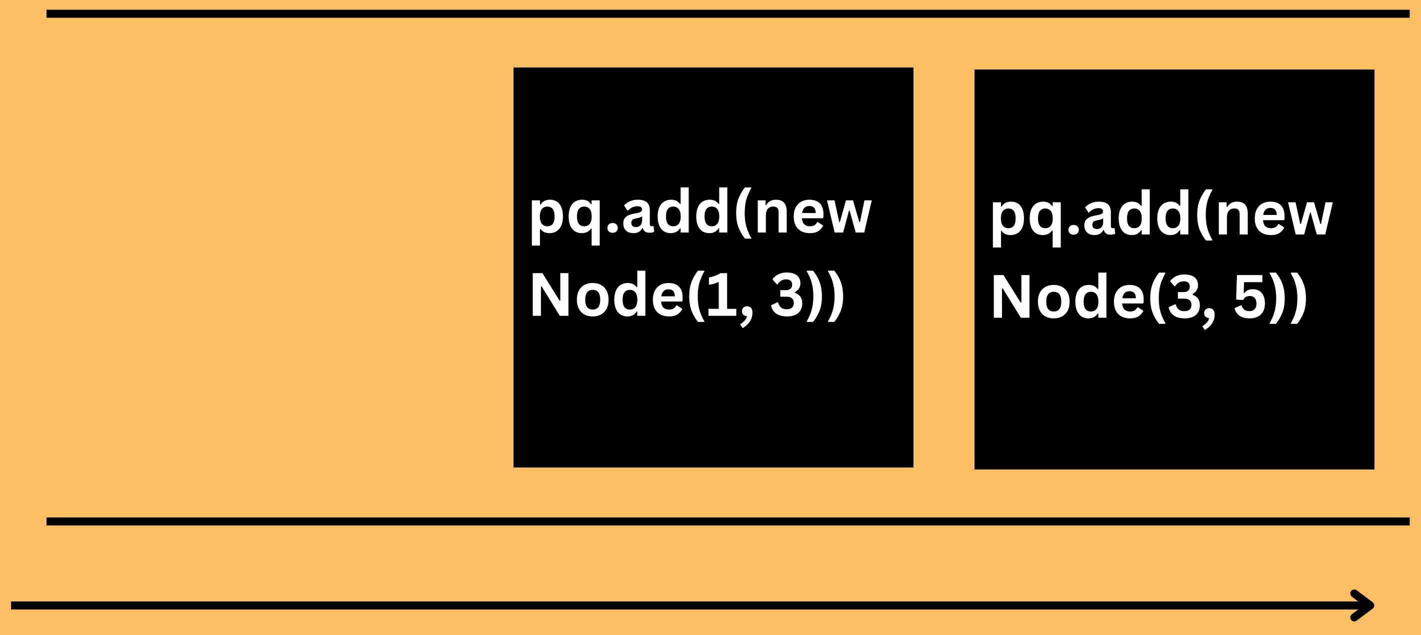


**distances : [0, 3, 1, 5 ]**

**u = 2**
**v = 3** (edge.dest)
**weight = 5** (edge.weight)

**distances[2** (u) **] = 1**

**1 + 5 < 5 (distances[3** (v) **]) ? False**

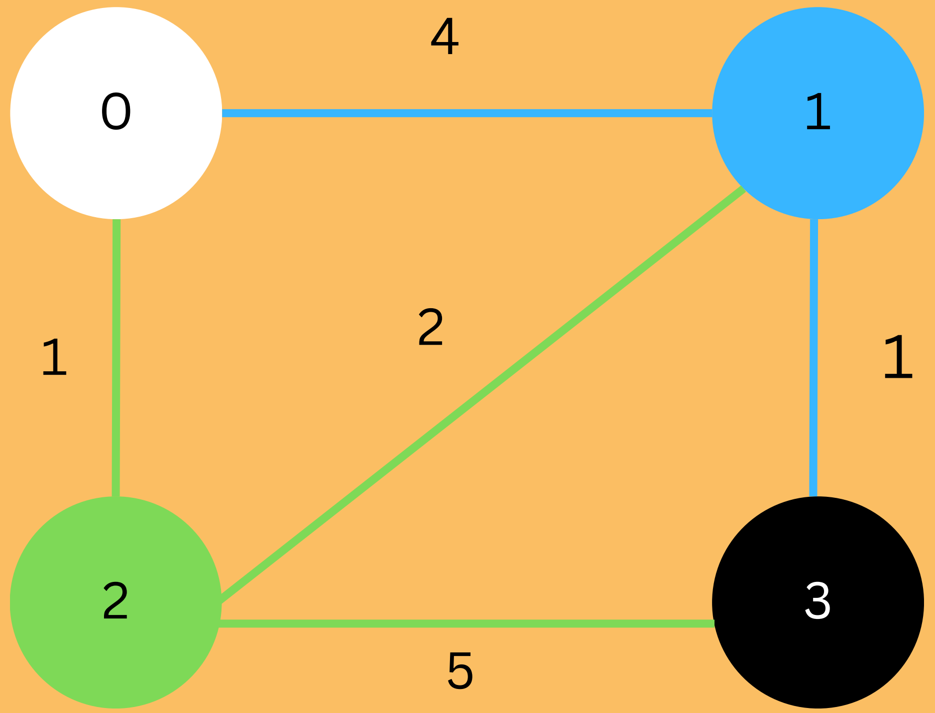


**pq.add(new Node(1, 3))**

**pq.add(new Node(3, 5))**

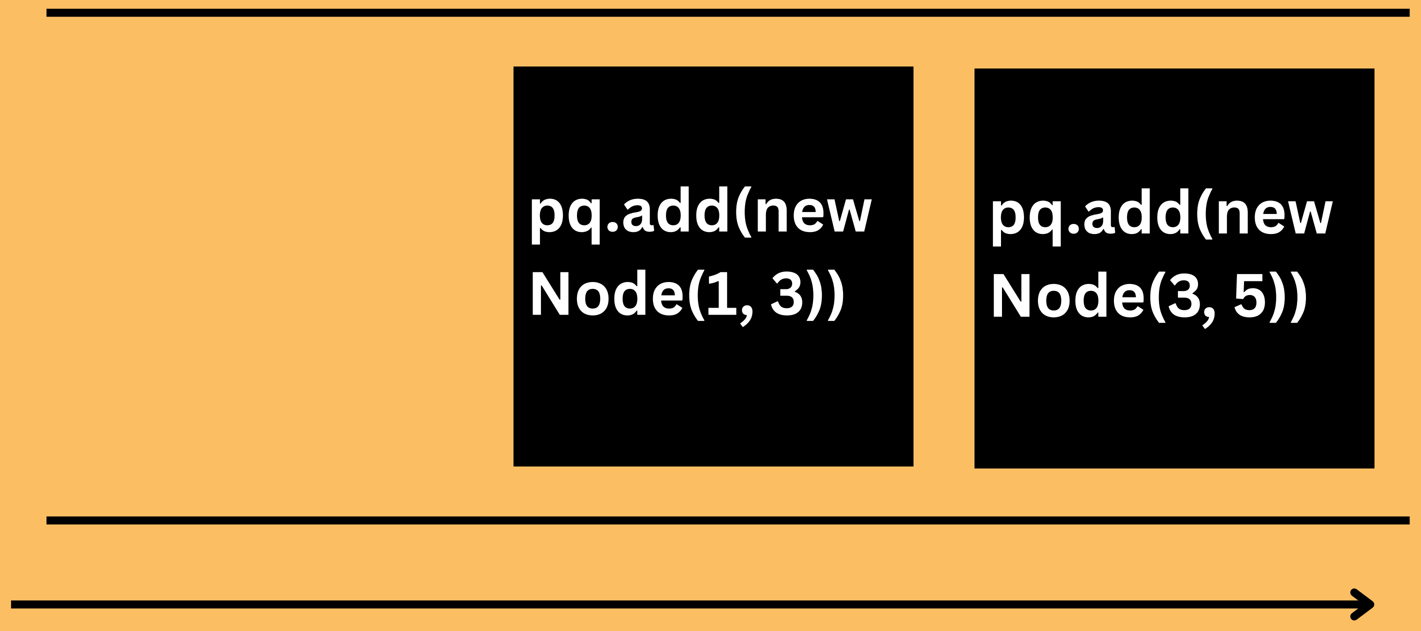
# JUMLAH NETWORK : 4

## DARI NETWORK 2



**distances : [0, 3, 1, 5 ]**

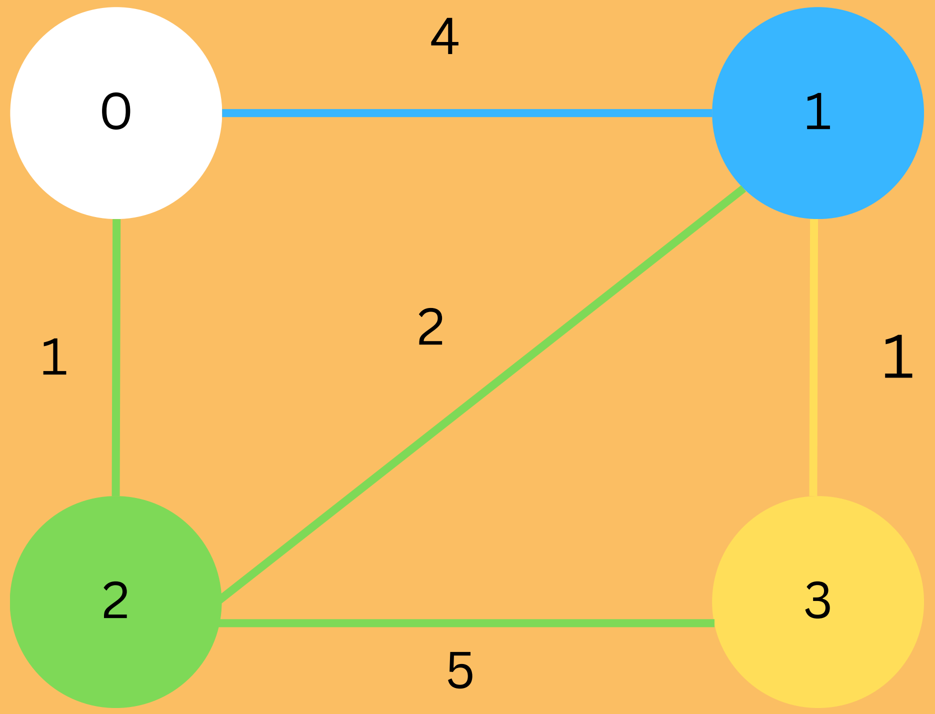
**pq.poll()**


**pq.add(new Node(1, 3))**

**pq.add(new Node(3, 5))**

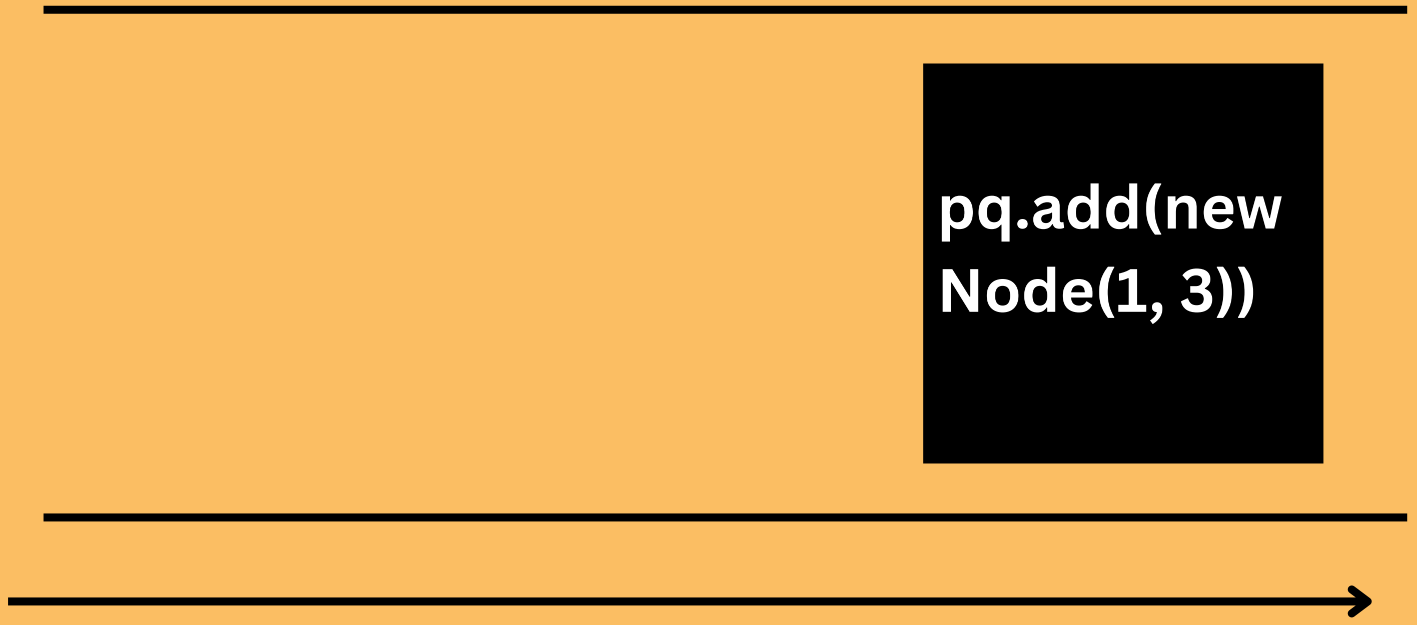
# JUMLAH NETWORK : 4

## DARI NETWORK 3



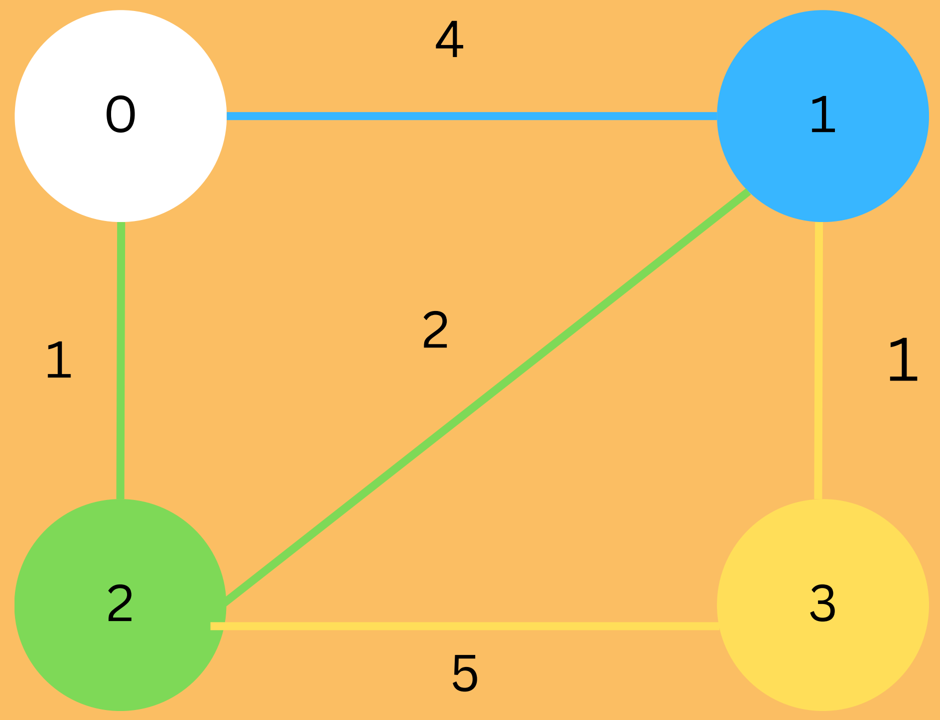
**distances : [0, 3, 1, 5 ]**

**u = 3**
**v = 1** (edge.dest)
**weight = 1** (edge.weight)
**distances[3** (u) **] = 5**
**1 + 5 < 5 (distances[3** (v) **]) ? False**



# JUMLAH NETWORK : 4

## DARI NETWORK 3



**distances : [0, 3, 1, 5 ]**

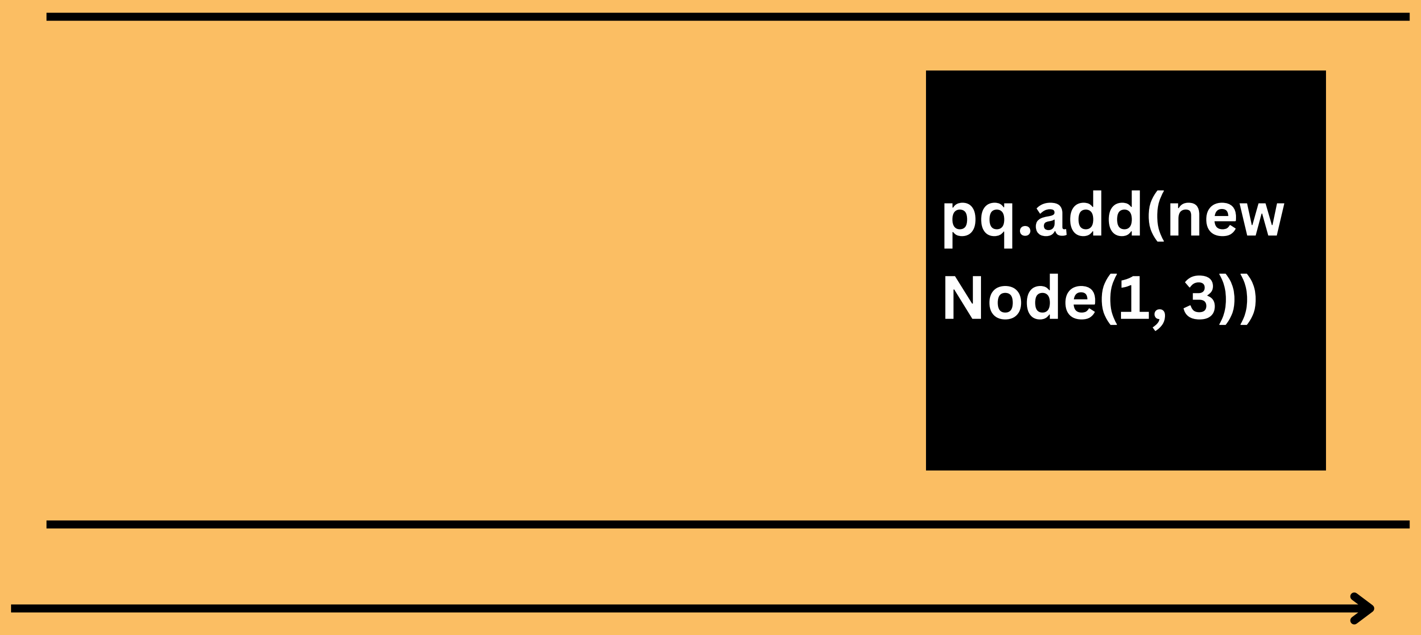
**u = 3**

**v = 2** (edge.dest)

**weight = 5** (edge.weight)

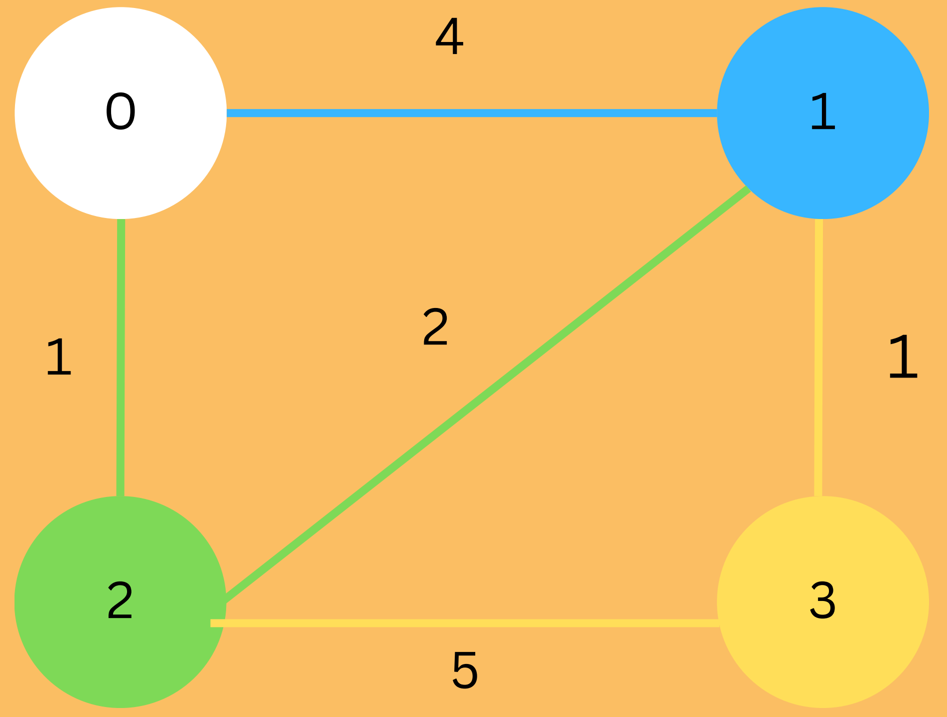
**distances[3** (u) **] = 5**

**5 + 5 < 1 (distances[2** (v) **]) ? False**



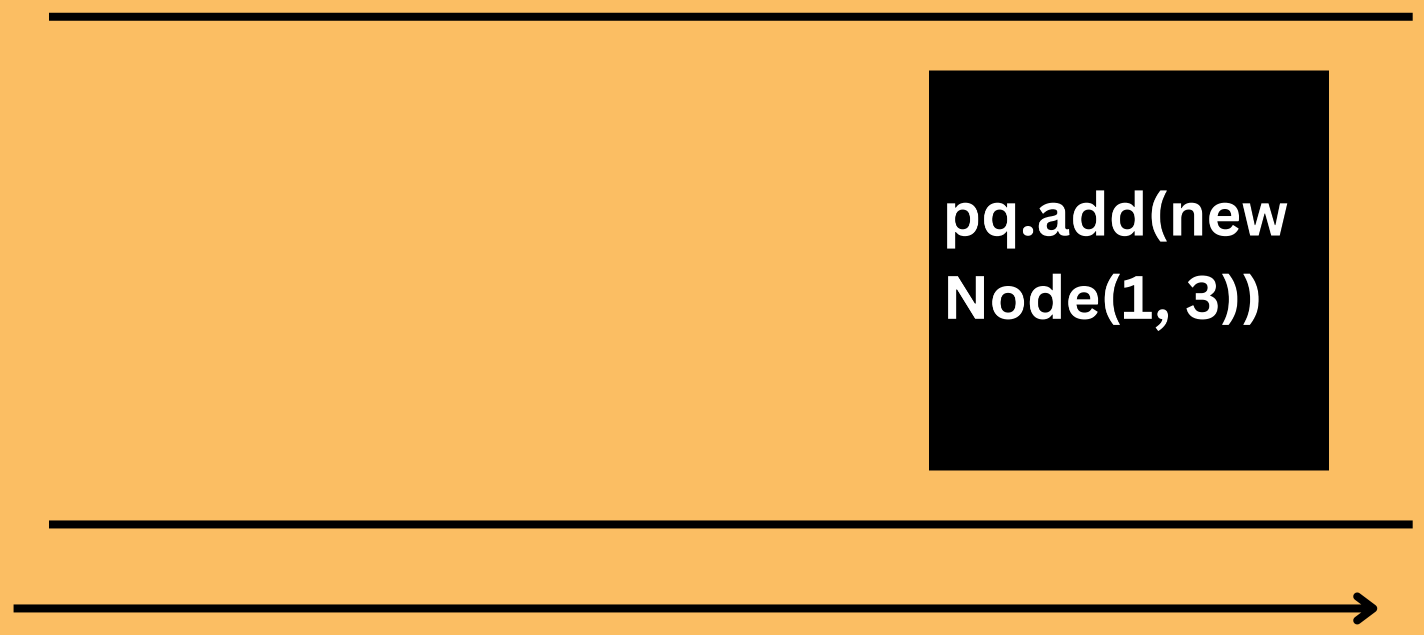
# JUMLAH NETWORK : 4

## DARI NETWORK 3



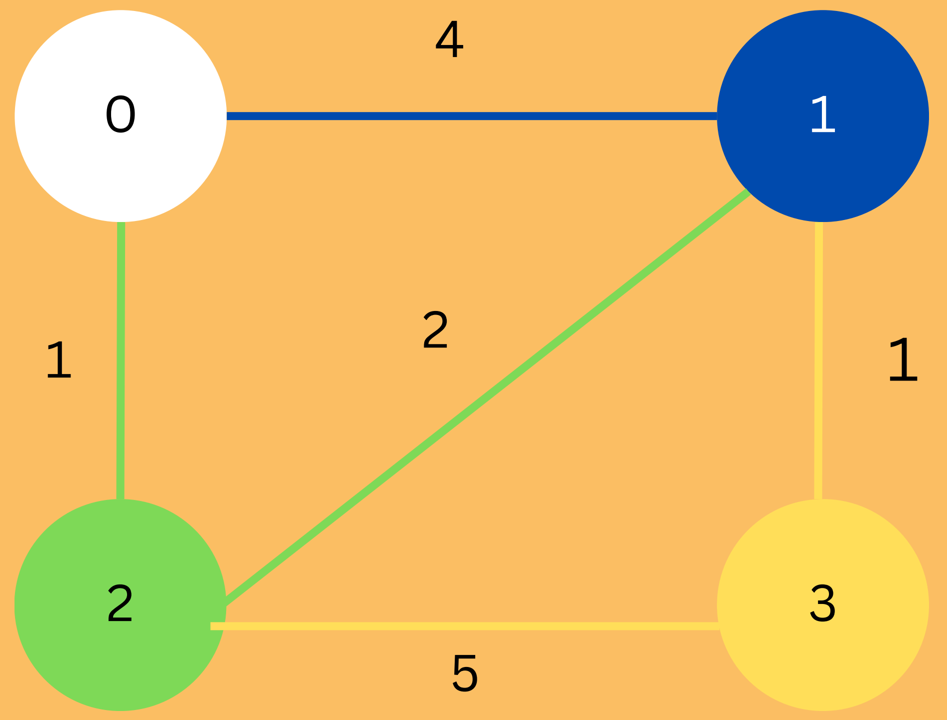
**distances : [0, 3, 1, 5 ]**

**pq.poll()**



# JUMLAH NETWORK : 4

## DARI NETWORK 1

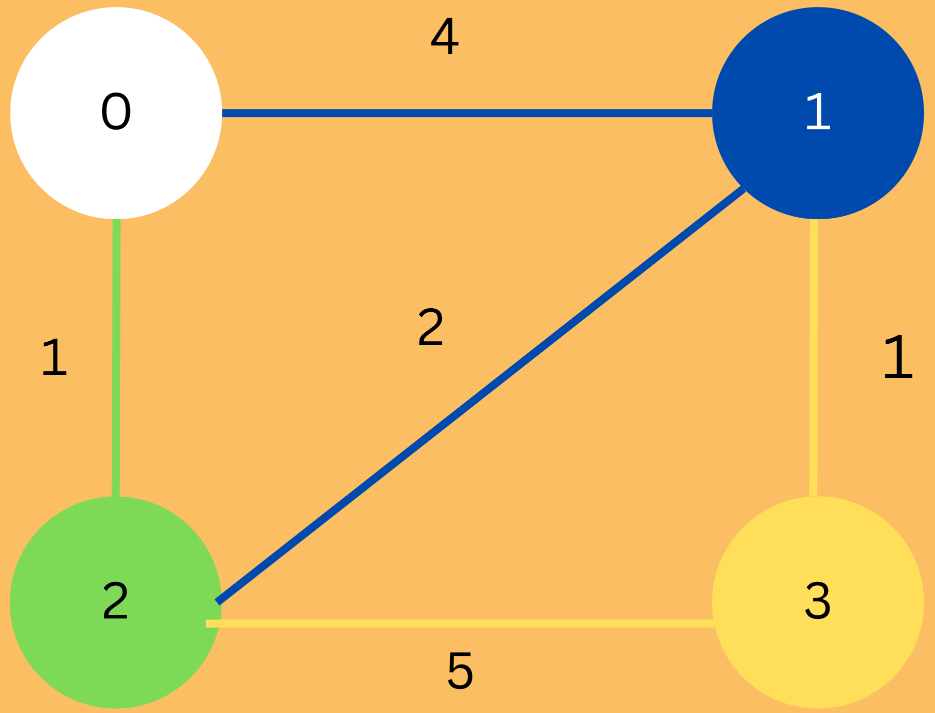


**distances : [0, 3, 1, 5 ]**

**u = 1**
**v = 0** (edge.dest)
**weight = 4** (edge.weight)
**distances[1** (u) **] = 3**
**3 + 4 < 0 (distances[0** (v) **]) ? False**

# JUMLAH NETWORK : 4

## DARI NETWORK 1



**distances : [0, 3, 1, 5 ]**

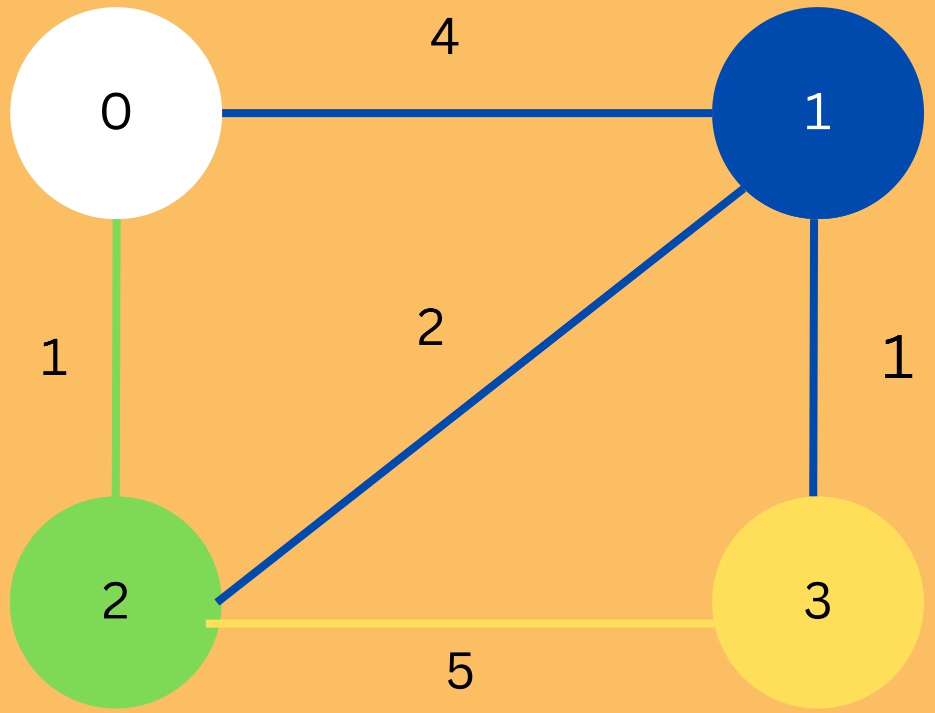
**u = 1**
**v = 2** (edge.dest)
**weight = 2** (edge.weight)

**distances[1** (u) **] = 3**

**3 + 2 < 1 (distances[2** (v) **]) ? False**

# JUMLAH NETWORK : 4

## DARI NETWORK 1

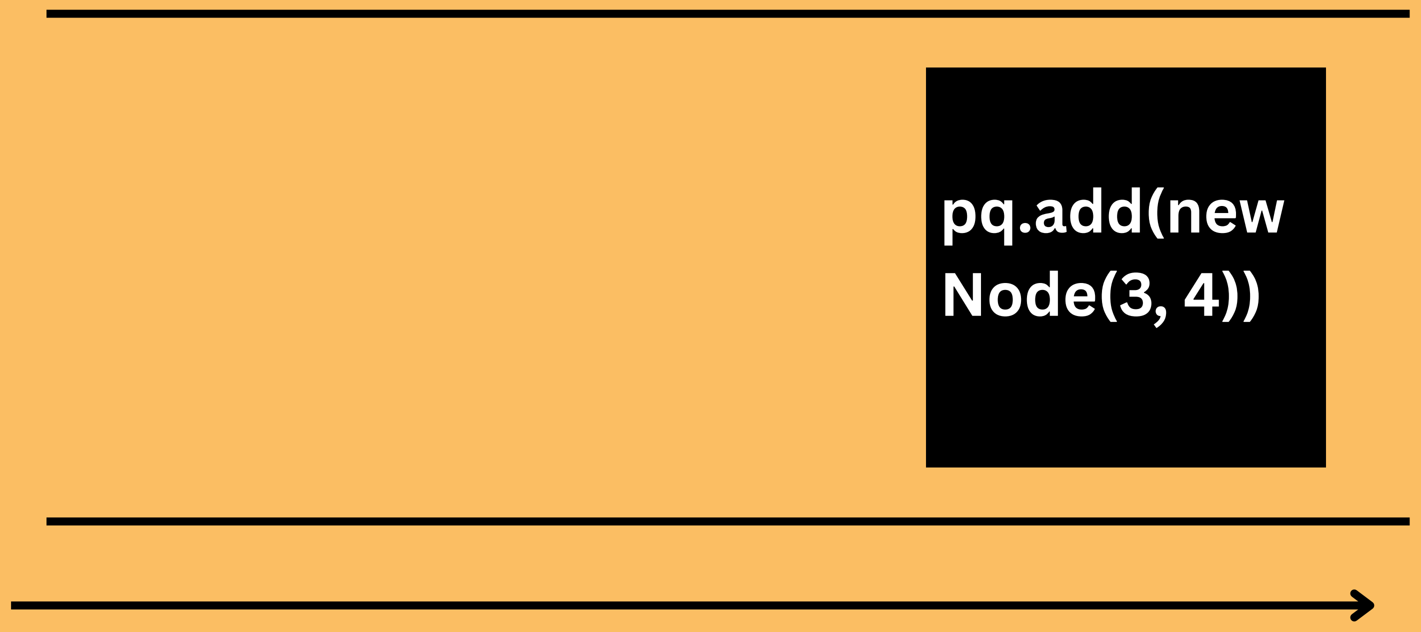


**distances : [0, 3, 1, 5 ]**

**u = 1**
**v = 3** (edge.dest)
**weight = 1** (edge.weight)
**distances[1** (u) **] = 3**
**3 + 1 < 5 (distances[3** (v) **]) ? True**
**distances[3 (v) ] = 3 + 1 = 4**

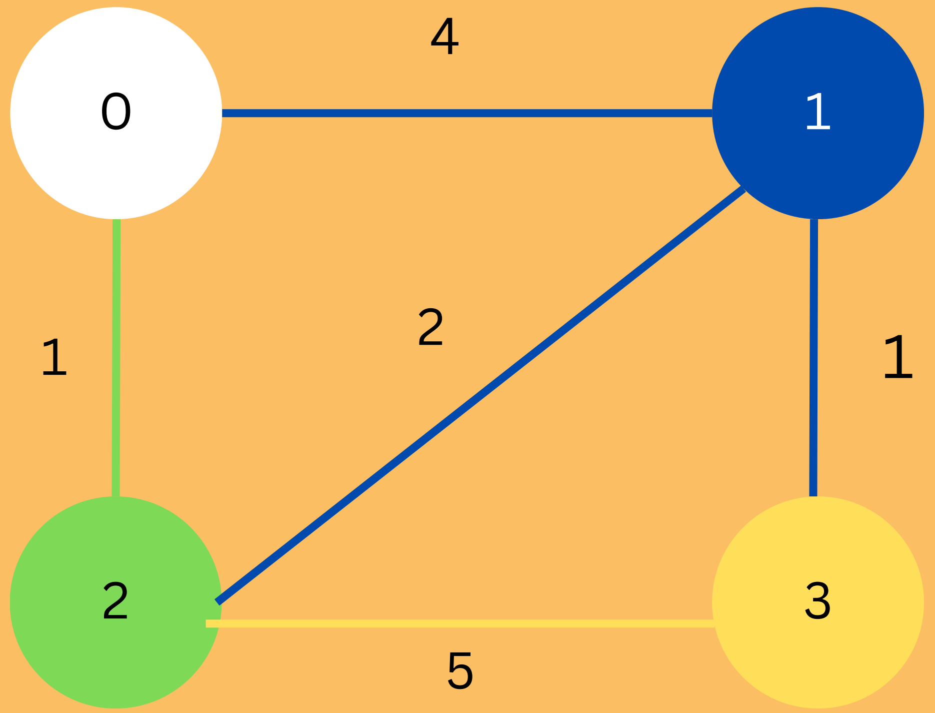
**pq.add(new Node(3, 4))**

**distances : [0, 3, 1, 4 ]**



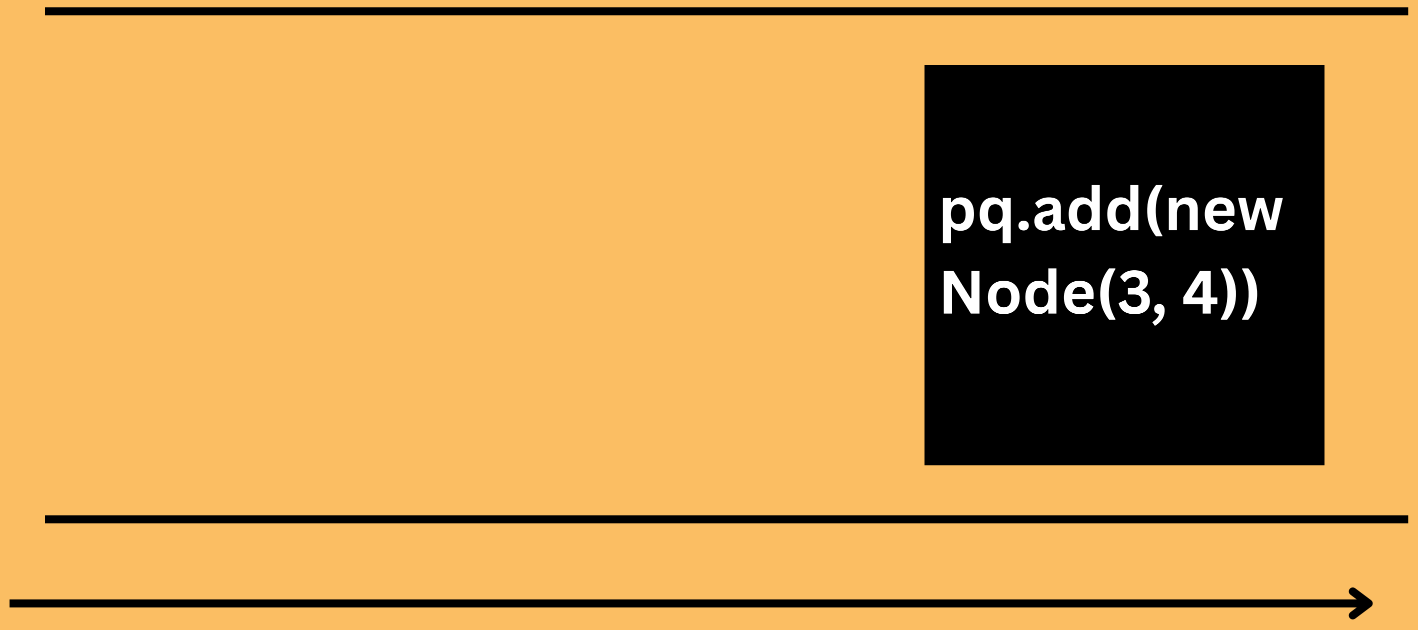
# JUMLAH NETWORK : 4

## DARI NETWORK 1



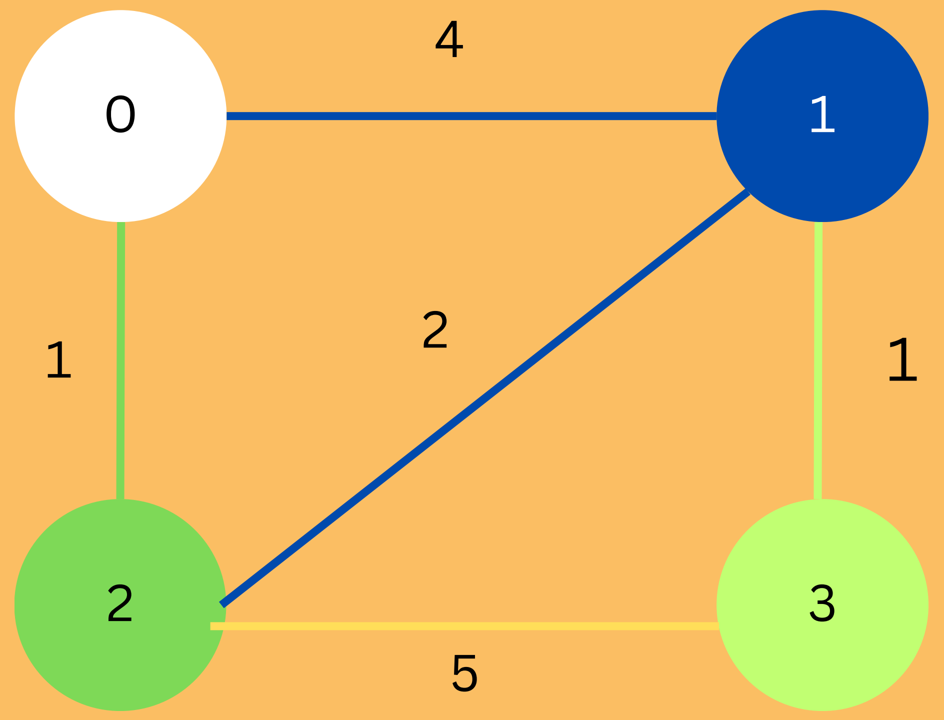
**distances : [0, 3, 1, 4 ]**

**pq.poll()**



# JUMLAH NETWORK : 4

## DARI NETWORK 3



**distances : [0, 3, 1, 4 ]**

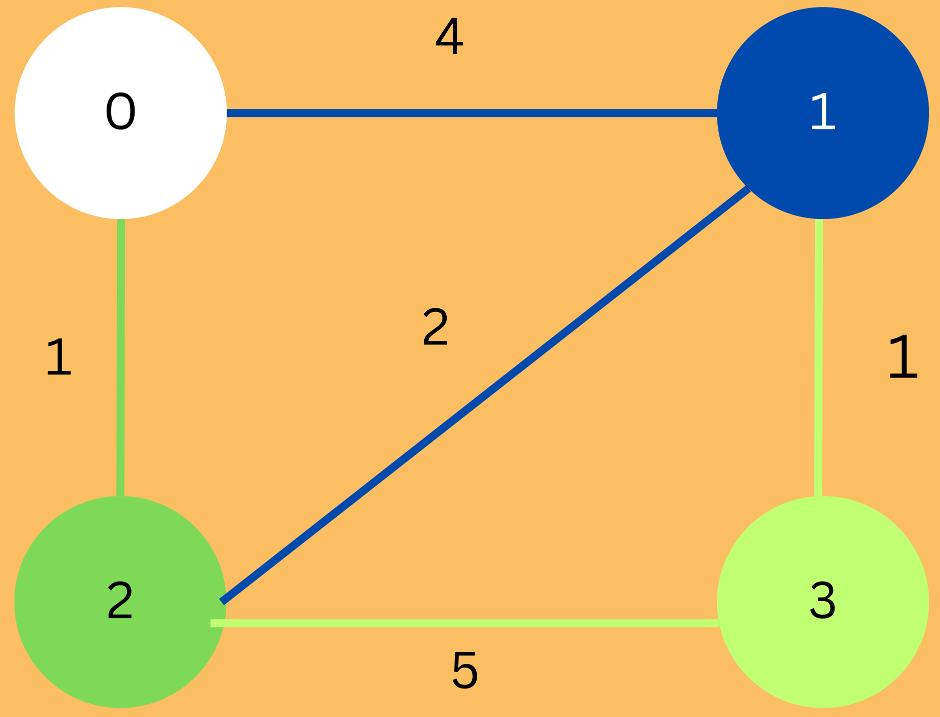
**u = 3**
**v = 1** (edge.dest)
**weight = 1** (edge.weight)

**distances[3** (u) **] = 4**

**4 + 1 < 3 (distances[1** (v) **]) ? False**

## JUMLAH NETWORK : 4

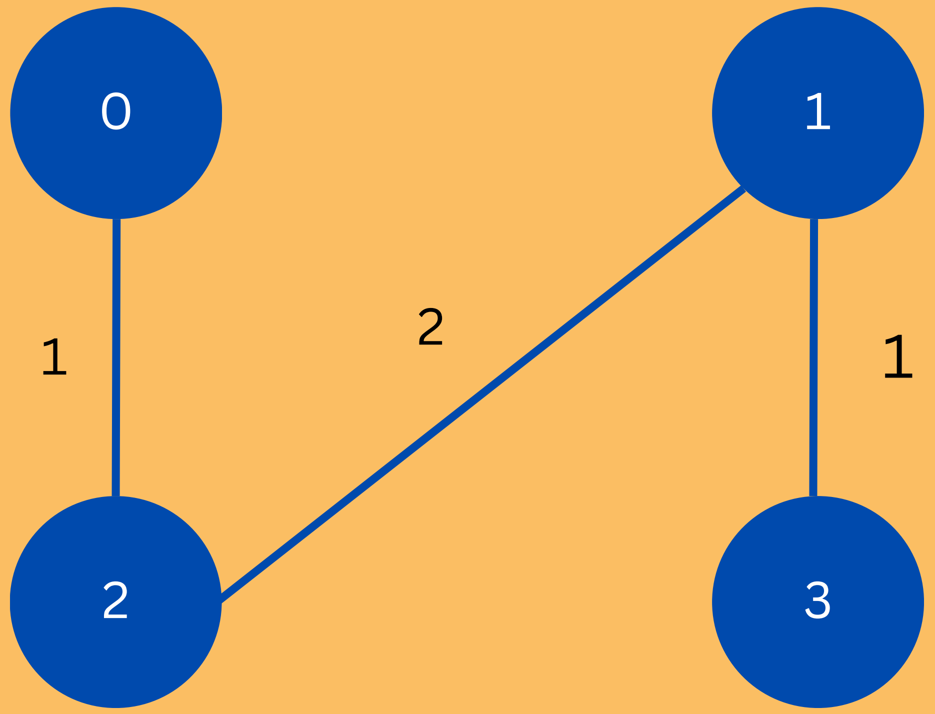
DARI NETWORK 3



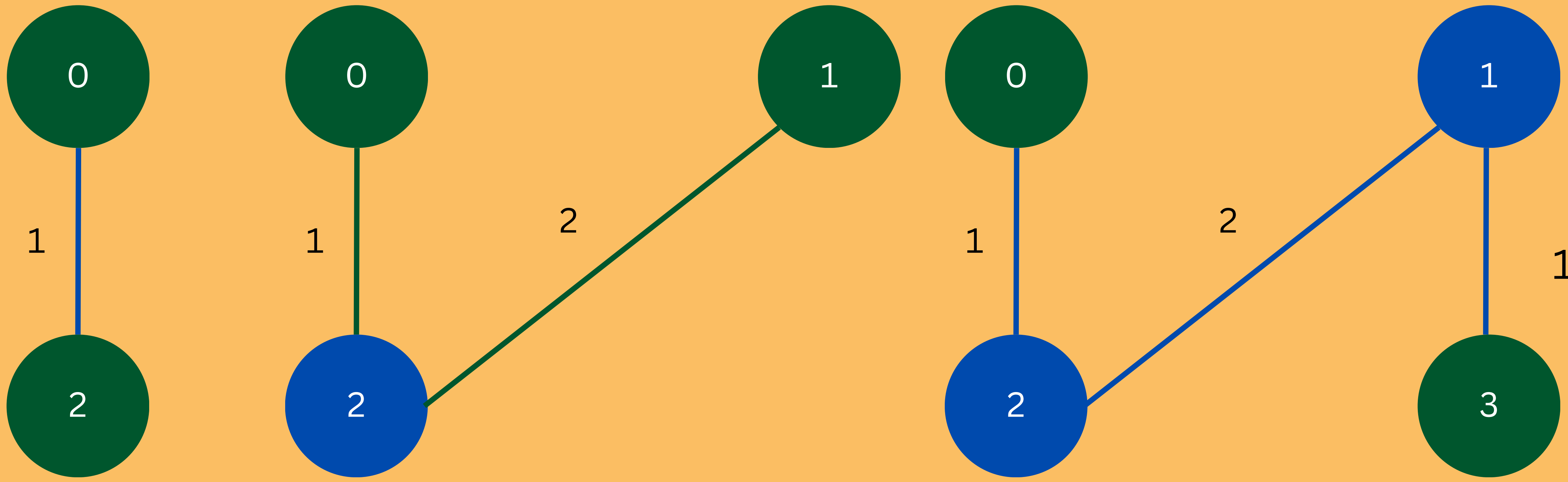
**distances : [0, 3, 1, 4 ]**

# JUMLAH NETWORK : 4

DARI NETWORK 3



**distances : [0, 3, 1, 4 ]**



0
2
1
0
1
2
2
1
0
1
2
2
1
3
1