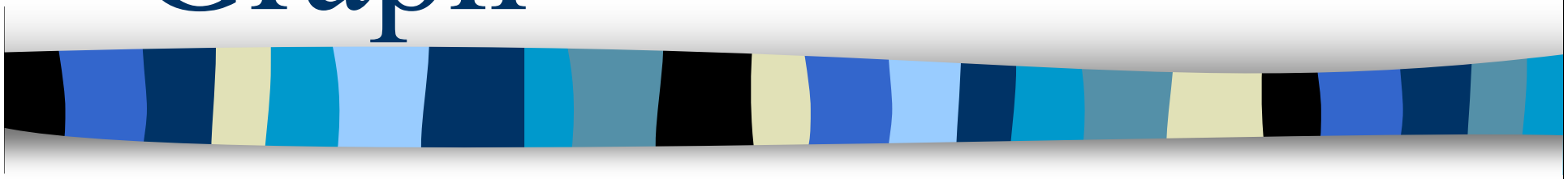
Graph



Graph

- Adalah kumpulan obyek atau aktivitas
- Direpresentasikan sebagai kumpulan titik (nodes/vertices) dan garis (arcs/edges)

- Contoh :
  - Travelling salesman problem
  - Shortest path problem

Notasi graph

- Garis pada graph dinotasikan sebagai

  $e = [u, v]$

  dimana

  e : garis

  u : titik asal

  v : titik tujuan

Notasi graph

Jalur pada graph dinotasikan sebagai

$$P = (v_0, v_1, \ldots, v_n)$$

dimana

P : jalur

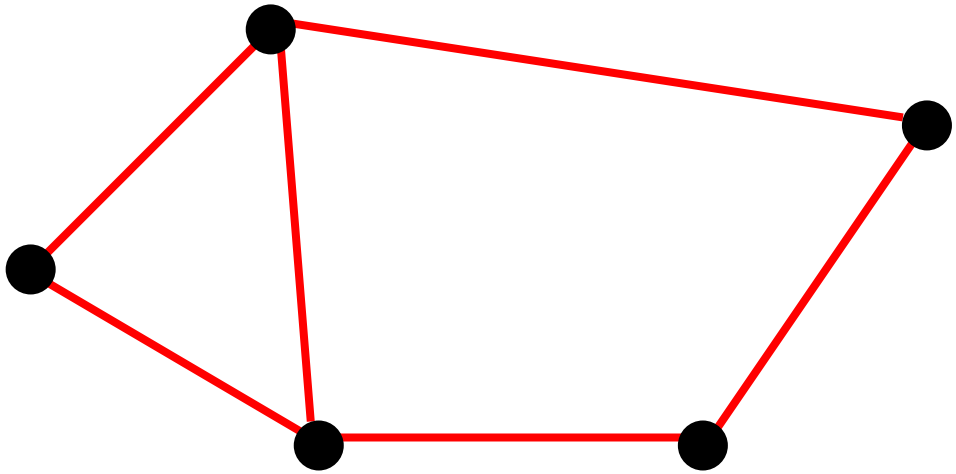
v_i : titik jalur

n : jumlah titik jalur

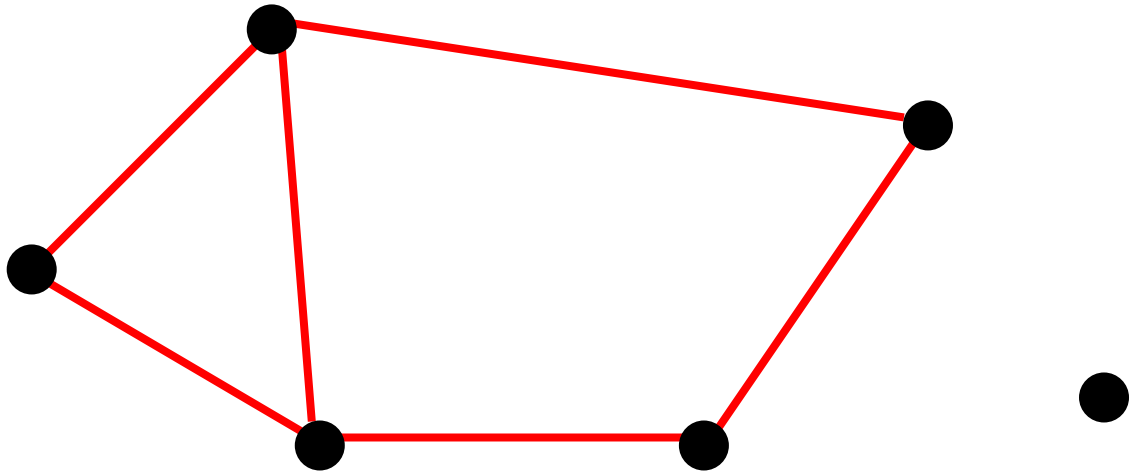
Bentuk-bentuk graph

- Connected graph
- Isolated graph
- Completed graph
- Directed graph

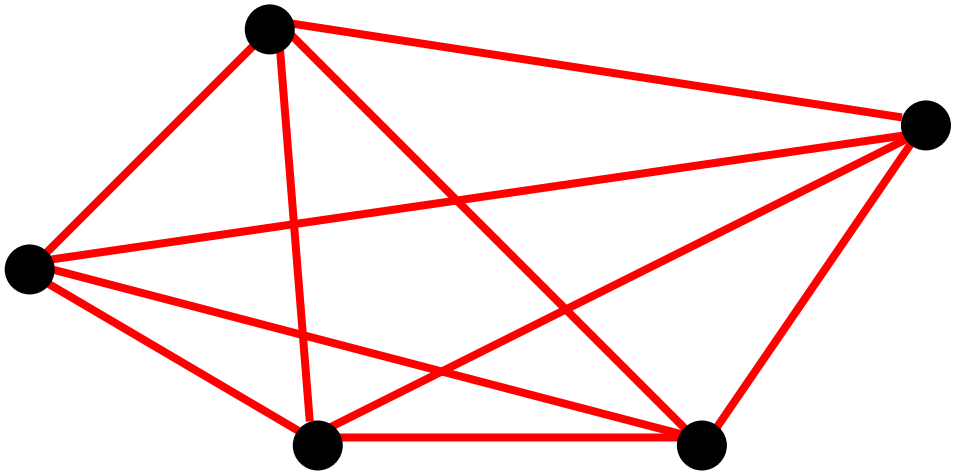
Connected graph



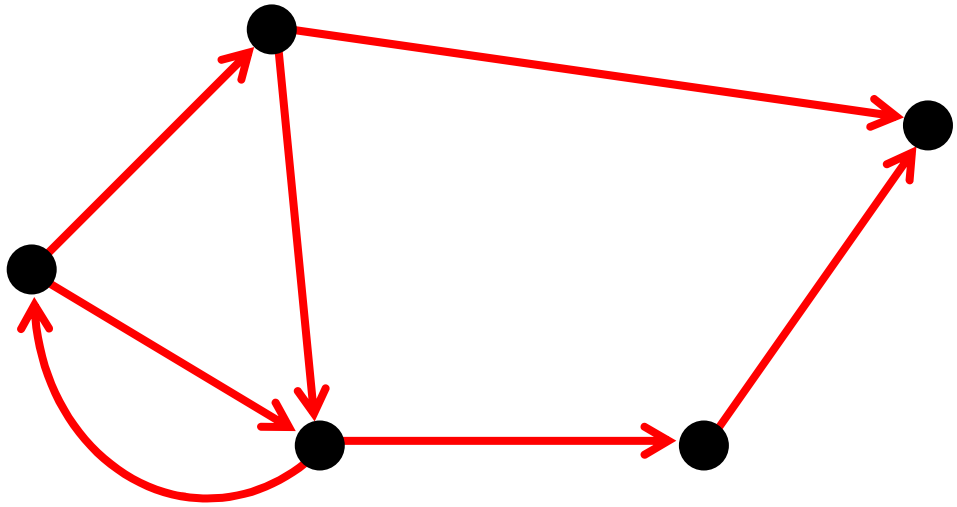
Isolated graph



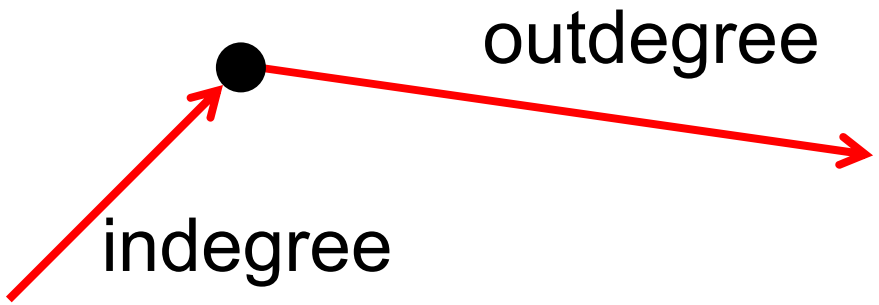
Completed graph



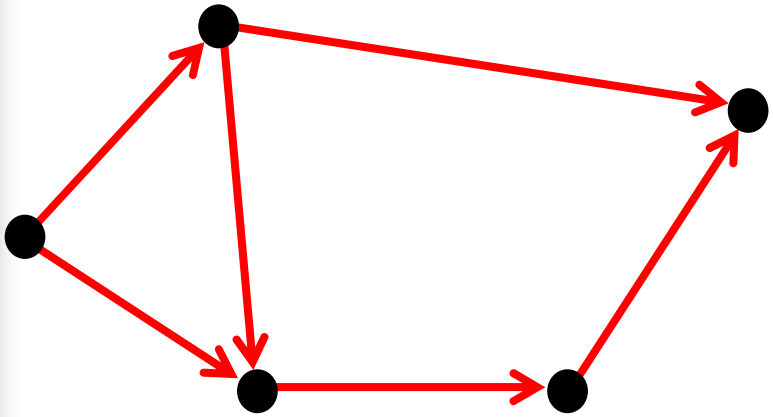
Directed graph



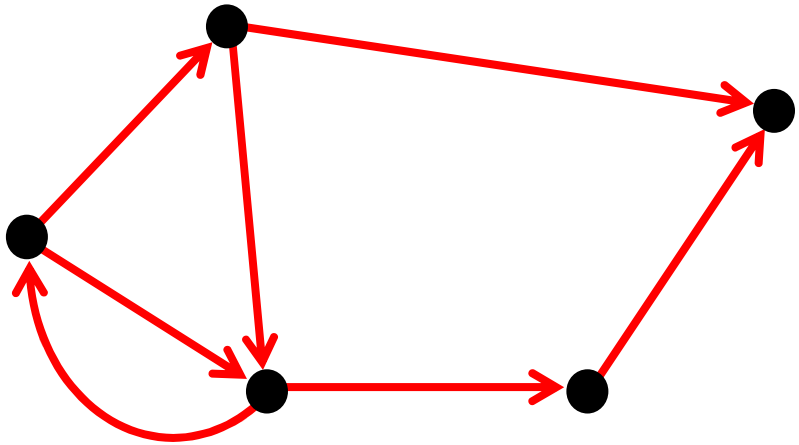
Directed graph



Directed graph

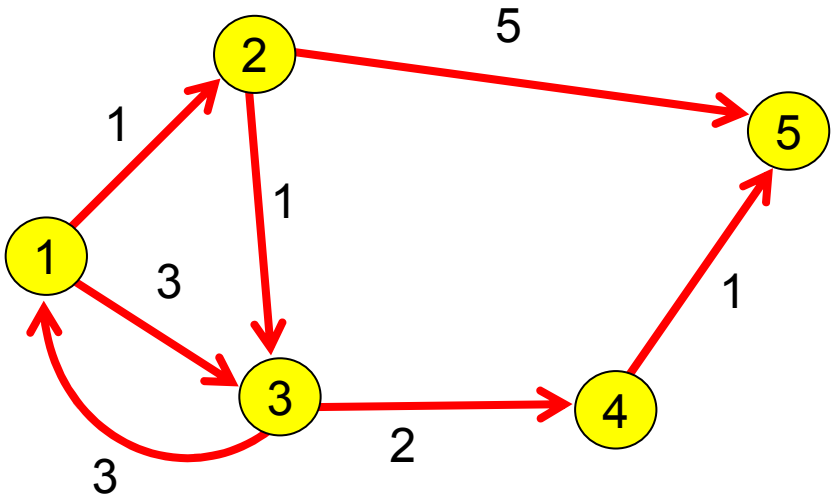


One-way traffic
(single path)



Two-way traffic
(multi path)

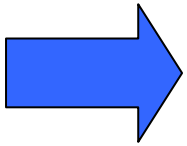
Representasi beban



	1	2	3	4	5
1		1	3	-	-
2	-		1	-	5
3	3	-		2	-
4	-	-	-		1
5	-	-	-	-	

Matriks beban

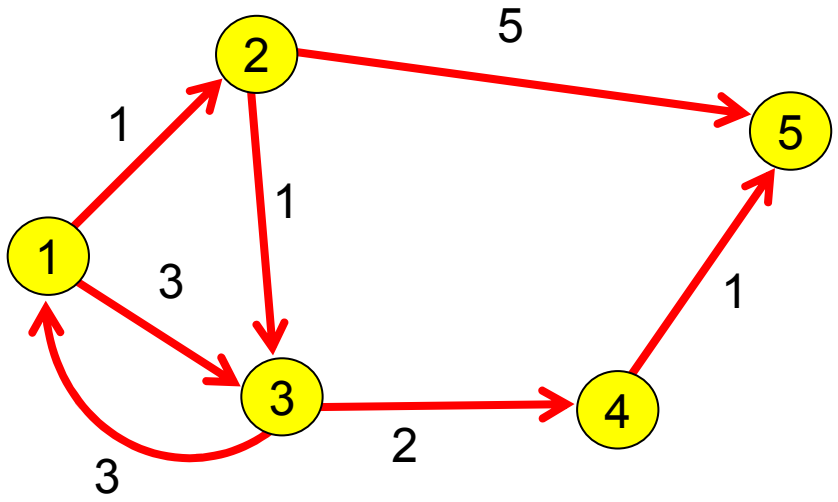
	1	2	3	4	5
1		1	3	-	-
2	-		1	-	5
3	3	-		2	-
4	-	-	-		1
5	-	-	-	-	



Q	1	2	3	4	5
1	M	1	3	M	M
2	M	M	1	M	5
3	3	M	M	2	M
4	M	M	M	M	1
5	M	M	M	M	M

```
#define M … //big integer
```

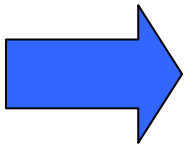
Representasi Jalur



	1	2	3	4	5
1		√	√	-	-
2	-		√	-	√
3	√	-		√	-
4	-	-	-		√
5	-	-	-	-	

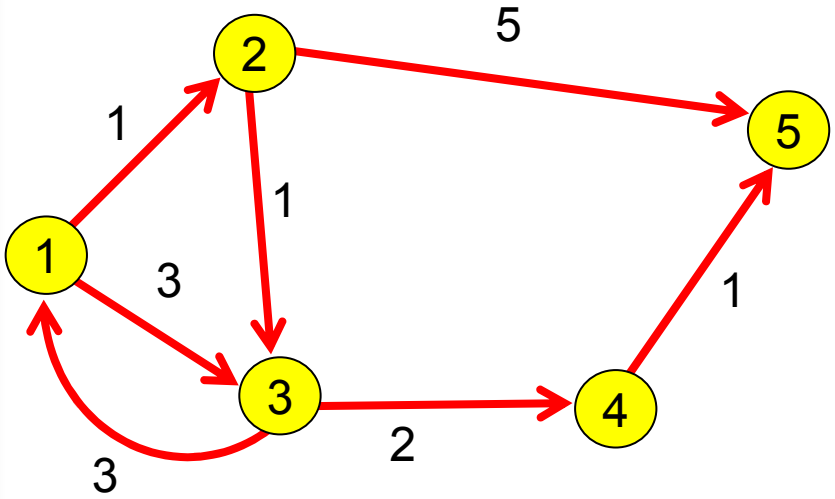
Matriks Jalur

	1	2	3	4	5
1		√	√	-	-
2	-		√	-	√
3	√	-		√	-
4	-	-	-		√
5	-	-	-	-	



P	1	2	3	4	5
1	0	1	1	0	0
2	0	0	1	0	1
3	1	0	0	1	0
4	0	0	0	0	1
5	0	0	0	0	0

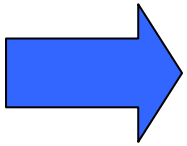
Representasi Rute



	1	2	3	4	5
1		0	0	-	-
2	-		0	-	0
3	0	-		0	-
4	-	-	-		0
5	-	-	-	-	

Matriks rute

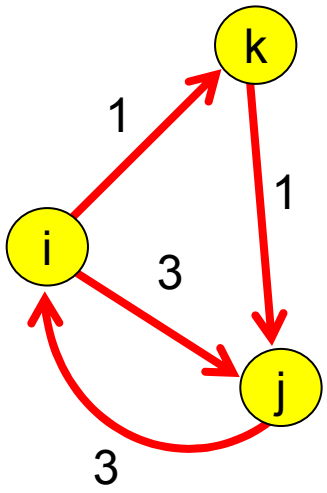
	1	2	3	4	5
1		0	0	-	-
2	-		0	-	0
3	0	-		0	-
4	-	-	-		0
5	-	-	-	-	



R	1	2	3	4	5
1	M	0	0	M	M
2	M	M	0	M	0
3	0	M	M	0	M
4	M	M	M	M	0
5	M	M	M	M	M

```
#define M … //big integer
```

Shortest path problem
Multi path
(Algoritma Warshall)

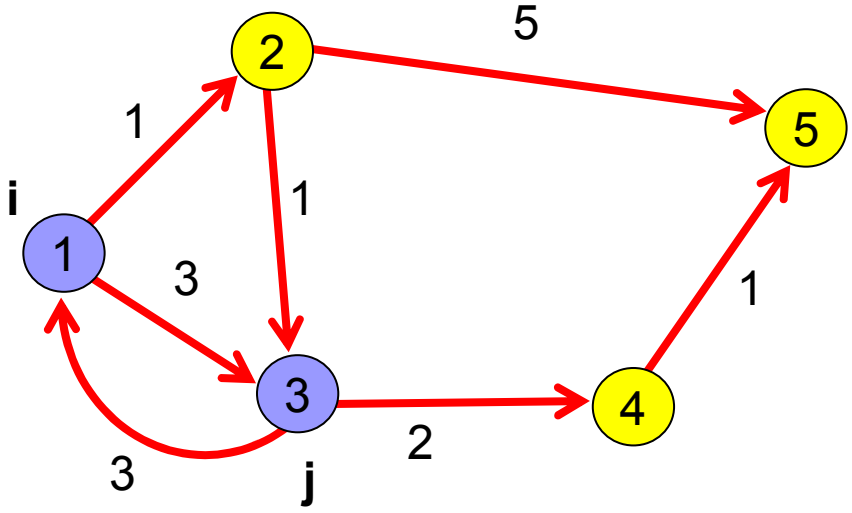


Melakukan pengecekan apakah beban langsung Q(i, j) memang lebih kecil daripada beban melalui titik perantara Q(i,k)+Q(k,j)

if ((Q(i,k)+Q(k,j))< Q(i, j))

Q(i, j) ← Q(i,k)+Q(k,j)

Q	1	2	3	4	5
1	M	1	3	M	M
2	M	M	1	M	5
3	3	M	M	2	M
4	M	M	M	M	1
5	M	M	M	M	M



Q(1,3) = 3 → Beban langsung

Q(1,1) + Q(1,3) = M+3
Q(1,2) + Q(2,3) = 2
Q(1,3) + Q(3,3) = 3+M
Q(1,4) + Q(4,3) = M+M
Q(1,5) + Q(5,3) = M+M

} Beban melalui perantara

Q(1,3) = 2

```
if ((Q(i,k)+Q(k,j))< Q(i, j))
        Q(i, j) ← Q(i,k)+Q(k,j)
```

Algoritma Warshall
(untuk beban)

```
for k=1 to n
    for i=1 to n
        for j=1 to n
            if ((Q(i,k)+Q(k,j)) < Q(i,j))
                Q(i,j) ← (Q(i,k)+Q(k,j)
```

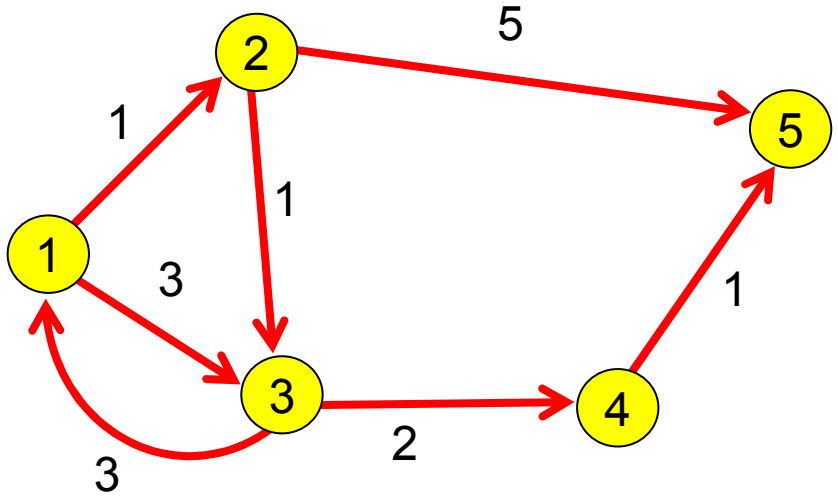
Algoritma Warshall
(untuk jalur)

```
for k=1 to n
    for i=1 to n
        for j=1 to n
            P(i,j) ← P(i,j) OR (P(i,k) AND P(k,j))
```

Pencarian rute

```
for k=1 to n
    for i=1 to n
        for j=1 to n
            if ((Q(i,k) + Q(k,j)) < Q(i,j)) {
               if (R(k,j) = 0)
                        R(i,j) ←k
               else
                        R(i,j) ← R(k,j)
            }
```

Cara membaca matriks rute



R	1	2	3	4	5
1	3	0	2	3	4
2	3	1	0	3	4
3	0	1	2	0	4
4	M	M	M	M	0
5	M	M	M	M	M

Rute 1-5 ?

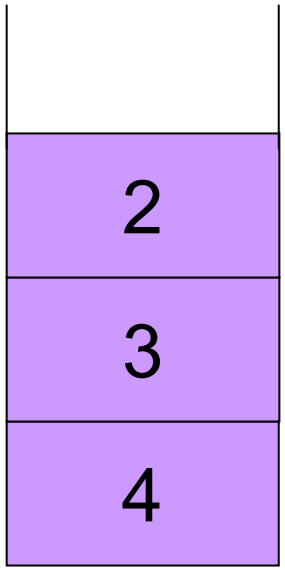
Rute 1-5?

Ambil nilai di baris 1, kolom 5 = 4 ➔ push

Ambil nilai di baris 1, kolom 4 = 3 ➔ push

Ambil nilai di baris 1, kolom 3 = 2 ➔ push

Ambil nilai di baris 1, kolom 2 = 0 (stop) ➔ pop



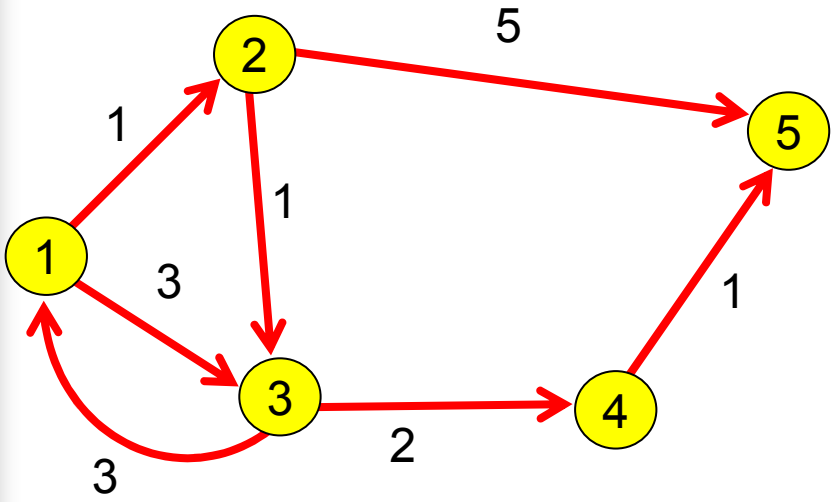
Rute =
1 – 2 – 3 – 4 - 5

R	1	2	3	4	5
1	3	0	2	3	4
2	3	1	0	3	4
3	0	1	2	0	4
4	M	M	M	M	0
5	M	M	M	M	M

Shortest path problem

Single path

(Algoritma Dijkstra)



- Tentukan titik asal dan titik tujuan sebelum proses
- Akumulasikan jarak minimal dan simpan ke titik berikutnya.
- Lakukan dari titik asal sampai titik tujuan

Algoritma Dijkstra

1. Buat matriks I(berukuran n x n) untuk menerima masukan beban dari semua node
2. Masukkan node `asal` & `tujuan`
3. Buat matriks Q (berukuran n) untuk menyimpan beban total minimal.

   Inisialisasi: node asal diberi nilai 0, selainnya diberi nilai M (big integer)
4. Buat matriks R (berukuran n) untuk menyimpan rute yang ditempuh.

   Inisialisasi: semua node diberi nilai : -1
5. Insialisasi Queue.
6. `enqueue(asal)`

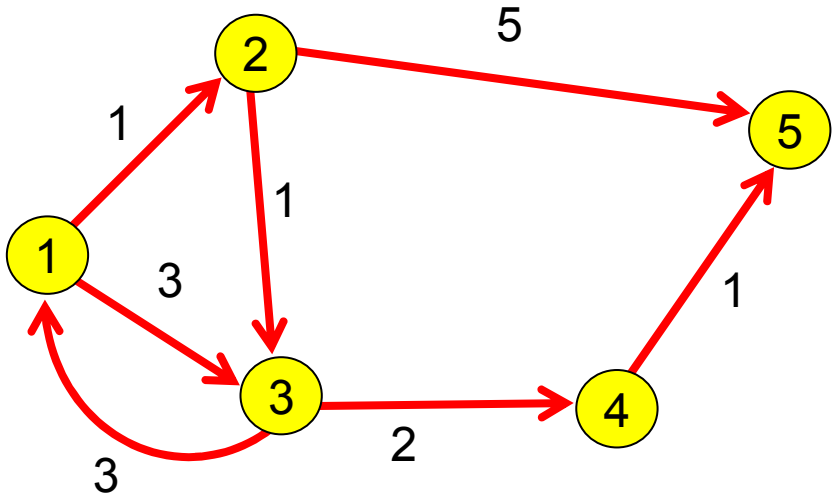
Algoritma Dijkstra
(continued)

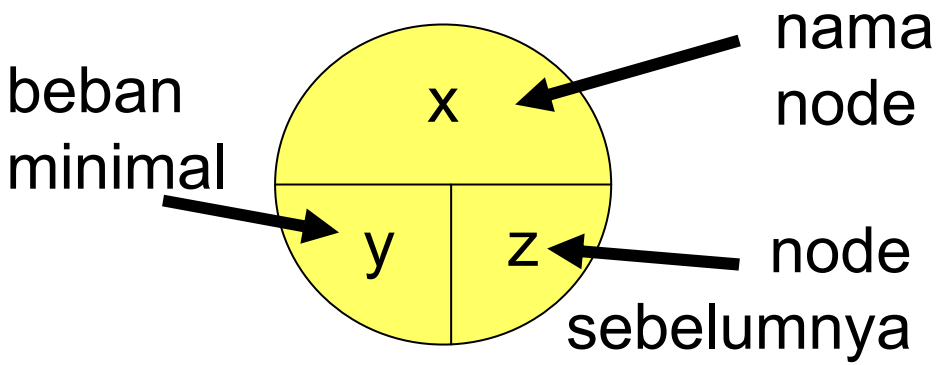
7. Selama Queue tidak kosong lakukan langkah 8 – 16
8. `currentNode` ← `dequeue()`
9. `i` ← `0`
10. Selama `i < n` lakukan langkah 11 – 16
11. Jika `I[currentNode][i] != M` lakukan langkah 12 – 15
12. Jika `I[currentNode][i] + Q[currentNode] < Q[i]` lakukan 13 & 15
13. `Q[i]` ← `I[currentNode][i] + Q[currentNode]`
14. `R[i]` ← `currentNode`
15. Jika `i != asal && i != tujuan` && i tdk ada dlm Queue

    `enqueue(i)`
16. `i++`

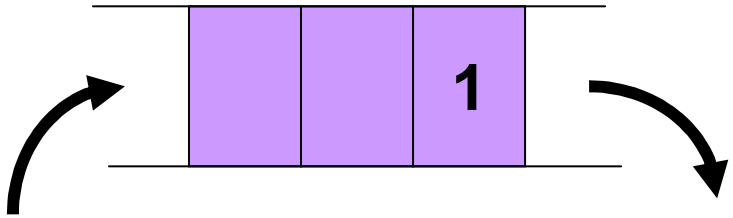
Algoritma Dijkstra

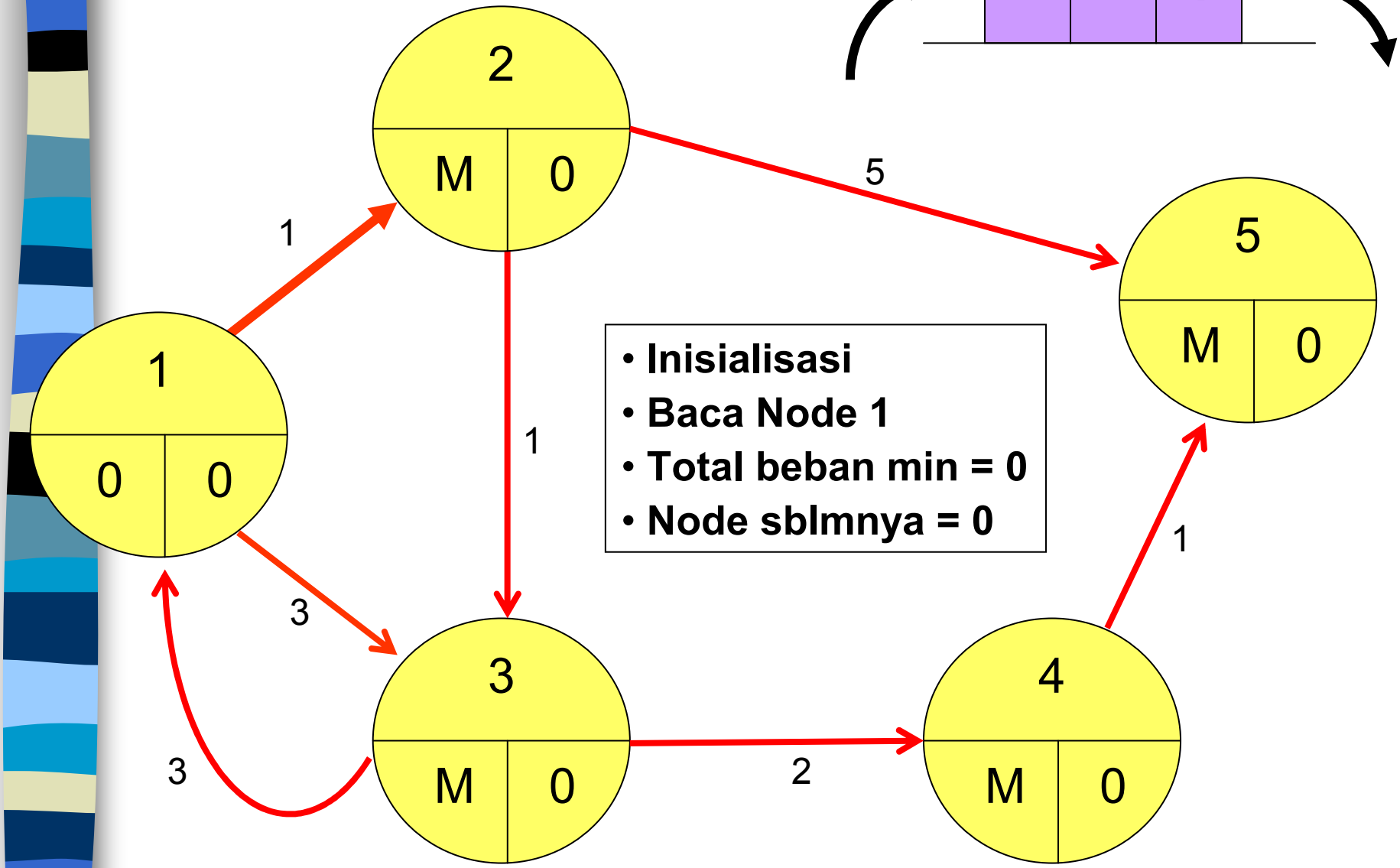
Titik asal = 1
Titik tujuan = 5



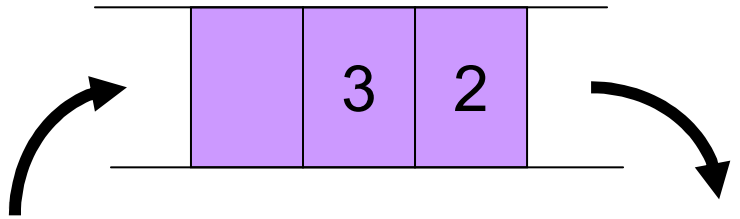


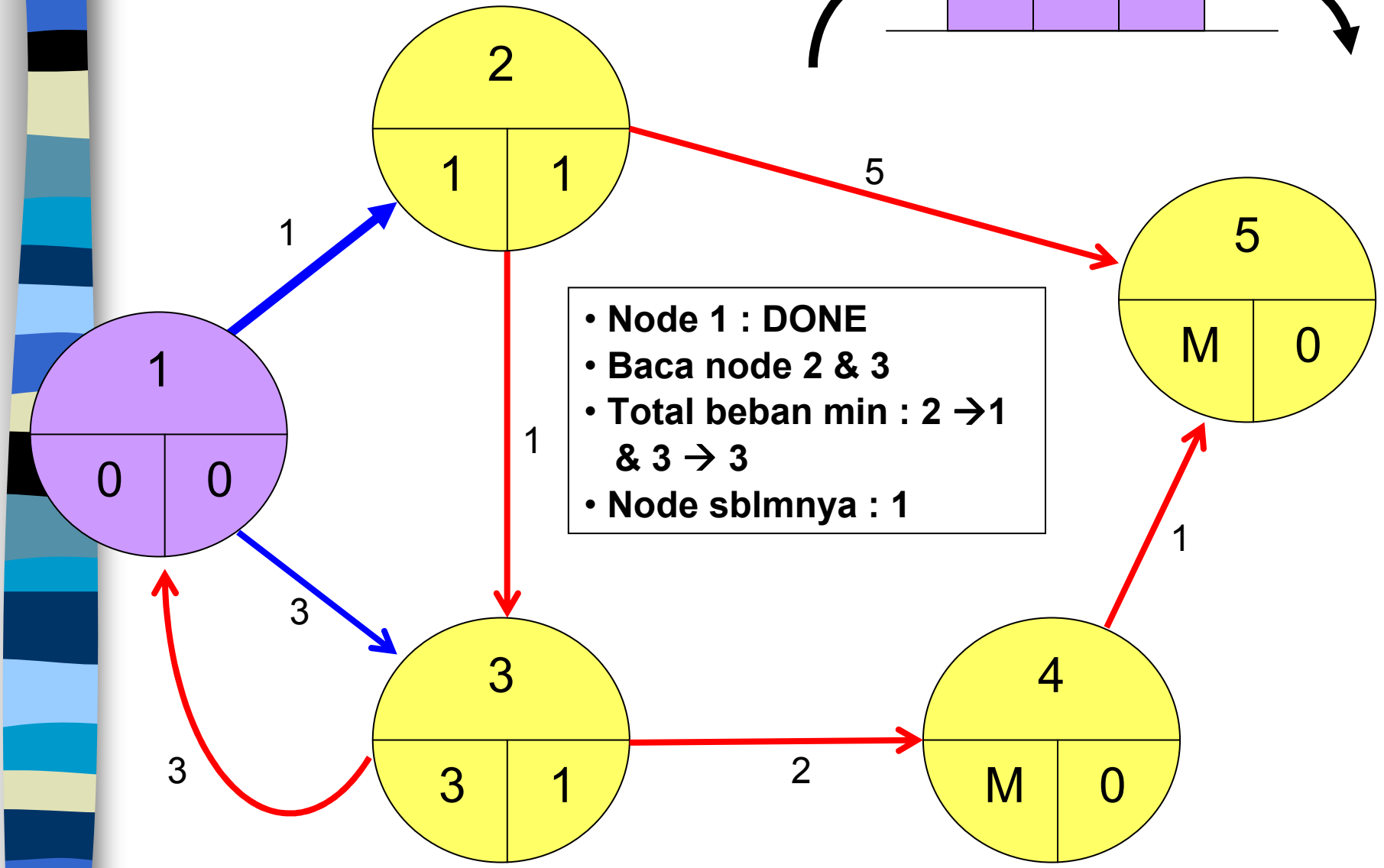
Algoritma Dijkstra





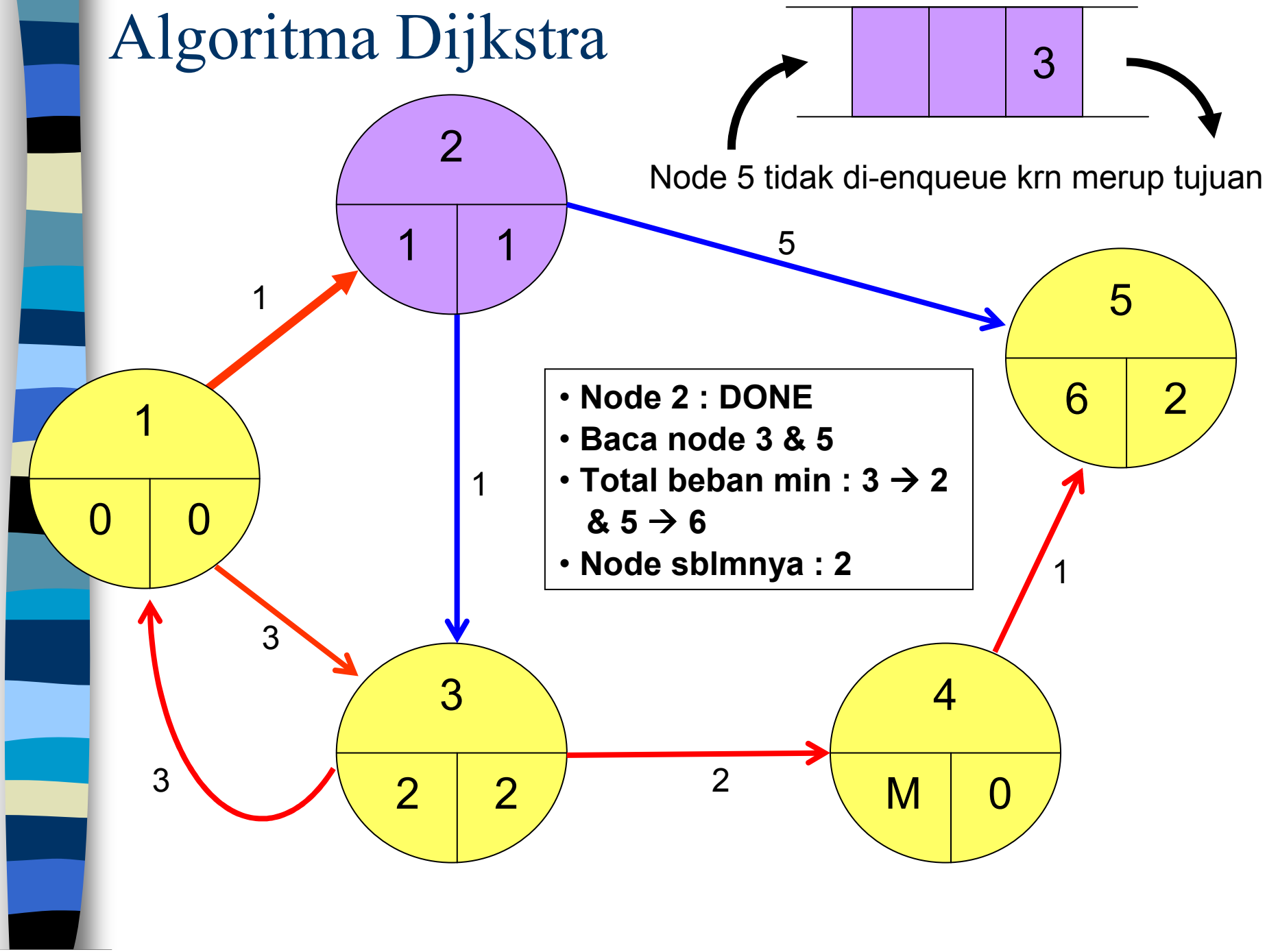
Algoritma Dijkstra



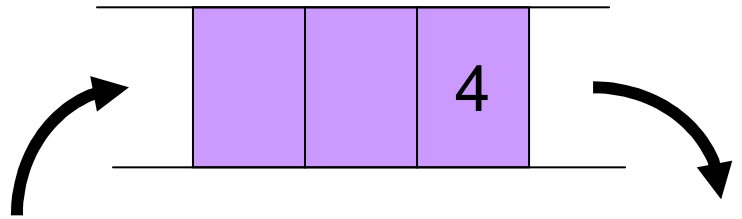


- **Node 1 : DONE**
- **Baca node 2 & 3**
- **Total beban min : 2 →1 & 3 → 3**
- **Node sblmnya : 1**

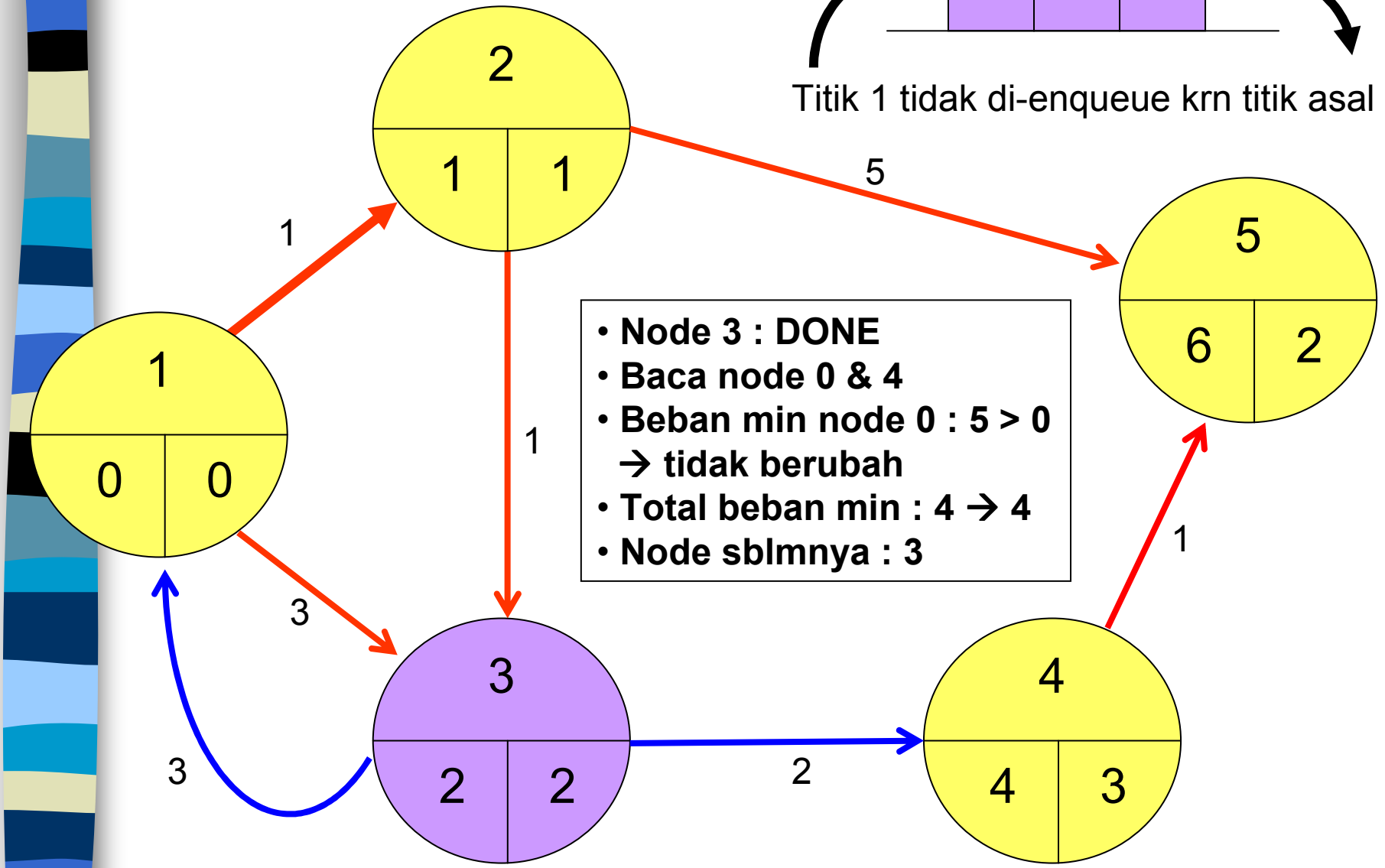
Algoritma Dijkstra



Algoritma Dijkstra

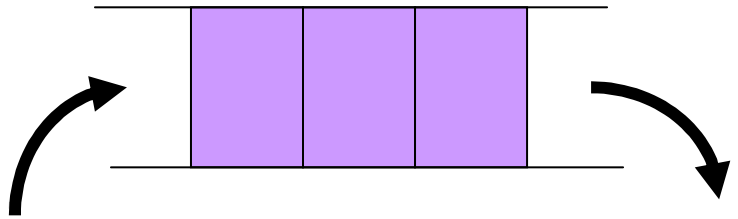


Titik 1 tidak di-enqueue krn titik asal

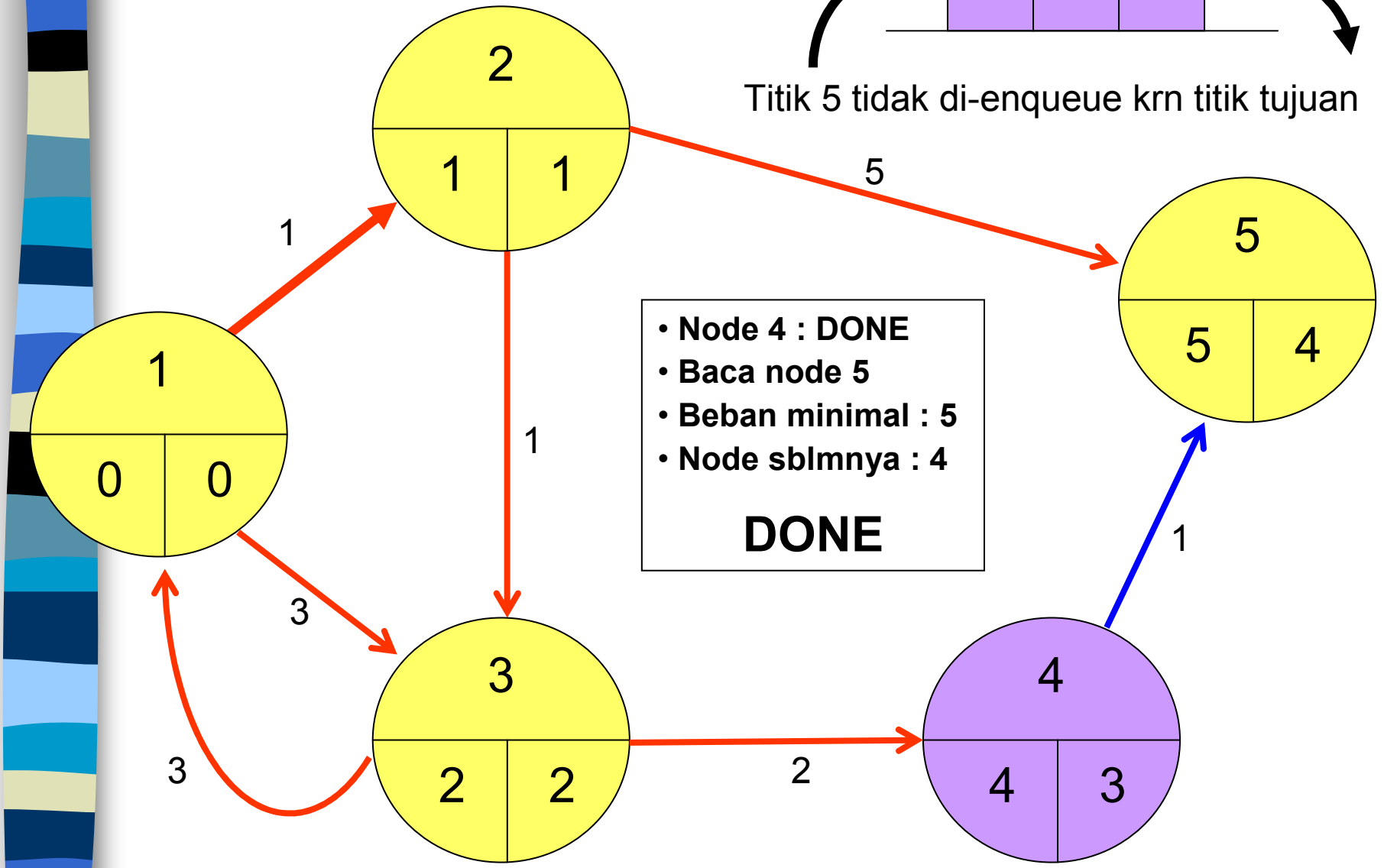


- **Node 3 : DONE**
- **Baca node 0 & 4**
- **Beban min node 0 : 5 > 0 → tidak berubah**
- **Total beban min : 4 → 4**
- **Node sblmnya : 3**

Algoritma Dijkstra



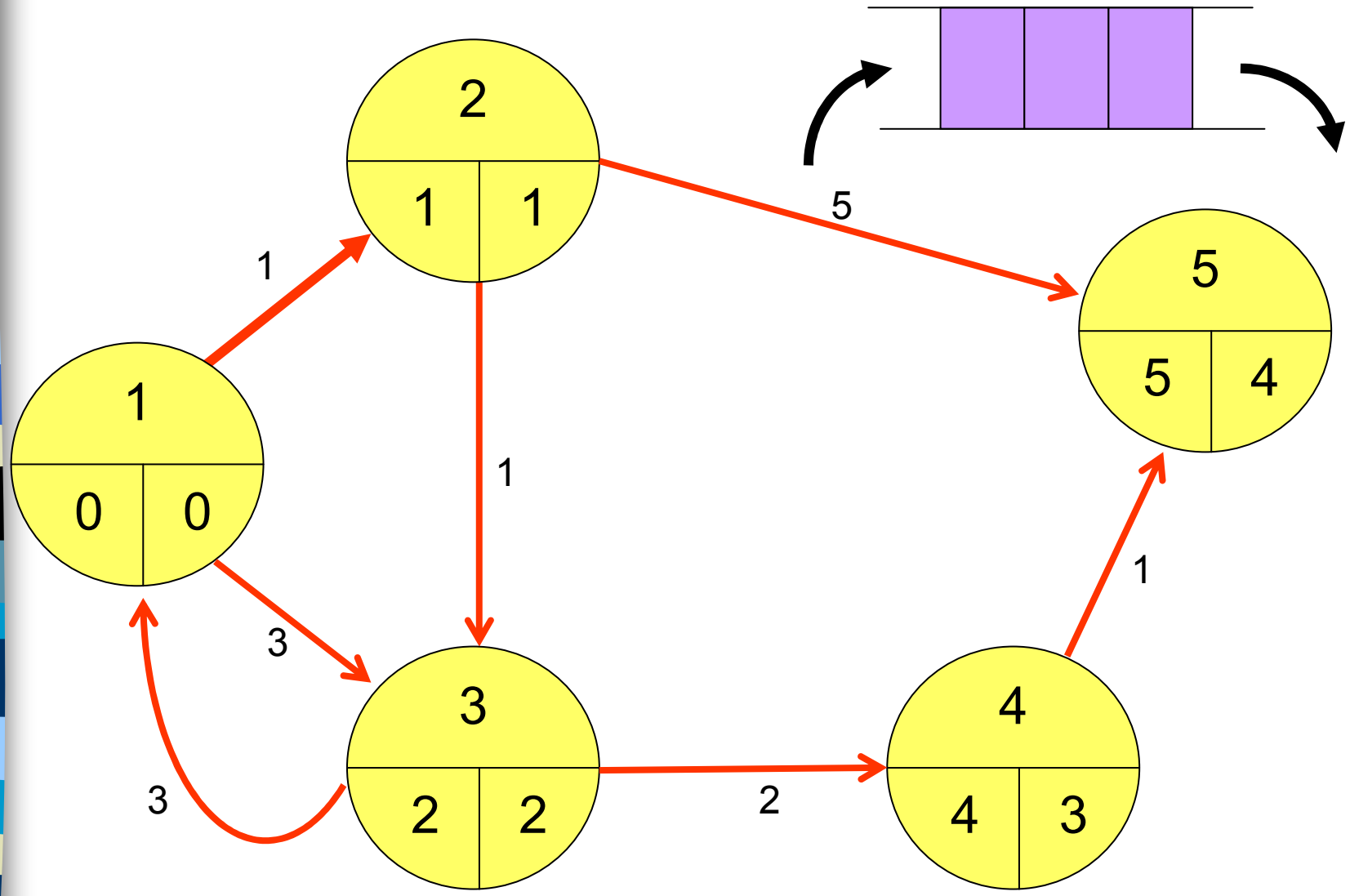
Titik 5 tidak di-enqueue krn titik tujuan



- **Node 4 : DONE**
- **Baca node 5**
- **Beban minimal : 5**
- **Node sblmnya : 4**

DONE

Algoritma Dijkstra



Rute : 1 – 2 – 3 – 4 – 5 dengan beban minimal = 5