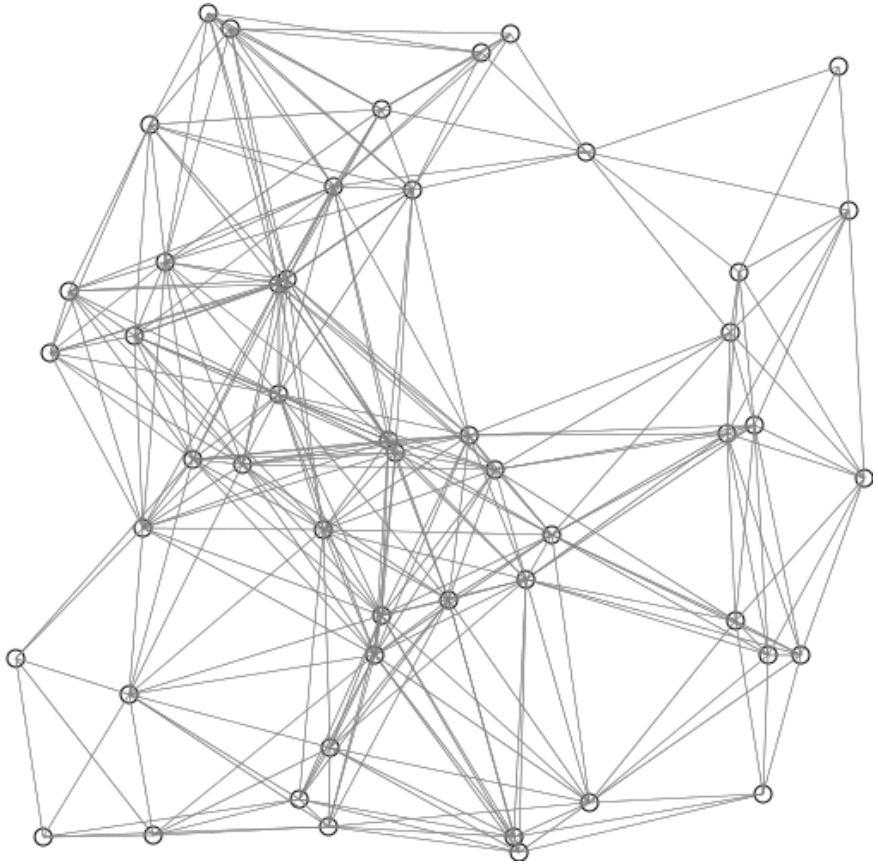
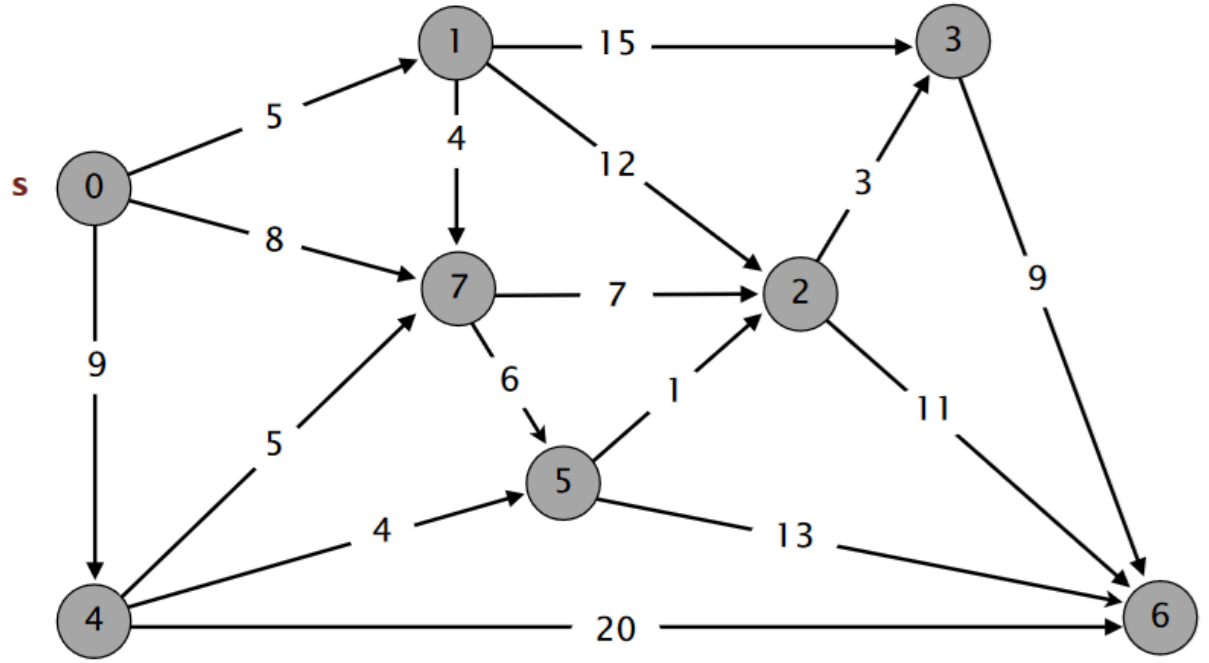
Pseudo Código Dijkstra (Con MinPQ)

```
1) function Dijkstra(Graph, source):
2)     dist[source] := 0                           // Initialization
3)     // create vertex priority queue Q
4)     for each vertex v in Graph.Vertices:
5)         if v ≠ source
6)             dist[v] := INFINITY                 // Unknown distance from source to v
7)             prev[v] := UNDEFINED                // Predecessor of v
8)         Q.add_with_priority(v, dist[v])
9)     while Q is not empty:                      // The main loop
10)        u := Q.extract_min()                    // Remove and return best vertex
11)        for each neighbor v of u:               // only v that are still in Q
12)            alt := dist[u] + Graph.Edges(u, v)
13)            if alt < dist[v]
14)                dist[v] := alt
15)                prev[v] := u
16)                Q.decrease_priority(v, alt)
17)    return dist, prev
```

Demo Dijkstra (con MinPQ)

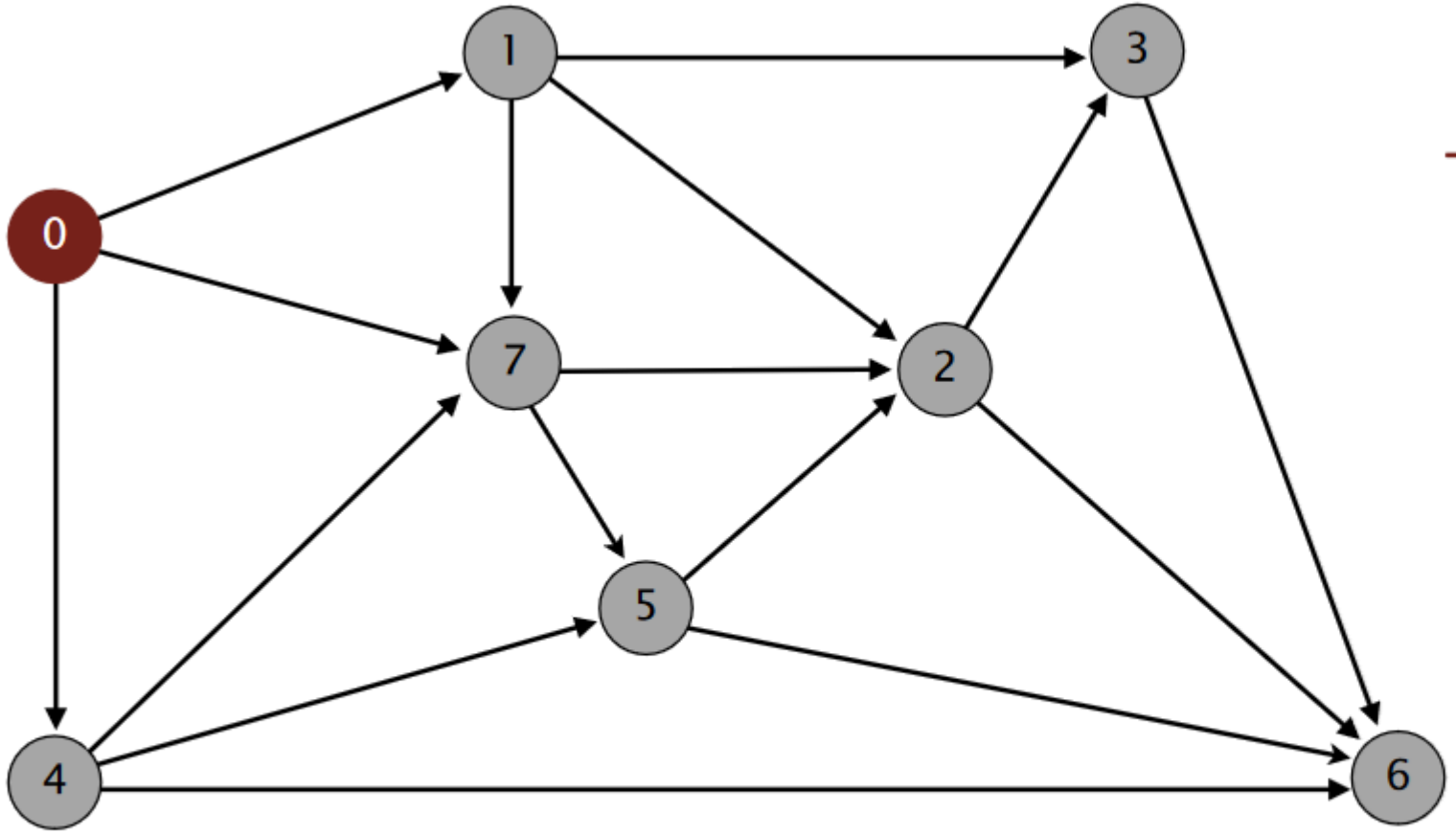


Ejemplo de ejecución Dijkstra (con MinPQ)



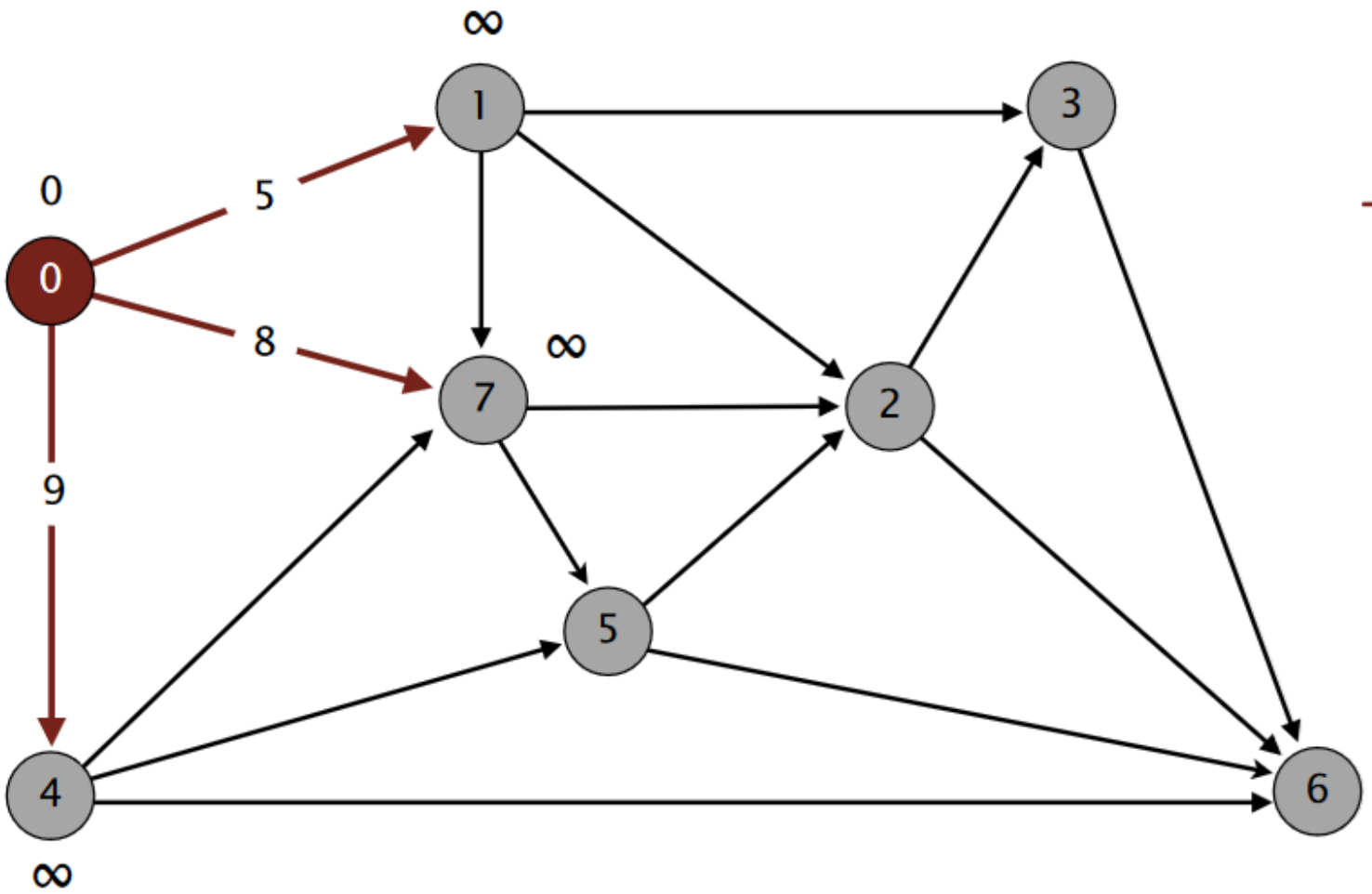
an edge-weighted digraph

```
0→1   5.0
0→4   9.0
0→7   8.0
1→2  12.0
1→3  15.0
1→7   4.0
2→3   3.0
2→6  11.0
3→6   9.0
4→5   4.0
4→6  20.0
4→7   5.0
5→2   1.0
5→6  13.0
7→5   6.0
7→2   7.0
```



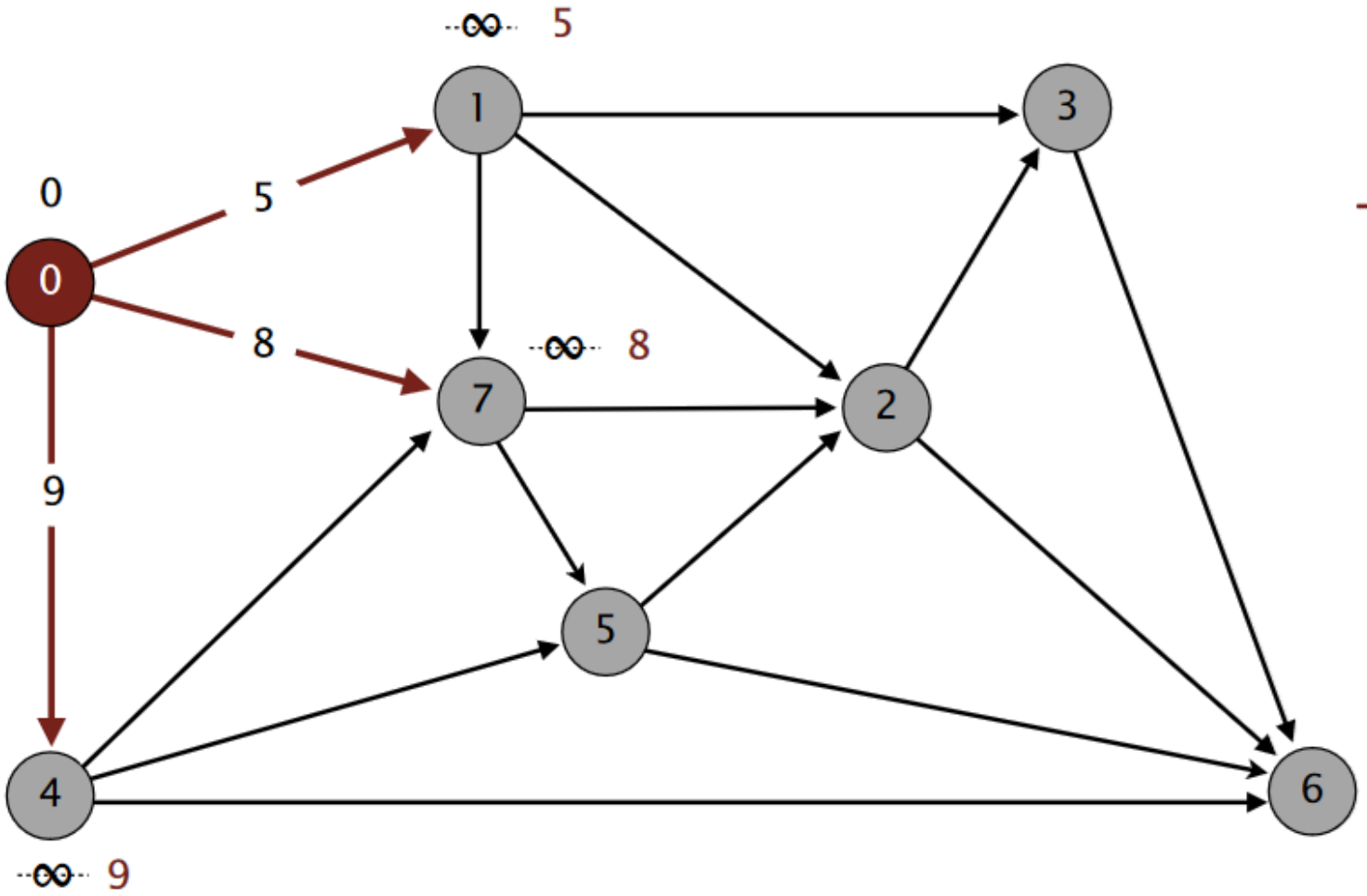
v	distTo[]	edgeTo[]
→ 0	0.0	-
1		
2		
3		
4		
5		
6		
7		

choose source vertex 0



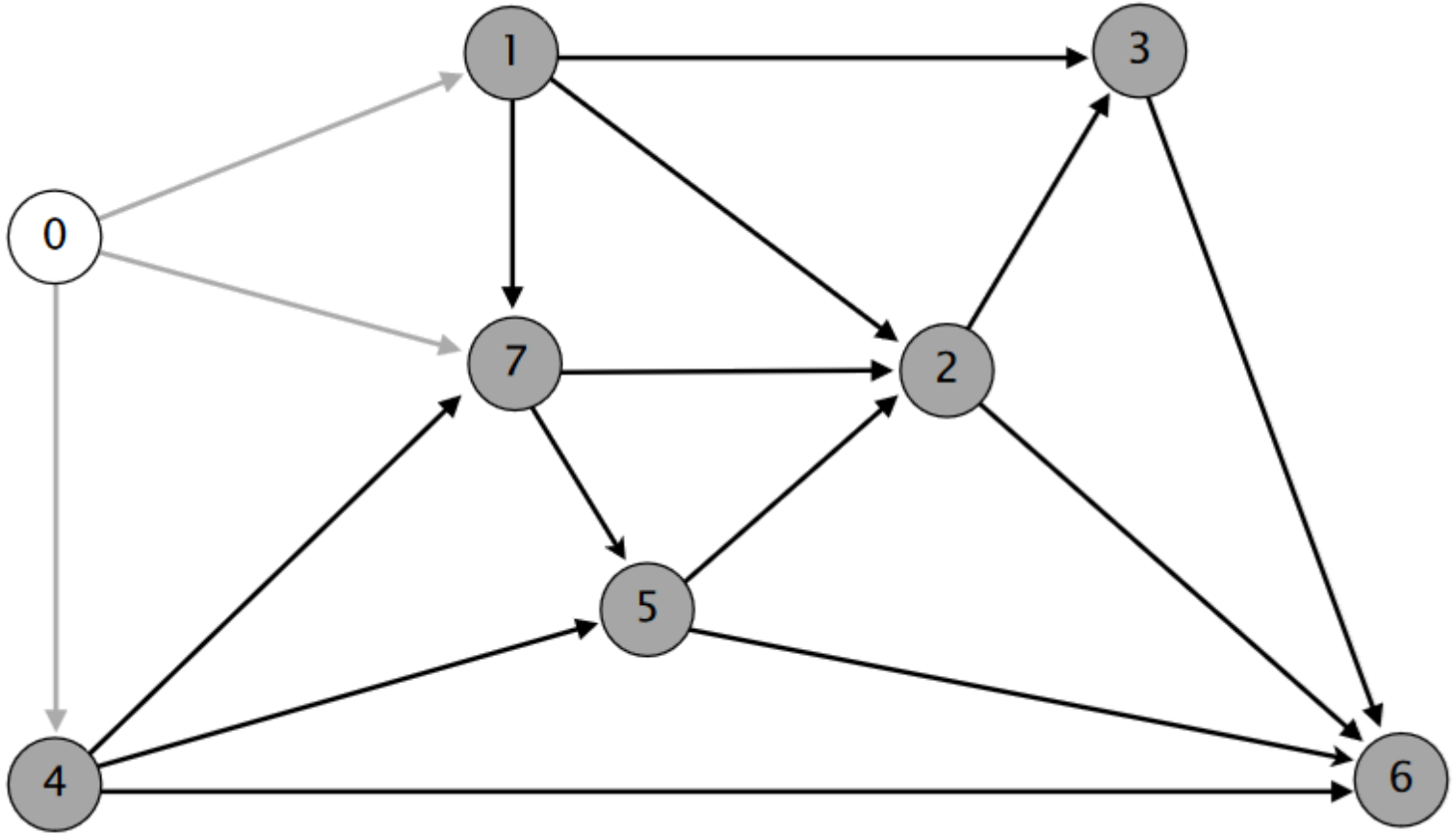
v	distTo[]	edgeTo[]
→ 0	0.0	-
1		
2		
3		
4		
5		
6		
7		

relax all edges pointing from 0

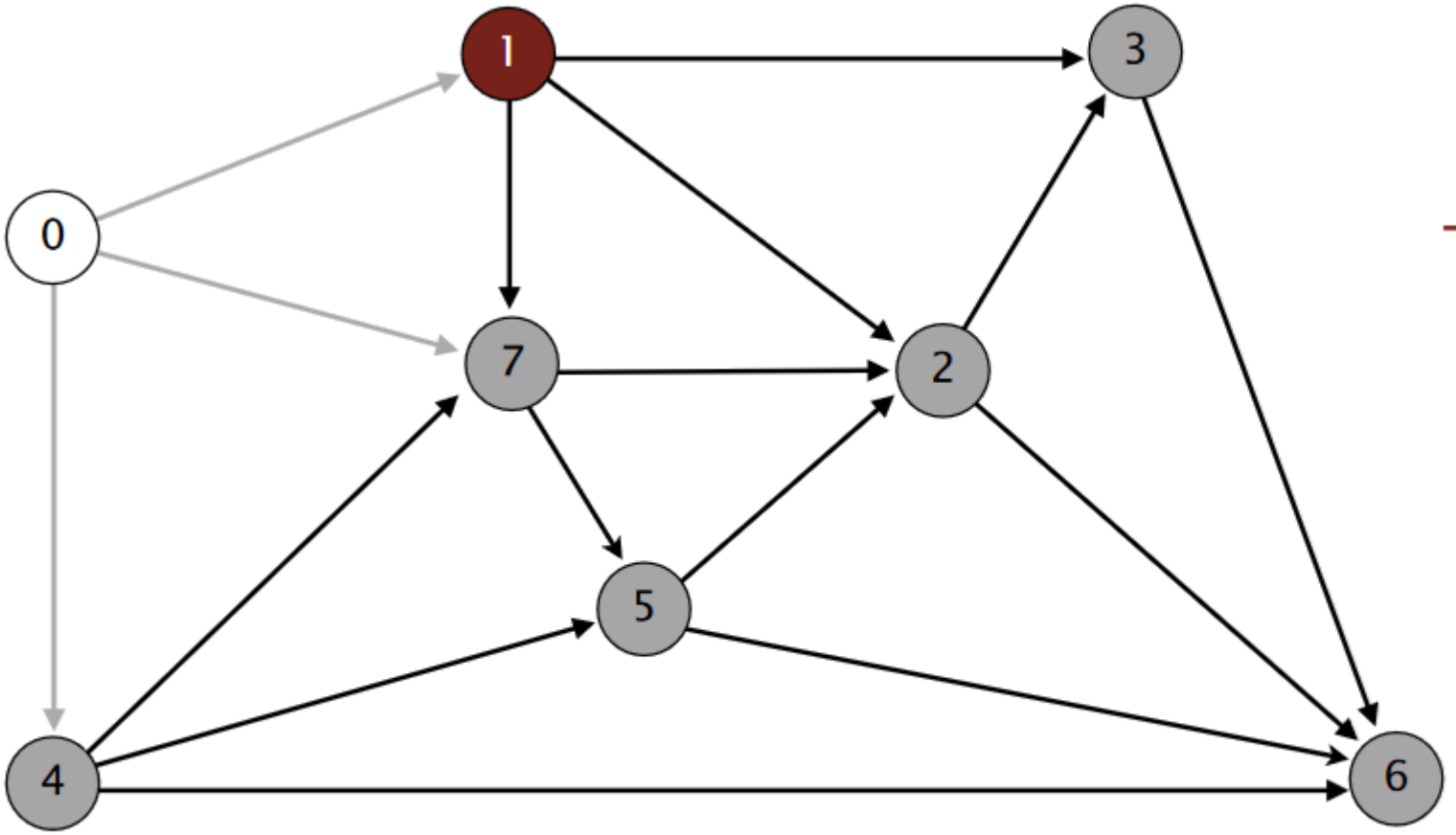


v	distTo[]	edgeTo[]
→ 0	0.0	-
1	5.0	0→1
2		
3		
4	9.0	0→4
5		
6		
7	8.0	0→7

relax all edges pointing from 0

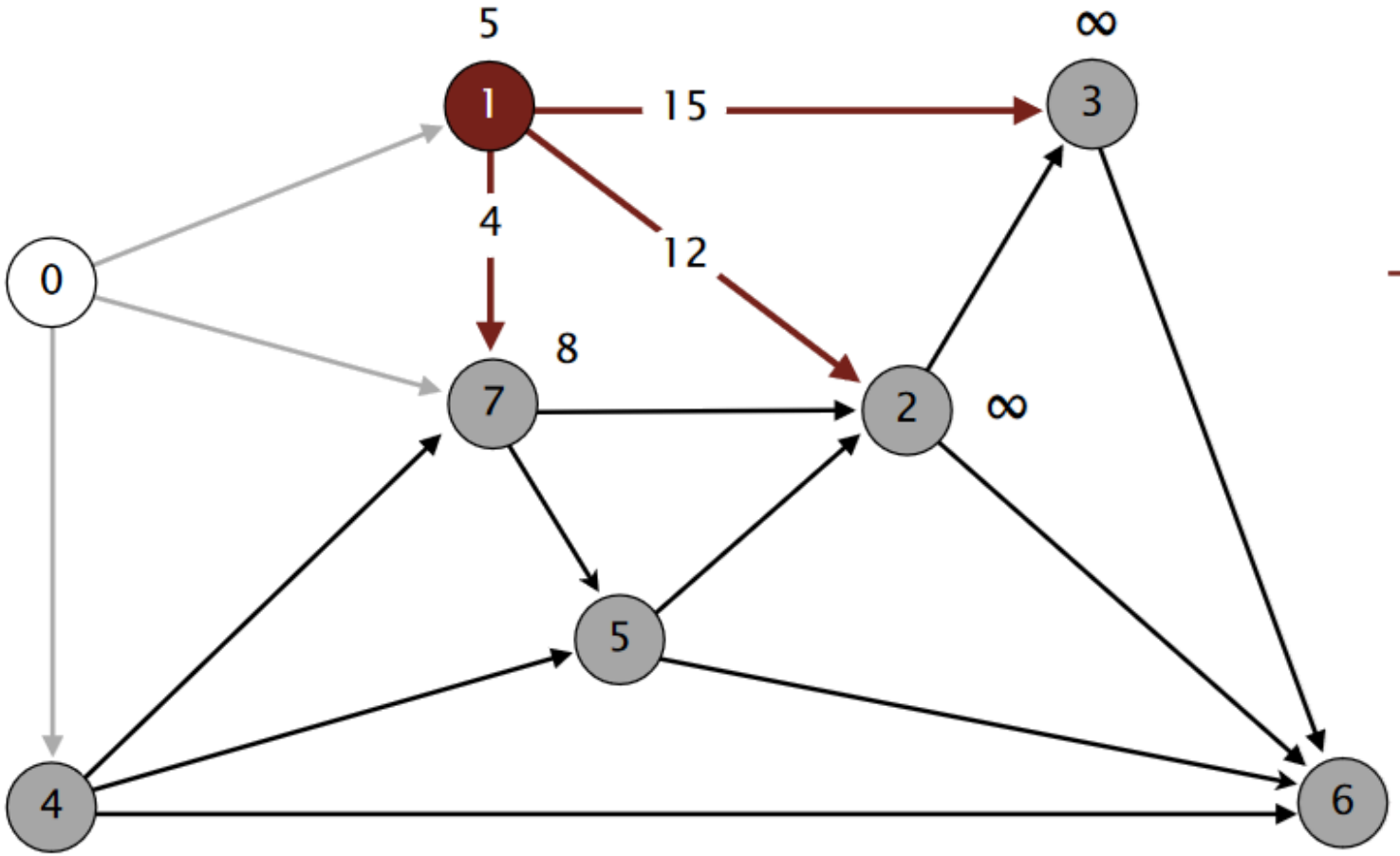


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2		
3		
4	9.0	0→4
5		
6		
7	8.0	0→7



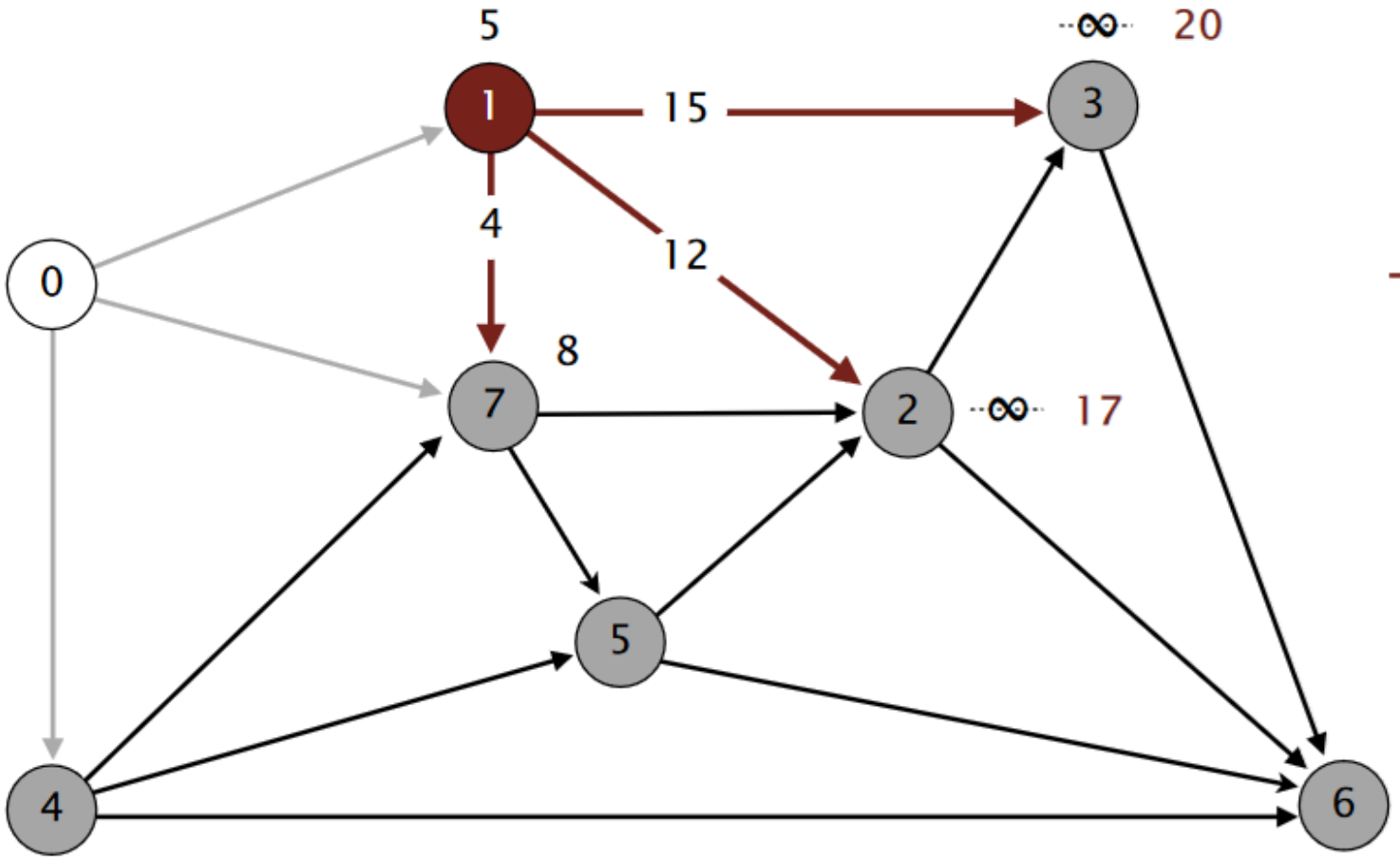
v	distTo[]	edgeTo[]
0	0.0	-
→ 1	5.0	0→1
2		
3		
4	9.0	0→4
5		
6		
7	8.0	0→7

choose vertex 1



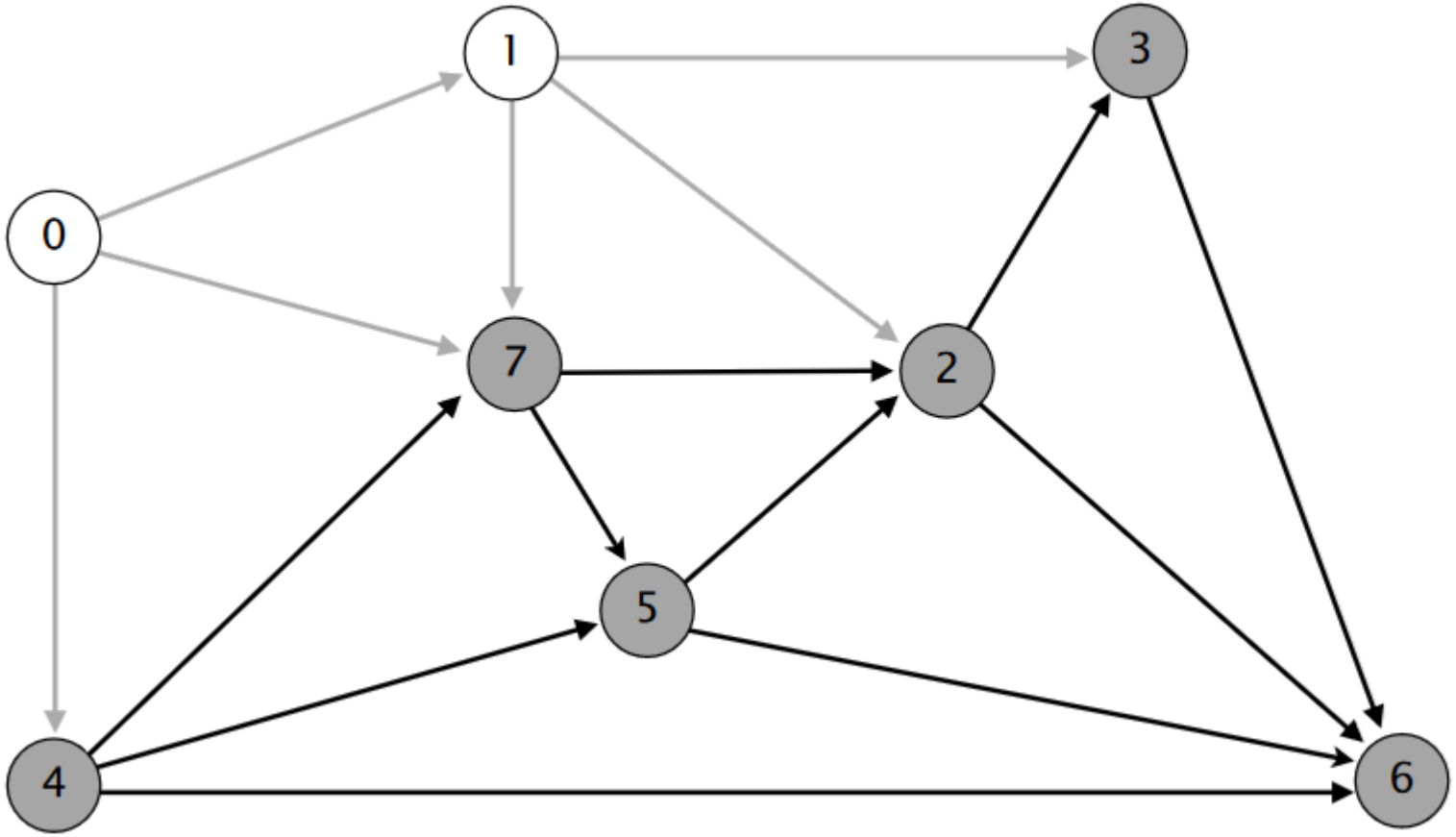
v	distTo[]	edgeTo[]
0	0.0	-
→ 1	5.0	0→1
2		
3		
4	9.0	0→4
5		
6		
7	8.0	0→7

relax all edges pointing from 1

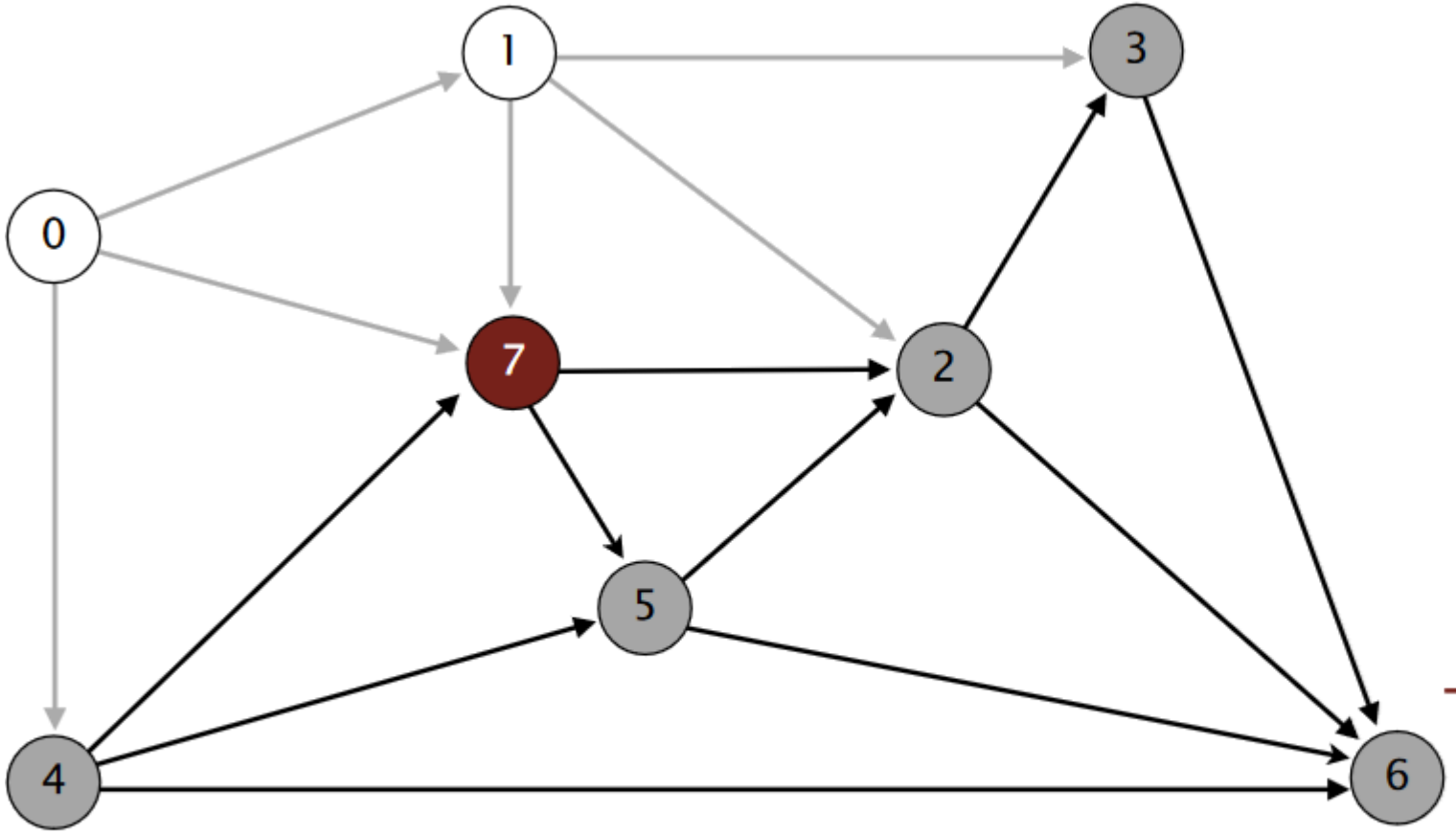


v	distTo[]	edgeTo[]
0	0.0	-
→ 1	5.0	0→1
2	17.0	1→2
3	20.0	1→3
4	9.0	0→4
5		
6		
7	8.0 ✔	0→7

relax all edges pointing from 1

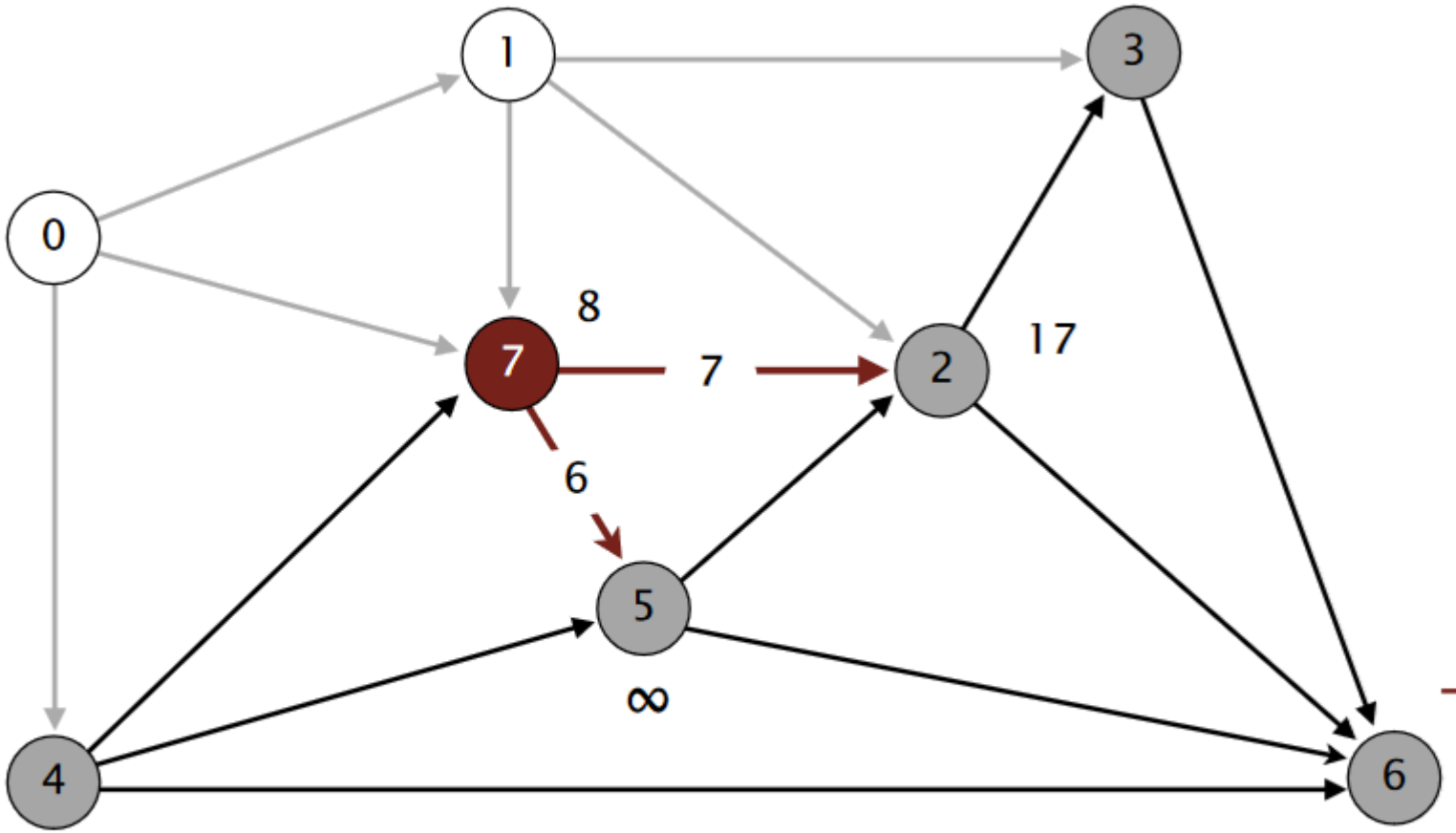


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	17.0	1→2
3	20.0	1→3
4	9.0	0→4
5		
6		
7	8.0	0→7



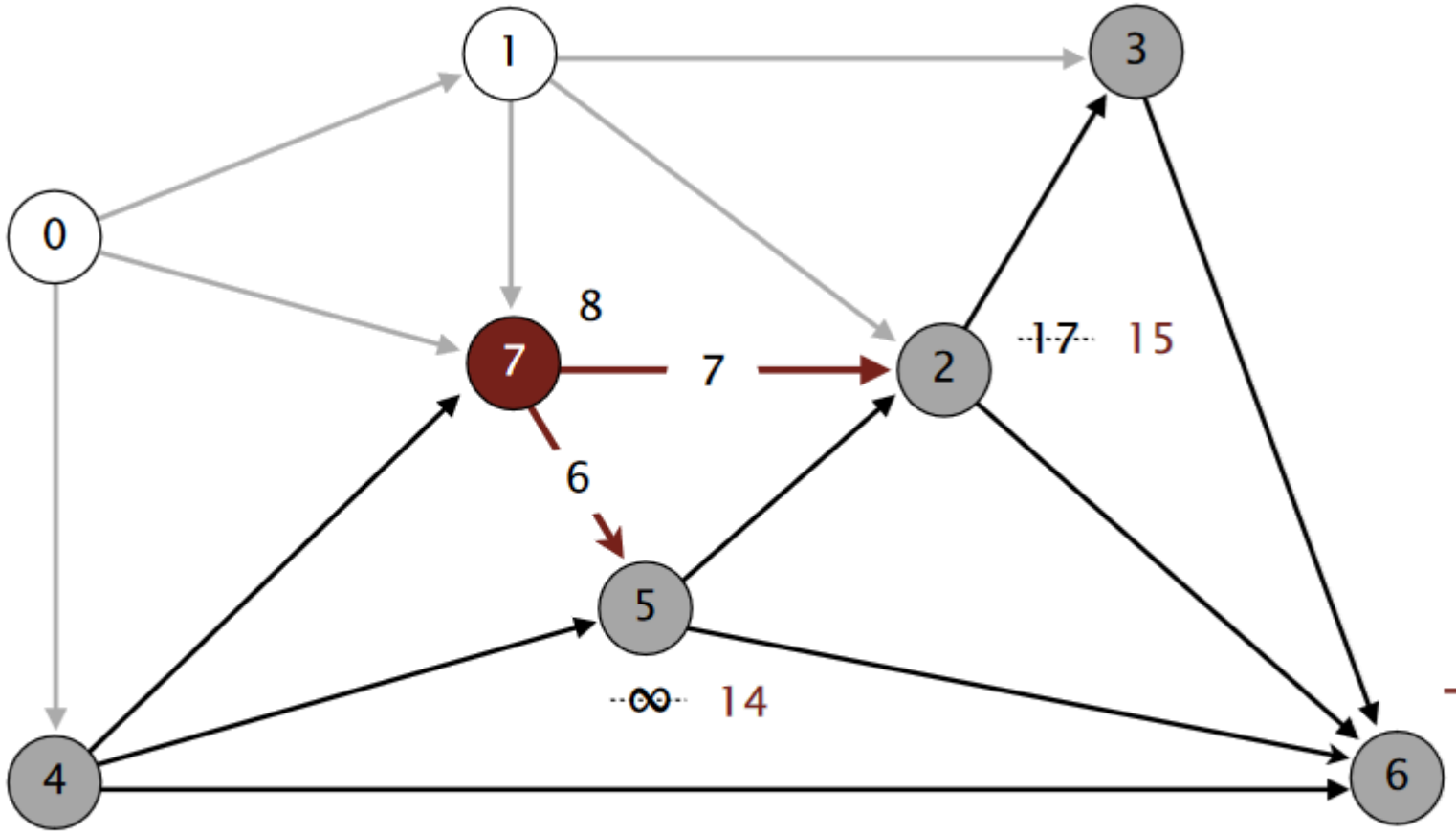
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	17.0	1→2
3	20.0	1→3
4	9.0	0→4
5		
6		
→ 7	8.0	0→7

choose vertex 7



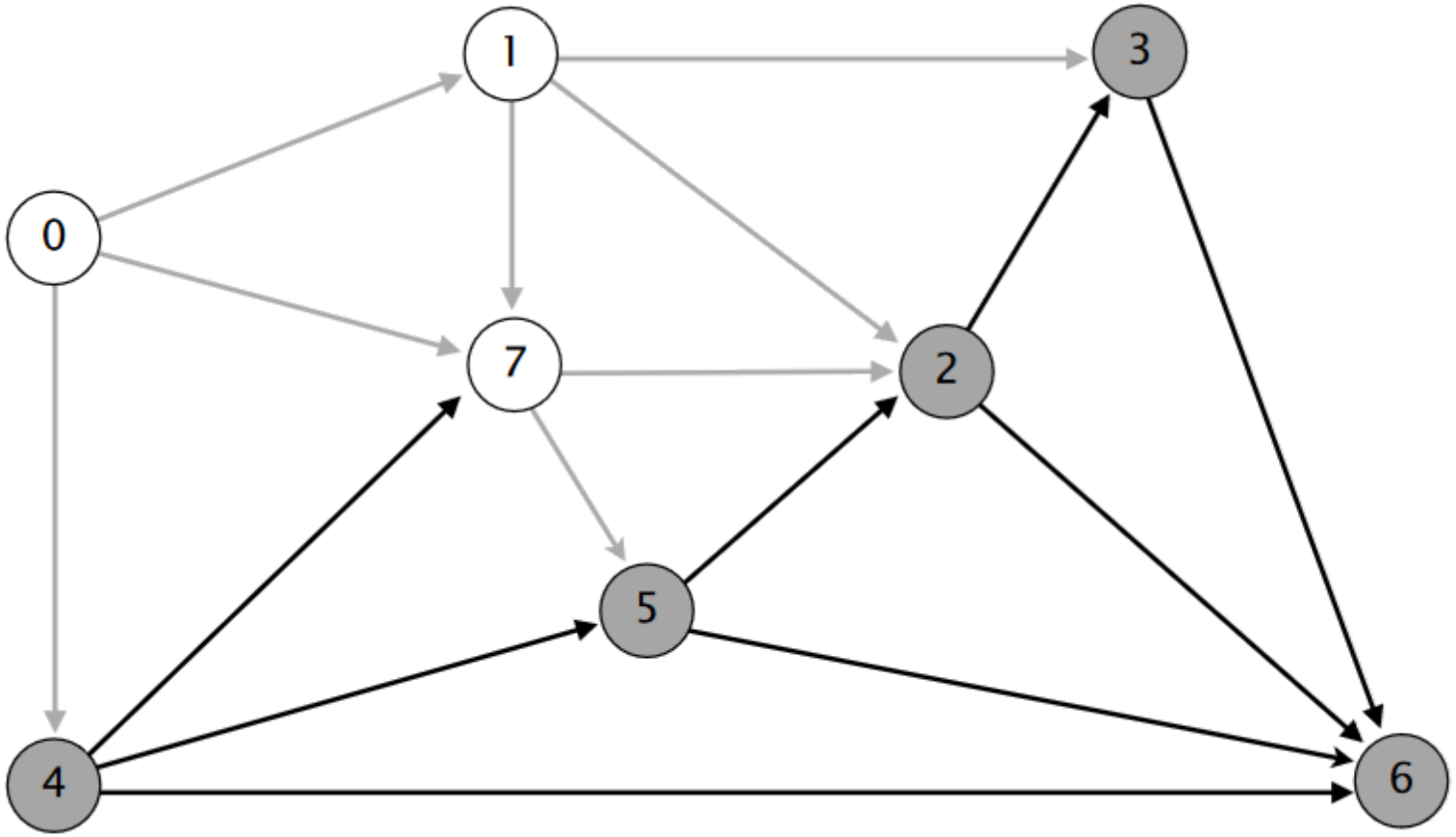
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	17.0	1→2
3	20.0	1→3
4	9.0	0→4
5		
6		
→ 7	8.0	0→7

relax all edges pointing from 7

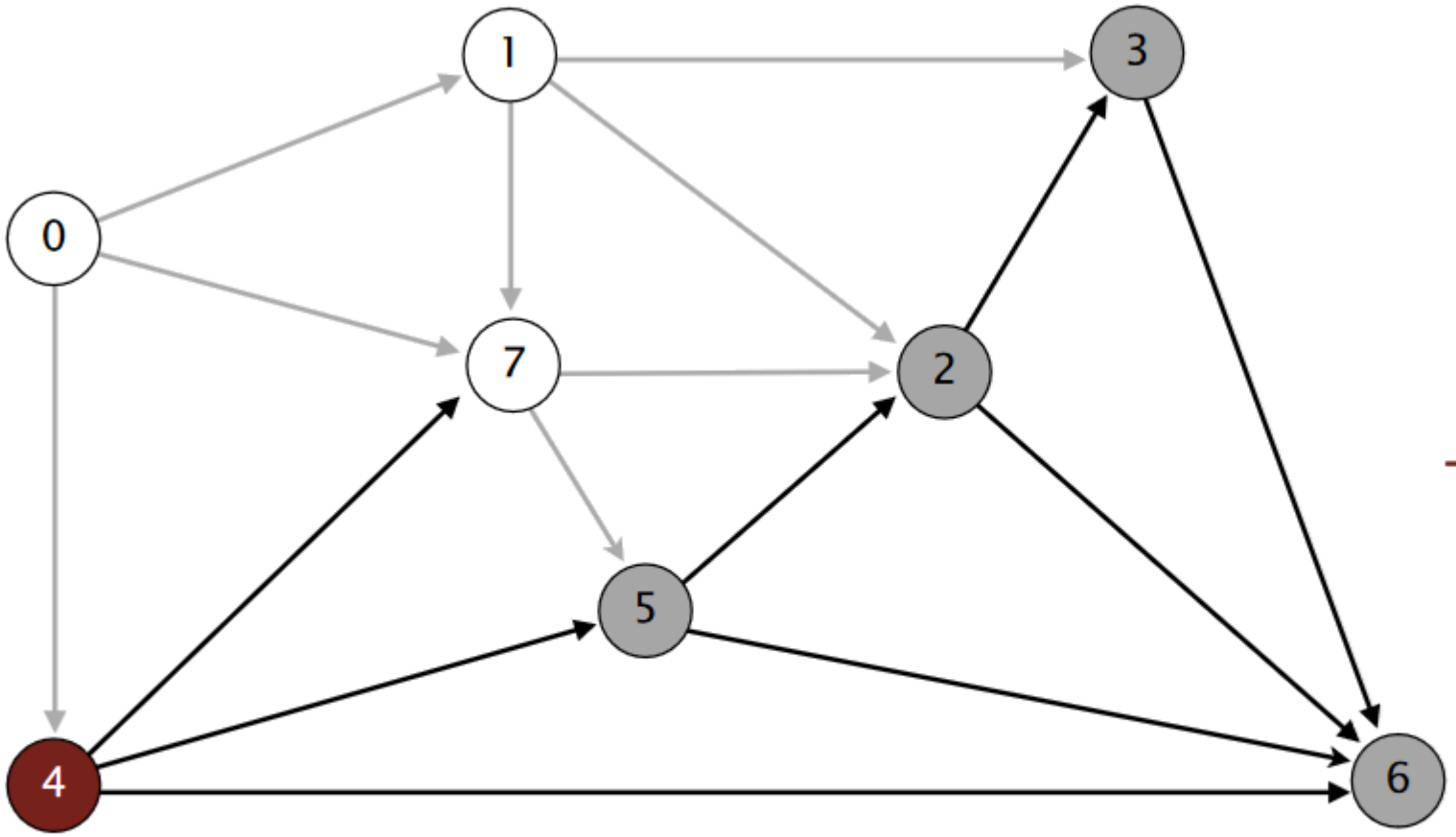


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	15.0	7→2
3	20.0	1→3
4	9.0	0→4
5	14.0	7→5
6		
→ 7	8.0	0→7

relax all edges pointing from 7

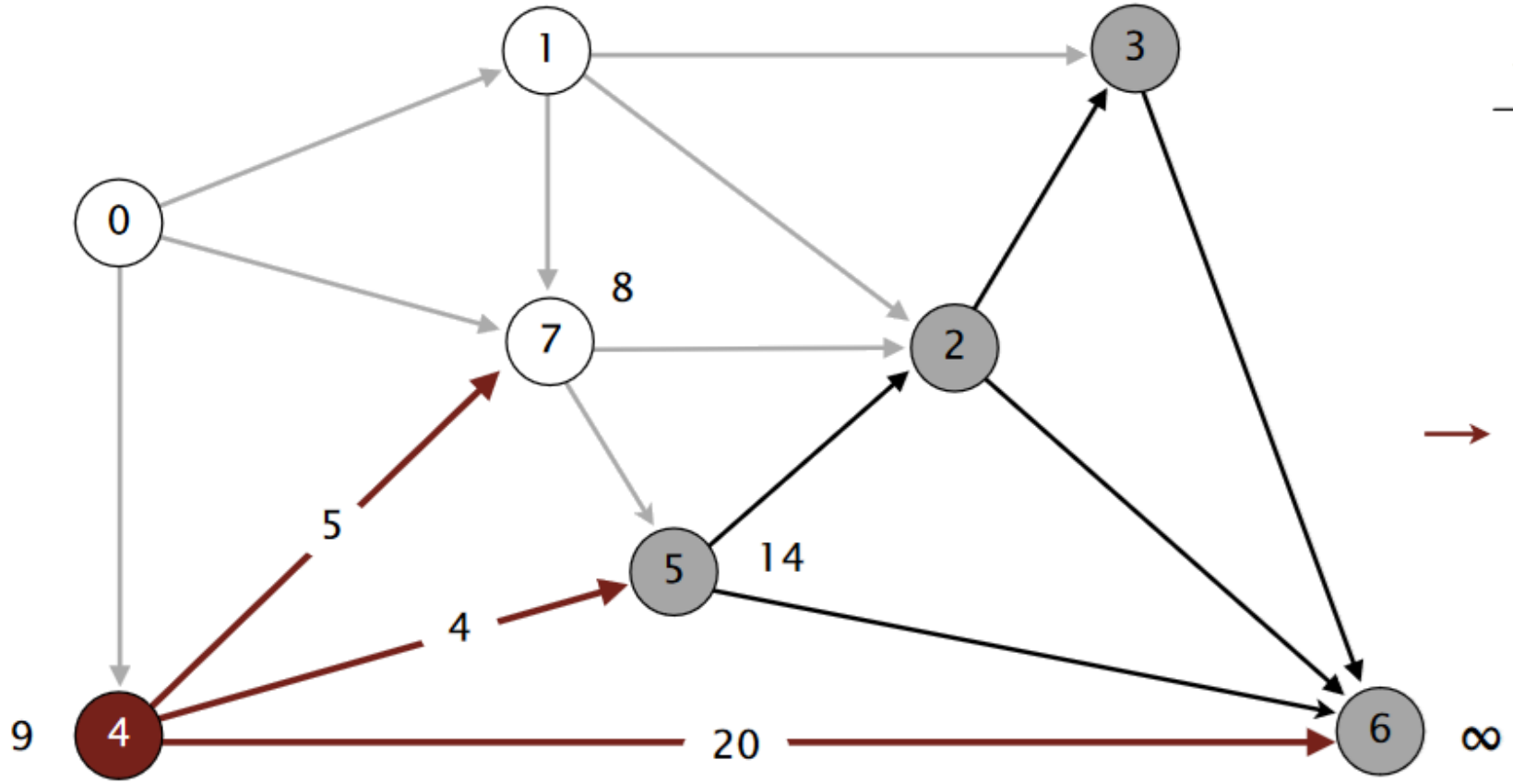


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	15.0	7→2
3	20.0	1→3
4	9.0	0→4
5	14.0	7→5
6		
7	8.0	0→7



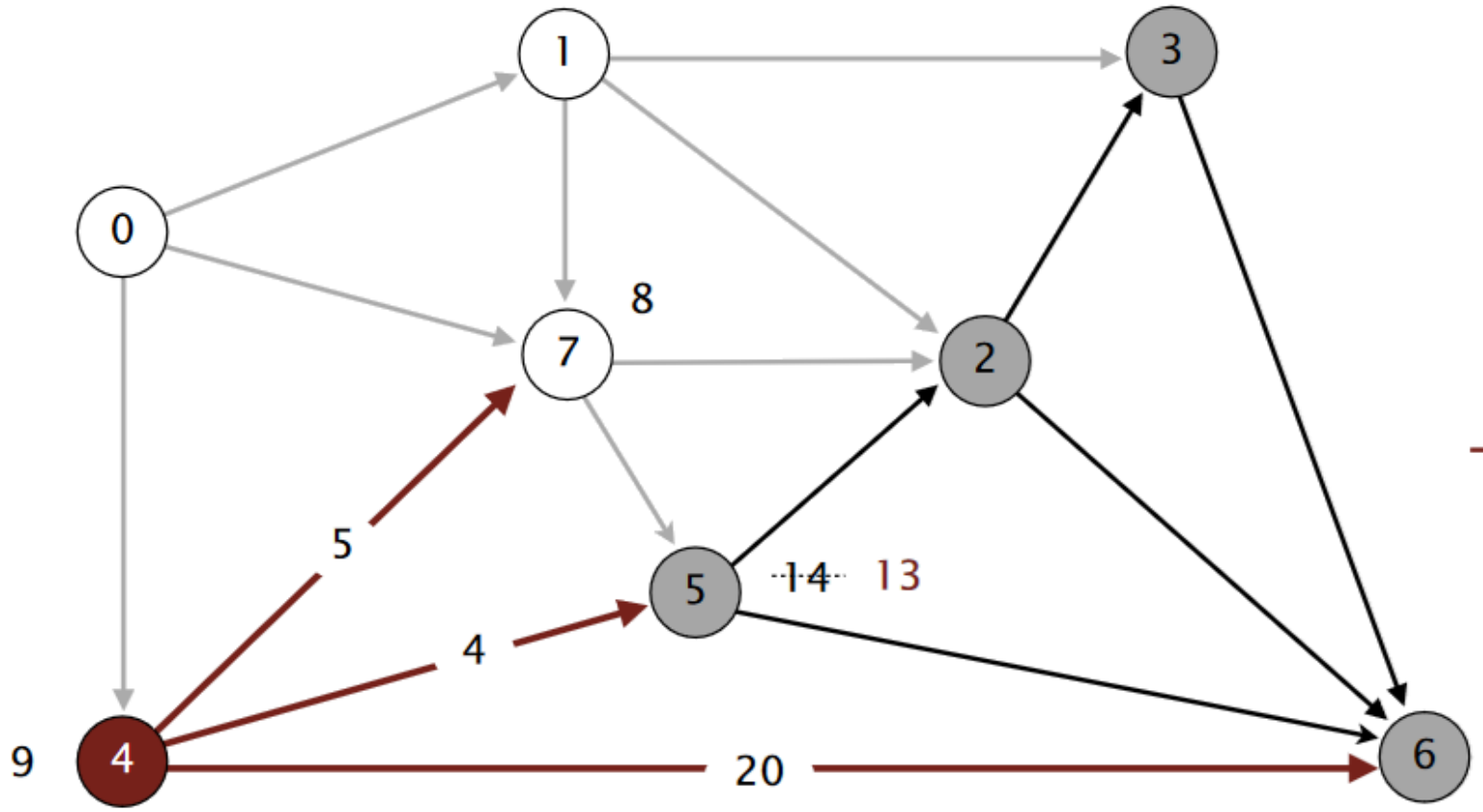
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	15.0	7→2
3	20.0	1→3
→ 4	9.0	0→4
5	14.0	7→5
6		
7	8.0	0→7

select vertex 4



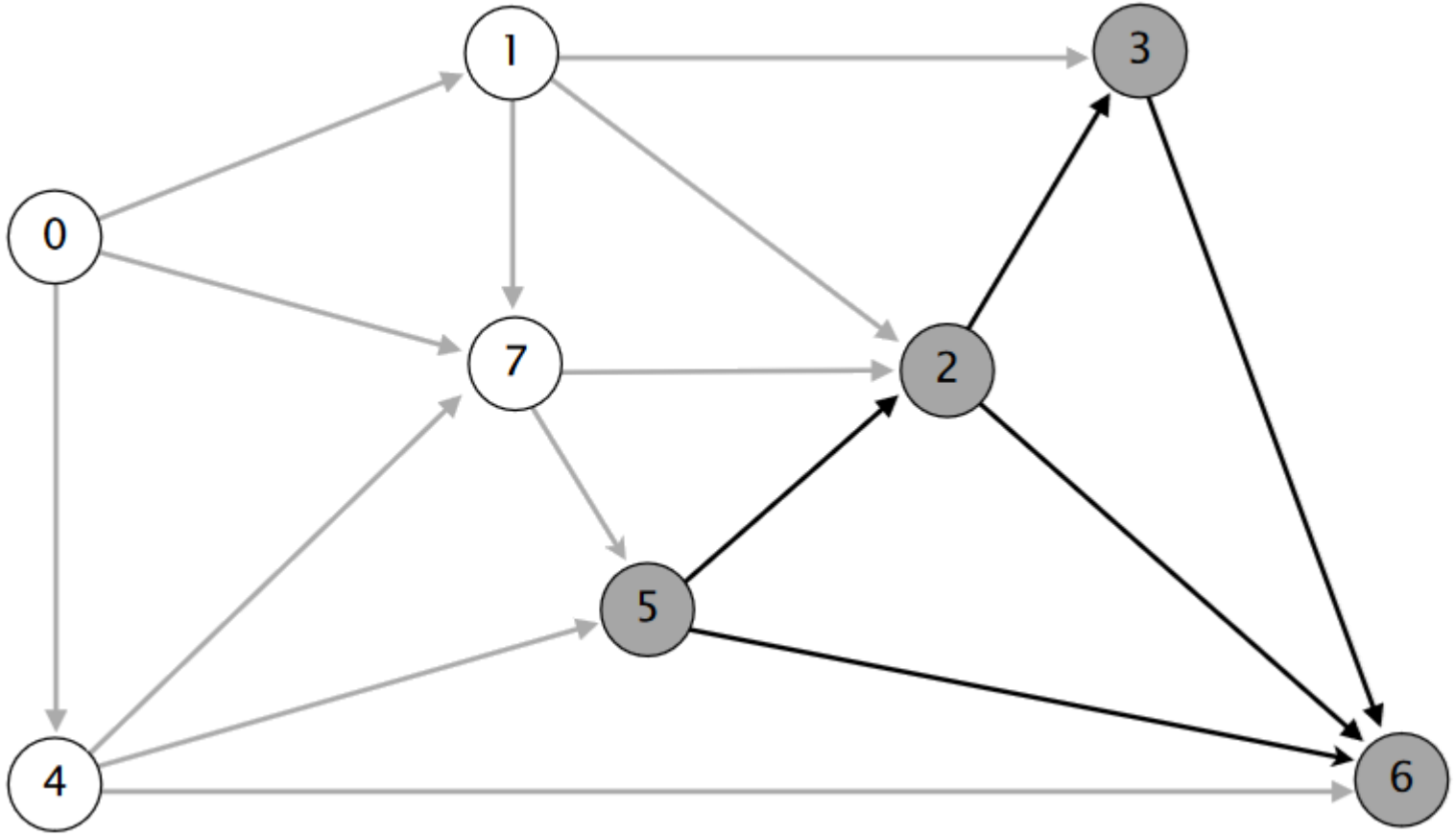
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	15.0	7→2
3	20.0	1→3
→ 4	9.0	0→4
5	14.0	7→5
6		
7	8.0	0→7

relax all edges pointing from 4

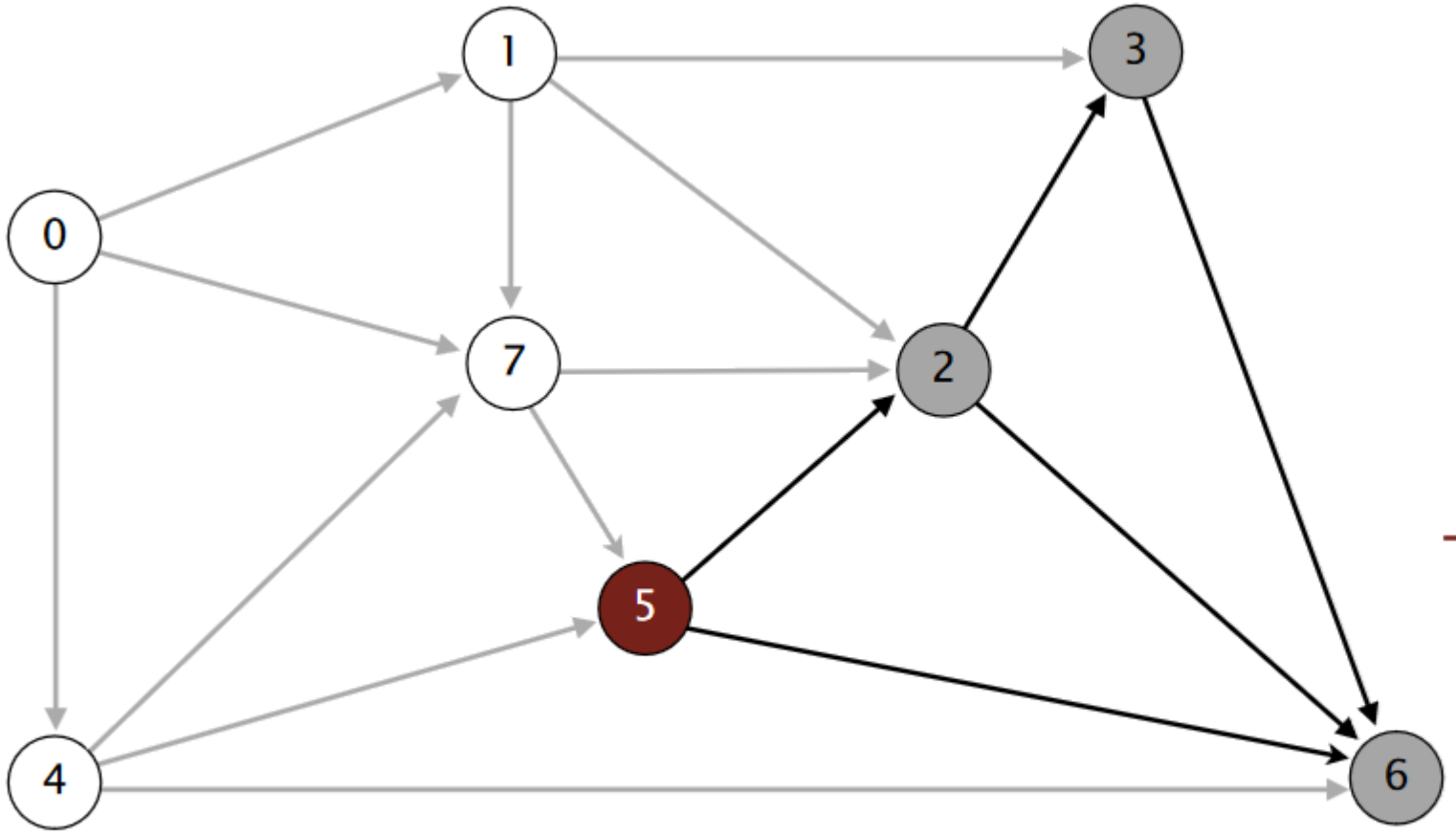


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	15.0	7→2
3	20.0	1→3
→ 4	9.0	0→4
5	13.0	4→5
6	29.0	4→6
7	8.0 ✔	0→7

relax all edges pointing from 4

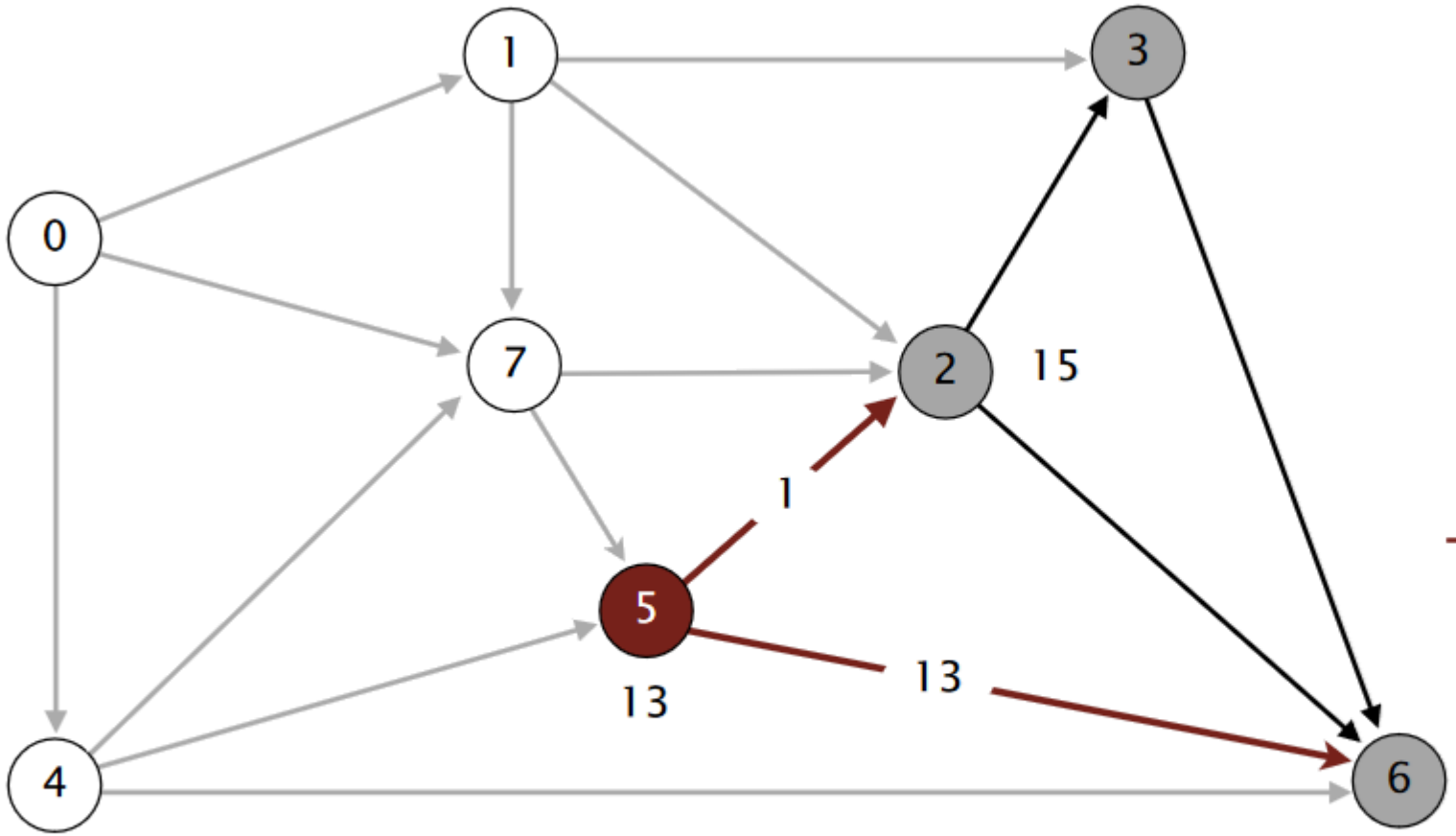


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	15.0	7→2
3	20.0	1→3
4	9.0	0→4
5	13.0	4→5
6	29.0	4→6
7	8.0	0→7



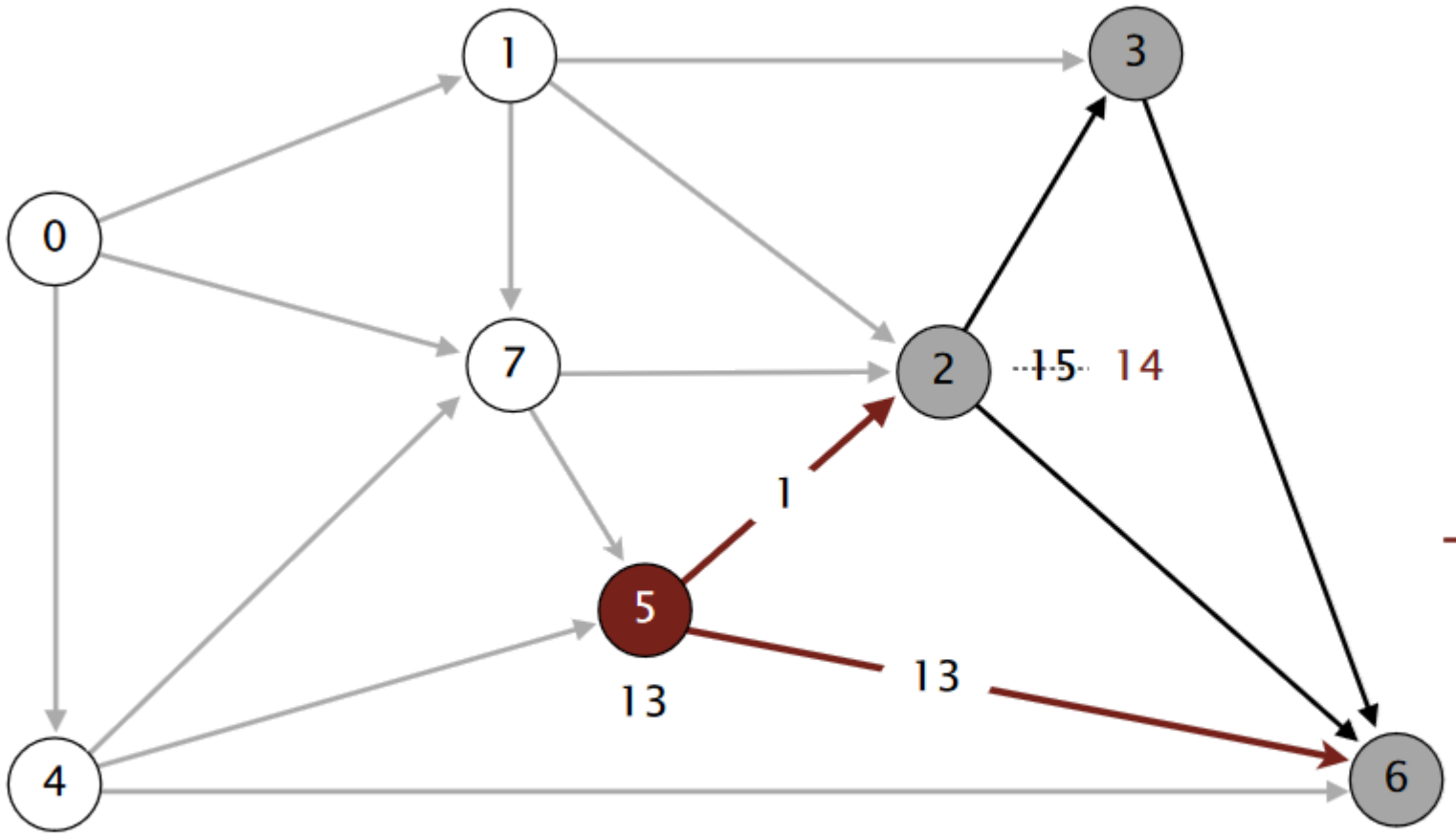
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	15.0	7→2
3	20.0	1→3
4	9.0	0→4
→ 5	13.0	4→5
6	29.0	4→6
7	8.0	0→7

select vertex 5



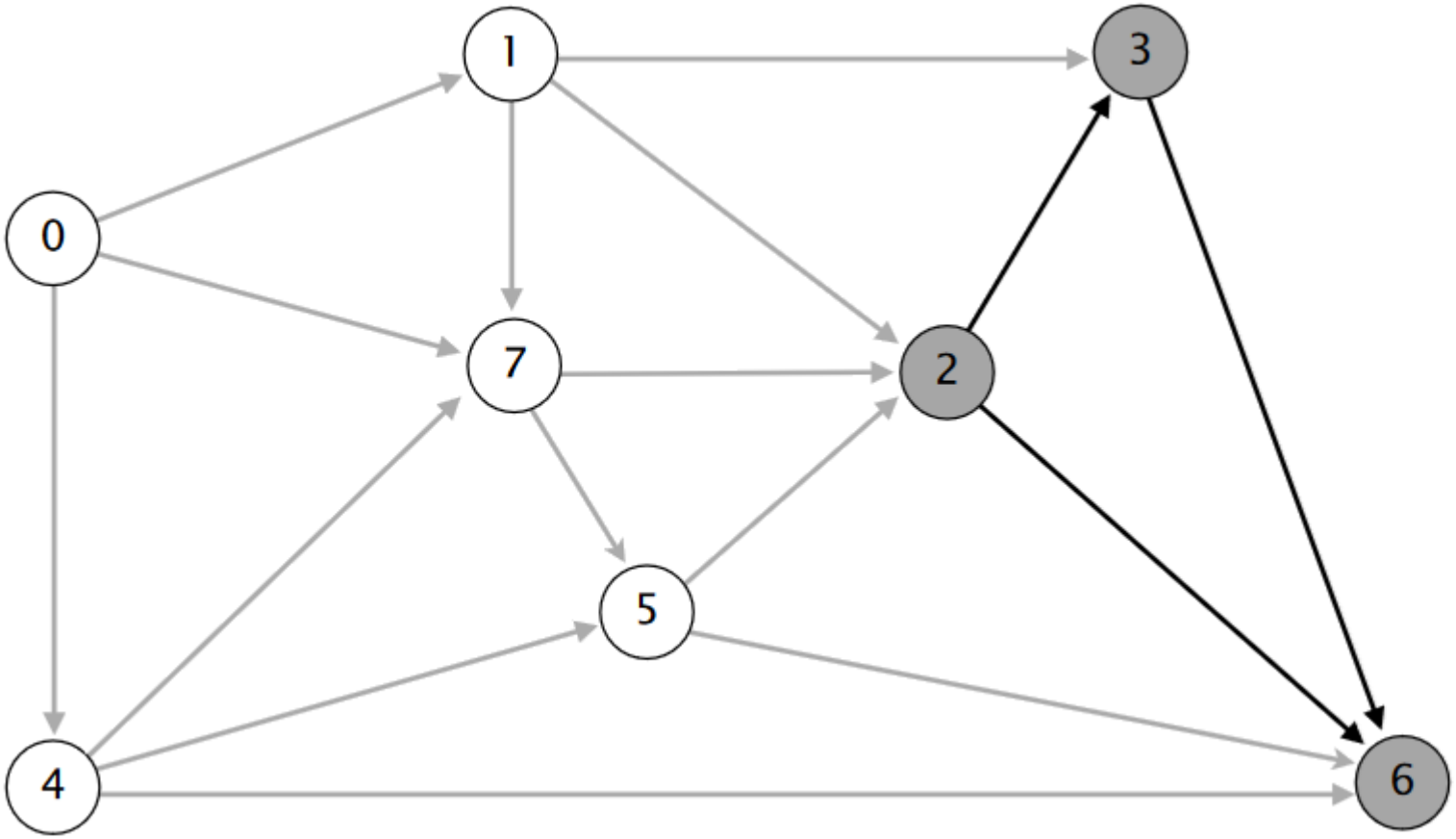
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	15.0	7→2
3	20.0	1→3
4	9.0	0→4
→ 5	13.0	4→5
6	29.0	4→6
7	8.0	0→7

relax all edges pointing from 5

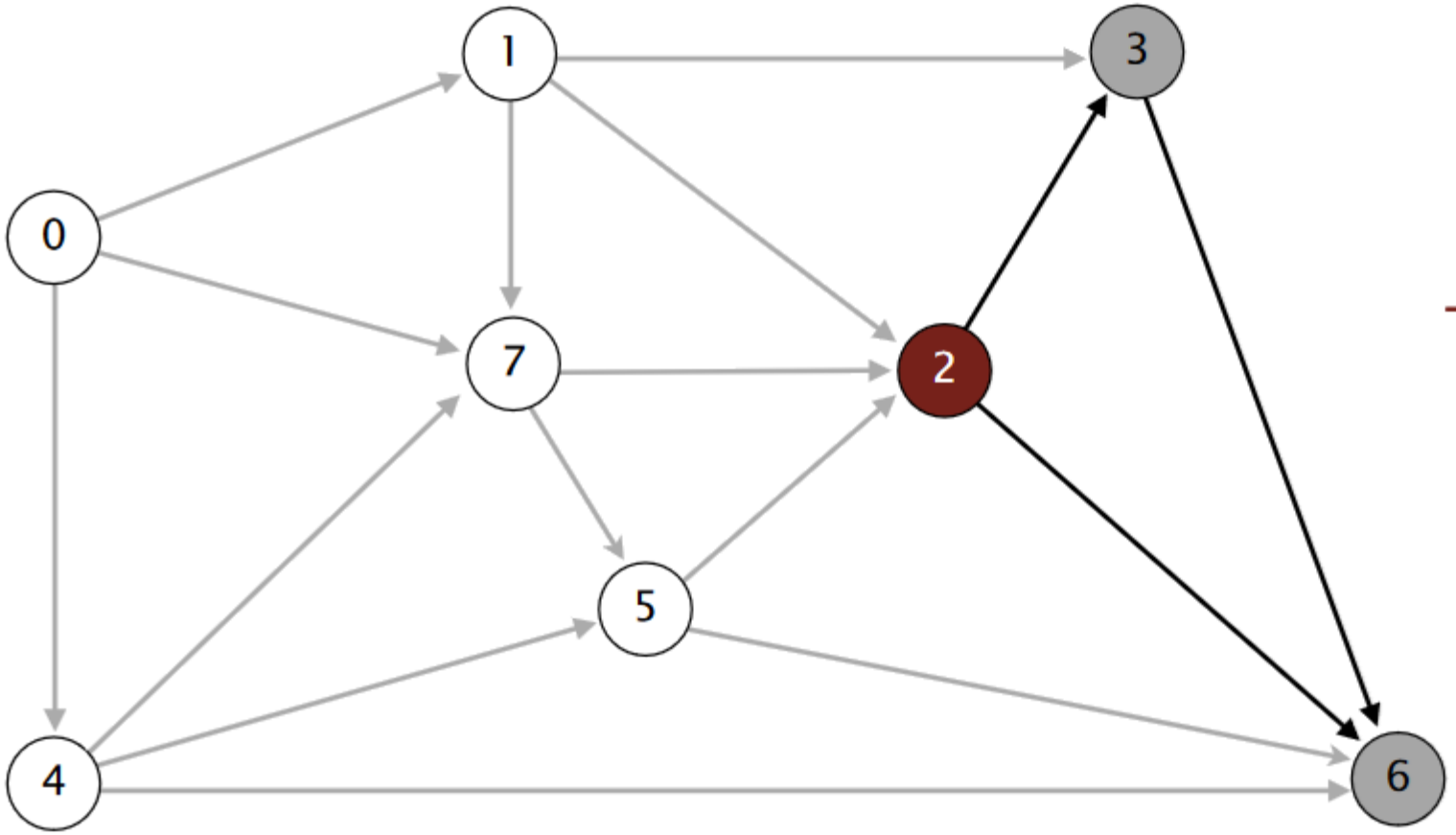


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
3	20.0	1→3
4	9.0	0→4
→ 5	13.0	4→5
6	26.0	5→6
7	8.0	0→7

relax all edges pointing from 5

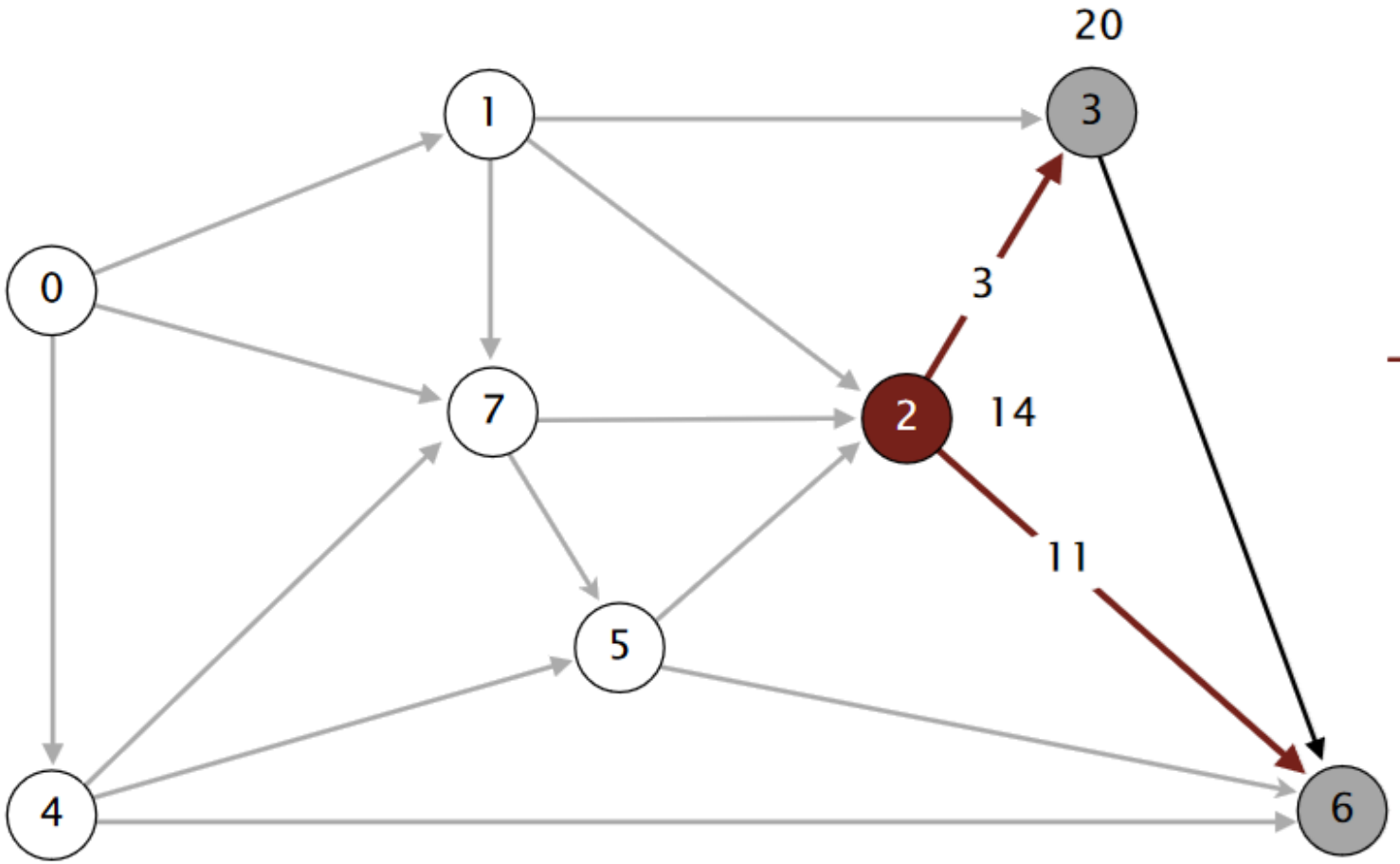


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
3	20.0	1→3
4	9.0	0→4
5	13.0	4→5
6	26.0	5→6
7	8.0	0→7



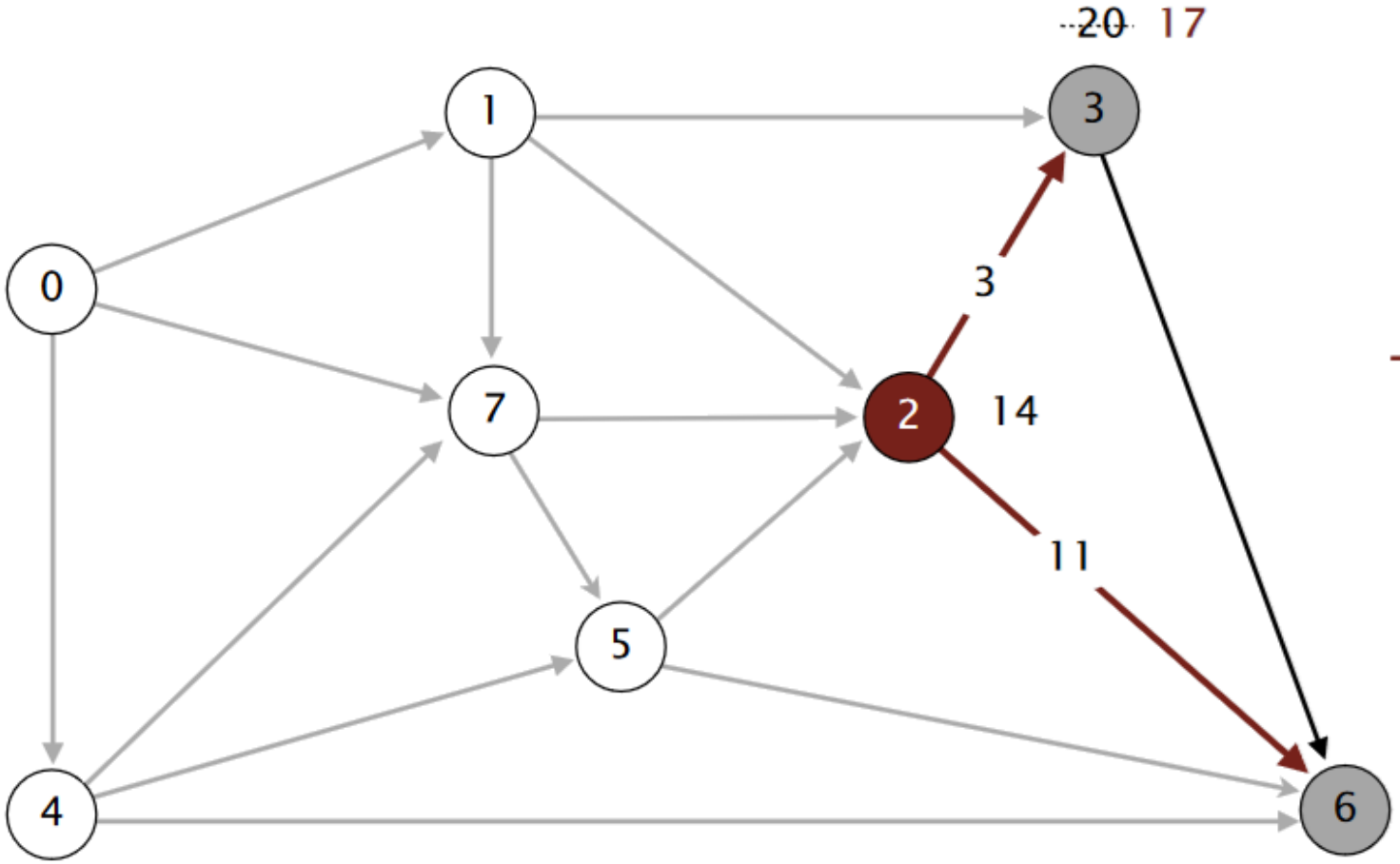
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
→ 2	14.0	5→2
3	20.0	1→3
4	9.0	0→4
5	13.0	4→5
6	26.0	5→6
7	8.0	0→7

select vertex 2



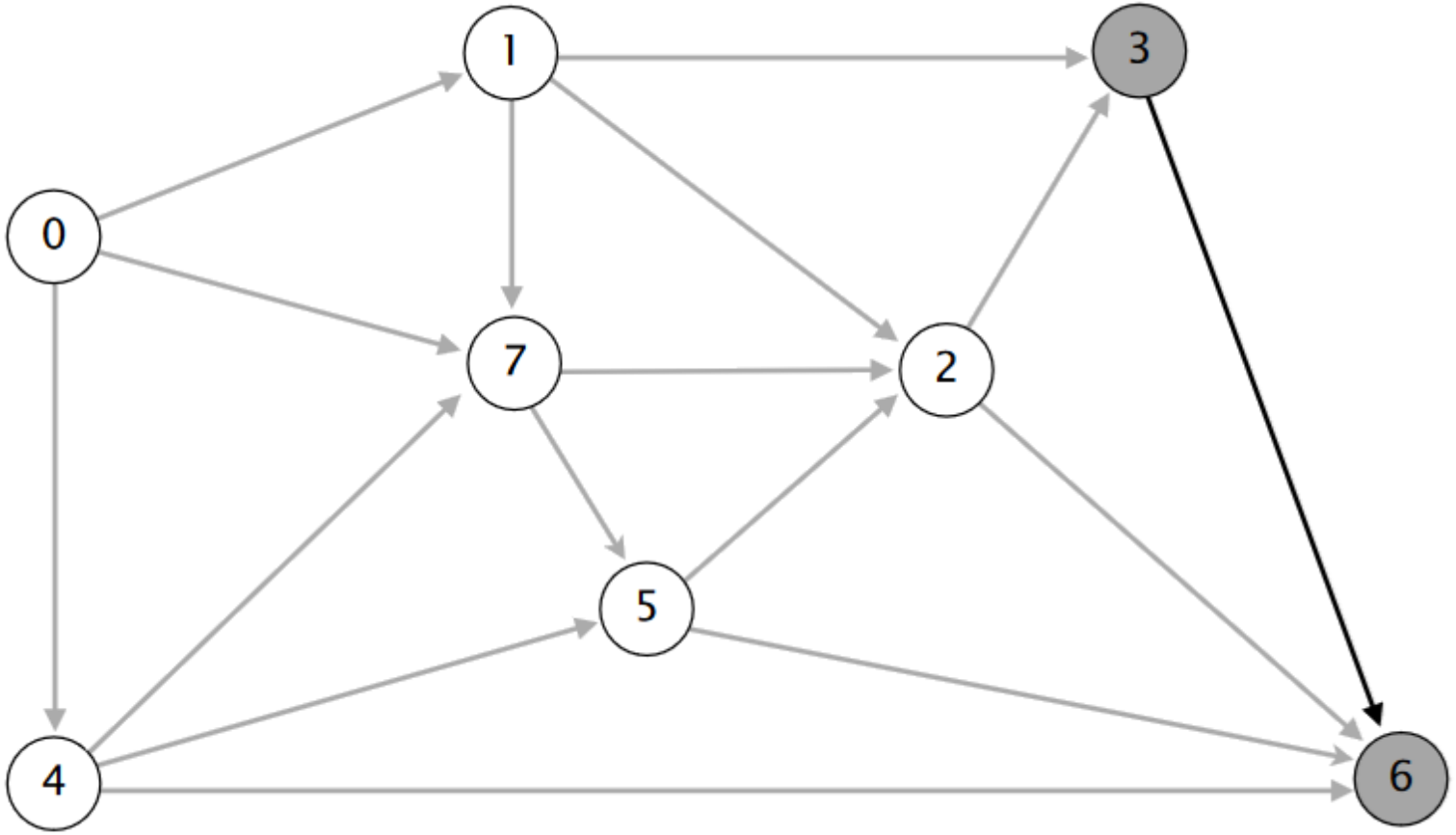
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
→ 2	14.0	5→2
3	20.0	1→3
4	9.0	0→4
5	13.0	4→5
6	26.0	5→6
7	8.0	0→7

relax all edges pointing from 2

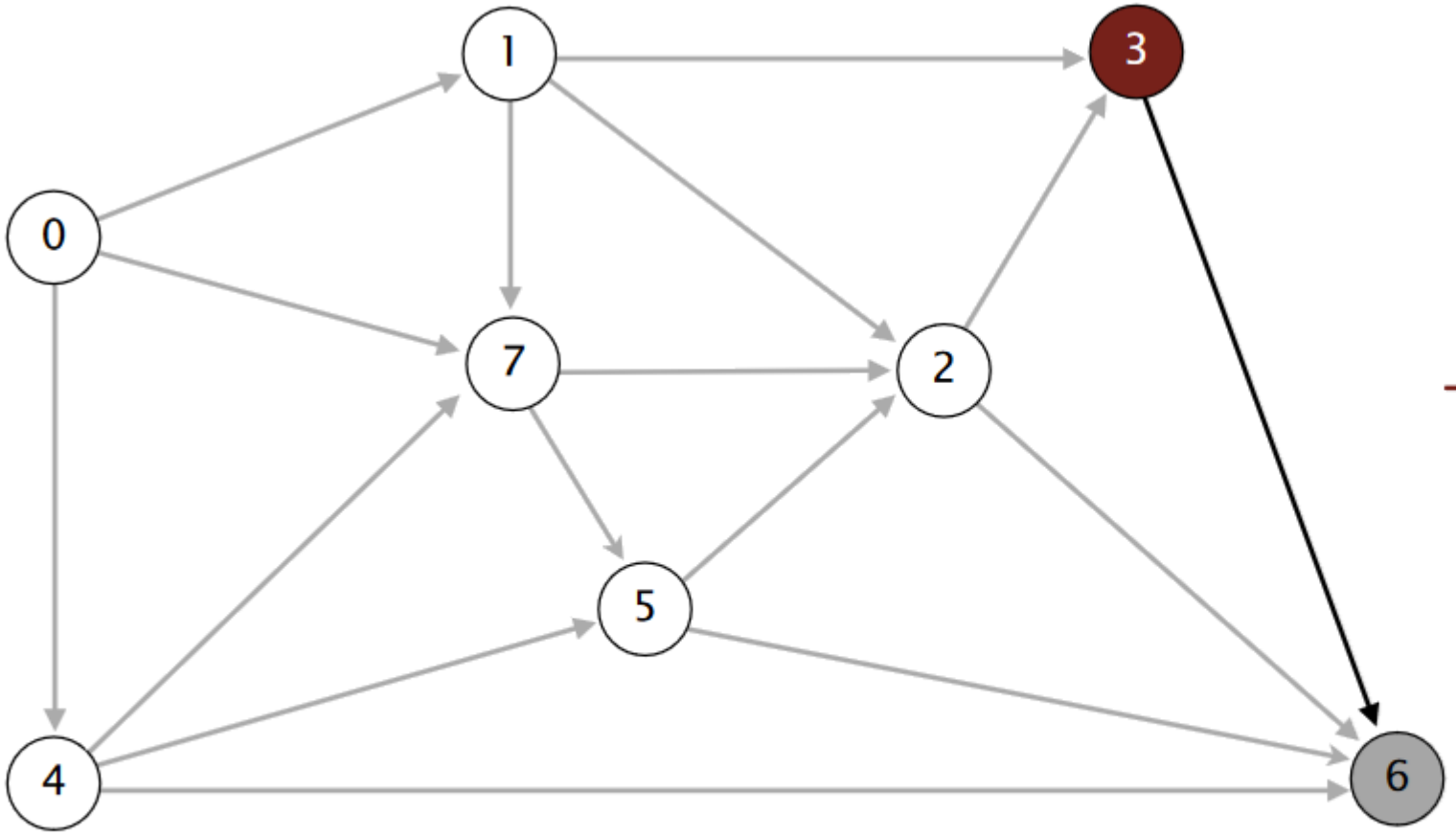


	v	distTo[]	edgeTo[]
	0	0.0	-
	1	5.0	0→1
→	2	14.0	5→2
	3	17.0	2→3
	4	9.0	0→4
	5	13.0	4→5
	6	25.0	2→6
	7	8.0	0→7

relax all edges pointing from 2

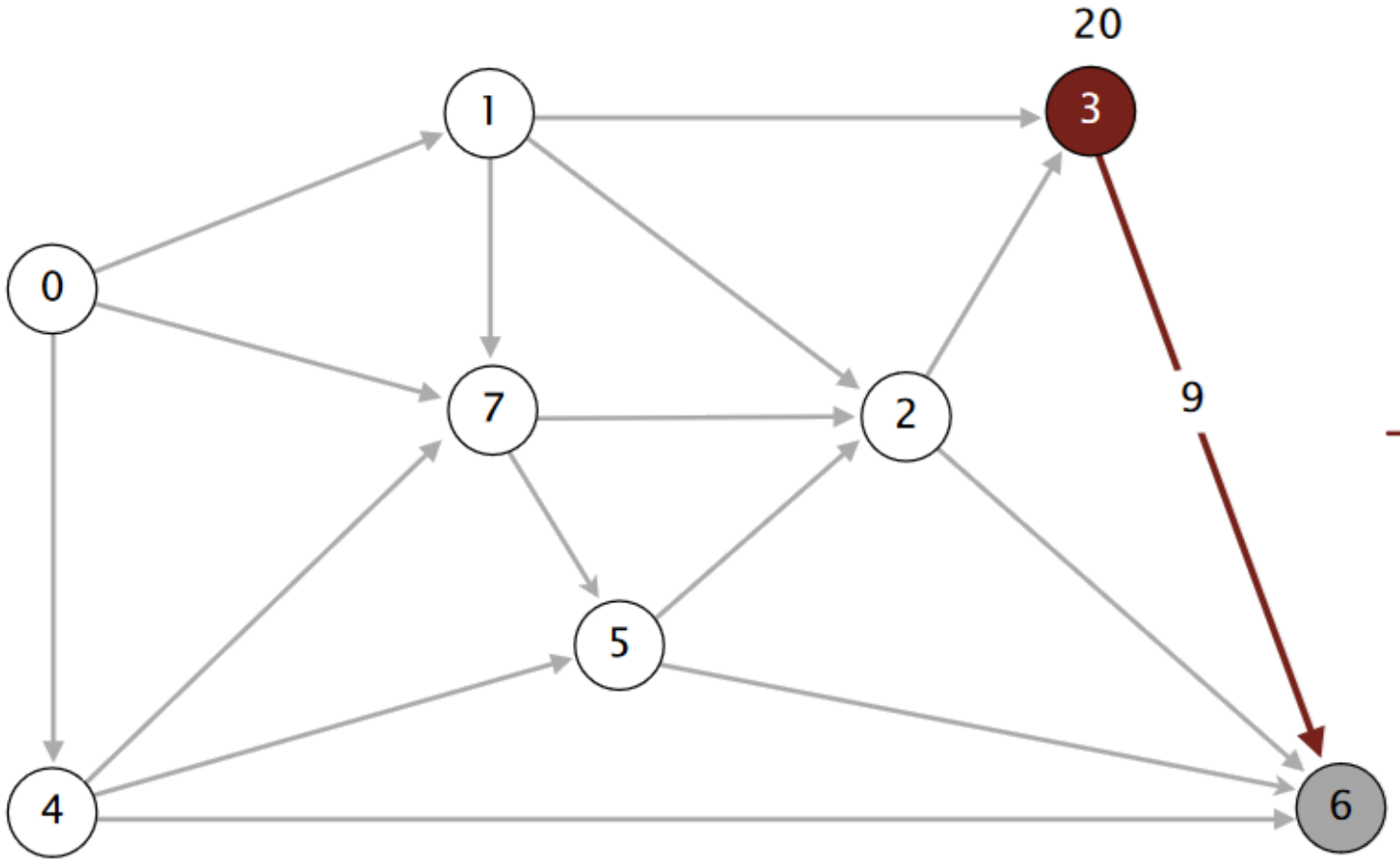


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
3	17.0	2→3
4	9.0	0→4
5	13.0	4→5
6	25.0	2→6
7	8.0	0→7



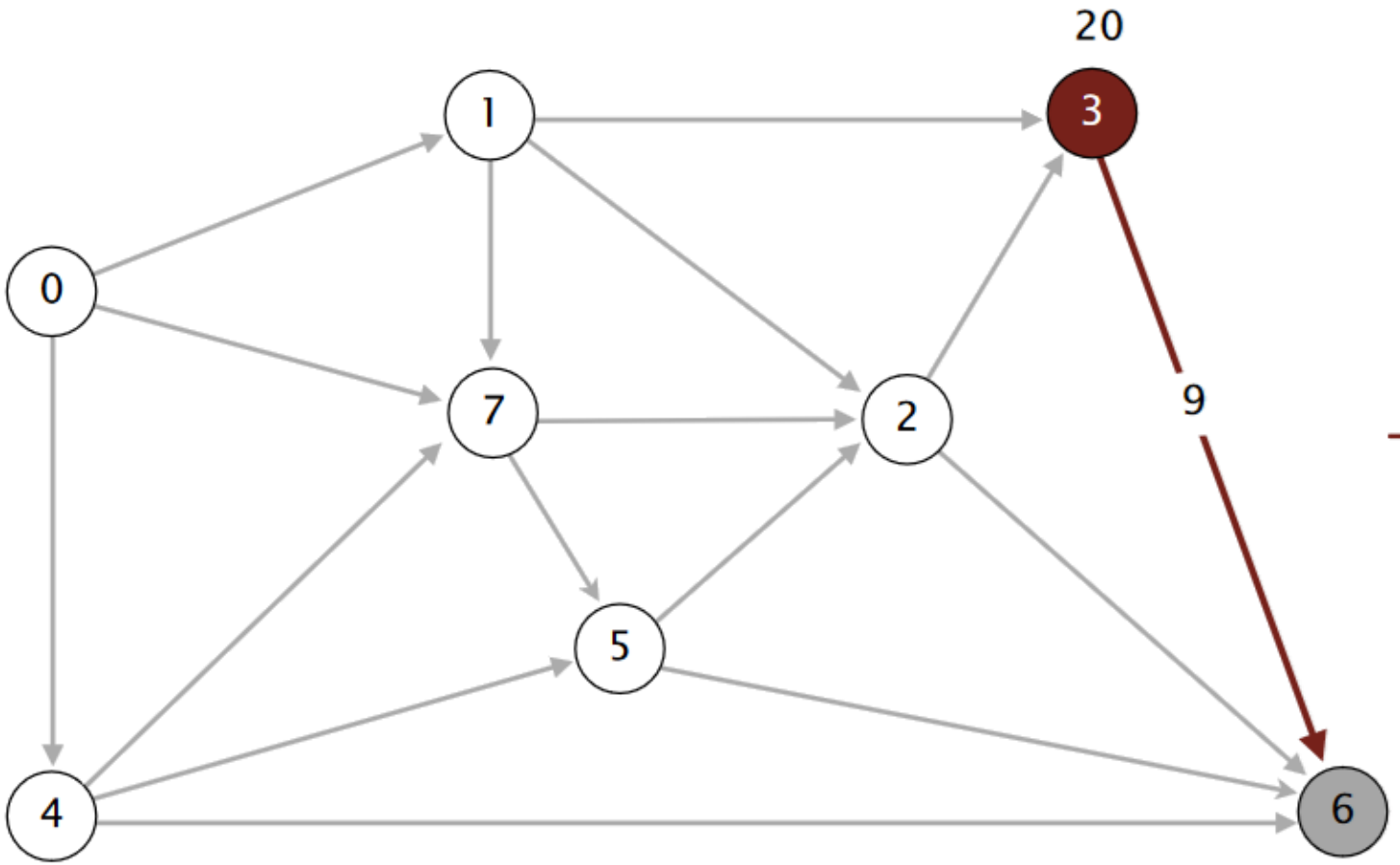
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
→ 3	17.0	2→3
4	9.0	0→4
5	13.0	4→5
6	25.0	2→6
7	8.0	0→7

select vertex 3



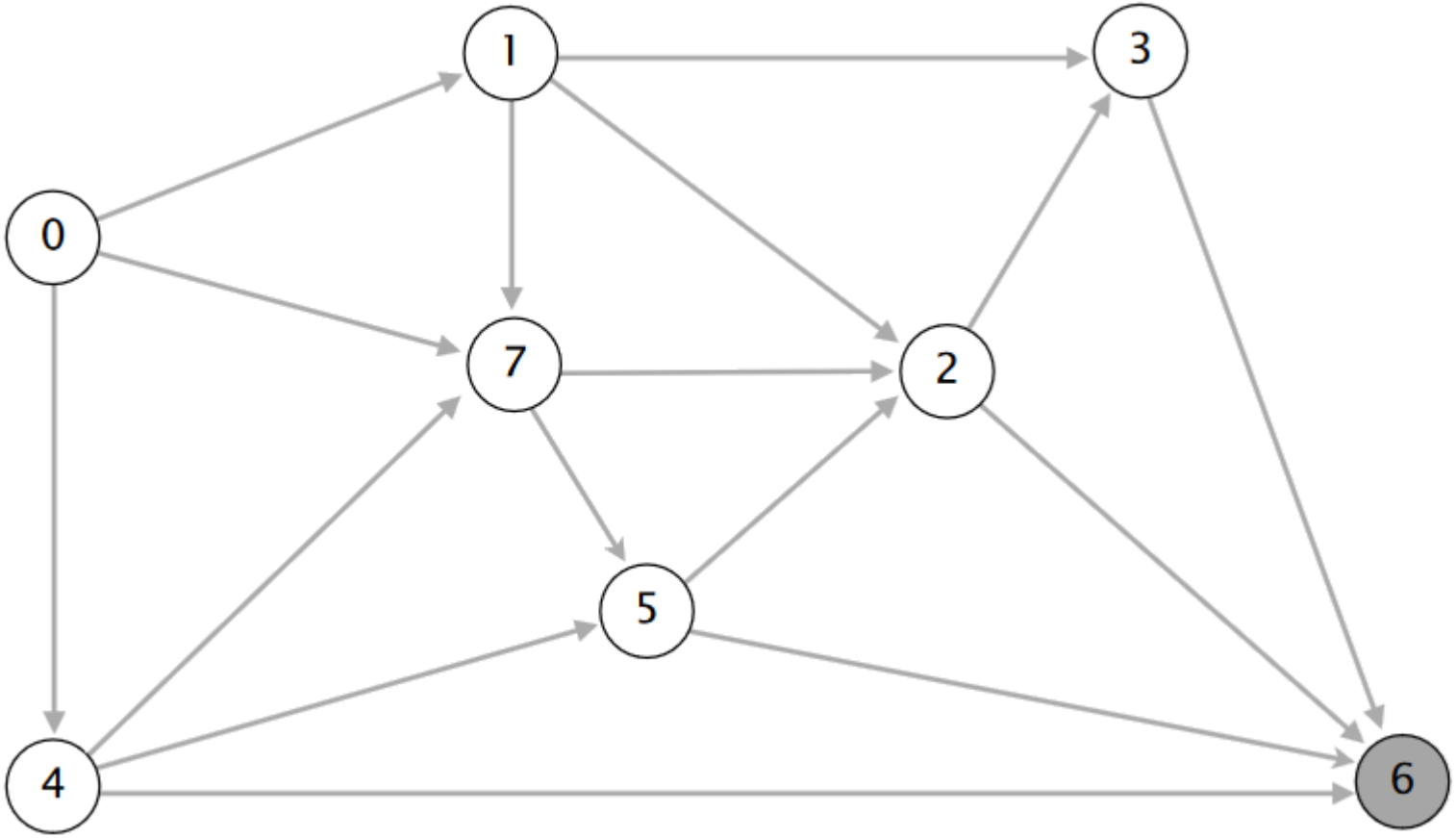
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
→ 3	17.0	2→3
4	9.0	0→4
5	13.0	4→5
6	25.0	2→6
7	8.0	0→7

relax all edges pointing from 3

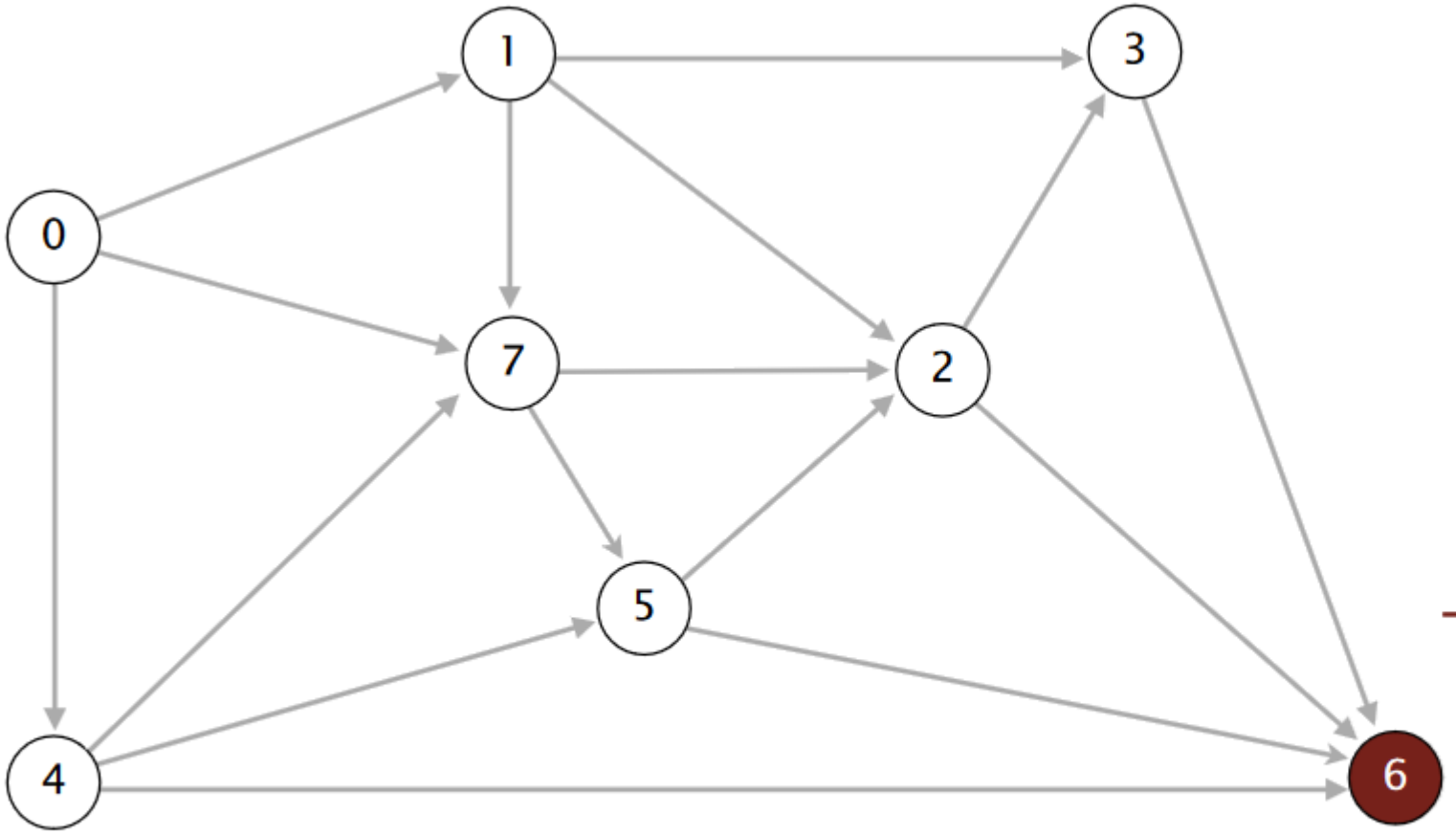


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
→ 3	17.0	2→3
4	9.0	0→4
5	13.0	4→5
6	25.0 ✔	2→6
7	8.0	0→7

relax all edges pointing from 3

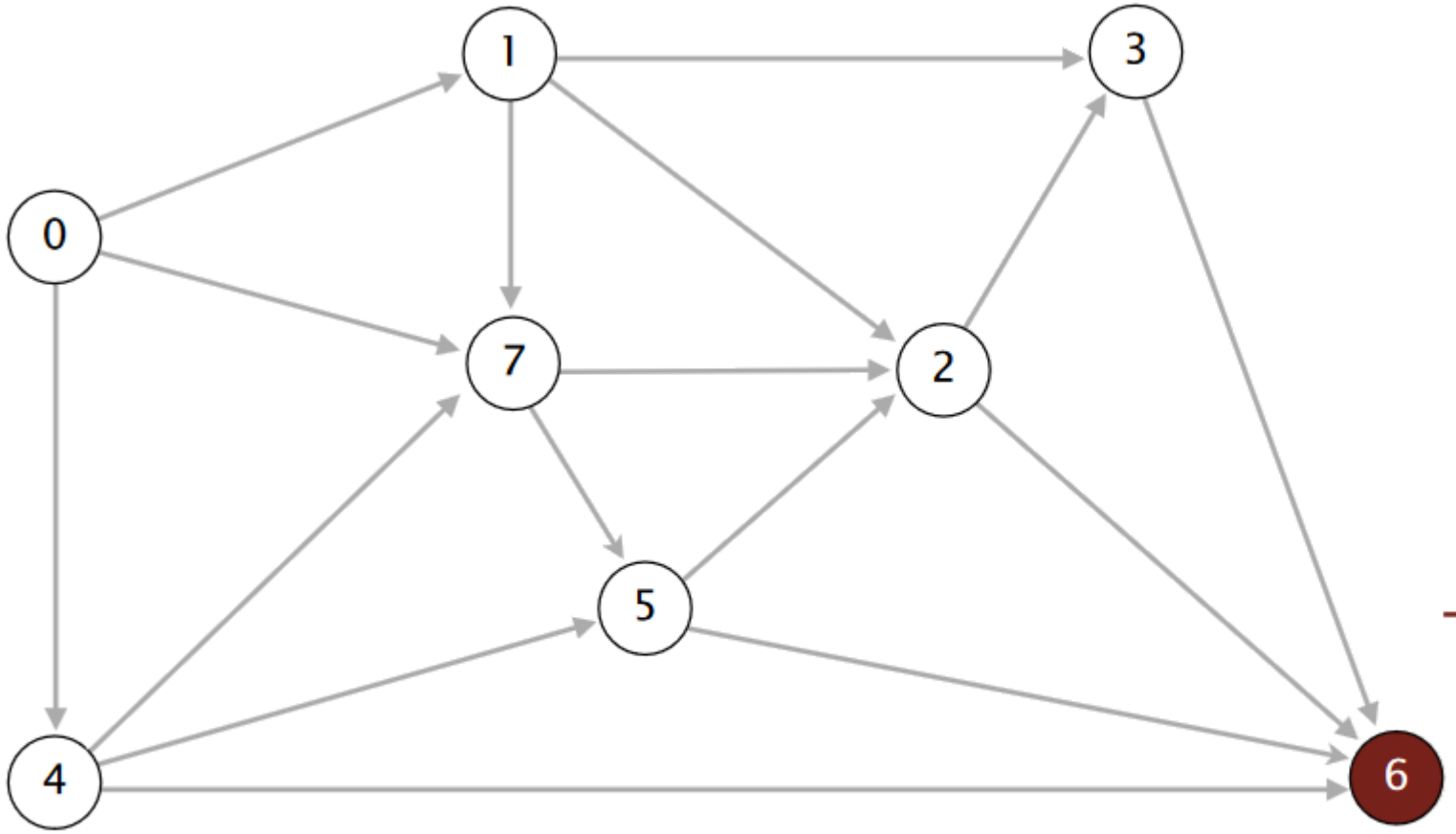


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
3	17.0	2→3
4	9.0	0→4
5	13.0	4→5
6	25.0	2→6
7	8.0	0→7



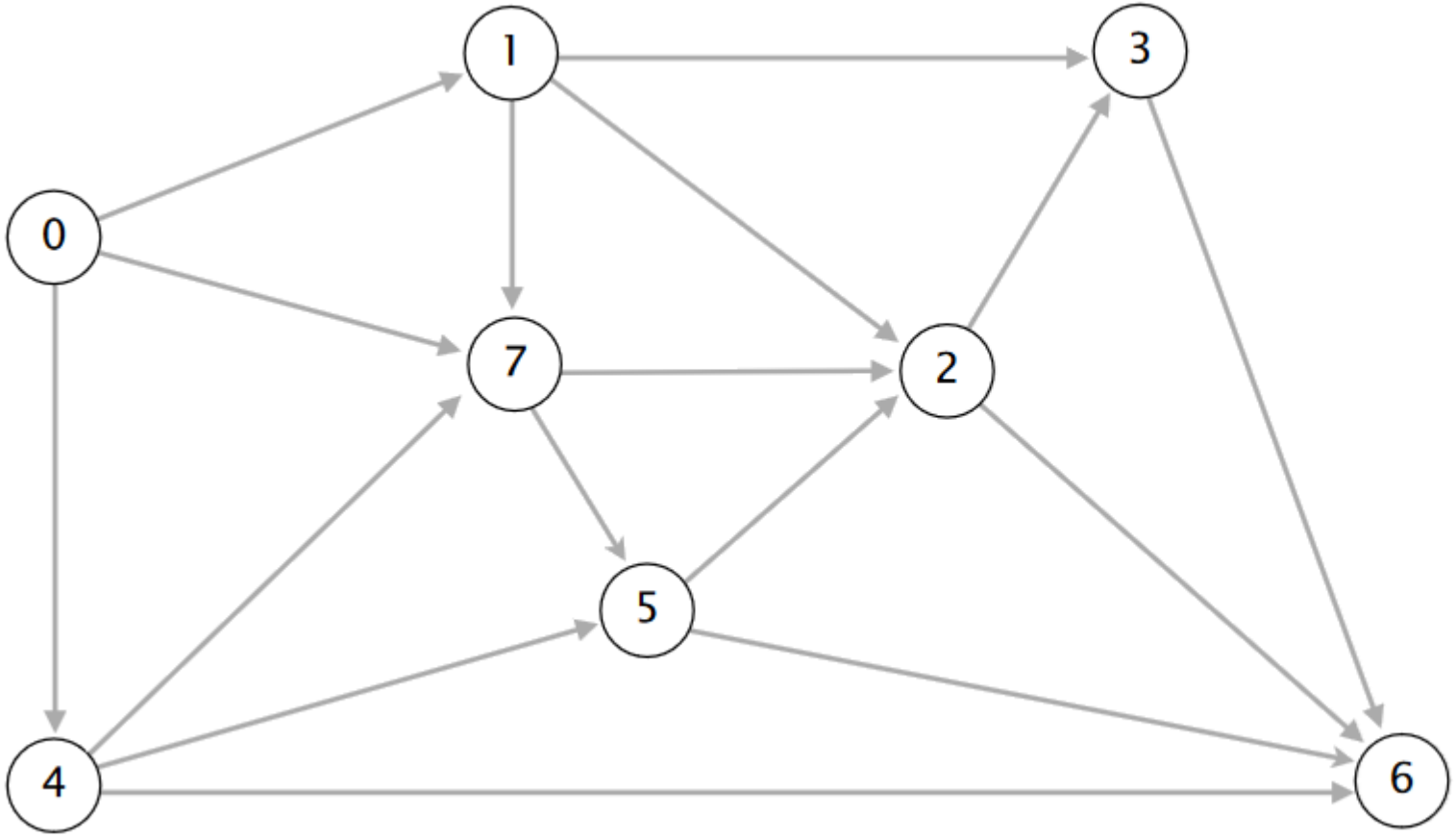
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
3	17.0	2→3
4	9.0	0→4
5	13.0	4→5
→ 6	25.0	2→6
7	8.0	0→7

select vertex 6

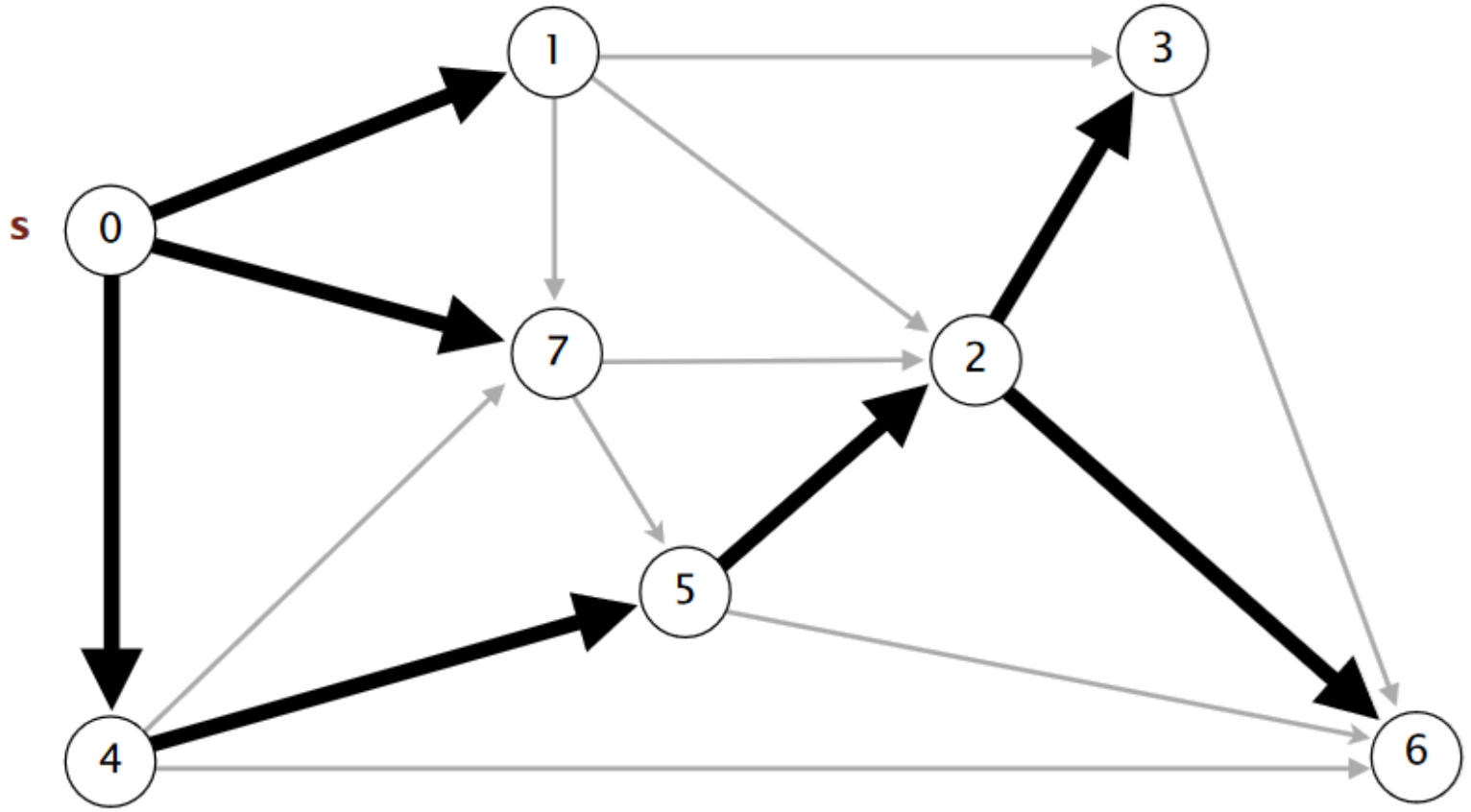


v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
3	17.0	2→3
4	9.0	0→4
5	13.0	4→5
→ 6	25.0	2→6
7	8.0	0→7

relax all edges pointing from 6



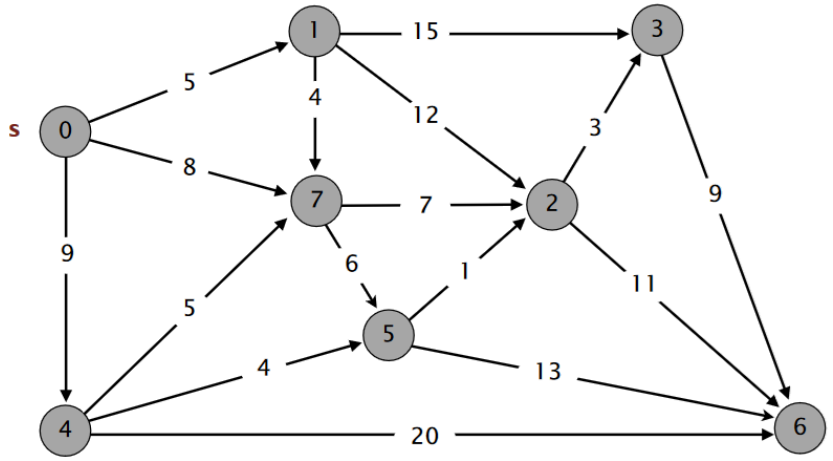
v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
3	17.0	2→3
4	9.0	0→4
5	13.0	4→5
6	25.0	2→6
7	8.0	0→7



v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
3	17.0	2→3
4	9.0	0→4
5	13.0	4→5
6	25.0	2→6
7	8.0	0→7

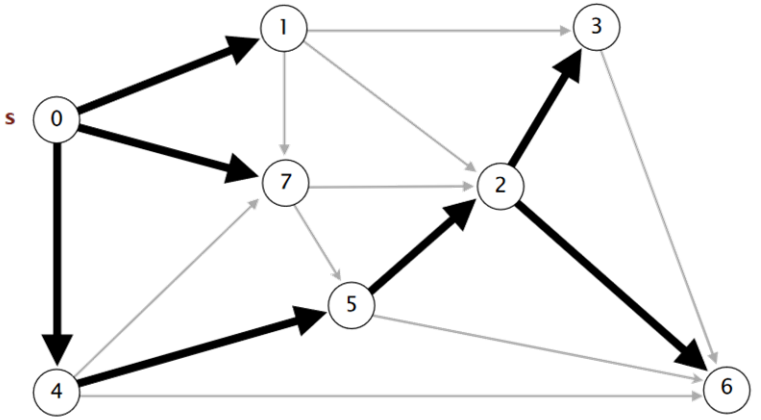
shortest-paths tree from vertex s

Solución con Dijkstra (con MinPQ)



an edge-weighted digraph

0→1	5.0
0→4	9.0
0→7	8.0
1→2	12.0
1→3	15.0
1→7	4.0
2→3	3.0
2→6	11.0
3→6	9.0
4→5	4.0
4→6	20.0
4→7	5.0
5→2	1.0
5→6	13.0
7→5	6.0
7→2	7.0



shortest-paths tree from vertex s

v	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0→1
2	14.0	5→2
3	17.0	2→3
4	9.0	0→4
5	13.0	4→5
6	25.0	2→6
7	8.0	0→7