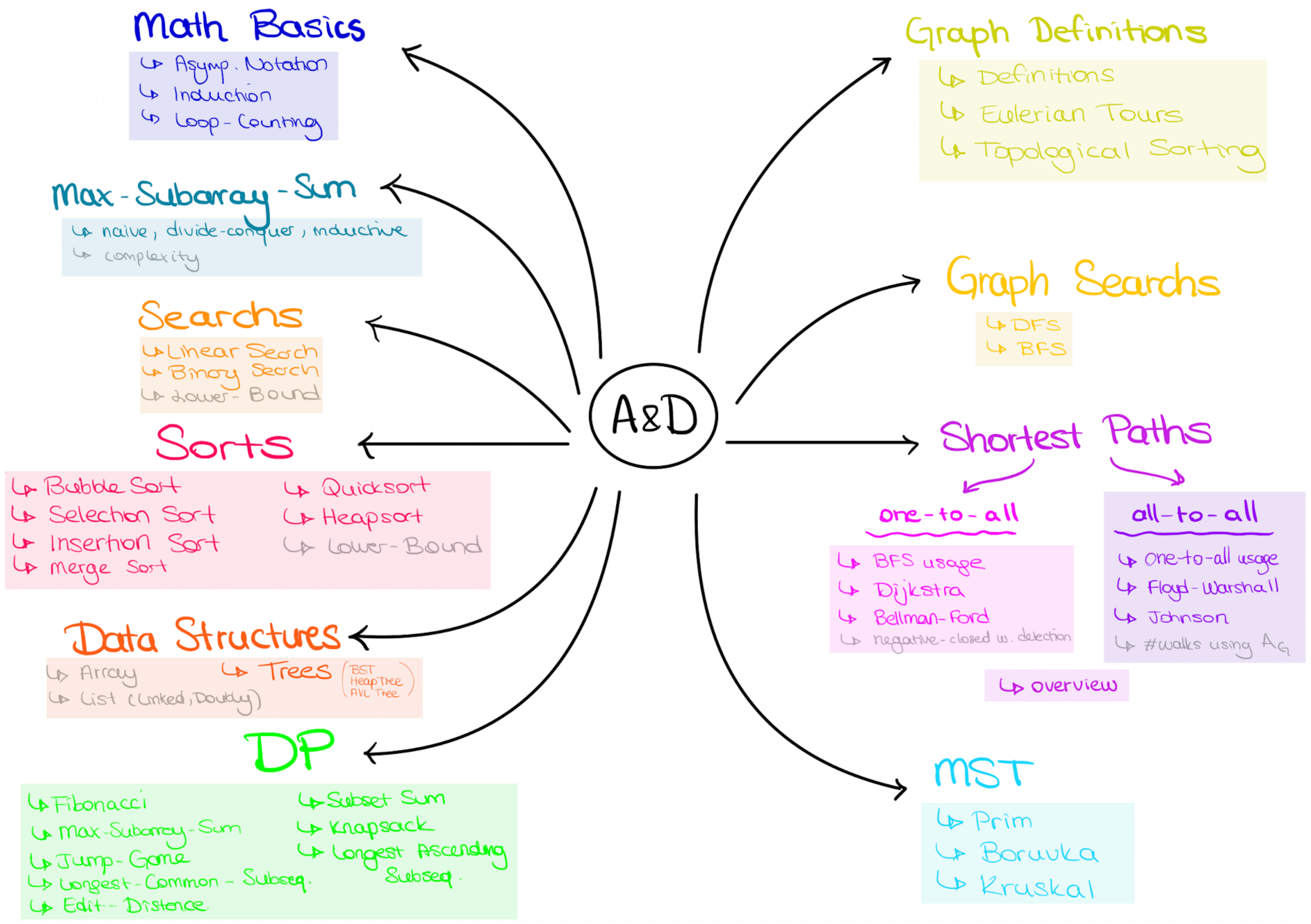
A&D

Exercise Session 11

Nil Ozer

A&D Overview


A&D
Math Basics
Asymp. Notation
Induction
Loop-Counting
Max-Subarray-Sum
naive, divide-conquer, inductive
complexity
Searchs
Linear Search
Binary Search
Lower-Bound
Sorts
Bubble Sort
Selection Sort
Insertion Sort
Merge Sort
Quicksort
Heapsort
Lower-Bound
Data Structures
Array
List (Linked, Doubly)
Trees
BST
Heap Tree
AVL Tree
DP
Fibonacci
Max-Subarray-Sum
Jump-Game
Longest-Common-Subseq.
Edit-Distance
Subset Sum
Knapsack
Longest Ascending Subseq.
Graph Definitions
Definitions
Eulerian Tours
Topological Sorting
Graph Searchs
DFS
BFS
Shortest Paths
one-to-all
BFS usage
Dijkstra
Bellman-Ford
negative-closed w. detection
all-to-all
one-to-all usage
Floyd-Warshall
Johnson
#walks using A_G
overview
MST
Prim
Boruvka
Kruskal

Outline

- Quiz
- Shortest Paths - one to all
- Code Expert Graph Sets

Quiz

Peergrading and rest

- Exercise Sheet 10 peergrading
  - 10.4 this week
  - Emails will be sent

- If urgent feedback is needed, send me an email !

Shortest Paths

→ Shortest Paths

one-to-all

- ↳ BFS usage
- ↳ Dijkstra
- ↳ Bellman-Ford
- ↳ negative-closed w. detection

all-to-all

- ↳ one-to-all usage
- ↳ Floyd-Warshall
- ↳ Johnson
- ↳ #walks using A_G

↳ overview

Shortest Paths

one - to - all

G (directed/undirected)	Algorithm	Runtime
unweighted , all edges with the same positive weight	BFS usage	O (|V| + |E|)
weighted , nonnegative edge weights $c(e) \geq 0$	Dijsktra	O ((|V| + |E|) * log n)
weighted, positive and (possibly) negative edge weights $c(e) \in \mathbb{R}$	Belmann-Ford	O (|V| * |E|)
G has no cycles	topological sorting + DP	O (|V| + |E|)

Shortest Paths

BFS usage - with distances

Runtime : O (|V| + |E|)

Algorithm 5 BFS(s)

$Q \leftarrow \{s\}$

enter$[s] \leftarrow 0$; $T \leftarrow 1$ distance[s] = 0 ;

while $Q \neq \varnothing$ **do**

  $u \leftarrow$ dequeue(Q)

  leave$[u] \leftarrow T$; $T \leftarrow T + 1$

  for $(u, v) \in E$, enter$[v]$ nicht zugewiesen **do**

    enqueue(Q, v)

    enter$[v] \leftarrow T$; $T \leftarrow T + 1$ distance[v] <- distance[u] + 1 ;

Q is a FIFO queue

Shortest Paths

Dijkstra's Algorithm

Runtime : O ((|V| + |E|) * log n)

weighted , nonnegative edge weights

$c(e) \geq 0$

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
    $v^* \leftarrow$ extract-min(H)
    $S \leftarrow S \cup \{v^*\}$
    for $(v^*, v) \in E,\; v \notin S$ **do**
        $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
        decrease-key($H, v, d[v]$)

Shortest Paths

Dijkstra's Algorithm

Runtime : O ((|V| + |E|) * log n)

weighted , positive edge weights

c(e) ≥ 0

Algorithm 6 Dijkstra(s)

$\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
    $v^* \leftarrow$ extract-min(H)
    $S \leftarrow S \cup \{v^*\}$
    for $(v^*, v) \in E, \; v \notin S$ **do**
        $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
        decrease-key($H, v, \mathrm{d}[v]$)

d[] : distance array

S : visited set

H : min-heap

Shortest Paths

Dijkstra's Algorithm

Runtime : O ((|V| + |E|) * log n)

weighted , positive edge weights

$c(e) \geq 0$

d[] : distance array S : visited set

H : min-heap

Algorithm 6 Dijkstra(s)

$\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$

$S \leftarrow \varnothing$

$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)

while $S \neq V$ **do**

  $v^* \leftarrow$ extract-min(H)

  $S \leftarrow S \cup \{v^*\}$

  for $(v^*, v) \in E,\; v \notin S$ **do**

    $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$

    decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

```
d[s] ← 0;   d[v] ← ∞  ∀v ∈ V \ {s}
S ← ∅
H ← make-heap(V); decrease-key(H, s, 0)
while S ≠ V do
    v* ← extract-min(H)
    S ← S ∪ {v*}
    for (v*, v) ∈ E, v ∉ S do
        d[v] ← min{d[v], d[v*] + c(v*, v)}
        decrease-key(H, v, d[v])
```

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

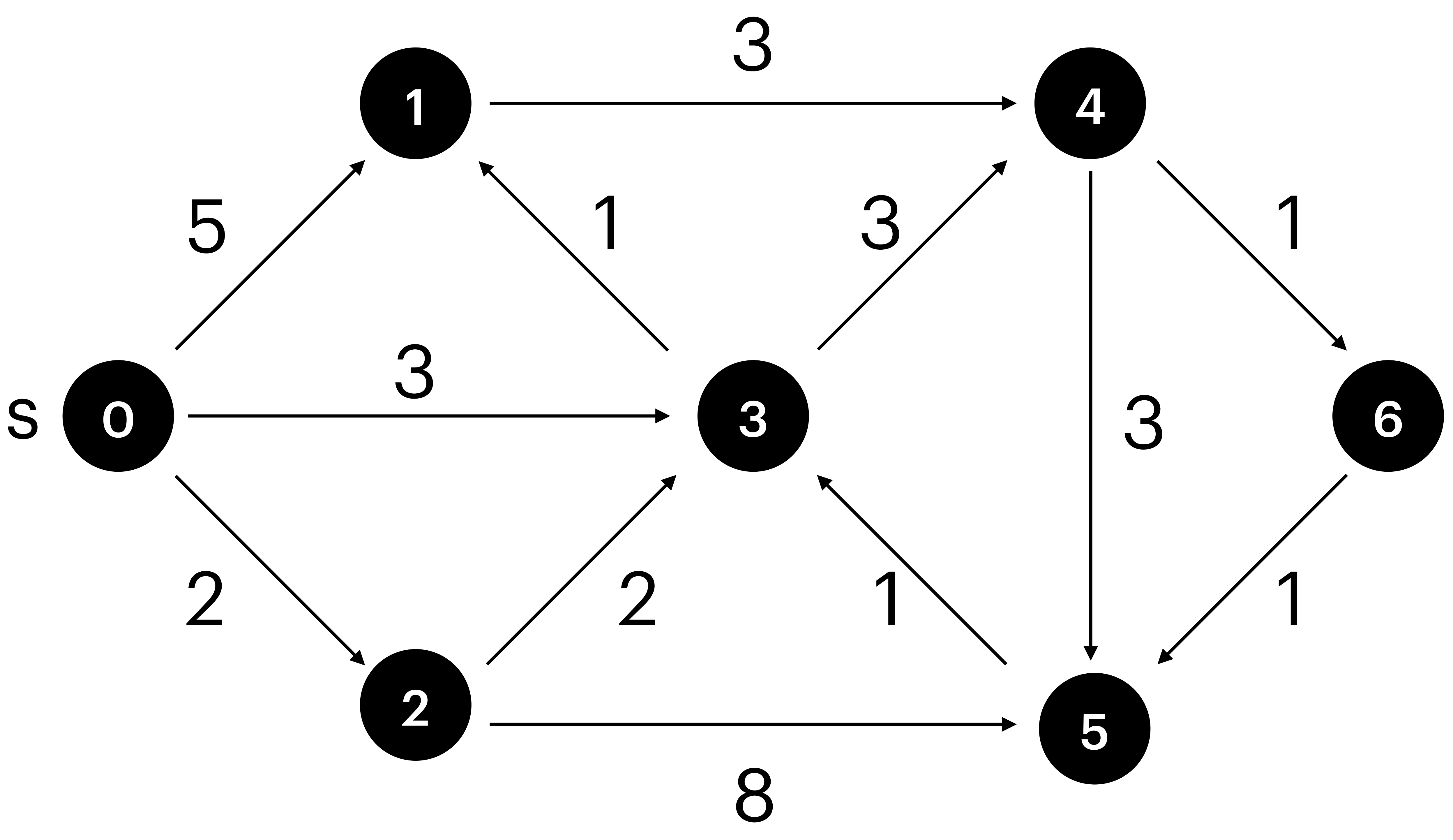
S :

d[] :

0	1	2	3	4	5	6

"Heap" :

0	1	2	3	4	5	6



Shortest Paths
Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $\quad v^* \leftarrow$ extract-min(H)
6: $\quad S \leftarrow S \cup \{v^*\}$
7: $\quad$ **for** $(v^*, v) \in E,\ v \notin S$ **do**
8: $\quad\quad \mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
9: $\quad\quad$ decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

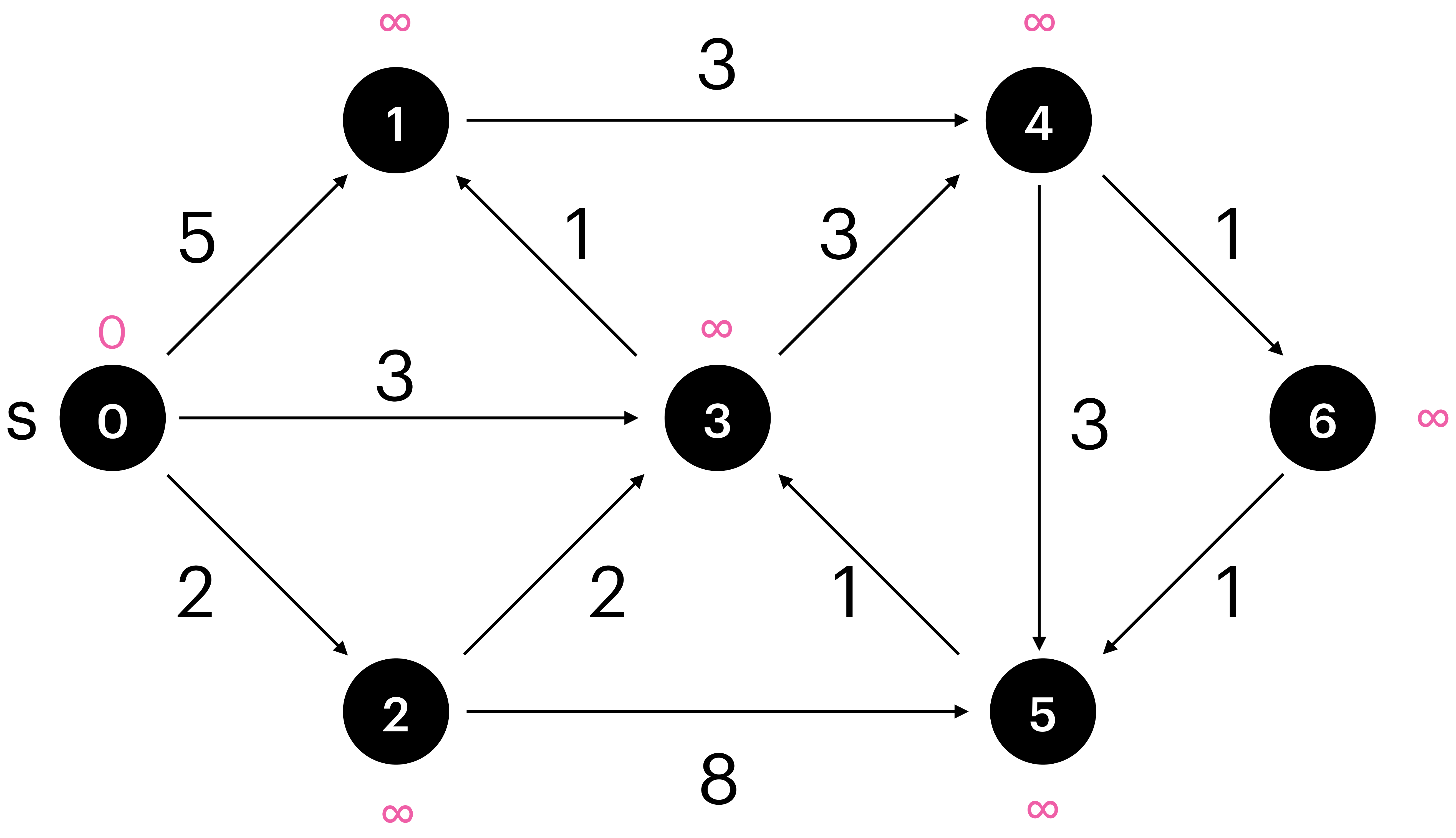
S : ∅

d[] :

0	1	2	3	4	5	6
0	∞	∞	∞	∞	∞	∞

"Heap" :

0	1	2	3	4	5	6
0	∞	∞	∞	∞	∞	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key$(H, s, 0)$
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
    decrease-key$(H, v, \mathrm{d}[v])$

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : ∅

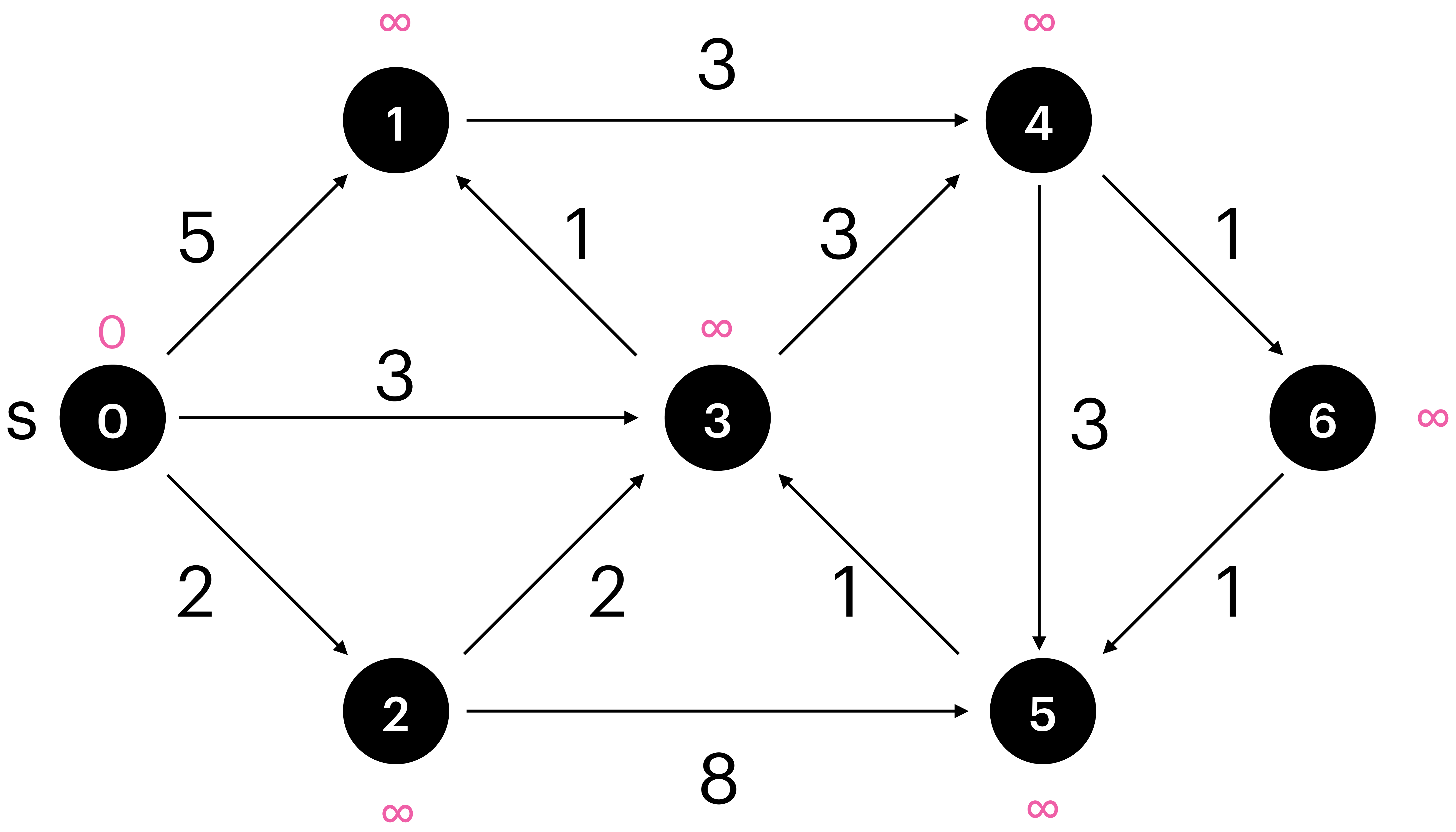
v* =

d[] :

0	1	2	3	4	5	6
0	∞	∞	∞	∞	∞	∞

"Heap" :

0	1	2	3	4	5	6
0	∞	∞	∞	∞	∞	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
    decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

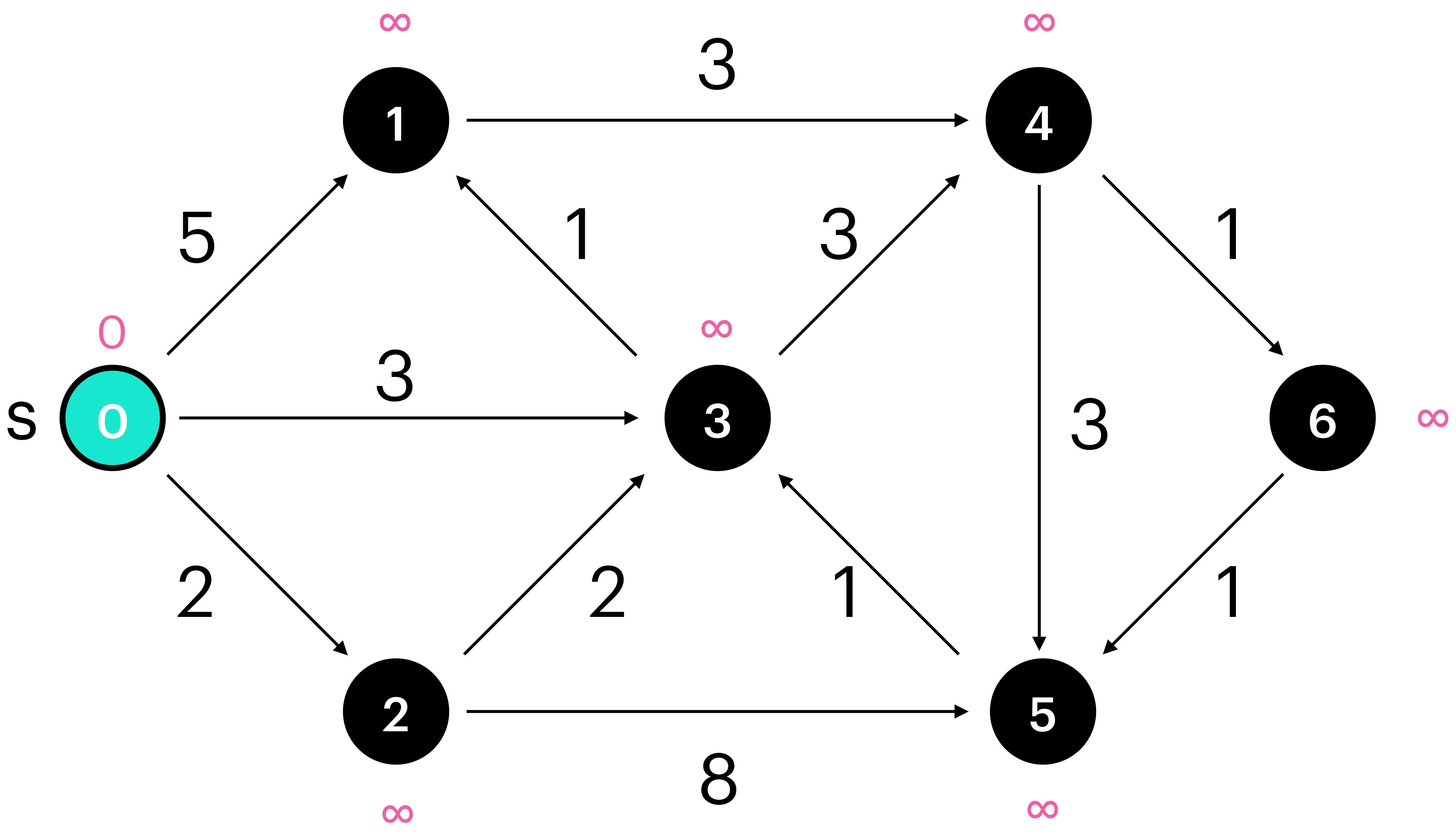
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : ∅

v* = 0



d[] :

0	1	2	3	4	5	6
0	∞	∞	∞	∞	∞	∞

"Heap" :

1	2	3	4	5	6
∞	∞	∞	∞	∞	∞

Shortest Paths

Dijkstra’s Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
    $v^* \leftarrow$ extract-min(H)
    $S \leftarrow S \cup \{v^*\}$
    for $(v^*, v) \in E,\; v \notin S$ **do**
        $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
        decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 }

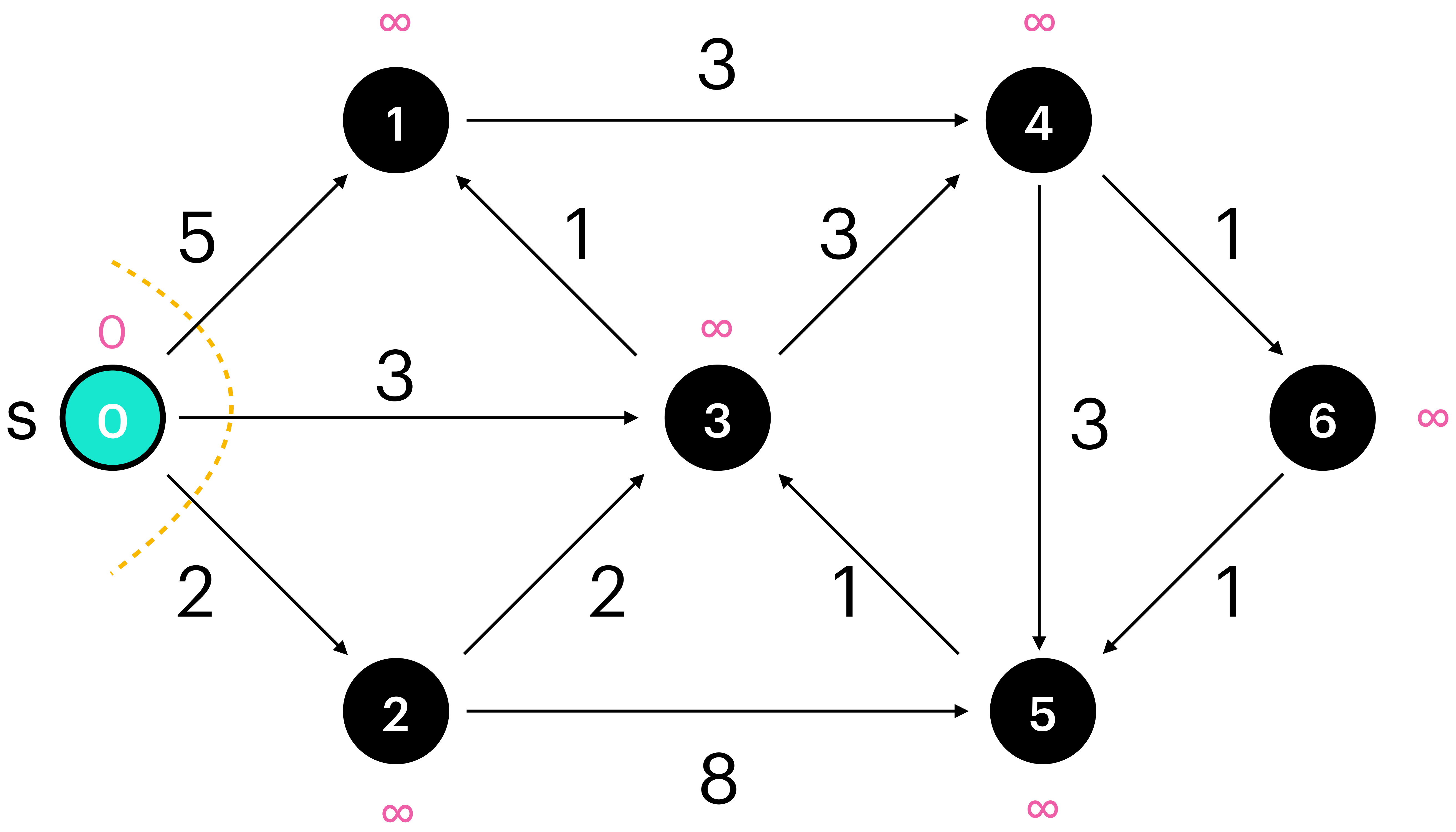
$v^* = 0$

d[] :

0	1	2	3	4	5	6
0	∞	∞	∞	∞	∞	∞

“Heap” :

1	2	3	4	5	6
∞	∞	∞	∞	∞	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\; v \notin S$ **do**
    $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
    decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

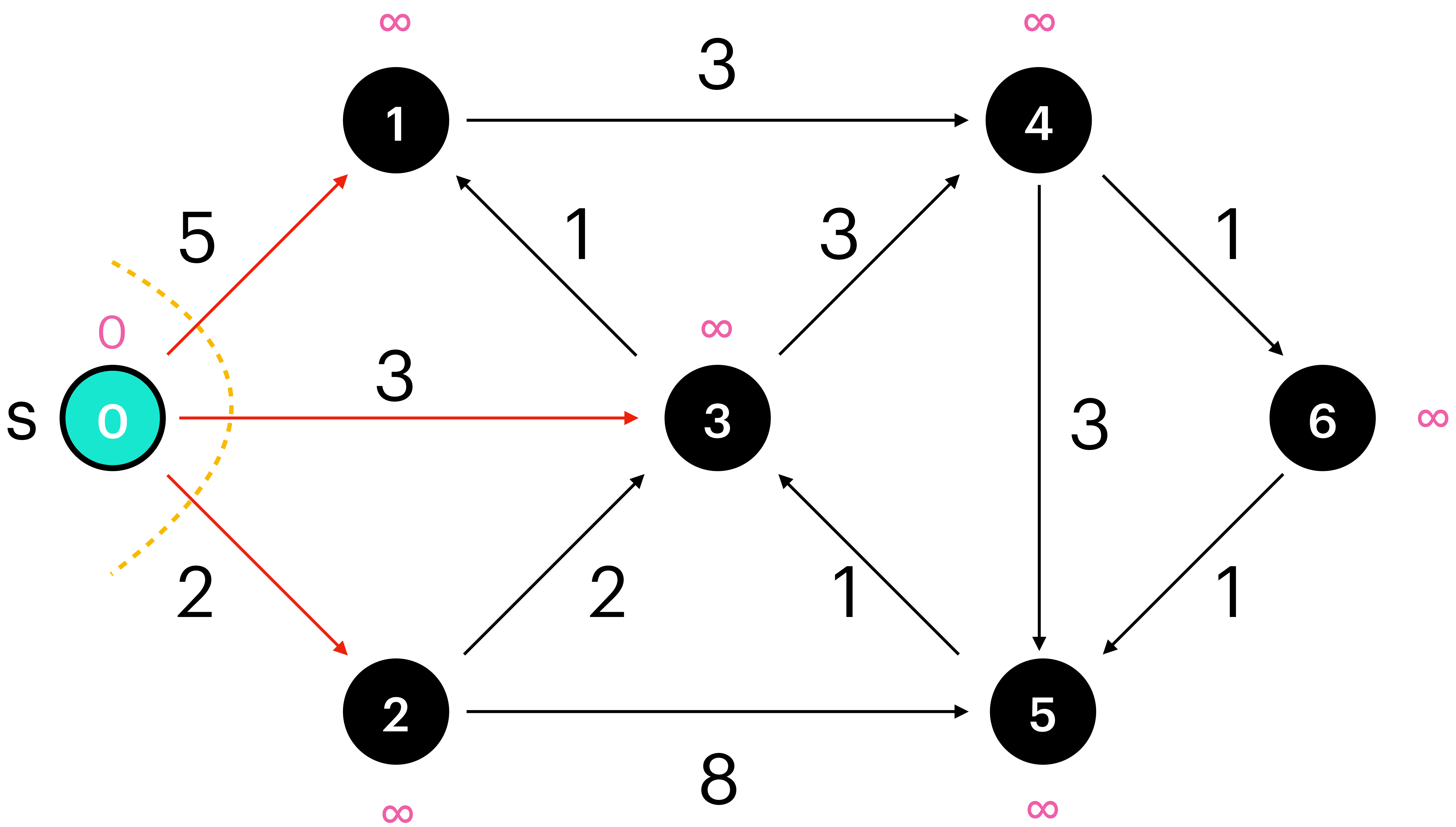
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 }

v* = 0



d[] :

0	1	2	3	4	5	6
0	∞	∞	∞	∞	∞	∞

"Heap" :

1	2	3	4	5	6
∞	∞	∞	∞	∞	∞

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\; v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

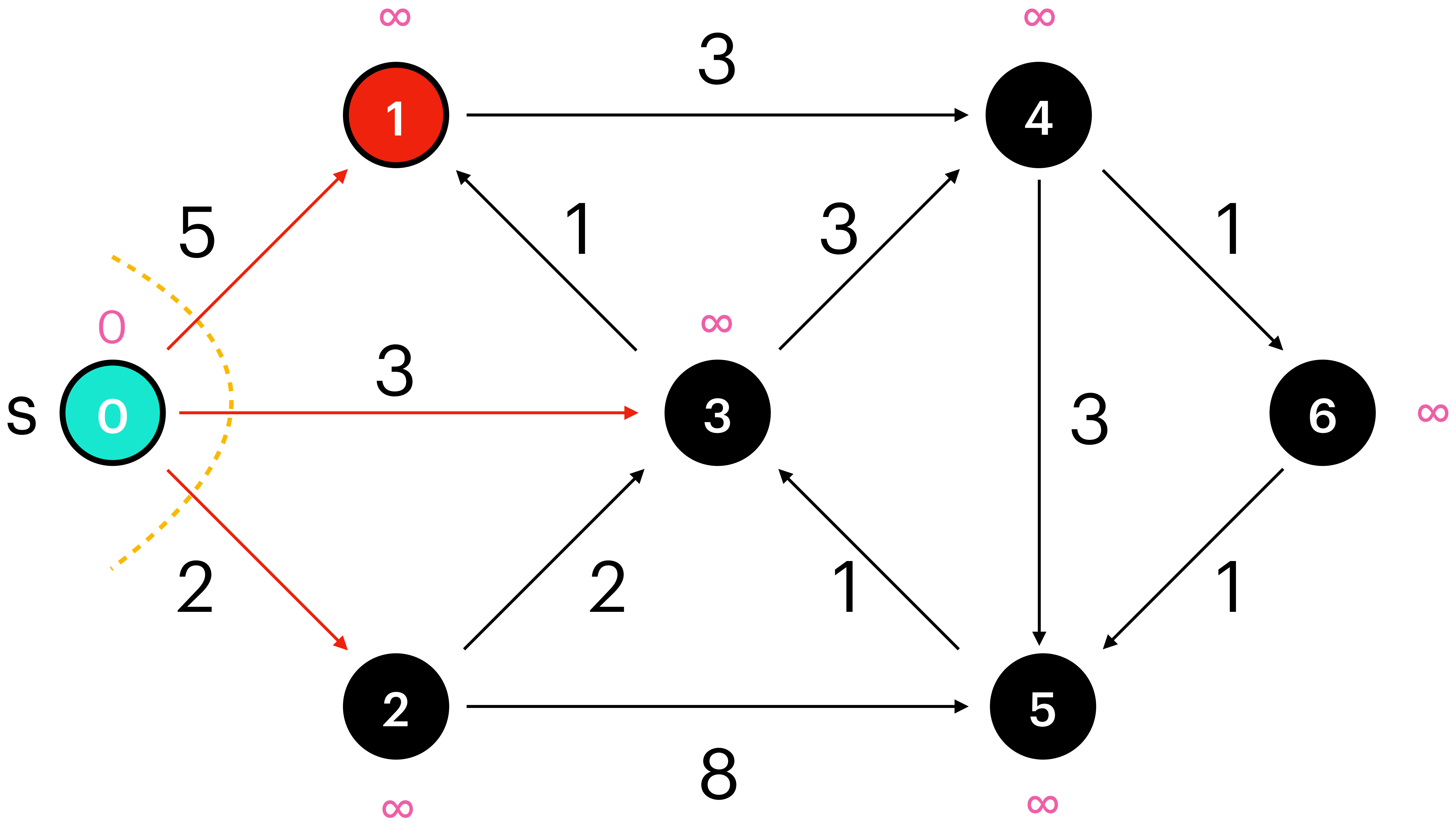
decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 }

$v^* = 0$

min {∞ , 0 + 5} = 5



d[] :

0	1	2	3	4	5	6
0	5	∞	∞	∞	∞	∞

"Heap" :

1	2	3	4	5	6
5	∞	∞	∞	∞	∞

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E, \; v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

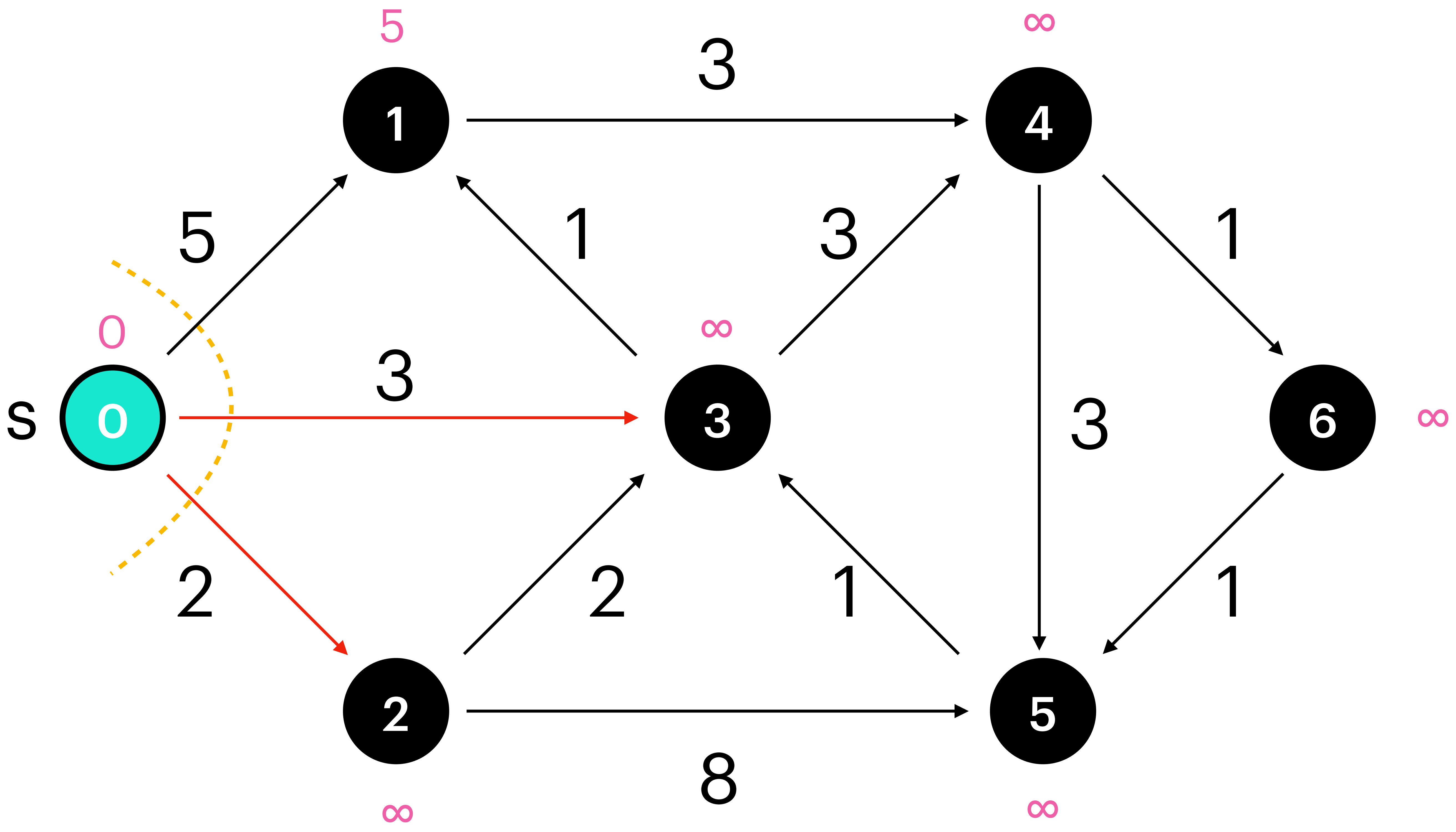
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 }

$v^* = 0$



d[] :

0	1	2	3	4	5	6
0	5	∞	∞	∞	∞	∞

"Heap" :

1	2	3	4	5	6
5	∞	∞	∞	∞	∞

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

```
d[s] ← 0;   d[v] ← ∞  ∀v ∈ V \ {s}
S ← ∅
H ← make-heap(V); decrease-key(H, s, 0)
while S ≠ V do
    v* ← extract-min(H)
    S ← S ∪ {v*}
    for (v*, v) ∈ E, v ∉ S do
        d[v] ← min{d[v], d[v*] + c(v*, v)}
        decrease-key(H, v, d[v])
```

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 }

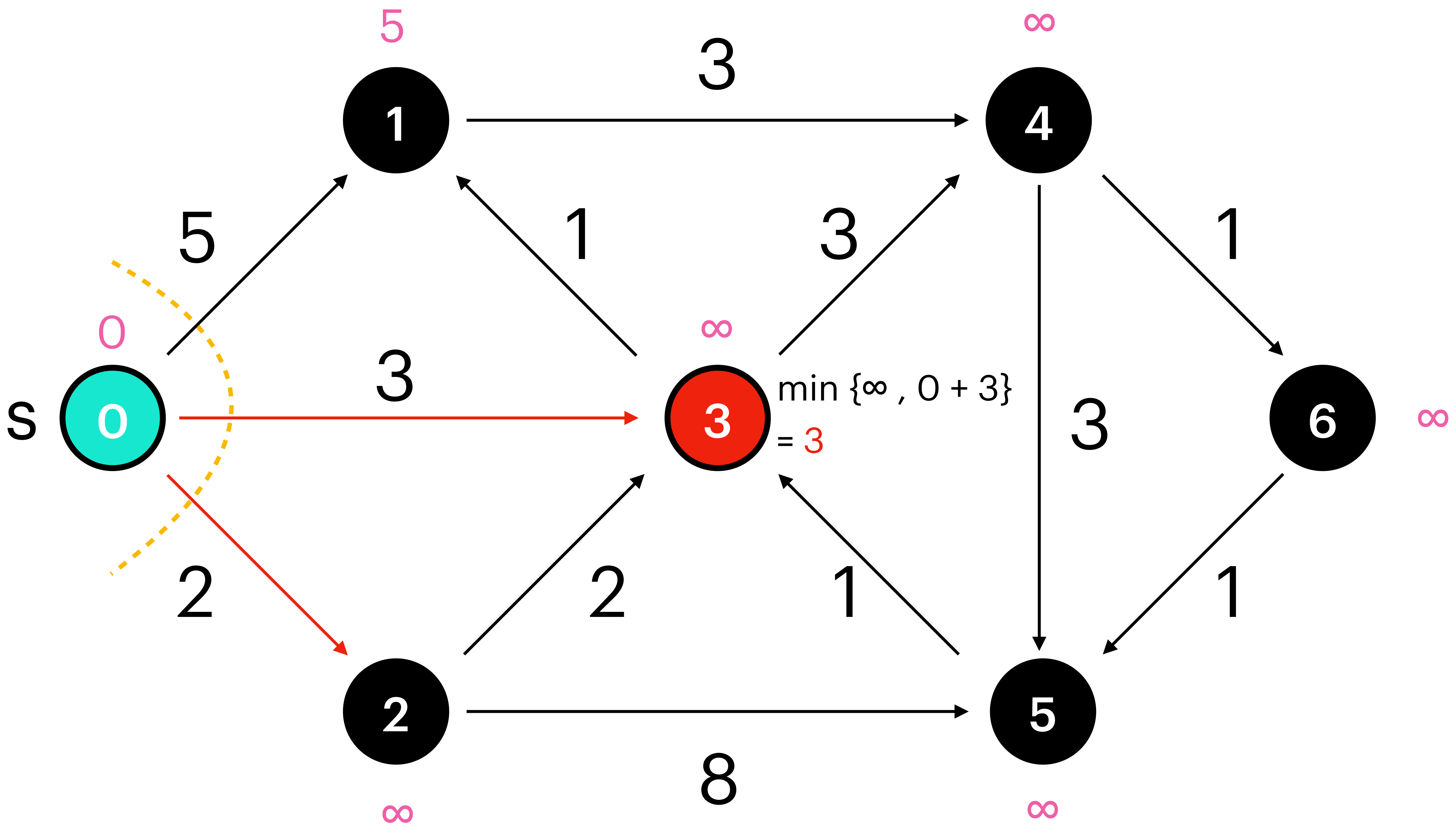
$v^* = 0$

d[] :

0	1	2	3	4	5	6
0	5	∞	3	∞	∞	∞

"Heap" :

1	2	3	4	5	6
5	∞	3	∞	∞	∞



Shortest Paths
Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

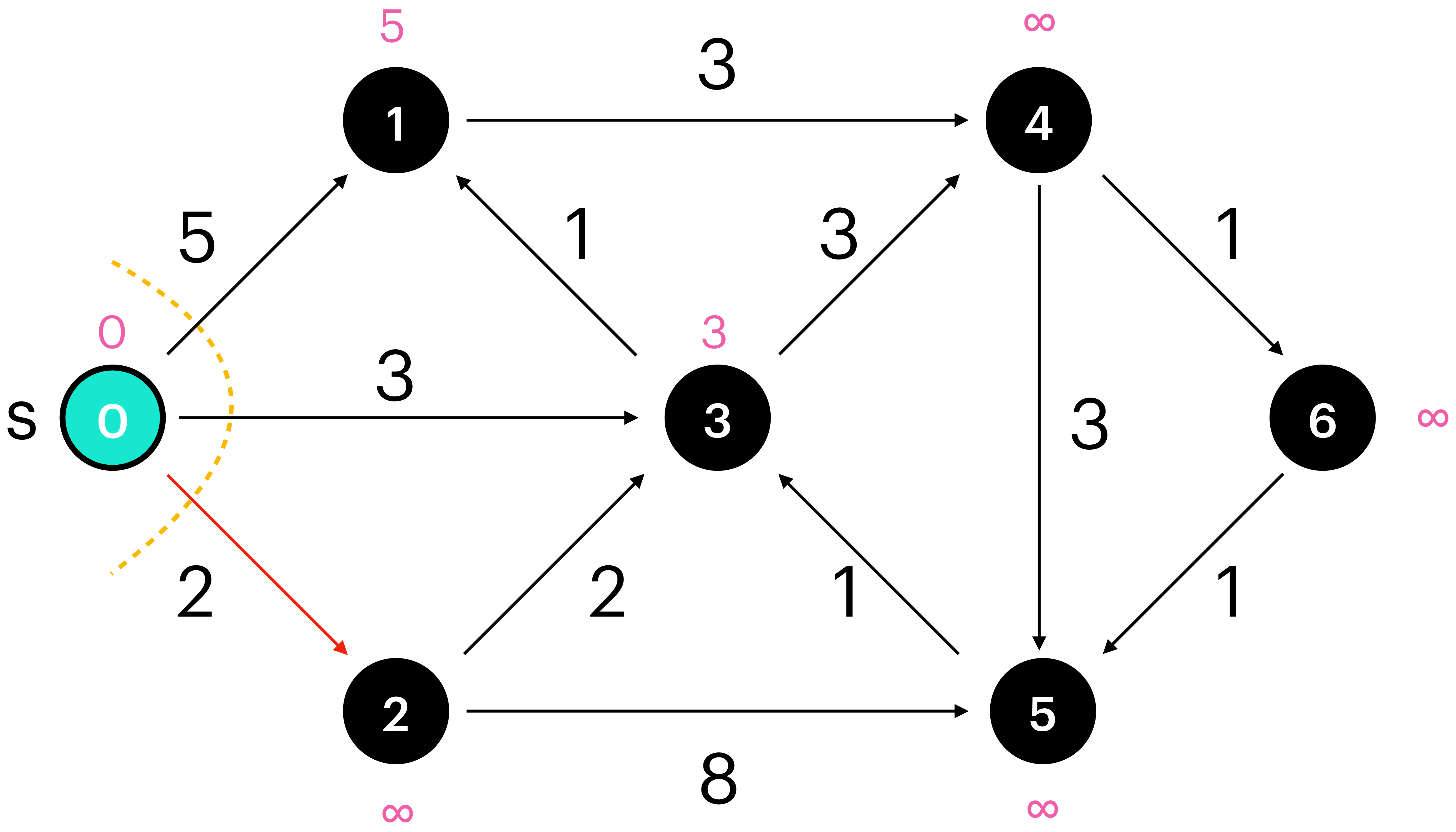
make-heap(V) :
Create a min heap of the vertices

extract-min(H) :
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :
Update the distance of v in heap H to the key k

S : { 0 }

$v^* = 0$



d[] :

0	1	2	3	4	5	6
0	5	∞	3	∞	∞	∞

"Heap" :

1	2	3	4	5	6
5	∞	3	∞	∞	∞

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :
Create a min heap of the vertices

extract-min(H) :
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :
Update the distance of v in heap H to the key k

S : { 0 }

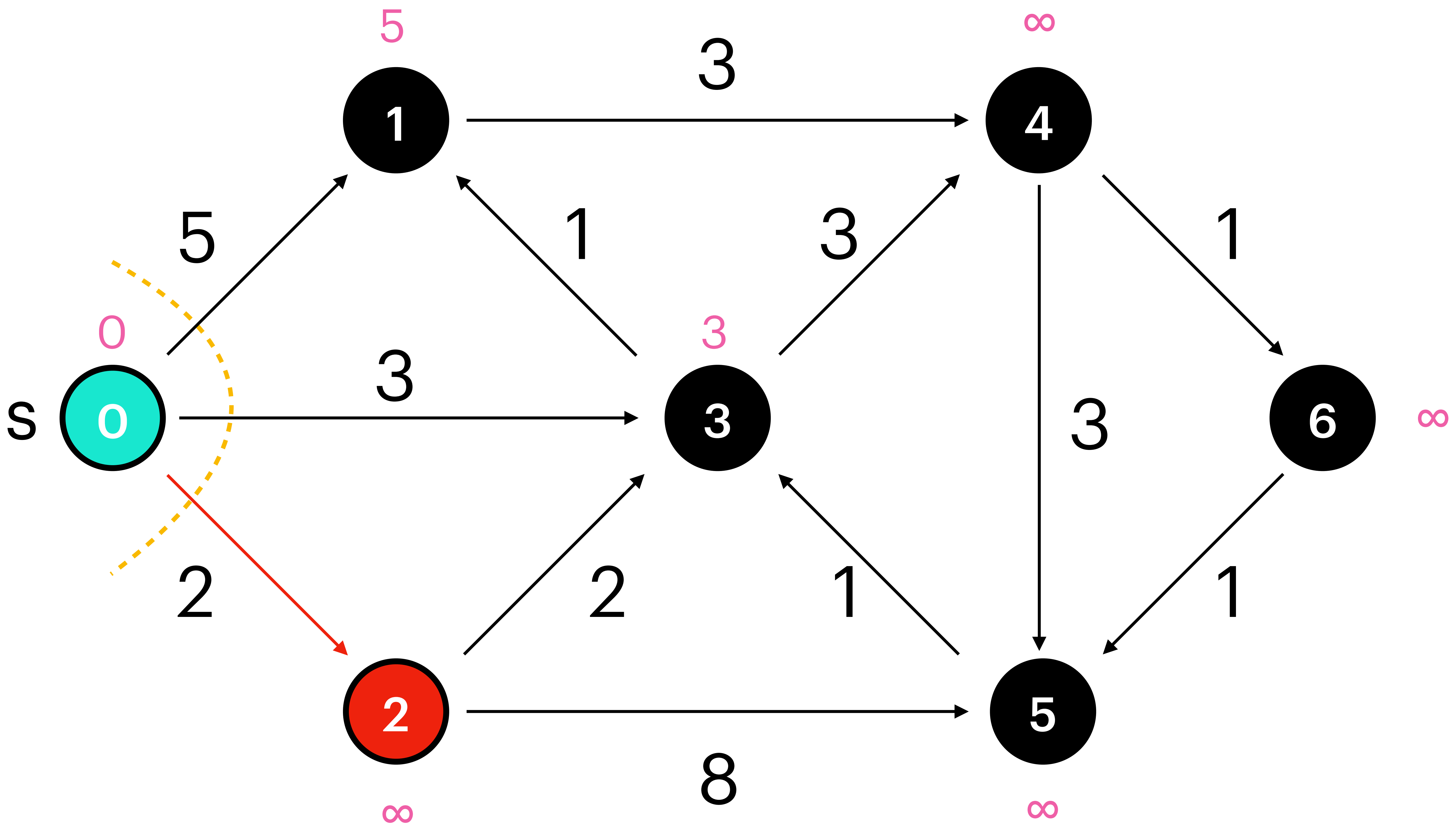
$v^* = 0$

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	∞	∞

"Heap" :

1	2	3	4	5	6
5	2	3	∞	∞	∞



min {∞ , 0 + 2} = 2

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

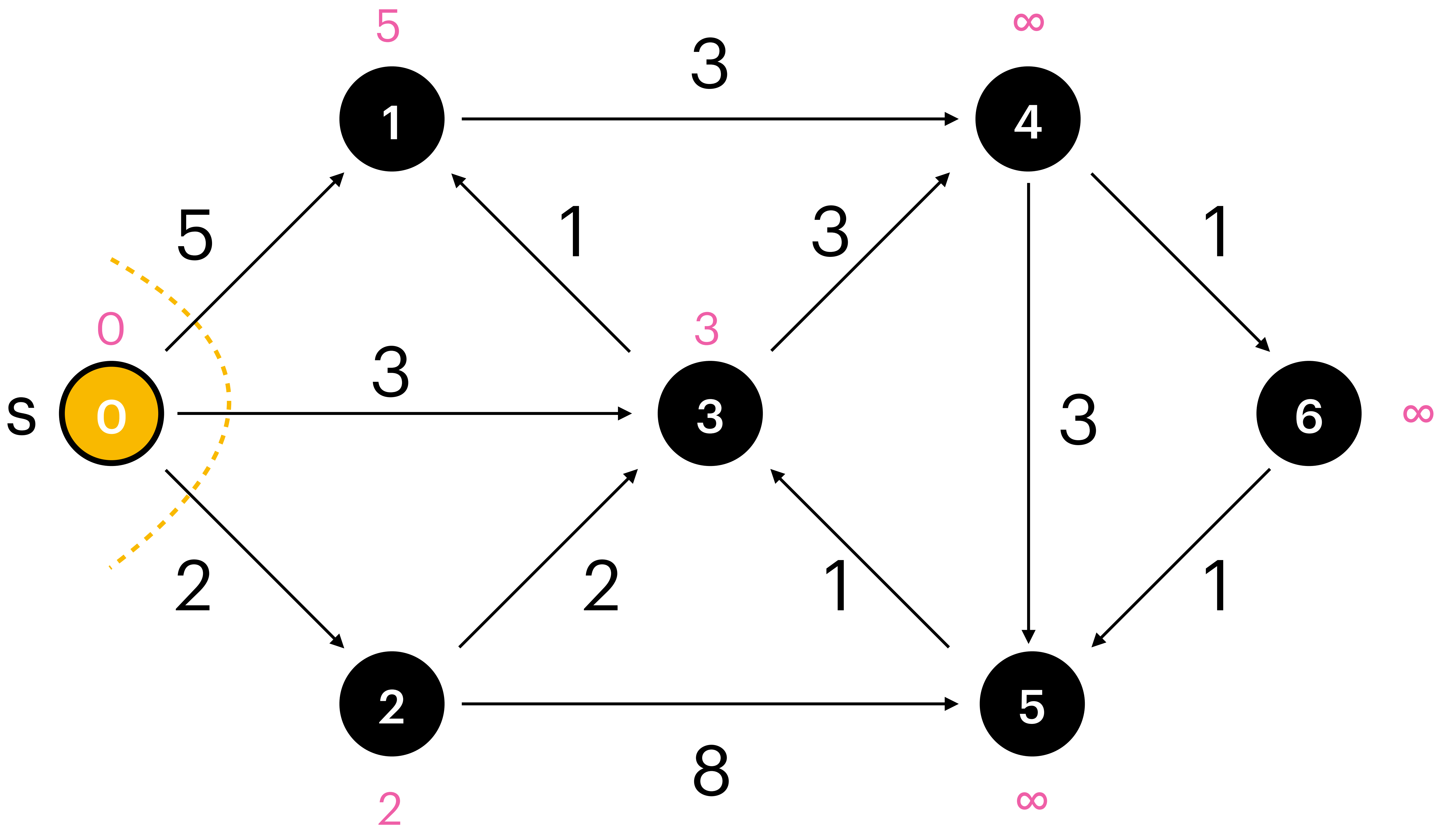
extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 }



d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	∞	∞

"Heap" :

1	2	3	4	5	6
5	2	3	∞	∞	∞

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
    decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 }

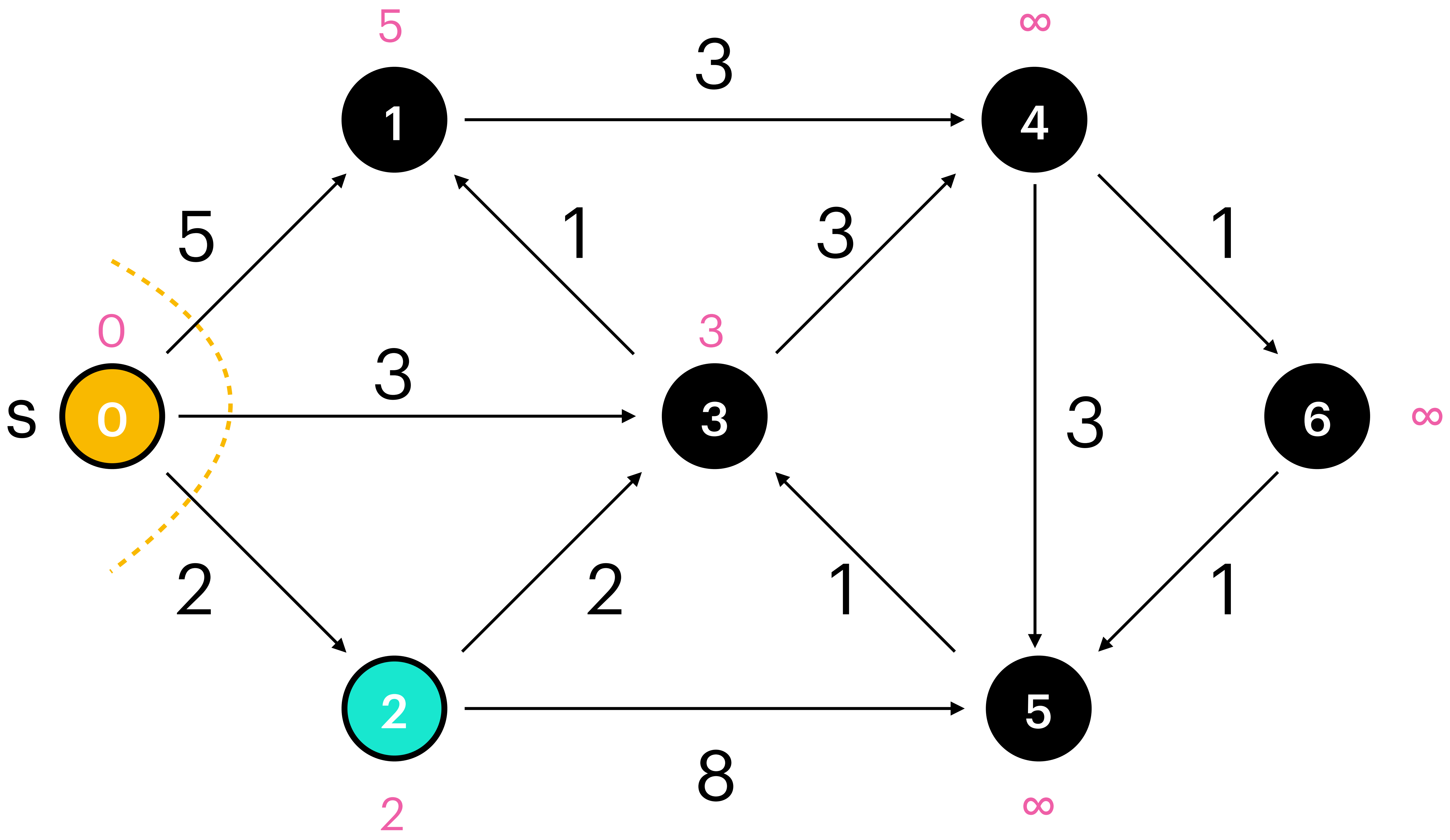
v* = 2

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	∞	∞

"Heap" :

1	3	4	5	6
5	3	∞	∞	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\; v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 }

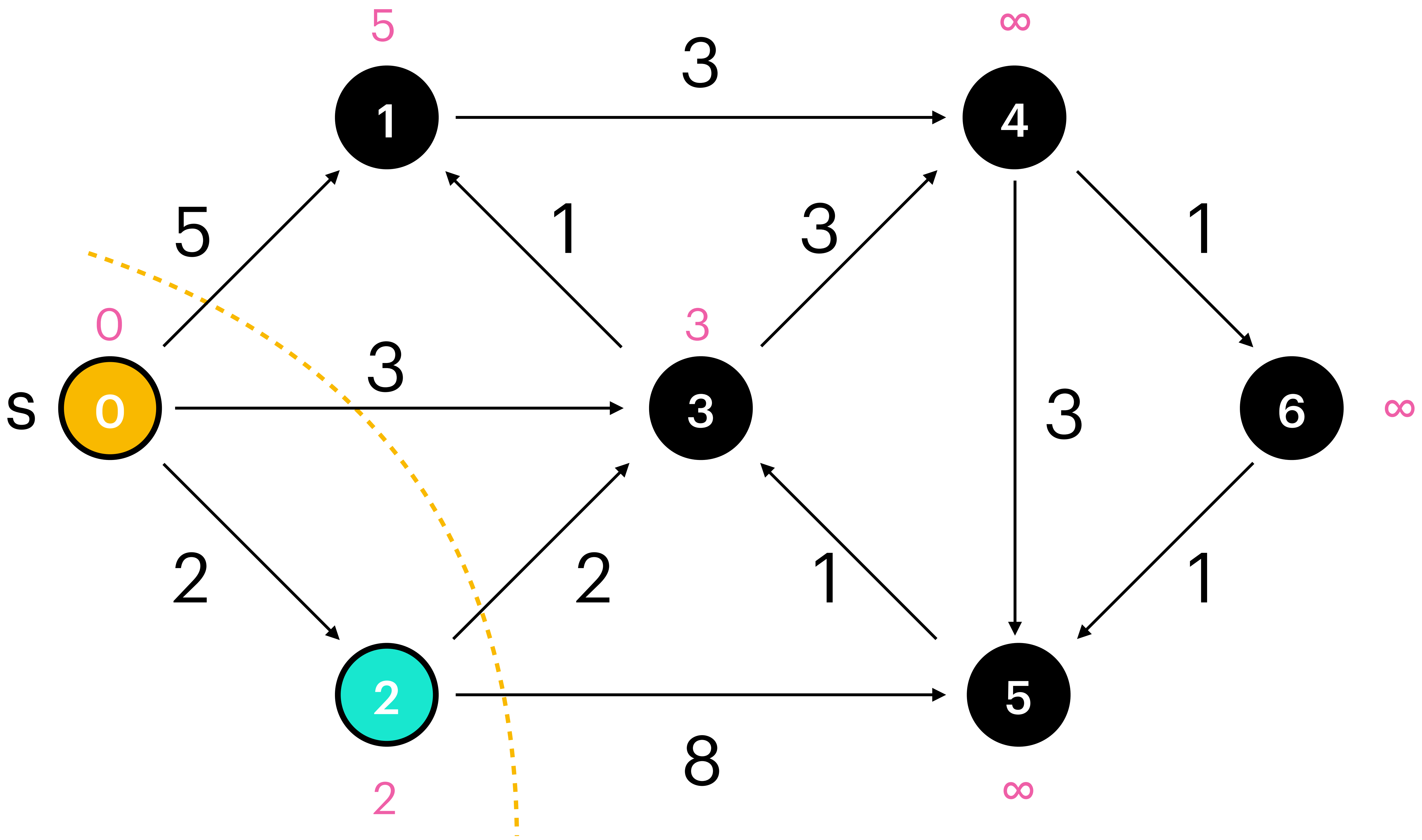
$v^* = 2$

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	∞	∞

"Heap" :

1	3	4	5	6
5	3	∞	∞	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $\mathrm{d}[s] \leftarrow 0$; $\mathrm{d}[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $v^* \leftarrow$ extract-min(H)
6: $S \leftarrow S \cup \{v^*\}$
7: **for** $(v^*, v) \in E,\ v \notin S$ **do**
8: $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
9: decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 }

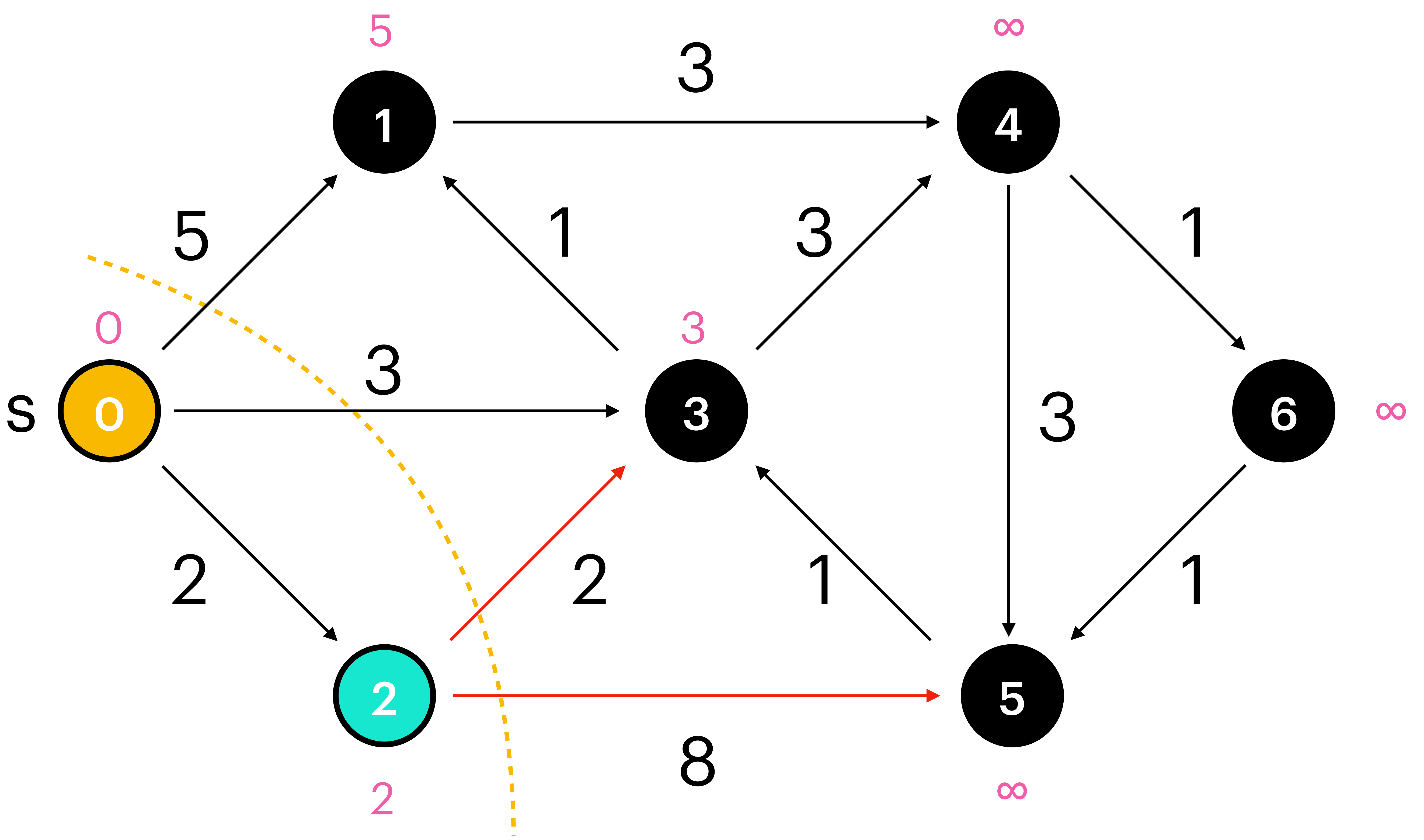
v* = 2

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	∞	∞

"Heap" :

1	3	4	5	6
5	3	∞	∞	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $\quad v^* \leftarrow$ extract-min(H)
6: $\quad S \leftarrow S \cup \{v^*\}$
7: $\quad$ **for** $(v^*, v) \in E,\ v \notin S$ **do**
8: $\quad\quad \mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
9: $\quad\quad$ decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 }

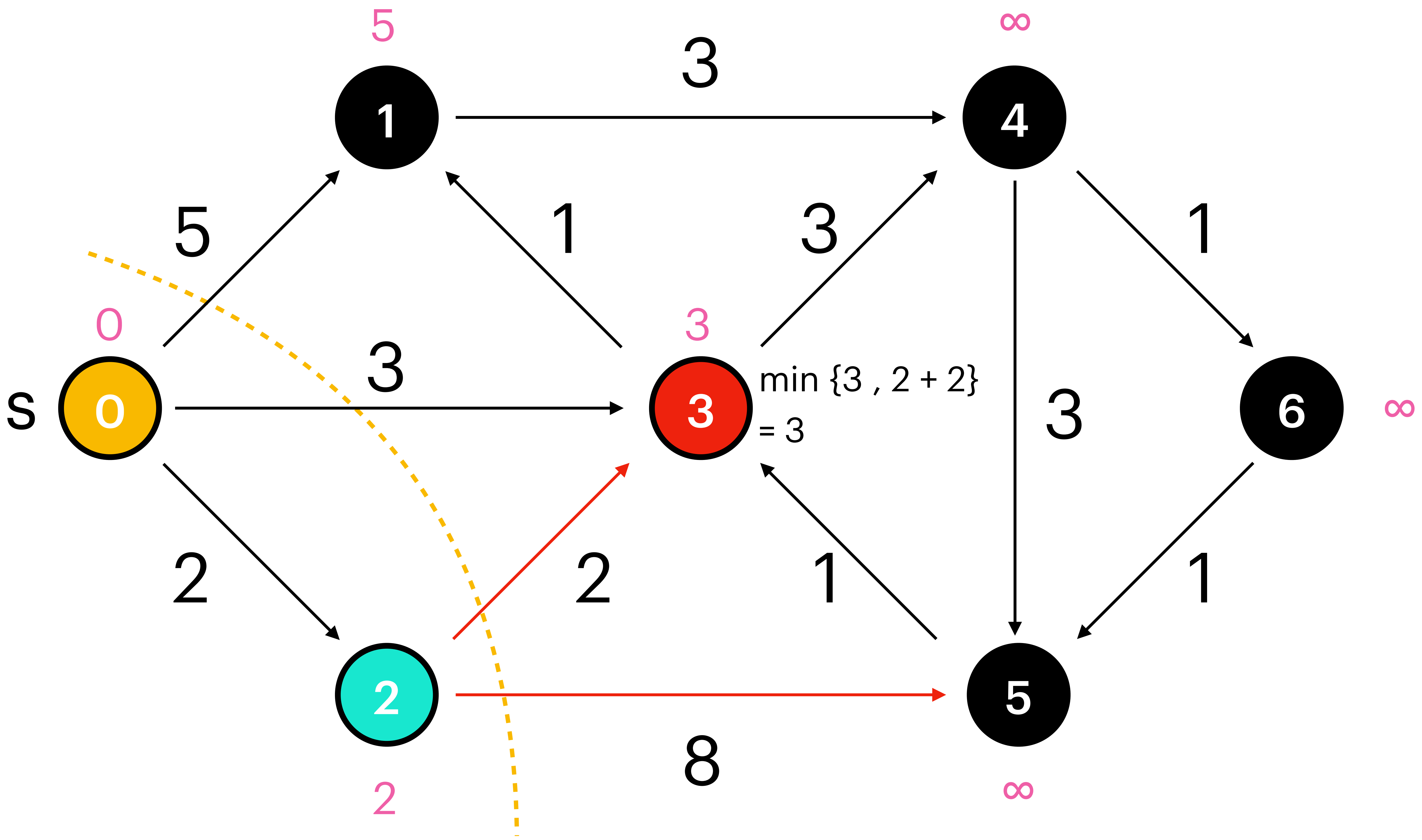
v* = 2

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	∞	∞

"Heap" :

1	3	4	5	6
5	3	∞	∞	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $\mathrm{d}[s] \leftarrow 0;\quad \mathrm{d}[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $\quad v^* \leftarrow$ extract-min(H)
6: $\quad S \leftarrow S \cup \{v^*\}$
7: $\quad$ **for** $(v^*, v) \in E,\ v \notin S$ **do**
8: $\quad\quad \mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
9: $\quad\quad$ decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 }

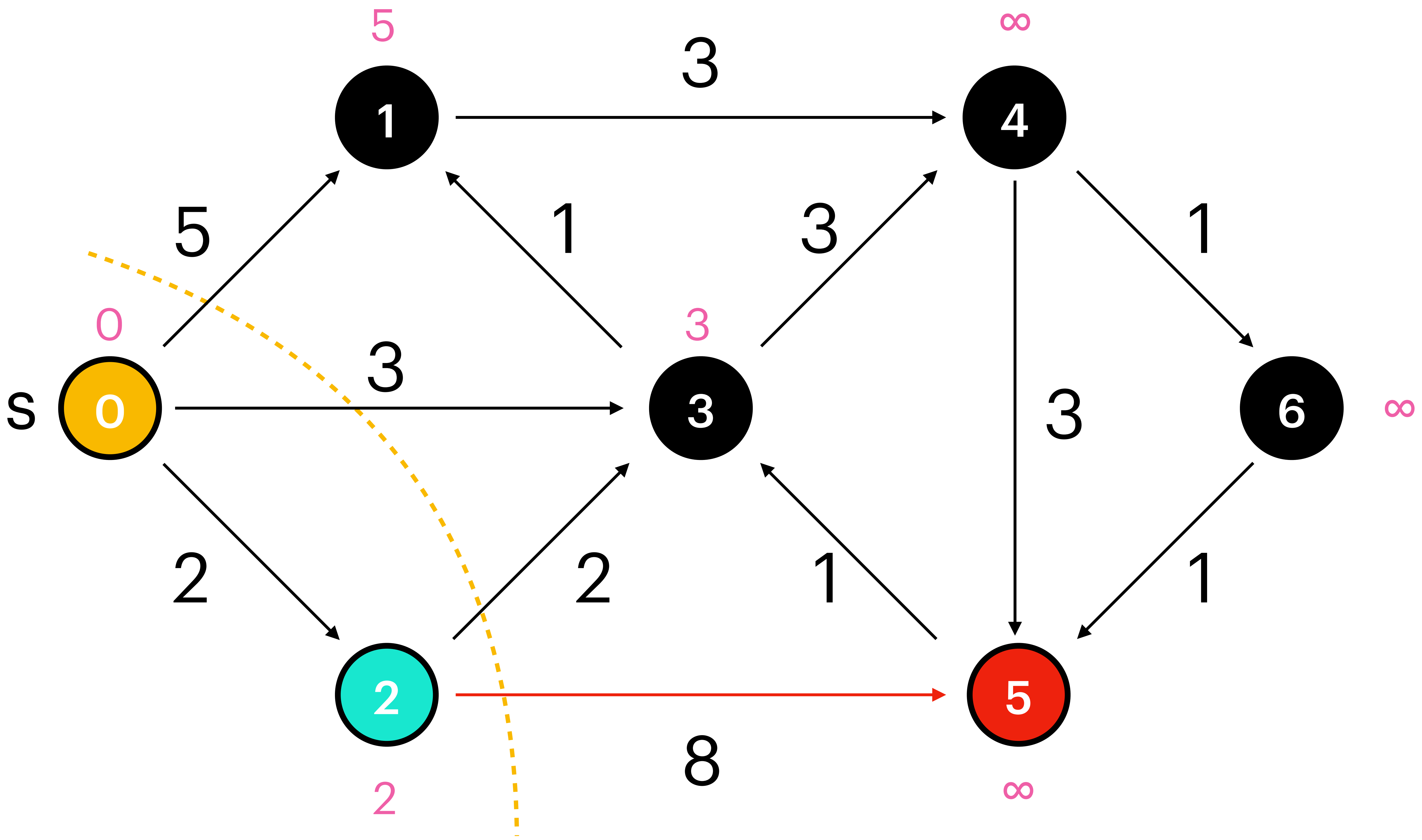
$v^* = 2$

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	∞	∞

"Heap" :

1	3	4	5	6
5	3	∞	∞	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

```
1: d[s] ← 0;   d[v] ← ∞  ∀v ∈ V \ {s}
2: S ← ∅
3: H ← make-heap(V); decrease-key(H, s, 0)
4: while S ≠ V do
5:     v* ← extract-min(H)
6:     S ← S ∪ {v*}
7:     for (v*, v) ∈ E, v ∉ S do
8:         d[v] ← min{d[v], d[v*] + c(v*, v)}
9:         decrease-key(H, v, d[v])
```

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 }

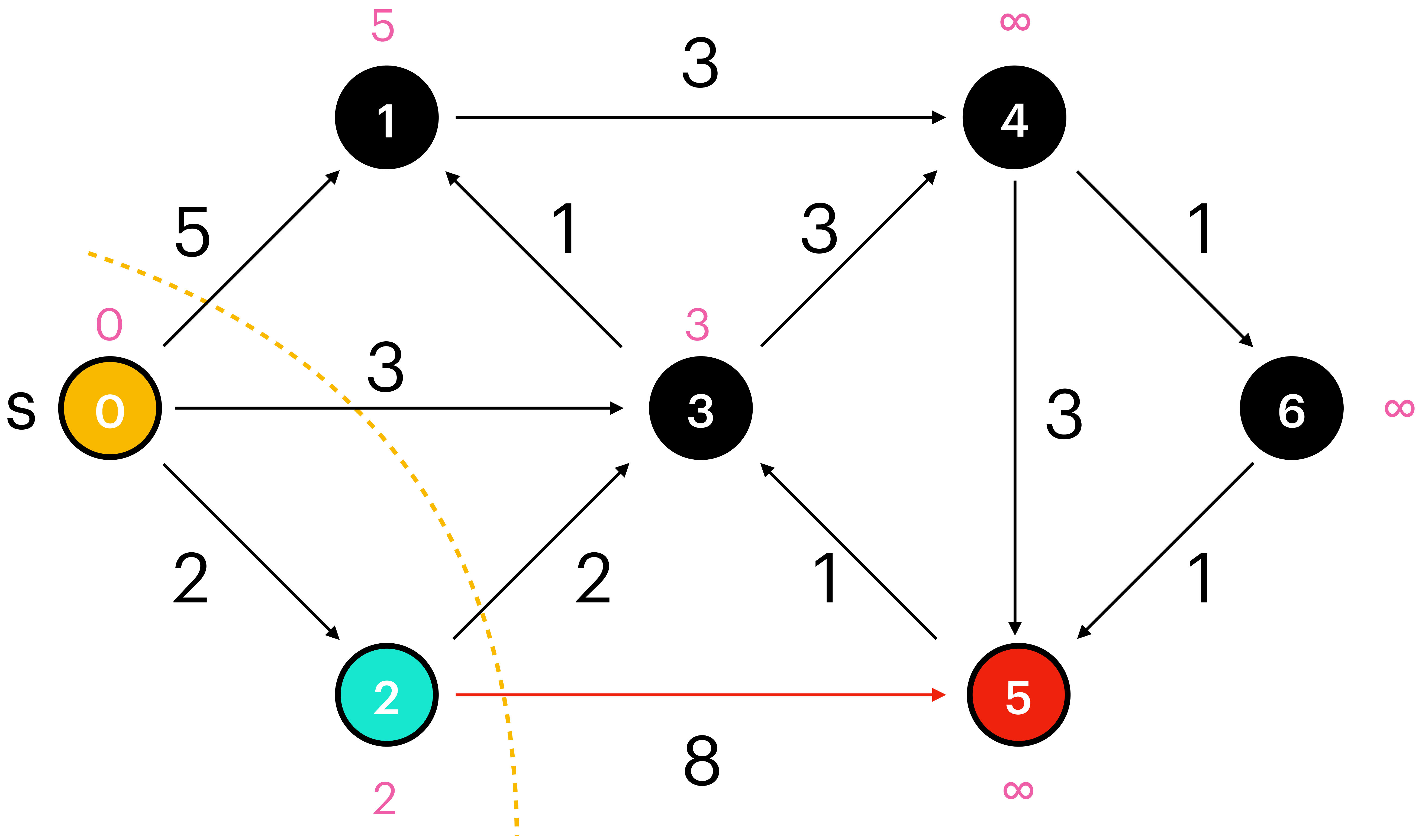
v* = 2

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	10	∞

"Heap" :

1	3	4	5	6
5	3	∞	10	∞



min {∞ , 2 + 8} = 10

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\; v \notin S$ **do**
    $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
    decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :
Create a min heap of the vertices

extract-min(H) :
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :
Update the distance of v in heap H to the key k

S : { 0 , 2 }

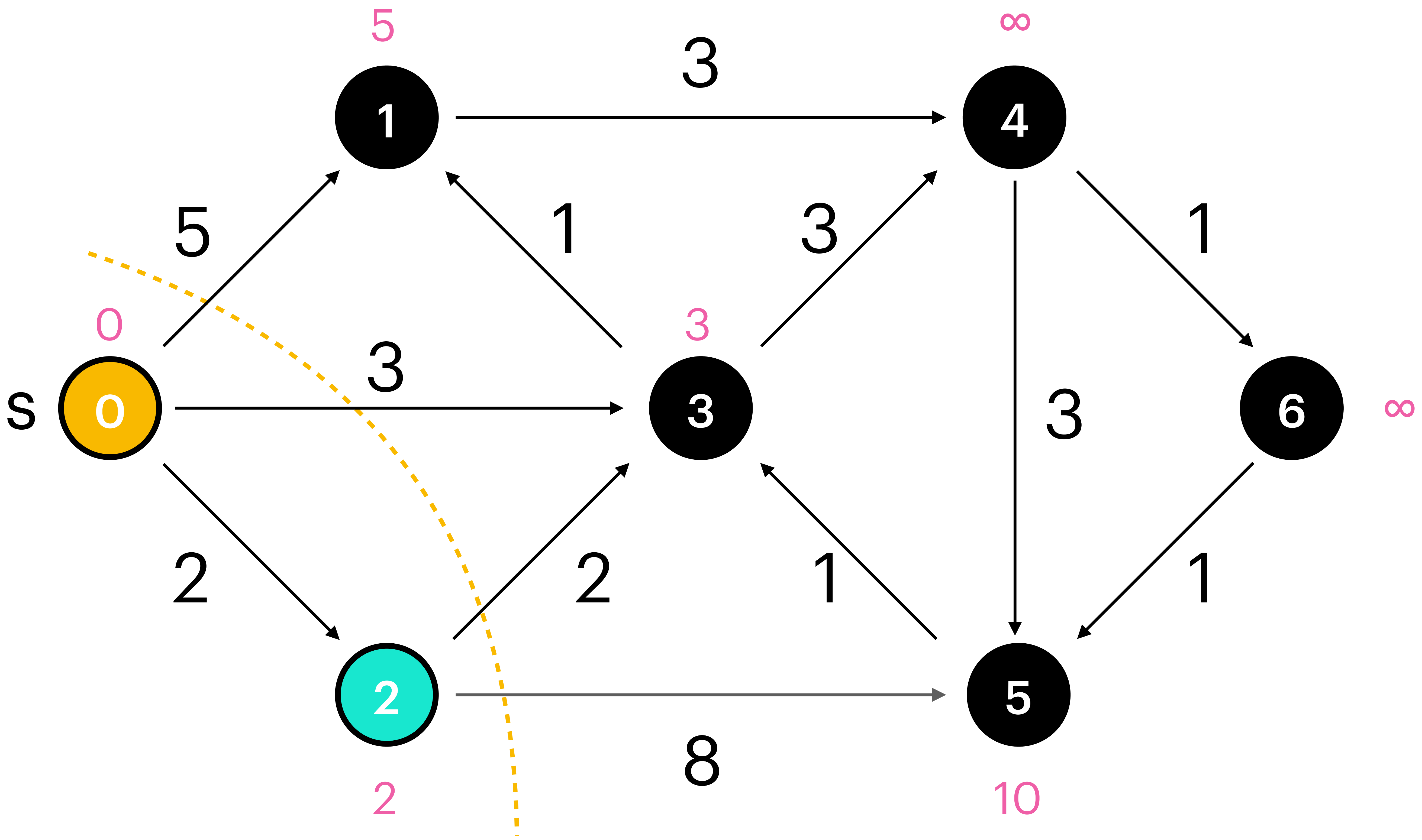
$v^* = 2$

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	10	∞

"Heap" :

1	3	4	5	6
5	3	∞	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

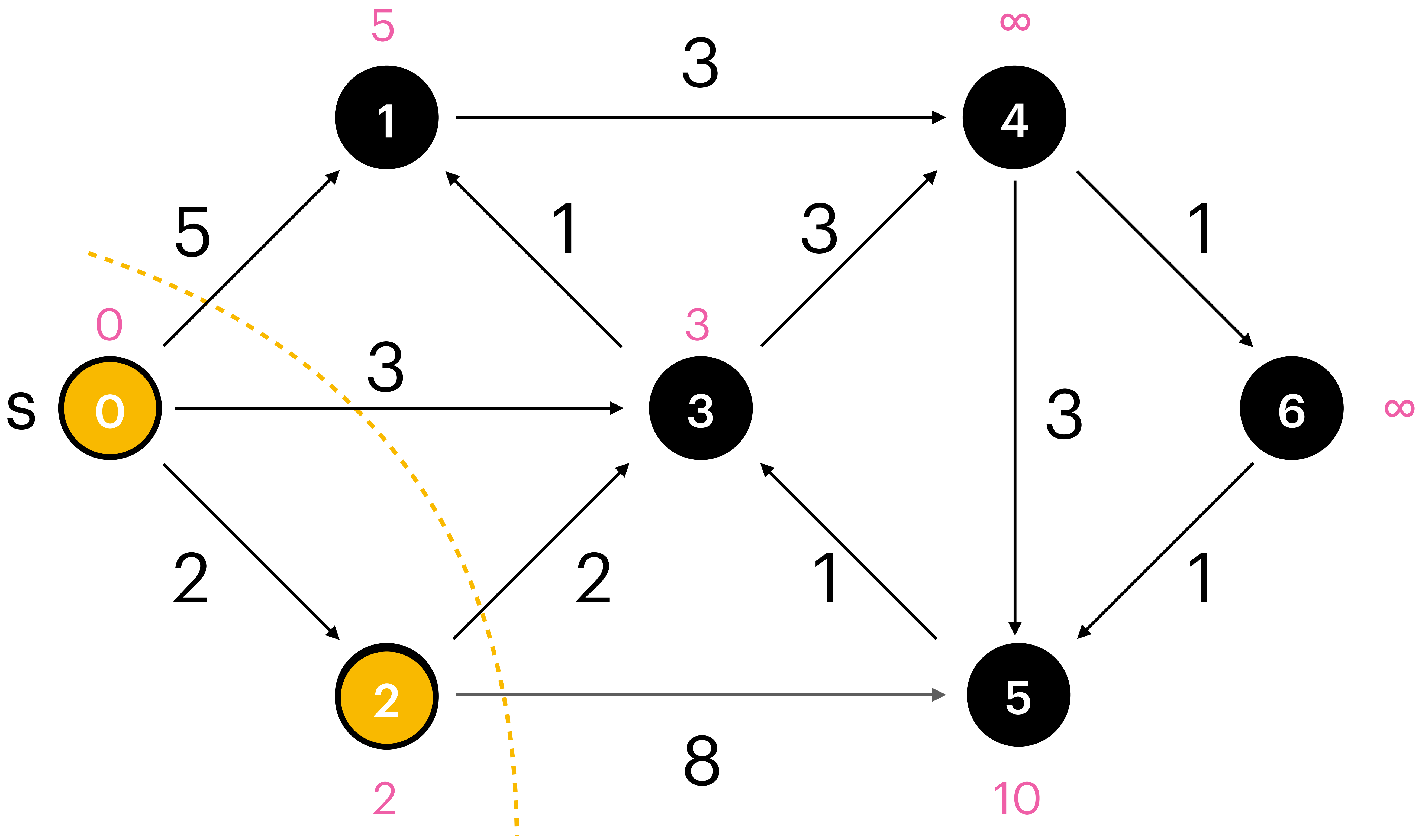
S : { 0 , 2 }

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	10	∞

"Heap" :

1	3	4	5	6
5	3	∞	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $d[s] \leftarrow 0$; $d[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $v^* \leftarrow$ extract-min(H)
6: $S \leftarrow S \cup \{v^*\}$
7: **for** $(v^*, v) \in E,\ v \notin S$ **do**
8: $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
9: decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

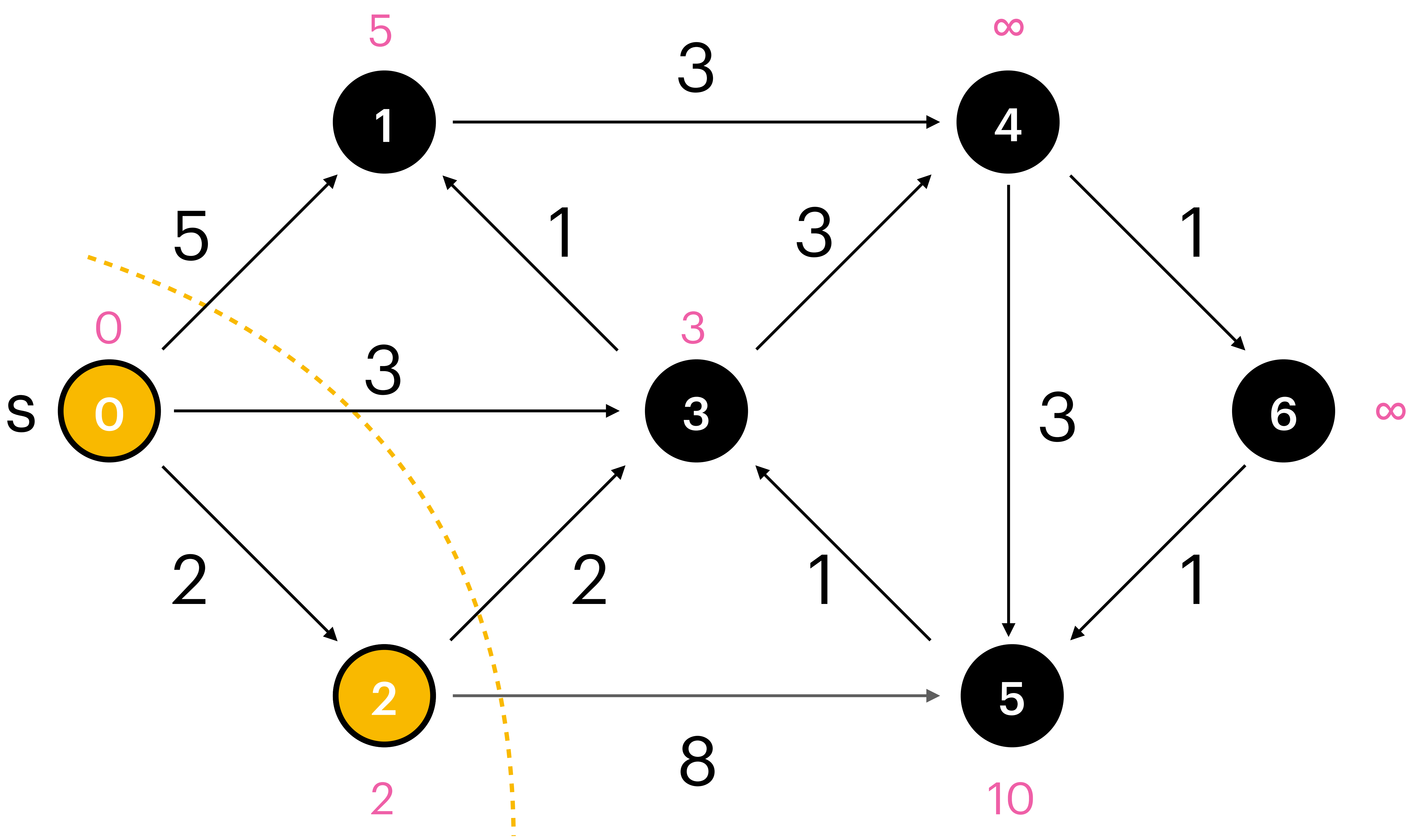
S : { 0 , 2 }

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	10	∞

"Heap" :

1	3	4	5	6
5	3	∞	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

```
d[s] ← 0;   d[v] ← ∞  ∀v ∈ V \ {s}
S ← ∅
H ← make-heap(V); decrease-key(H, s, 0)
while S ≠ V do
    v* ← extract-min(H)
    S ← S ∪ {v*}
    for (v*, v) ∈ E, v ∉ S do
        d[v] ← min{d[v], d[v*] + c(v*, v)}
        decrease-key(H, v, d[v])
```

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 }

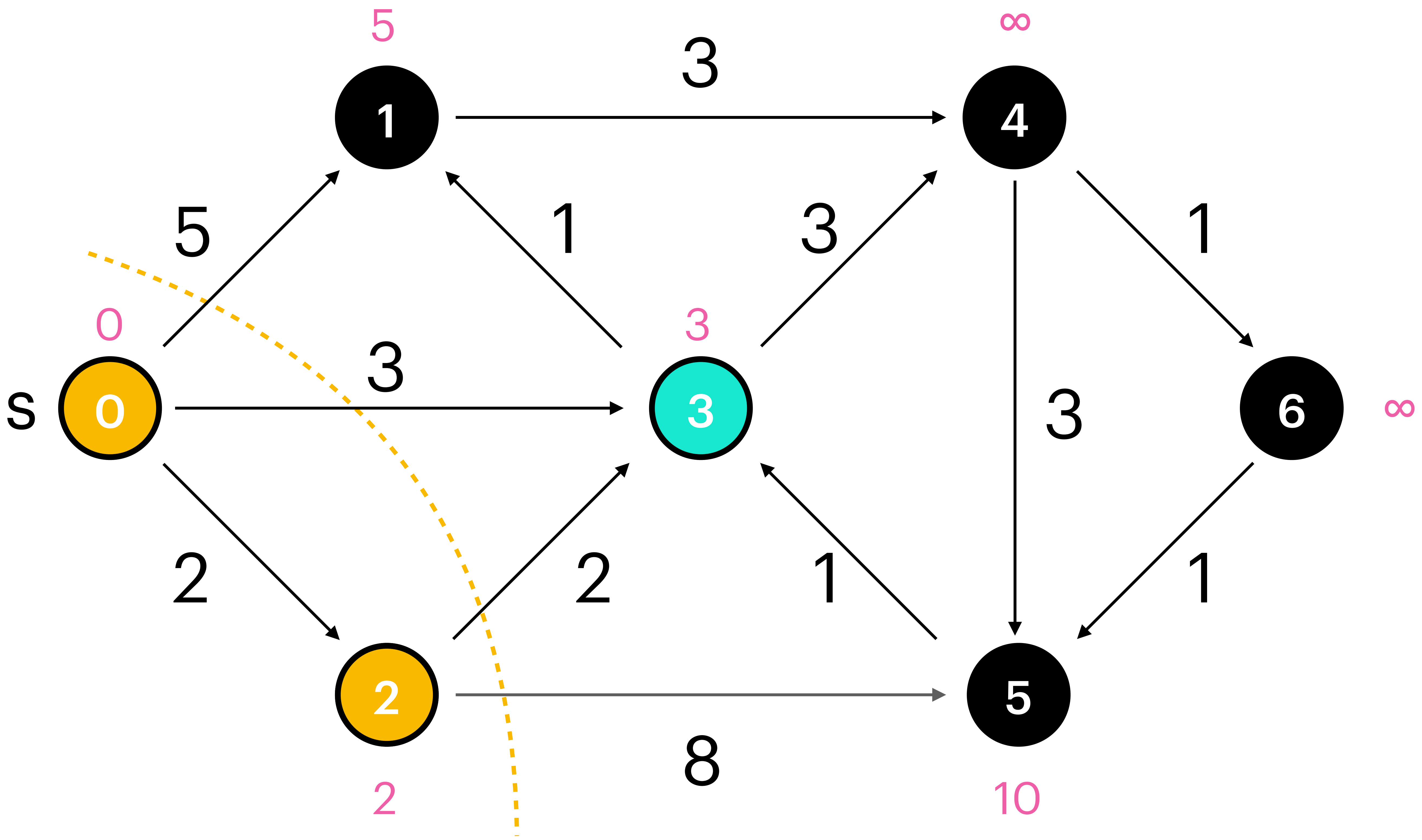
$v^* = 3$

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	10	∞

"Heap" :

1	4	5	6
5	∞	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 }

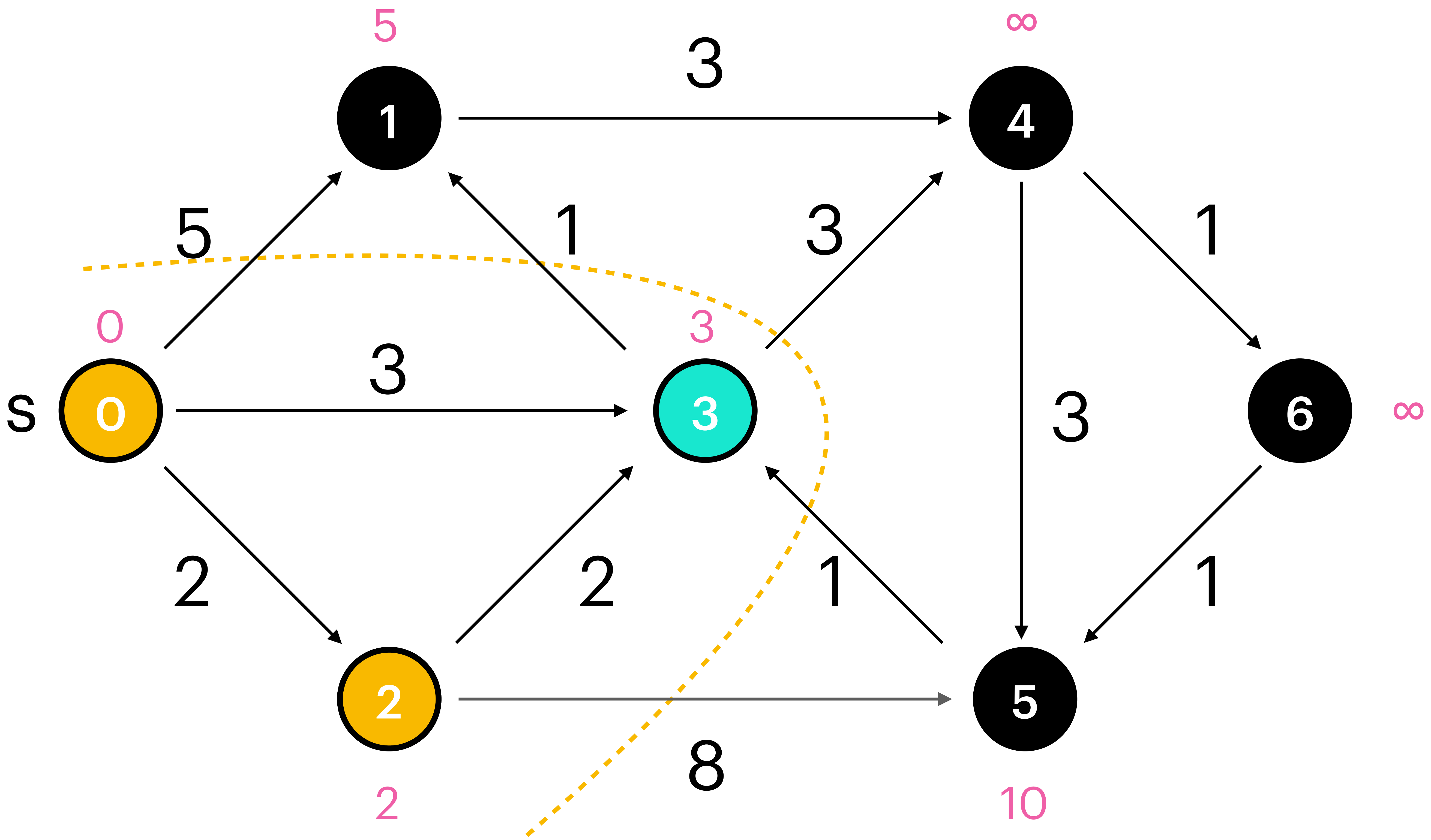
v* = 3

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	10	∞

"Heap" :

1	4	5	6
5	∞	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E, \ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 }

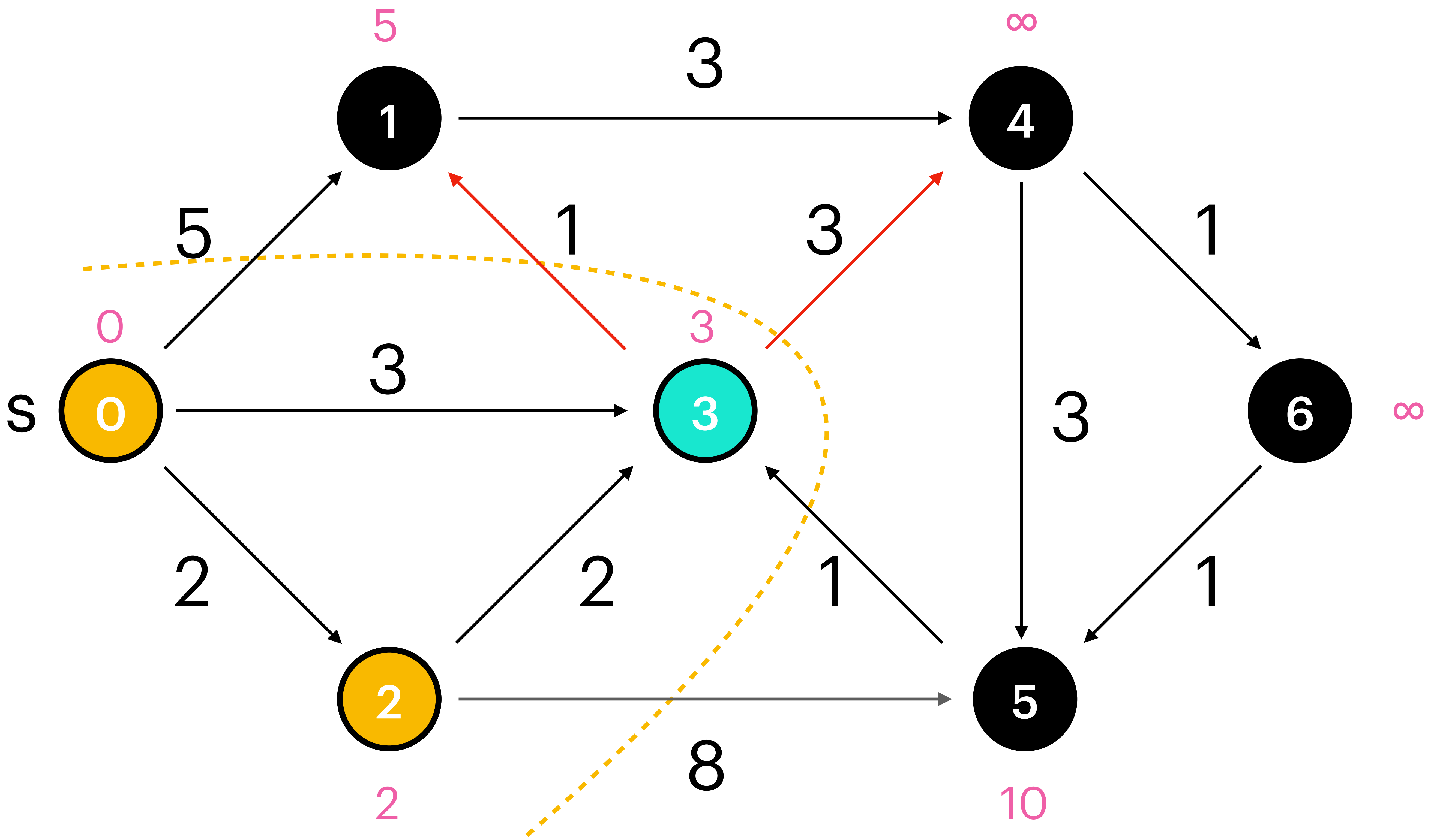
$v^* = 3$

d[] :

0	1	2	3	4	5	6
0	5	2	3	∞	10	∞

"Heap" :

1	4	5	6
5	∞	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $\text{d}[s] \leftarrow 0; \quad \text{d}[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $\quad v^* \leftarrow$ extract-min(H)
6: $\quad S \leftarrow S \cup \{v^*\}$
7: $\quad$ **for** $(v^*, v) \in E, \ v \notin S$ **do**
8: $\quad\quad \text{d}[v] \leftarrow \min\{\text{d}[v], \text{d}[v^*] + c(v^*, v)\}$
9: $\quad\quad$ decrease-key($H, v, \text{d}[v]$)

make-heap(V) :
Create a min heap of the vertices

extract-min(H) :
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :
Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 }

v* = 3

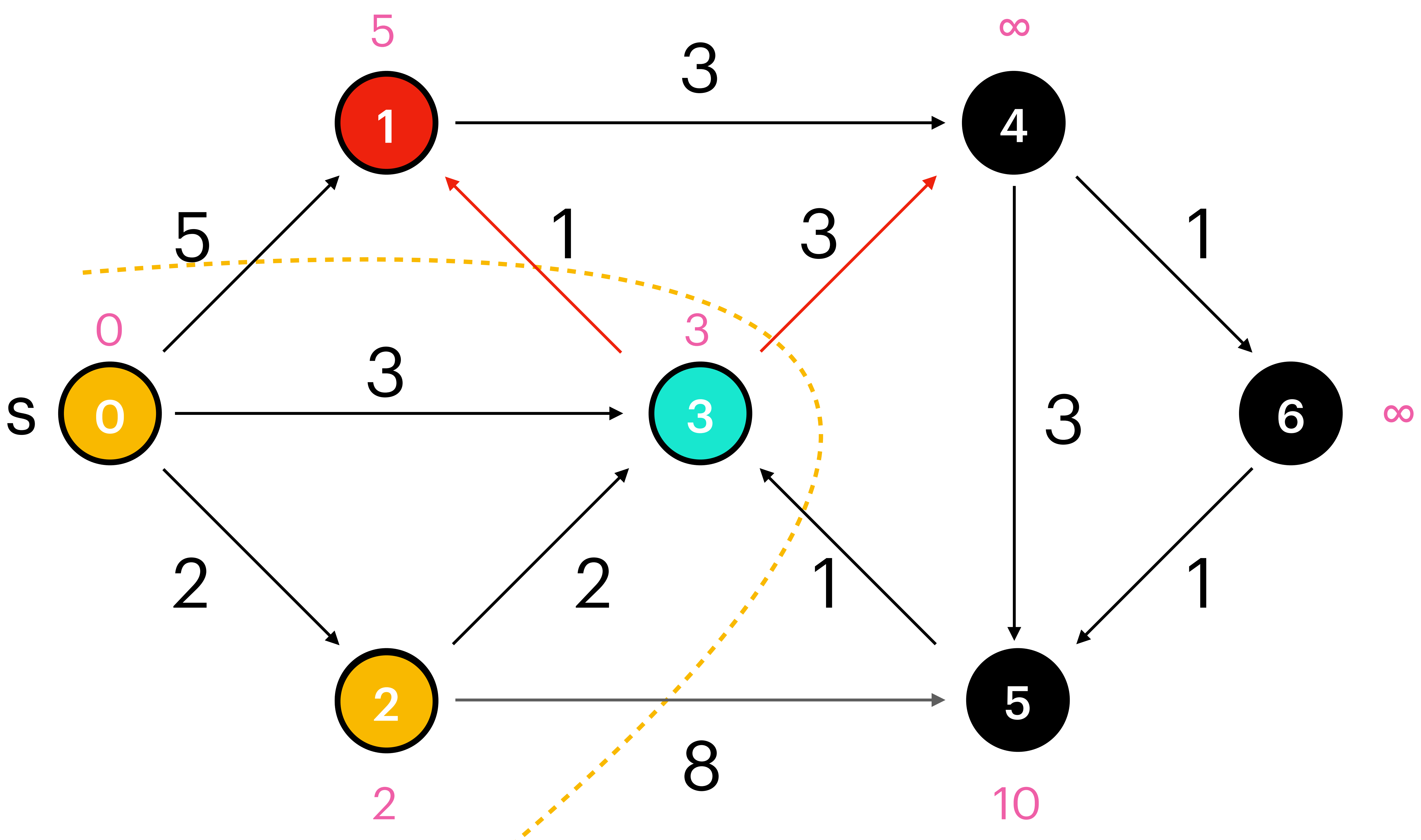
d[] :

0	1	2	3	4	5	6
0	4	2	3	∞	10	∞

"Heap" :

1	4	5	6
4	∞	10	∞

min {5 , 3 + 1} = 4



Shortest Paths

Dijkstra’s Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 }

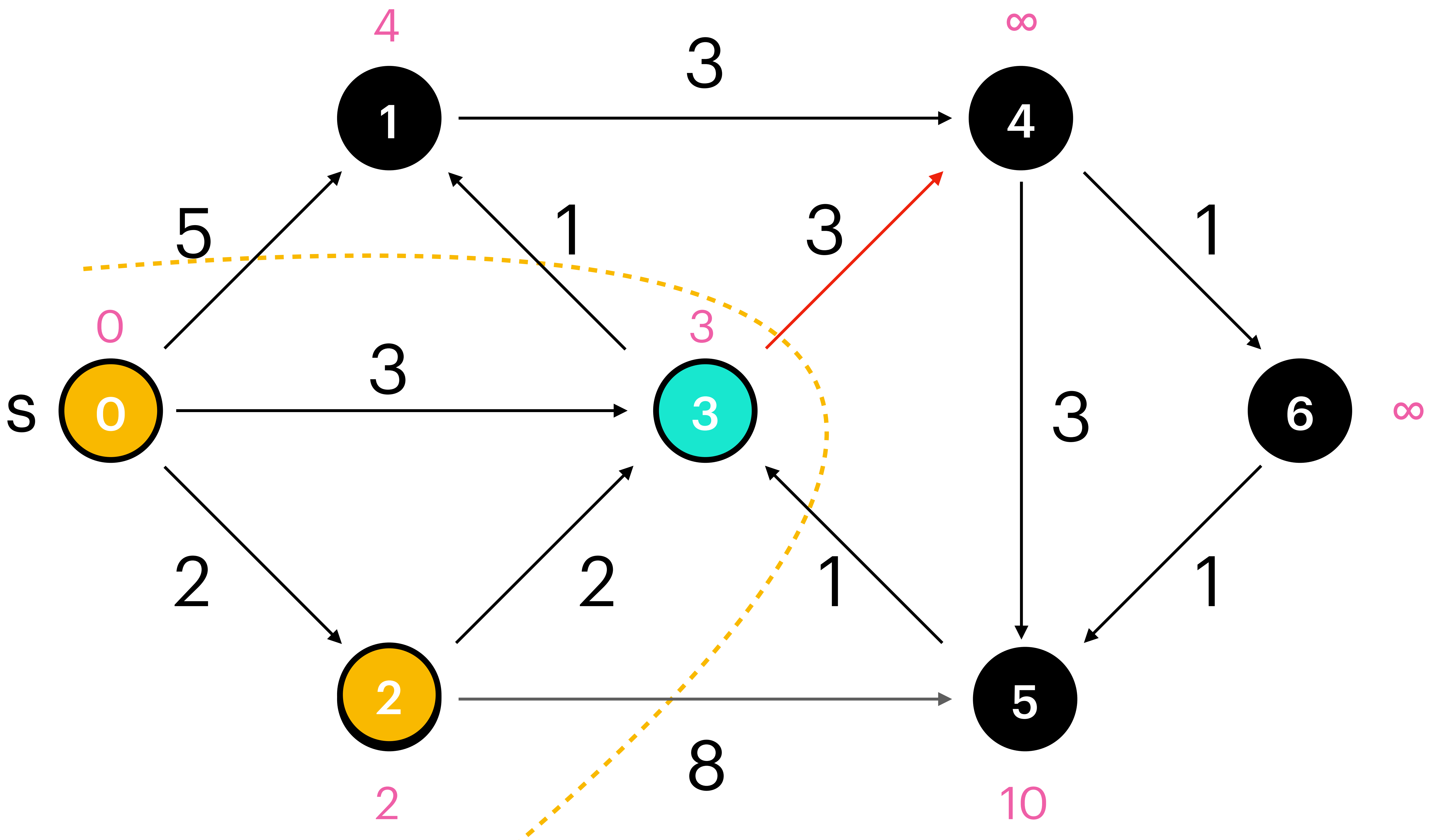
$v^* = 3$

d[] :

0	1	2	3	4	5	6
0	4	2	3	∞	10	∞

“Heap” :

1	4	5	6
4	∞	10	∞



Shortest Paths

Dijkstra’s Algorithm

Algorithm 6 Dijkstra(s)

$\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
    decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 }

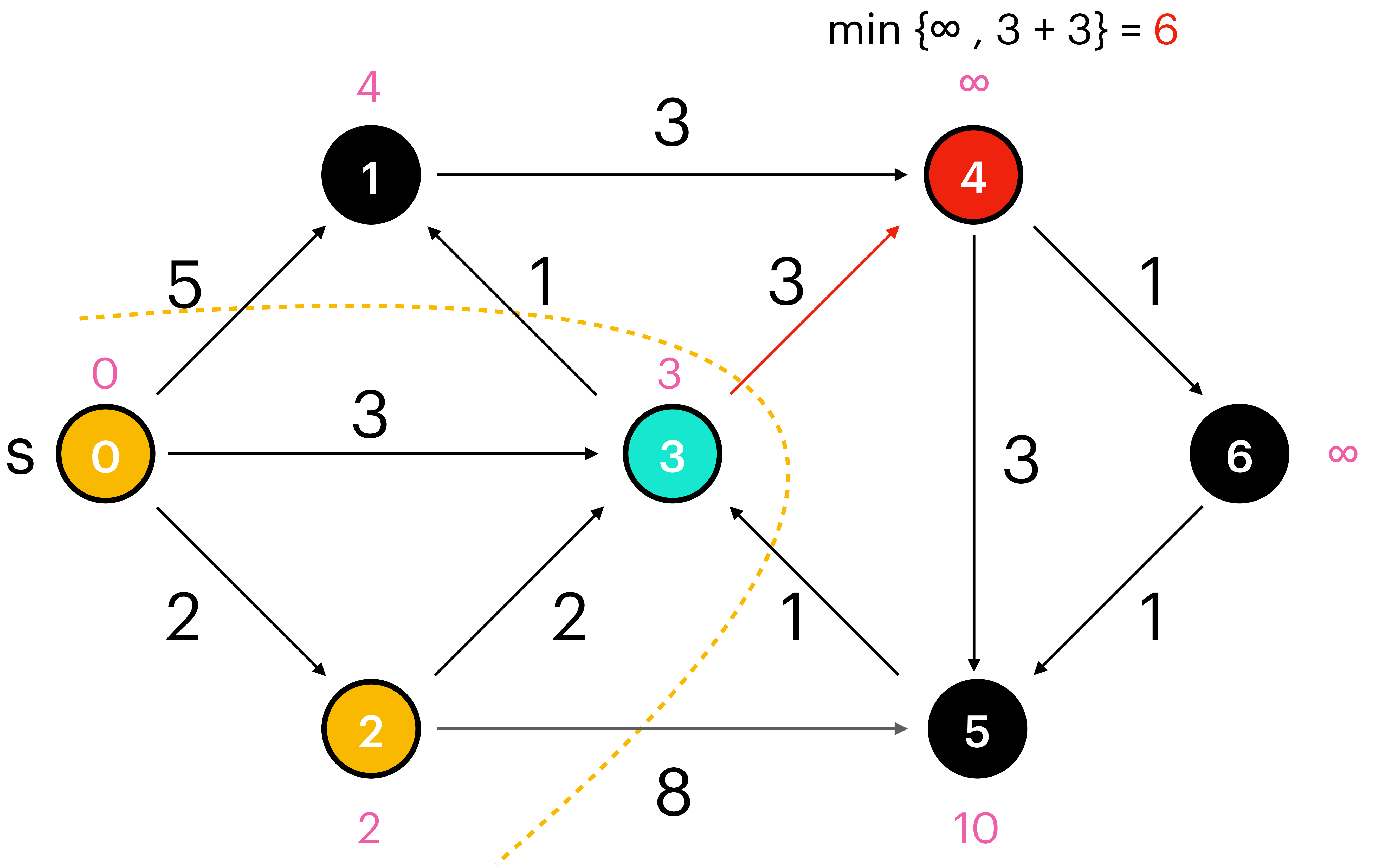
v* = 3

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

“Heap” :

1	4	5	6
4	6	10	∞



Shortest Paths

Dijkstra’s Algorithm

Algorithm 6 Dijkstra(s)

$\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\; v \notin S$ **do**
    $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
    decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

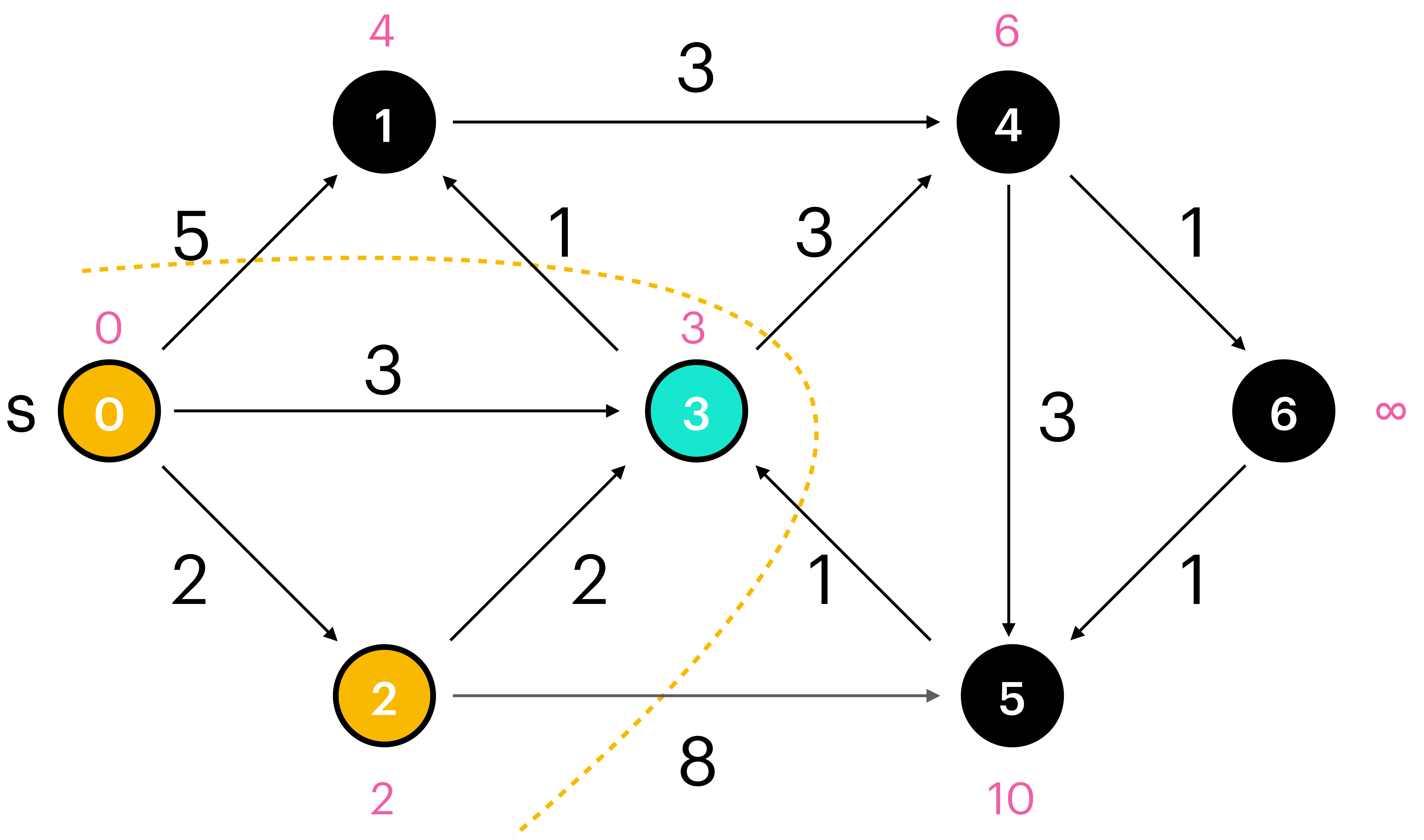
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 }

v* = 3



d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

“Heap” :

1	4	5	6
4	6	10	∞

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\; v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

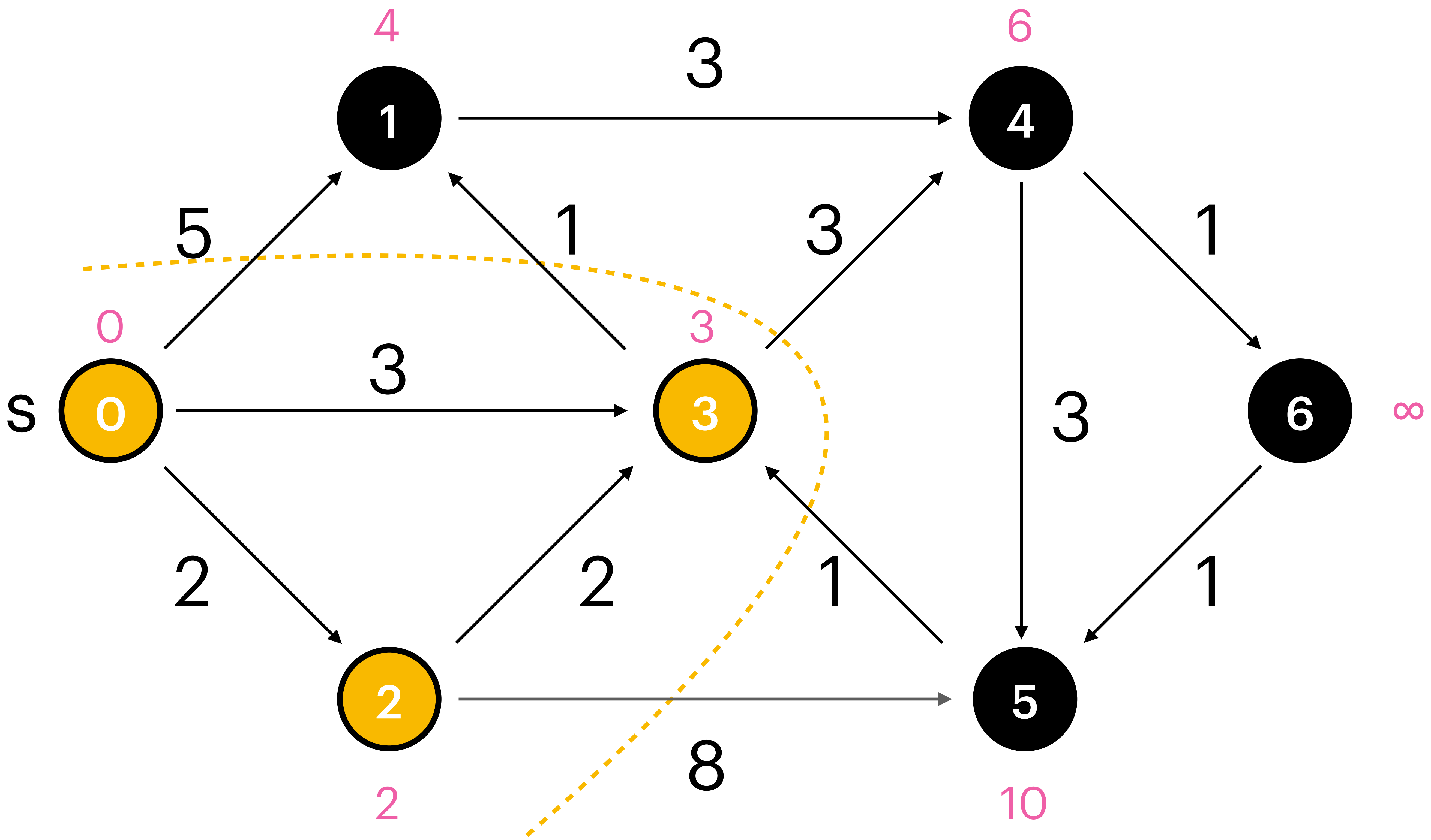
S : { 0 , 2 , 3 }

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

1	4	5	6
4	6	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

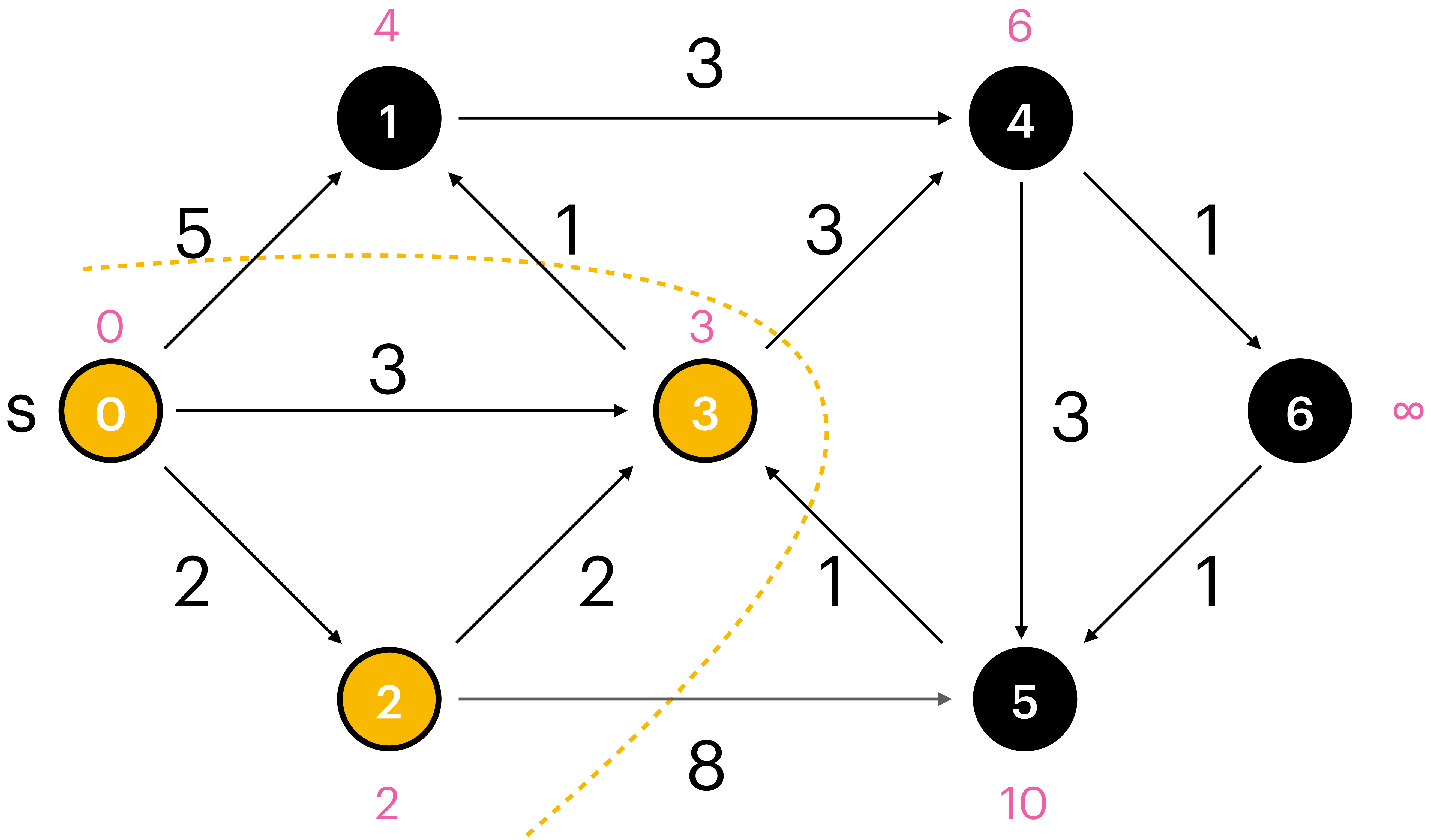
$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\; v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :
Create a min heap of the vertices

extract-min(H) :
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :
Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 }



d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

1	4	5	6
4	6	10	∞

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $d[s] \leftarrow 0$; $d[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $v^* \leftarrow$ extract-min(H)
6: $S \leftarrow S \cup \{v^*\}$
7: **for** $(v^*, v) \in E,\ v \notin S$ **do**
8: $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
9: decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 }

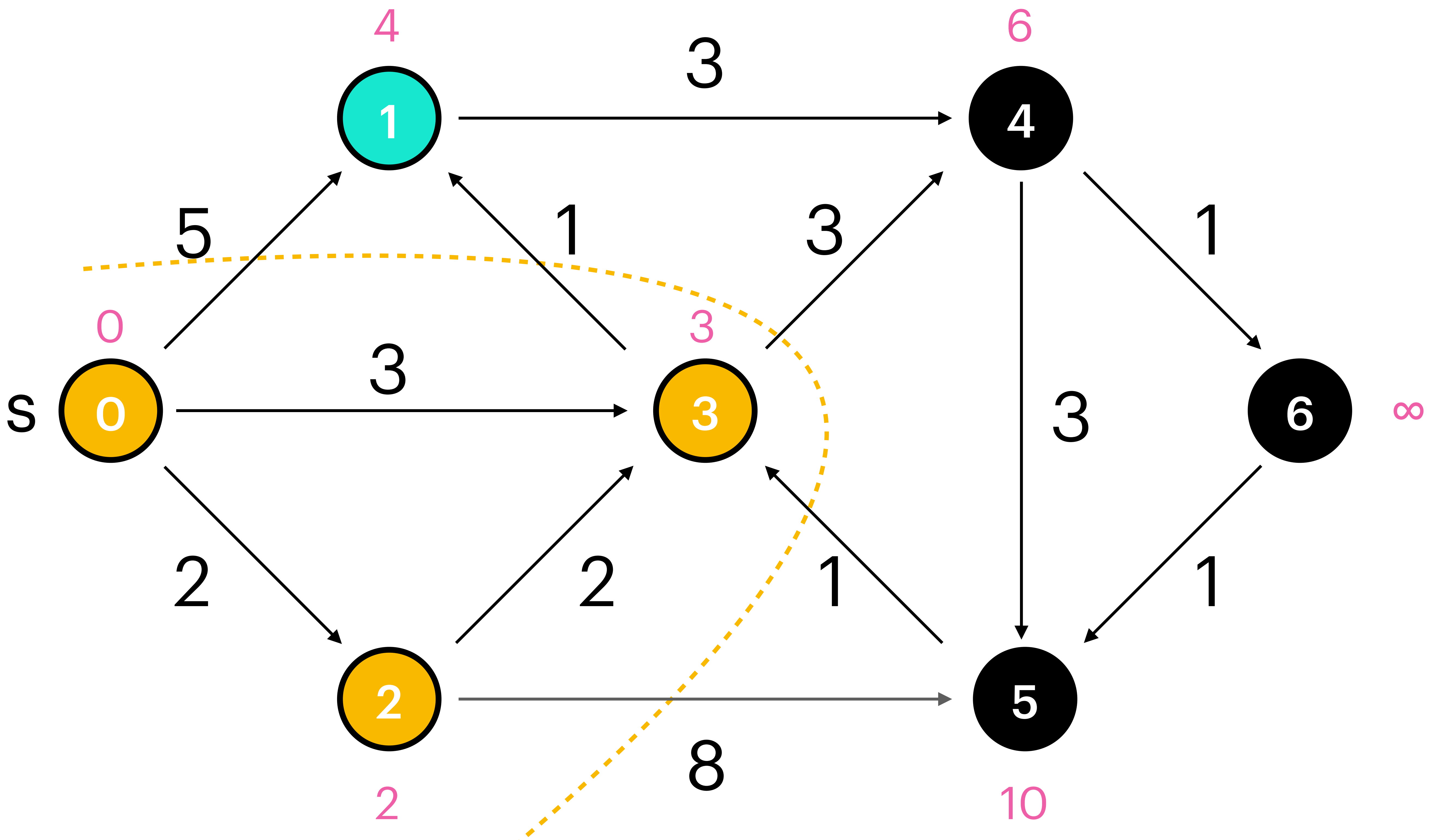
v* = 1

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

4	5	6
6	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
    decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 }

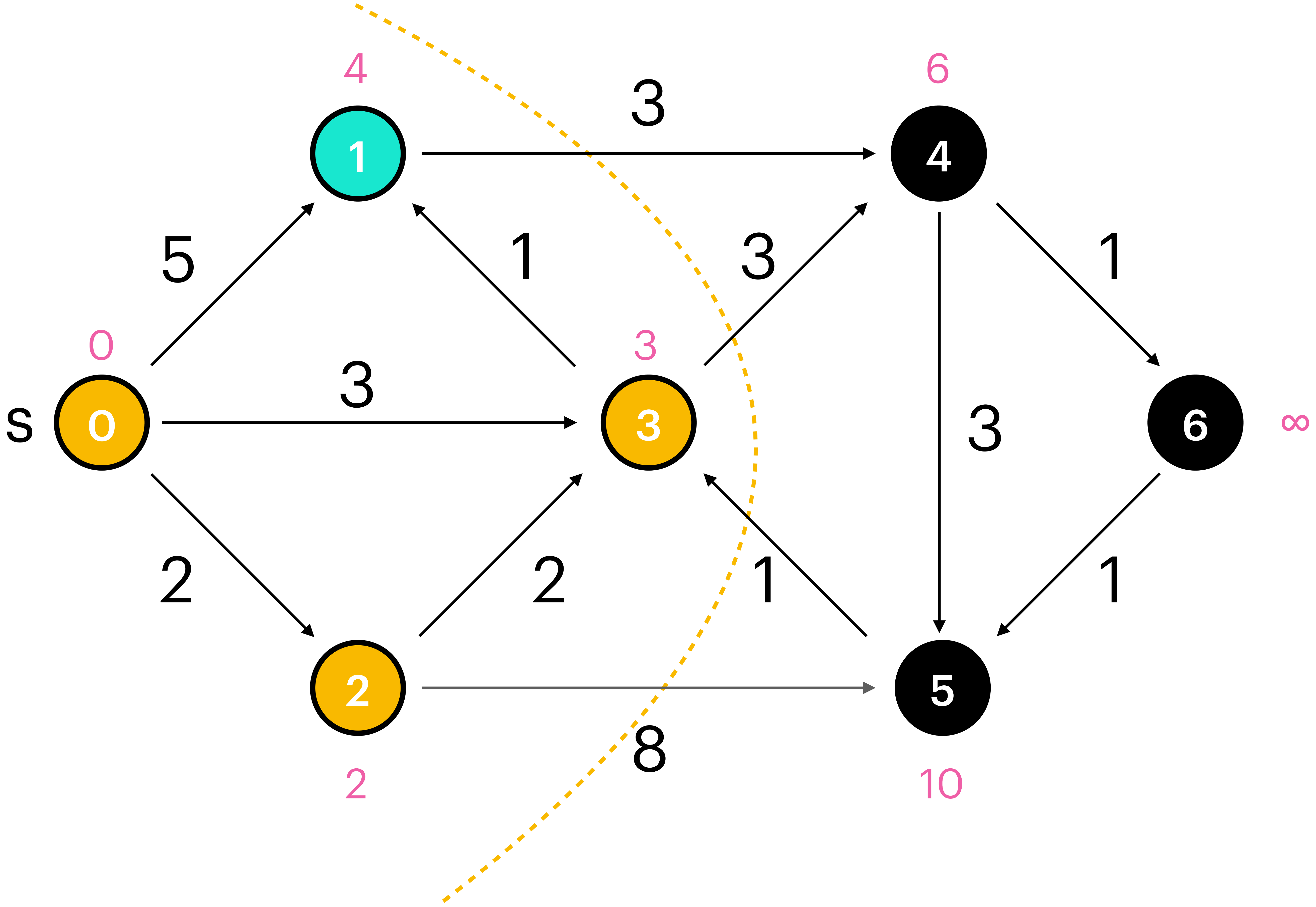
v* = 1

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

4	5	6
6	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

```
d[s] ← 0;   d[v] ← ∞  ∀v ∈ V \ {s}
S ← ∅
H ← make-heap(V); decrease-key(H, s, 0)
while S ≠ V do
    v* ← extract-min(H)
    S ← S ∪ {v*}
    for (v*, v) ∈ E, v ∉ S do
        d[v] ← min{d[v], d[v*] + c(v*, v)}
        decrease-key(H, v, d[v])
```

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 }

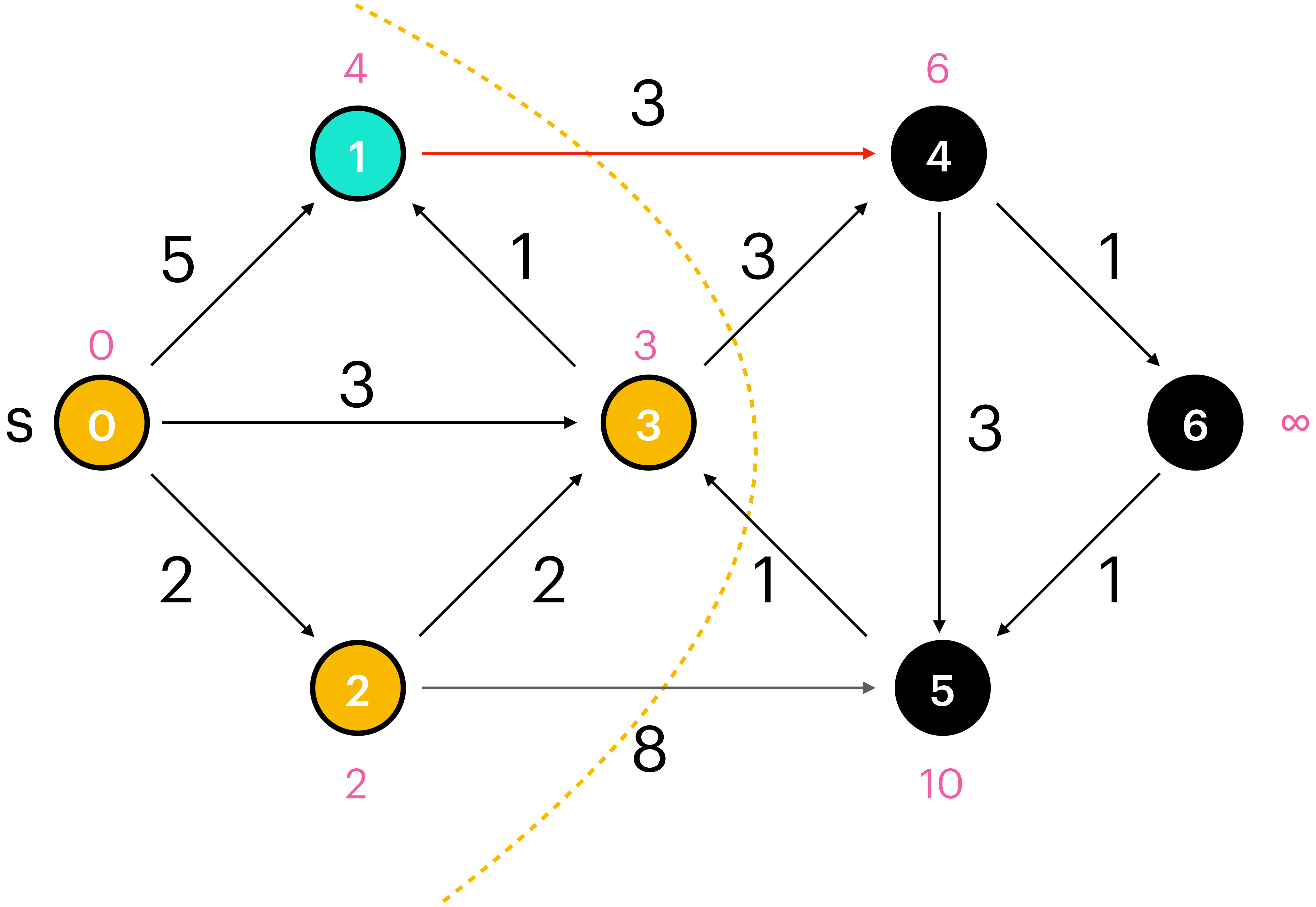
v* = 1

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

4	5	6
6	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

```
d[s] ← 0;   d[v] ← ∞  ∀v ∈ V \ {s}
S ← ∅
H ← make-heap(V); decrease-key(H, s, 0)
while S ≠ V do
    v* ← extract-min(H)
    S ← S ∪ {v*}
    for (v*, v) ∈ E, v ∉ S do
        d[v] ← min{d[v], d[v*] + c(v*, v)}
        decrease-key(H, v, d[v])
```

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 }

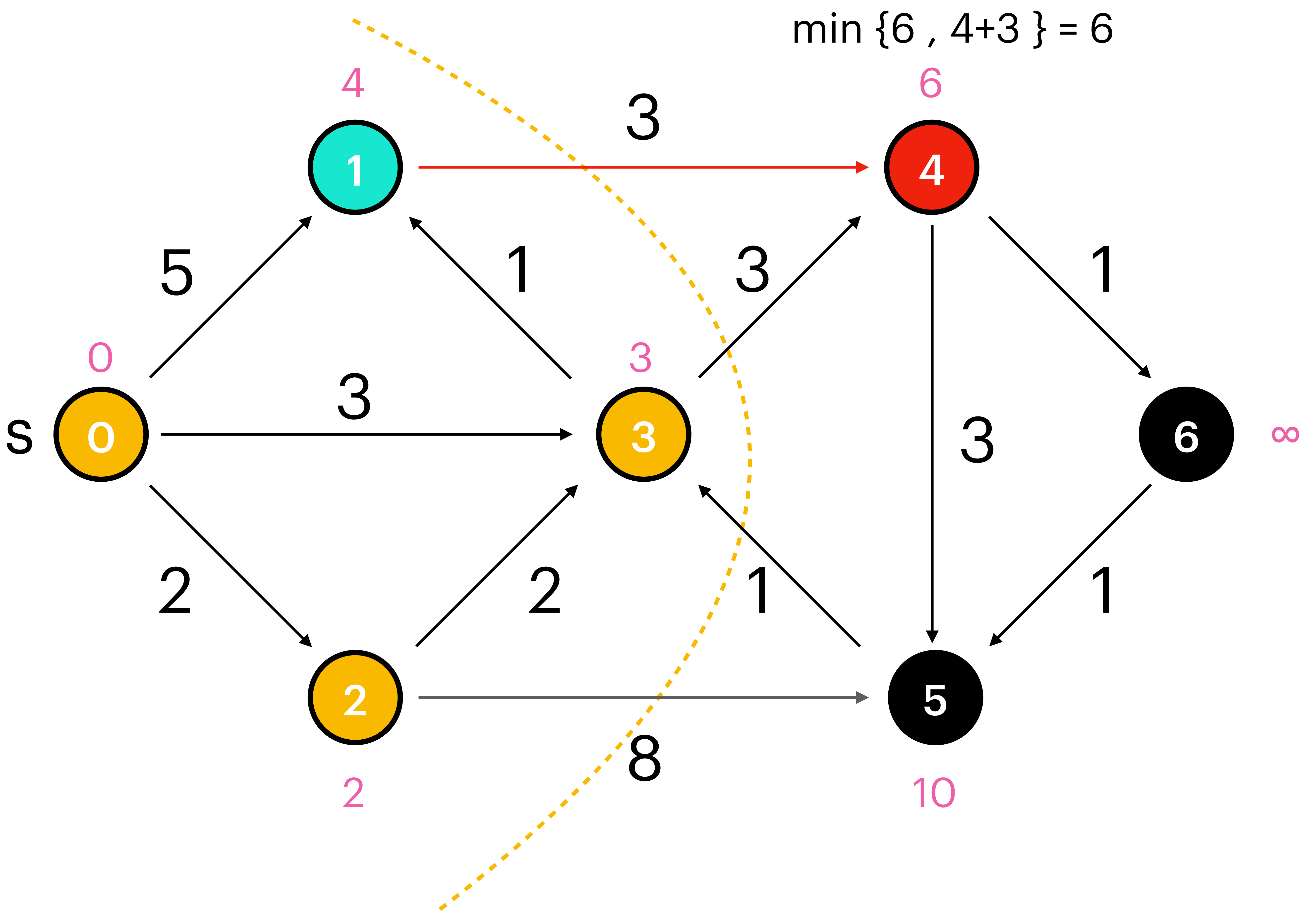
$v^* = 1$

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

4	5	6
6	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\; v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 }

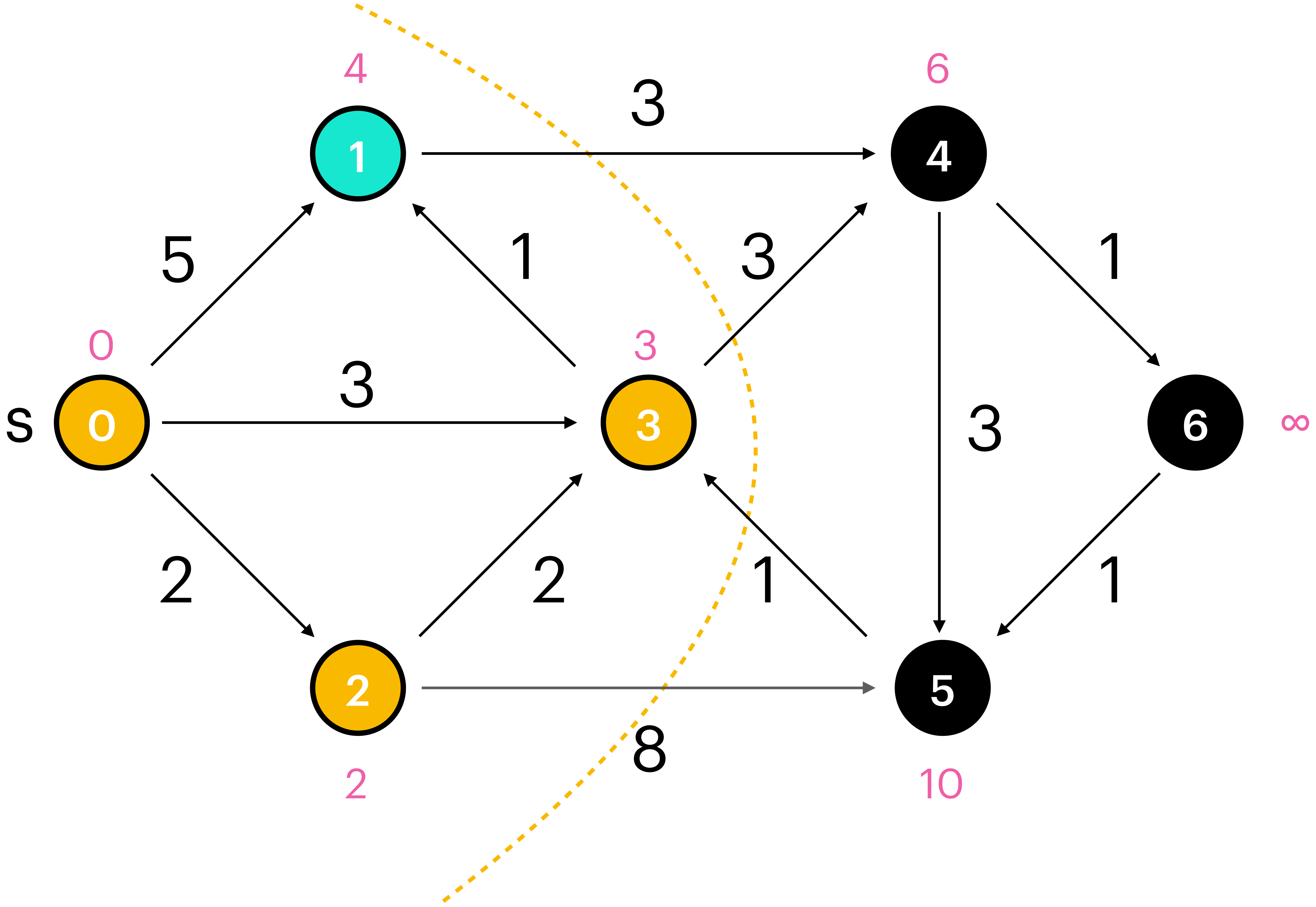
$v^* = 1$

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

4	5	6
6	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0$; $d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

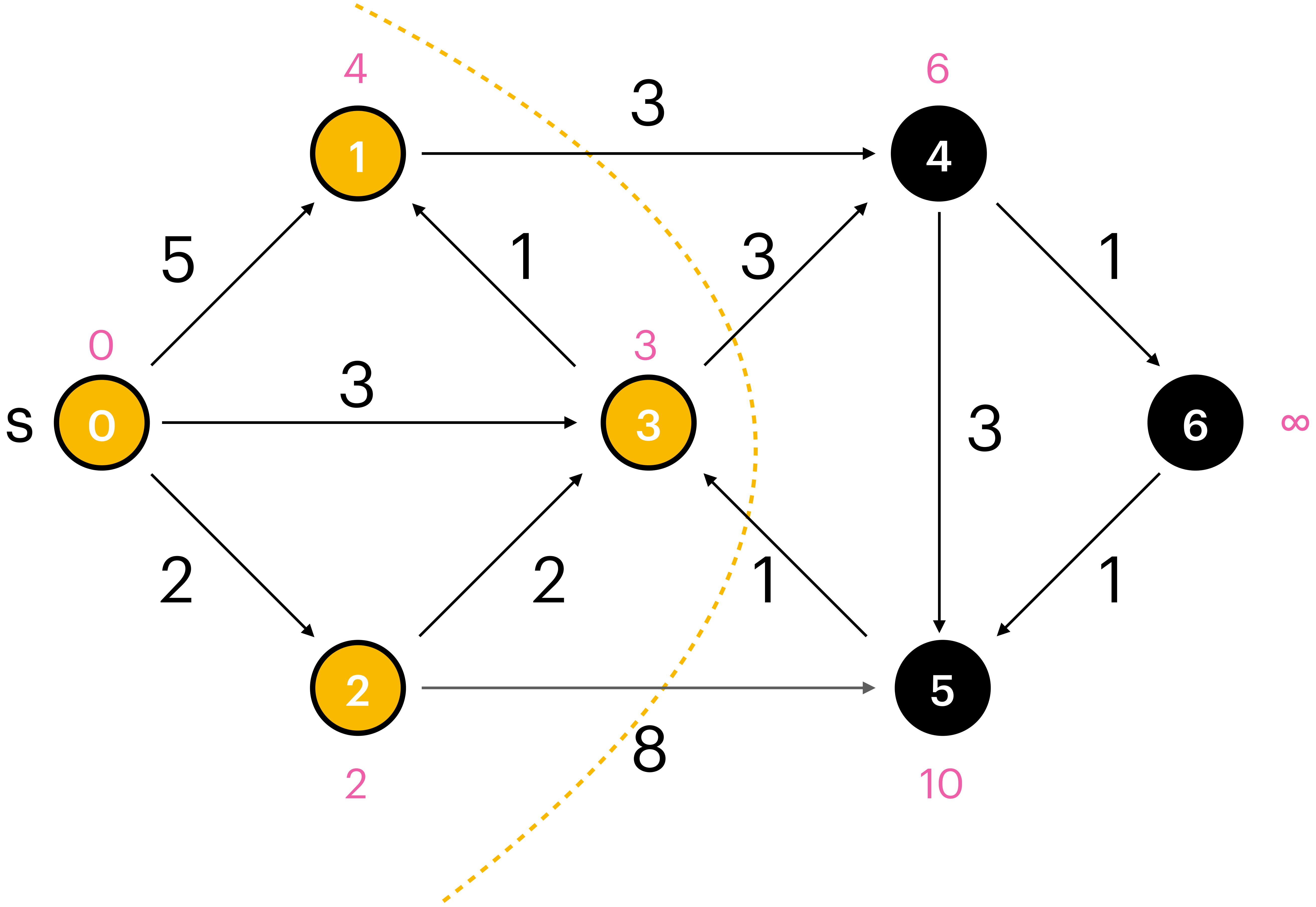
extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 }



d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

4	5	6
6	10	∞

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\; v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

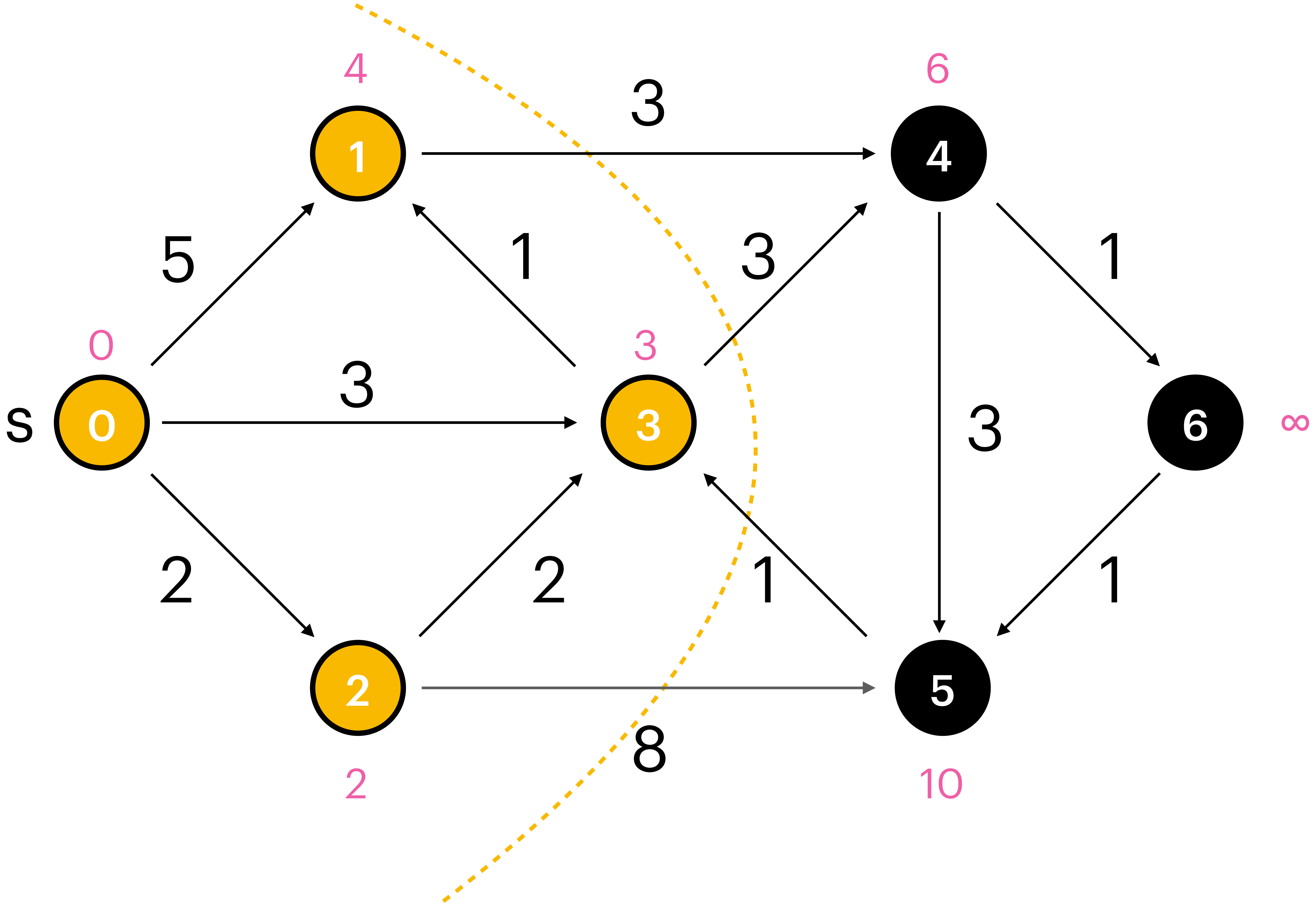
S : { 0 , 2 , 3 , 1 }

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

4	5	6
6	10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $\quad v^* \leftarrow$ extract-min(H)
6: $\quad S \leftarrow S \cup \{v^*\}$
7: $\quad$ **for** $(v^*, v) \in E,\; v \notin S$ **do**
8: $\quad\quad d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
9: $\quad\quad$ decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 }

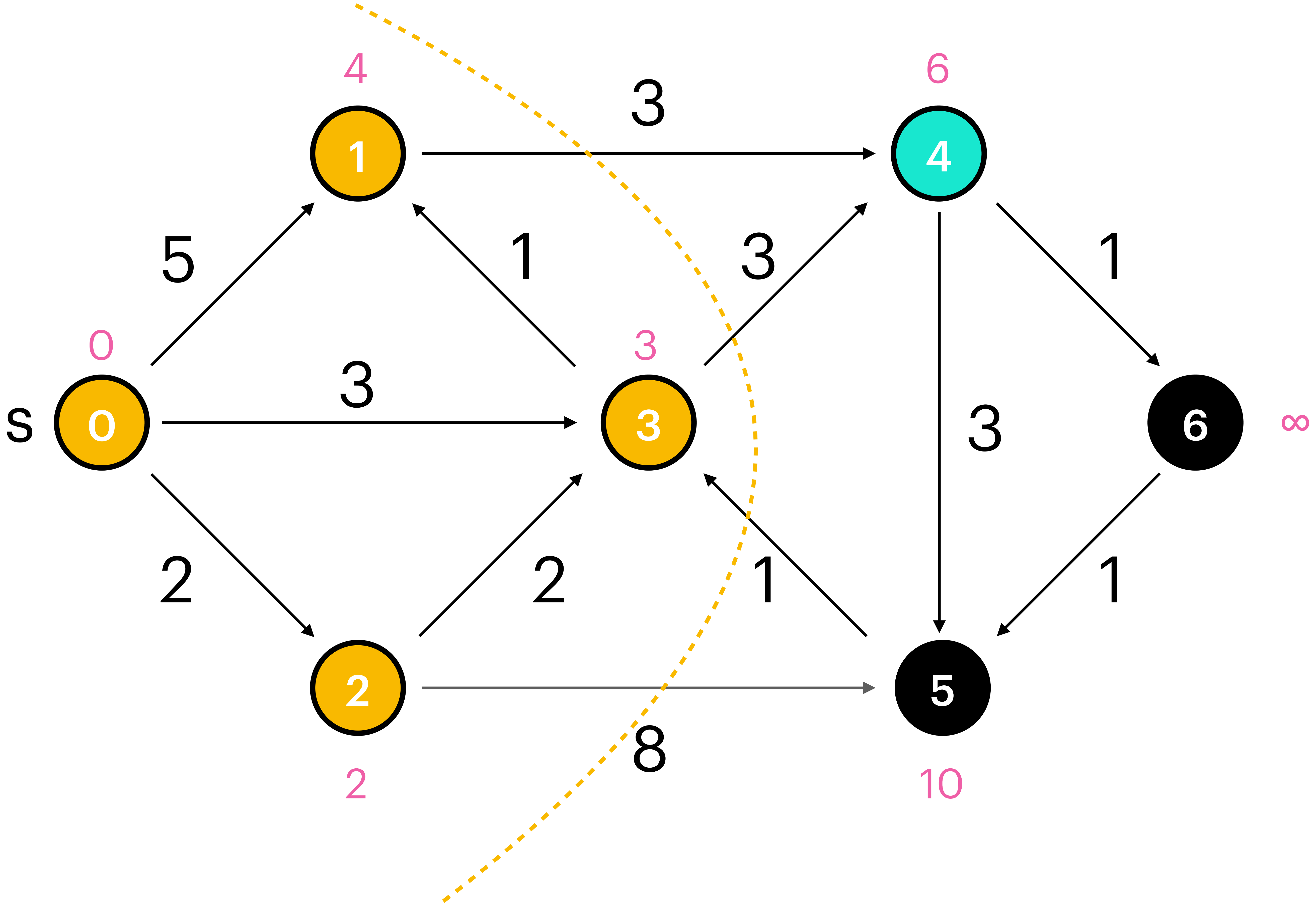
$v^* = 4$

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

5	6
10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

d[s] $\leftarrow 0$; d[v] $\leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    d[v] $\leftarrow$ min{d[v], d[v^*] + $c(v^*, v)$}
    decrease-key(H, v, d[v])

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

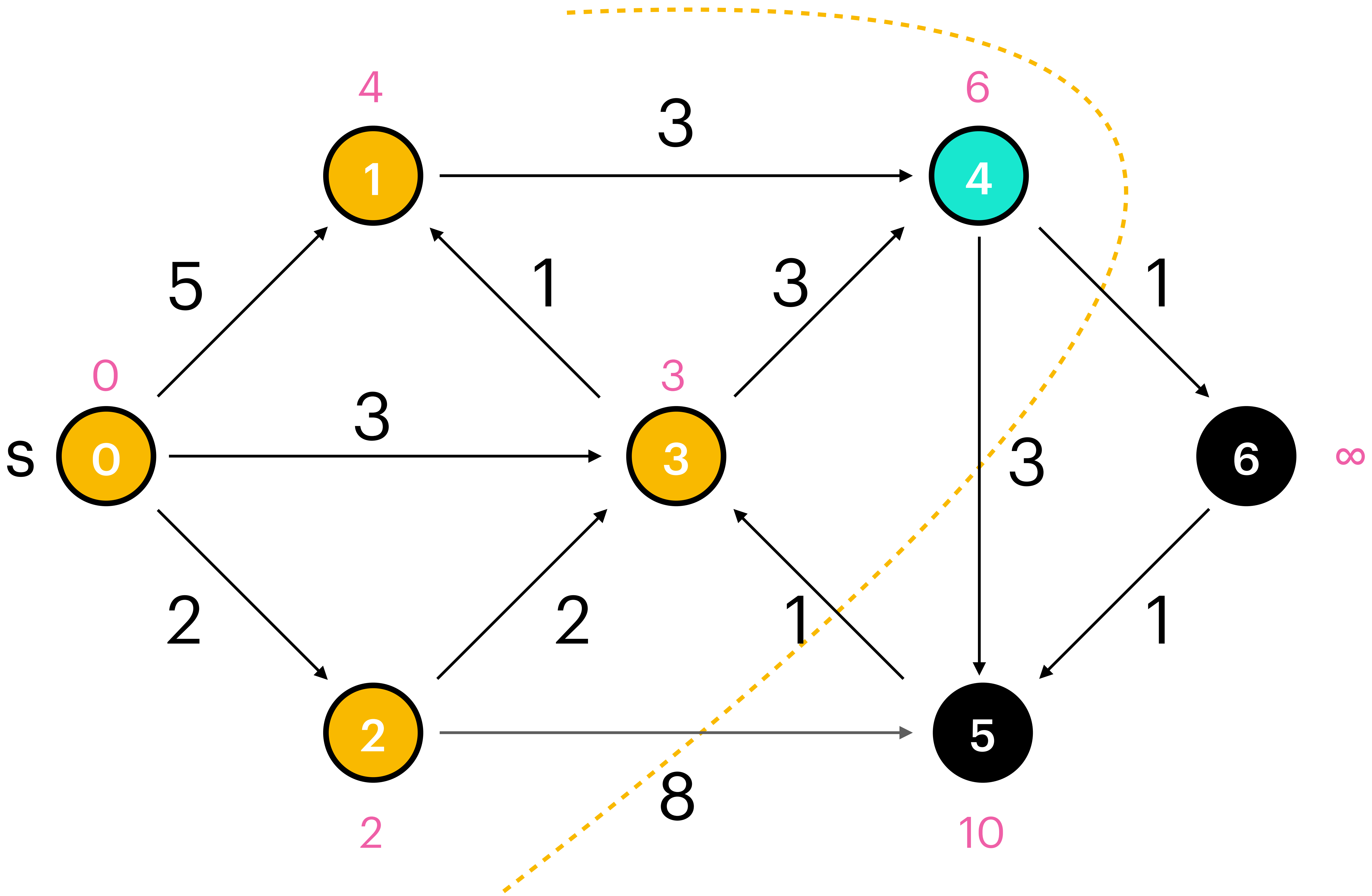
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 }

$v^* = 4$



d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

5	6
10	∞

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E, \ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 }

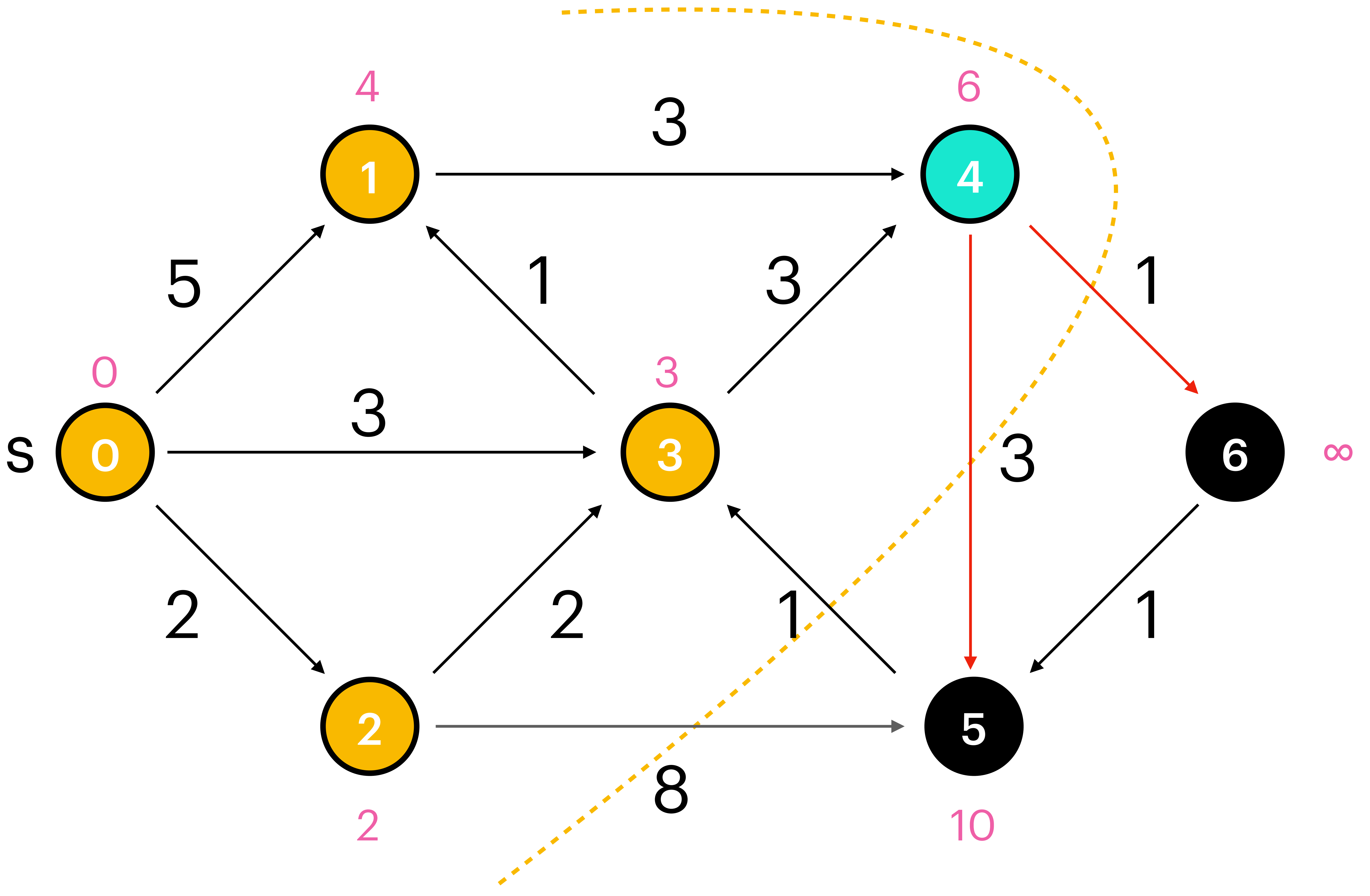
v* = 4

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	10	∞

"Heap" :

5	6
10	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 }

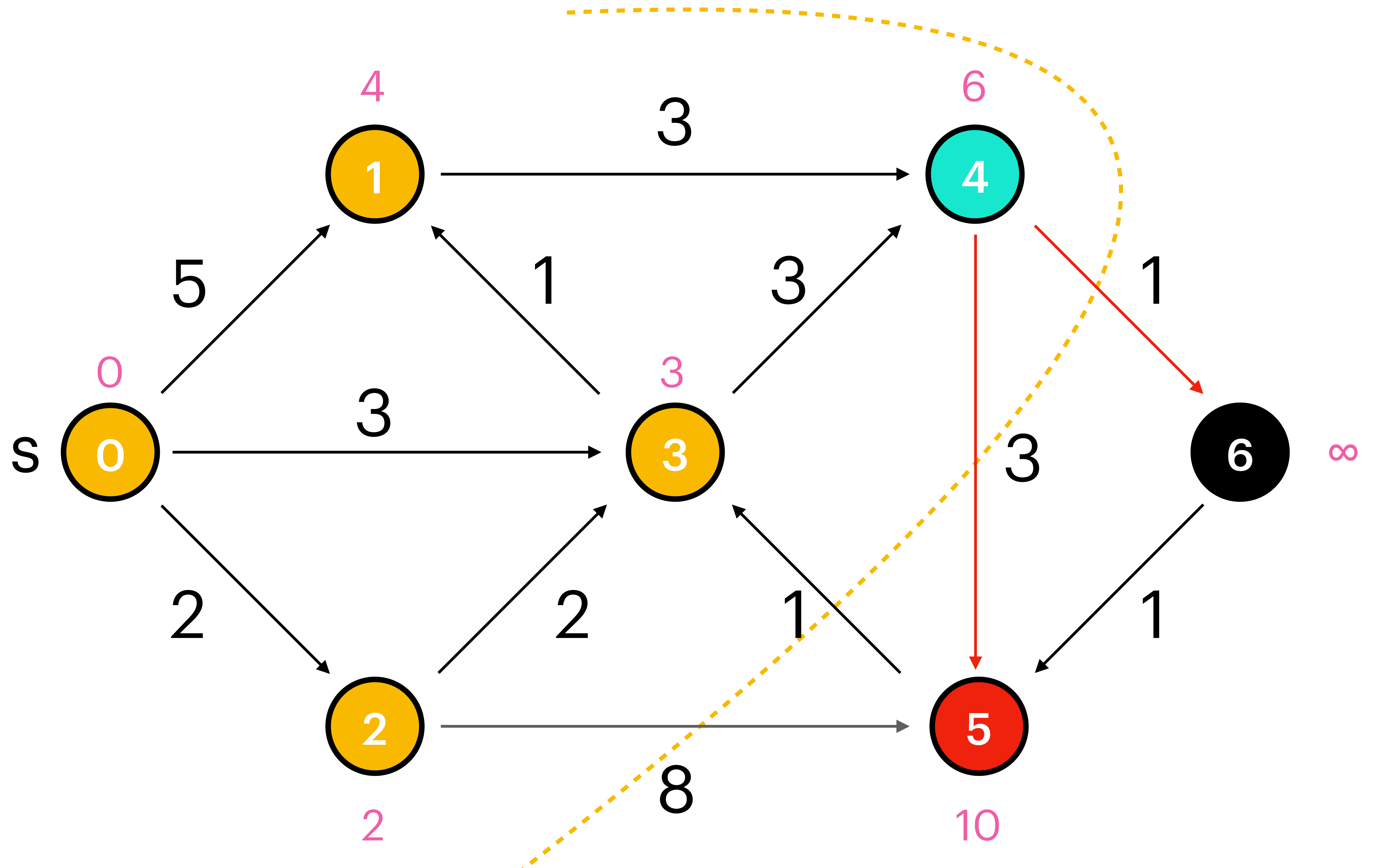
v* = 4

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	9	∞

"Heap" :

5	6
9	∞



min {10 , 6+3 } = 9

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 }

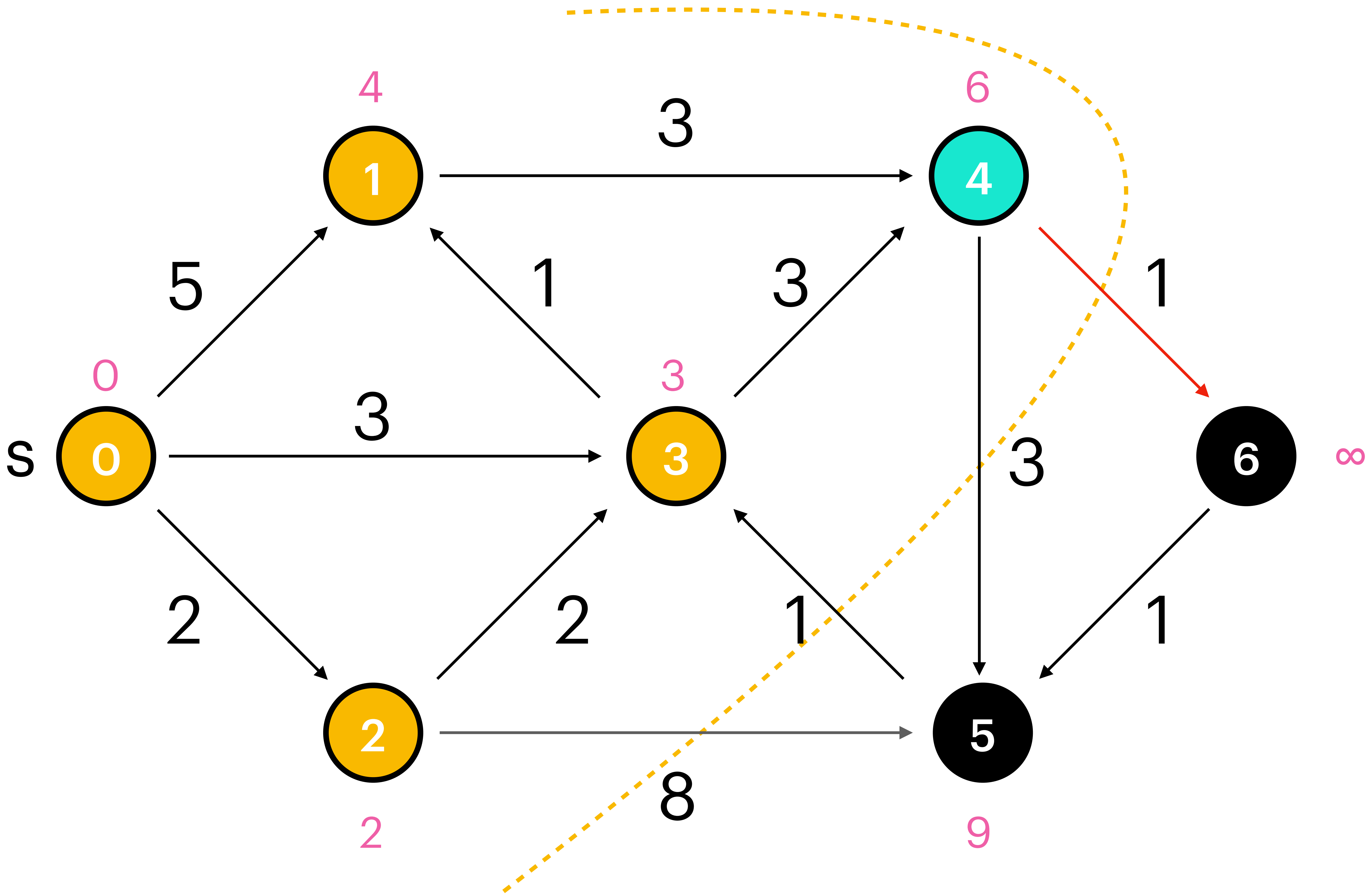
$v^* = 4$

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	9	∞

"Heap" :

5	6
9	∞



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

```
d[s] ← 0;   d[v] ← ∞  ∀v ∈ V \ {s}
S ← ∅
H ← make-heap(V); decrease-key(H, s, 0)
while S ≠ V do
    v* ← extract-min(H)
    S ← S ∪ {v*}
    for (v*, v) ∈ E, v ∉ S do
        d[v] ← min{d[v], d[v*] + c(v*, v)}
        decrease-key(H, v, d[v])
```

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 }

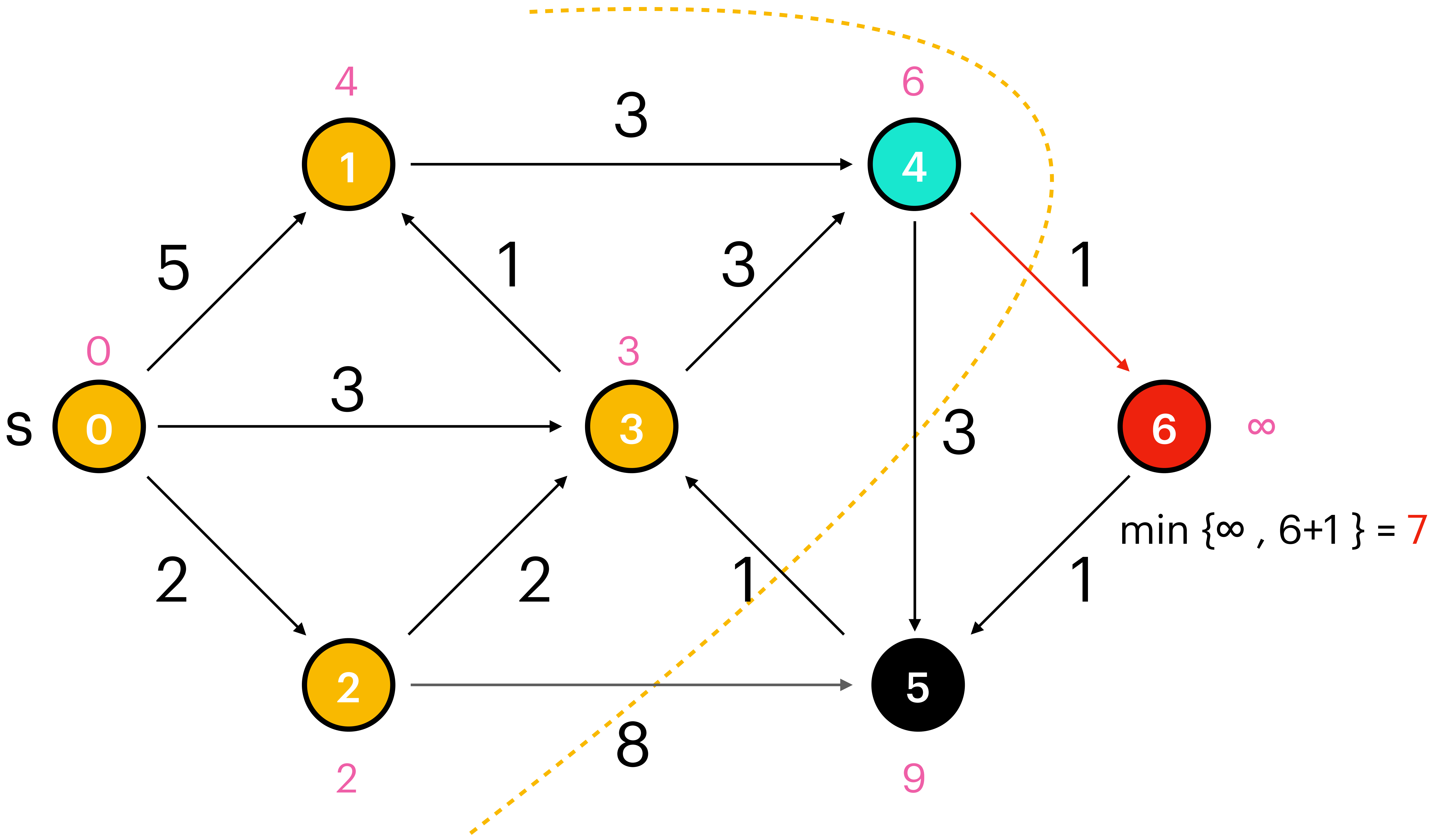
$v^* = 4$

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	9	7

"Heap" :

5	6
9	7



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $\quad v^* \leftarrow$ extract-min(H)
6: $\quad S \leftarrow S \cup \{v^*\}$
7: $\quad$ **for** $(v^*, v) \in E,\ v \notin S$ **do**
8: $\quad\quad \mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
9: $\quad\quad$ decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 }

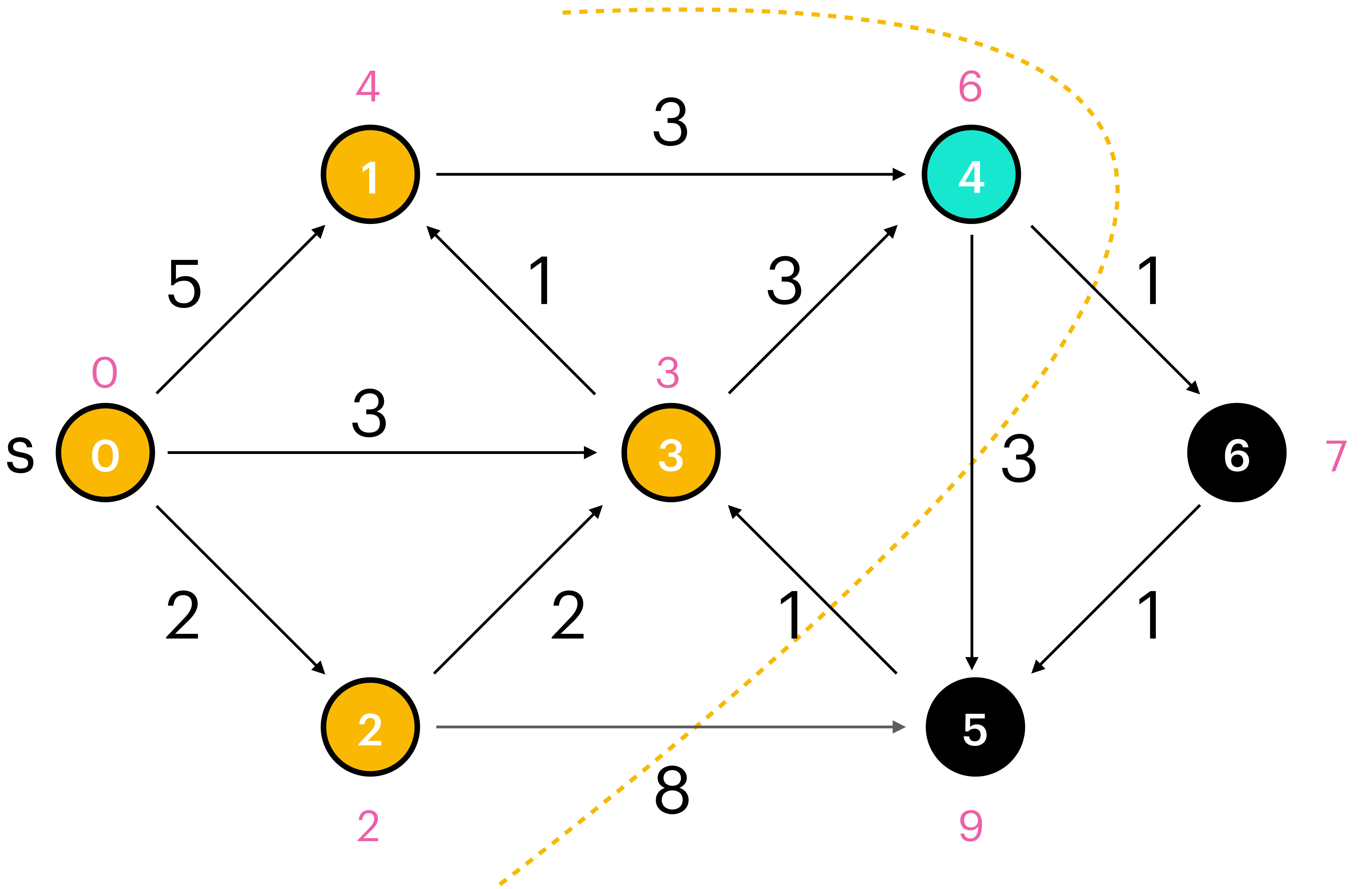
v* = 4

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	9	7

"Heap" :

5	6
9	7



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

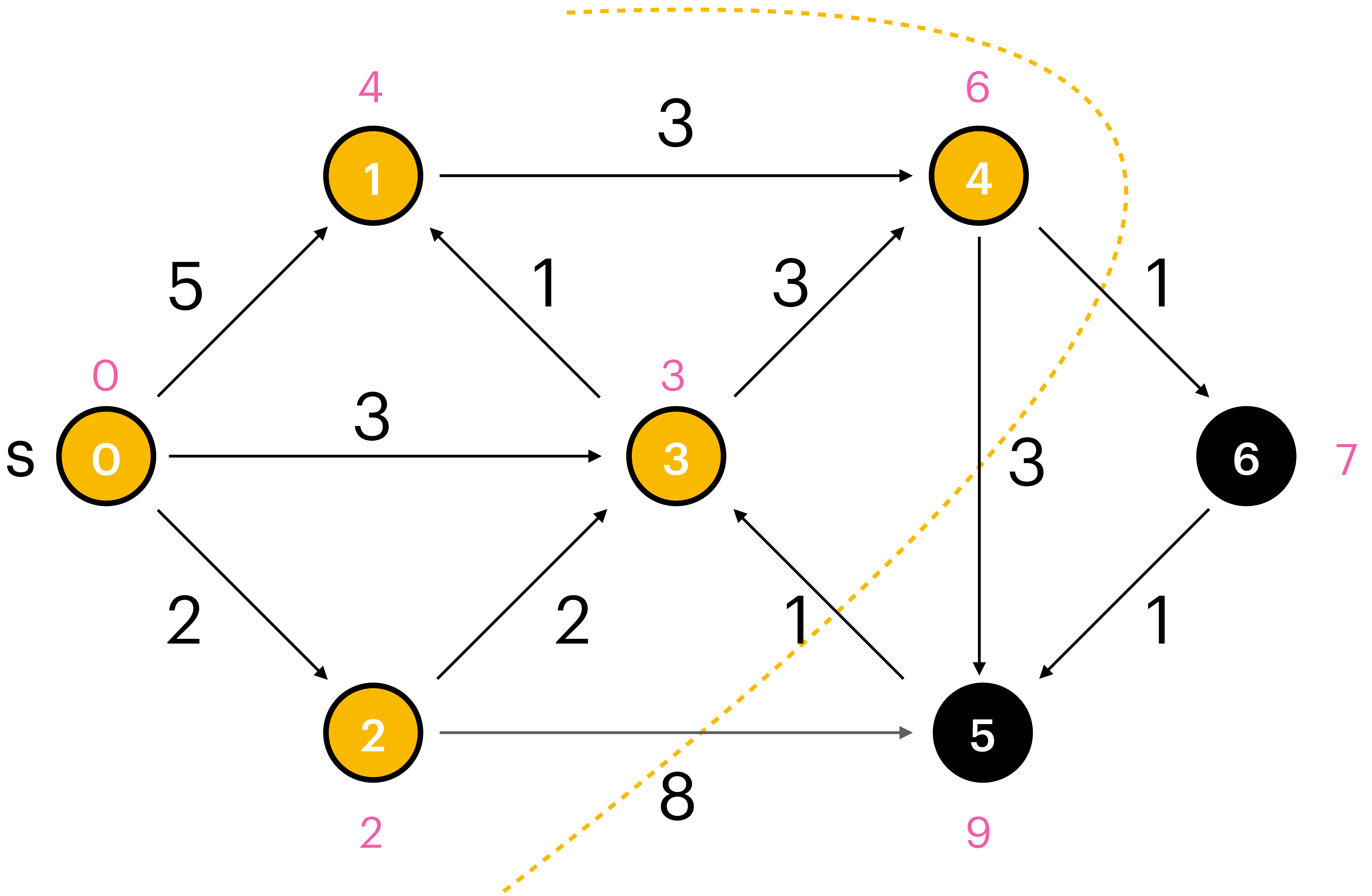
1: $d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $\quad v^* \leftarrow$ extract-min(H)
6: $\quad S \leftarrow S \cup \{v^*\}$
7: $\quad$ **for** $(v^*, v) \in E,\ v \notin S$ **do**
8: $\quad\quad d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
9: $\quad\quad$ decrease-key($H, v, d[v]$)

make-heap(V) :
Create a min heap of the vertices

extract-min(H) :
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :
Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 }



d[] :

0	1	2	3	4	5	6
0	4	2	3	6	9	7

"Heap" :

5	6
9	7

Shortest Paths
Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

```
1: d[s] ← 0;   d[v] ← ∞  ∀v ∈ V \ {s}
2: S ← ∅
3: H ← make-heap(V); decrease-key(H, s, 0)
4: while S ≠ V do
5:     v* ← extract-min(H)
6:     S ← S ∪ {v*}
7:     for (v*, v) ∈ E, v ∉ S do
8:         d[v] ← min{d[v], d[v*] + c(v*, v)}
9:         decrease-key(H, v, d[v])
```

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

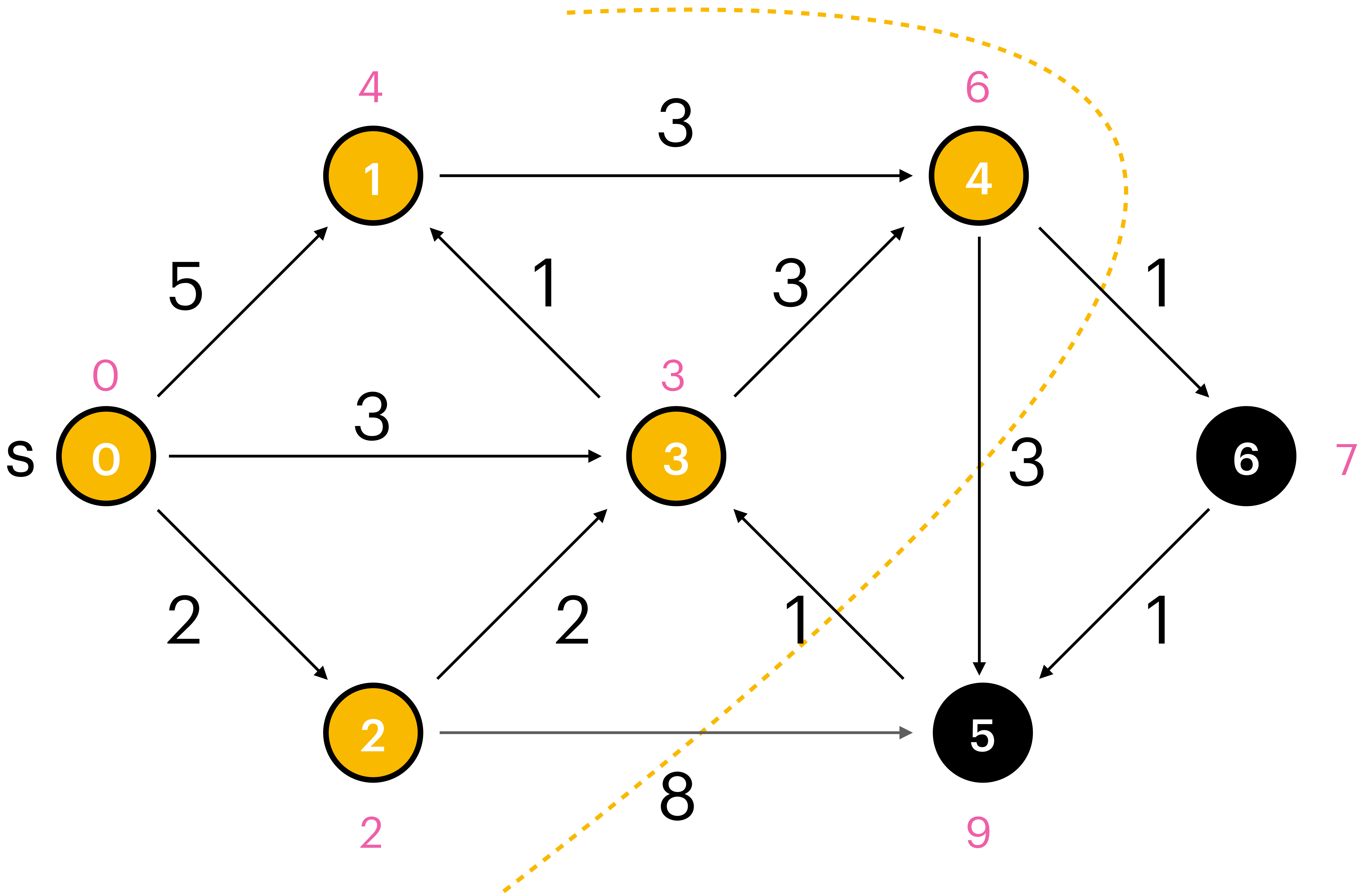
S : { 0 , 2 , 3 , 1 , 4 }

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	9	7

"Heap" :

5	6
9	7



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $\quad v^* \leftarrow$ extract-min(H)
6: $\quad S \leftarrow S \cup \{v^*\}$
7: $\quad$ **for** $(v^*, v) \in E,\; v \notin S$ **do**
8: $\quad\quad \mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
9: $\quad\quad$ decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 }

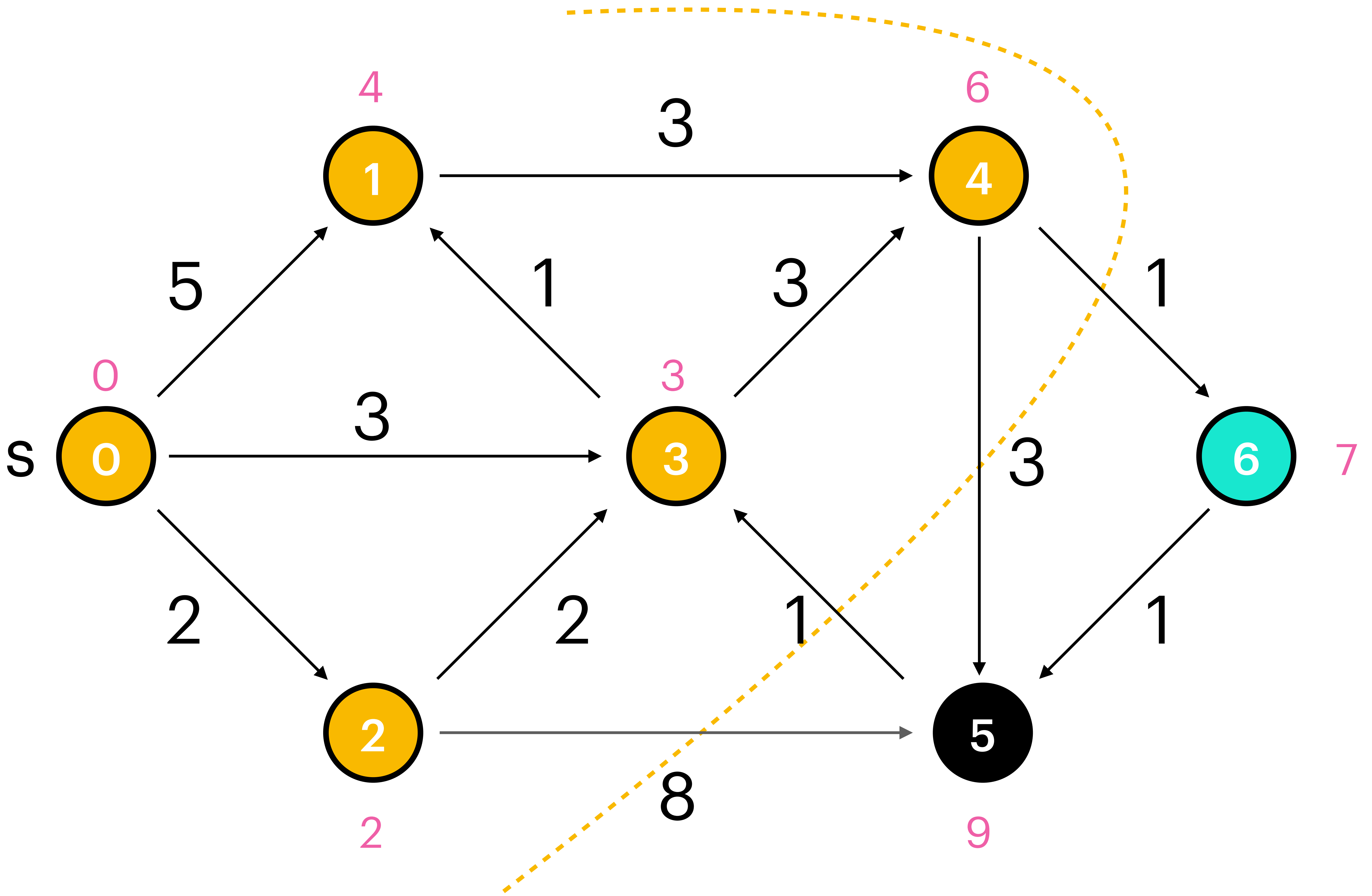
v* = 6

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	9	7

“Heap” :

5
9



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

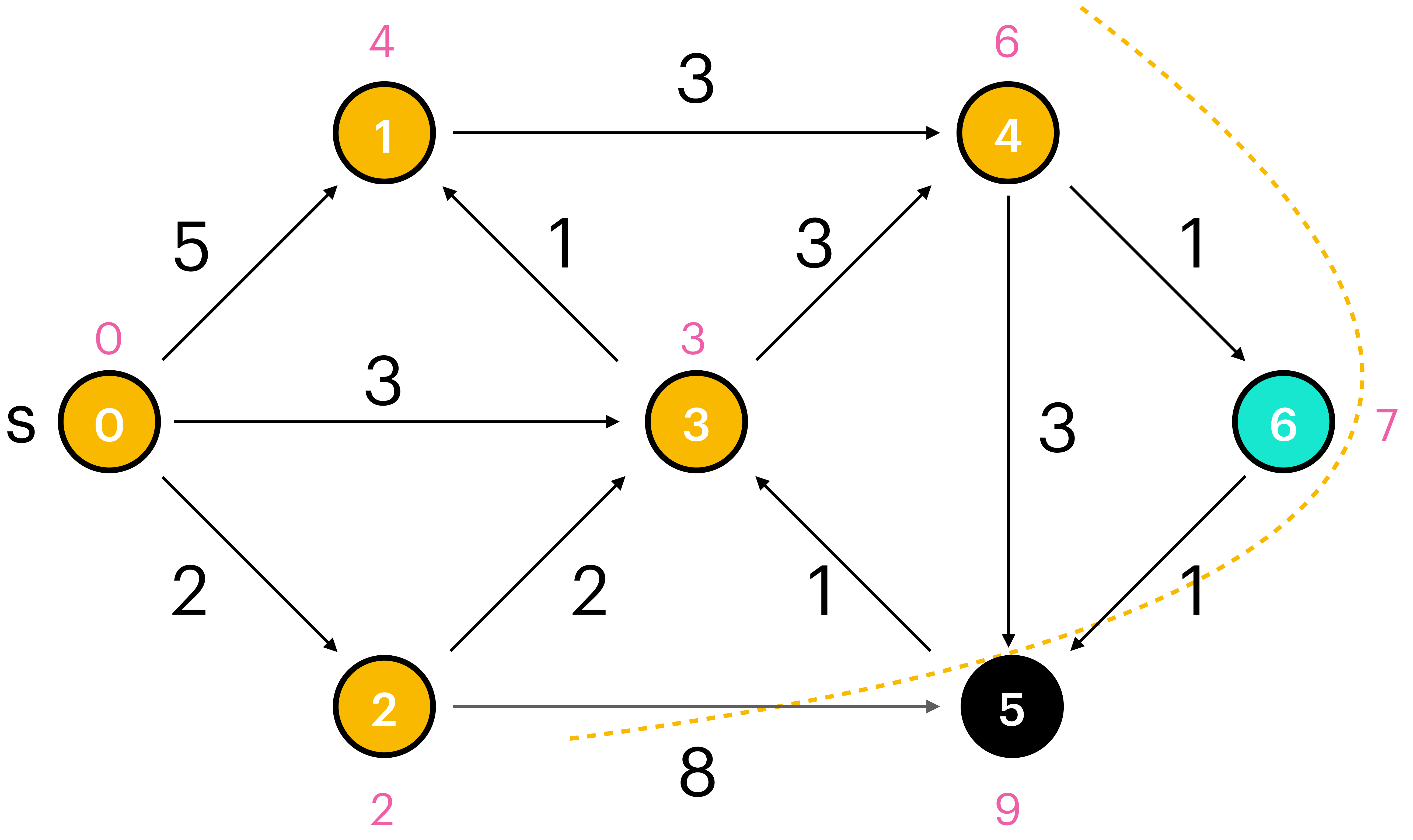
Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 , 6 }

v* = 6



d[] :

0	1	2	3	4	5	6
0	4	2	3	6	9	7

"Heap" :

5
9

Shortest Paths

Dijkstra’s Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 , 6 }

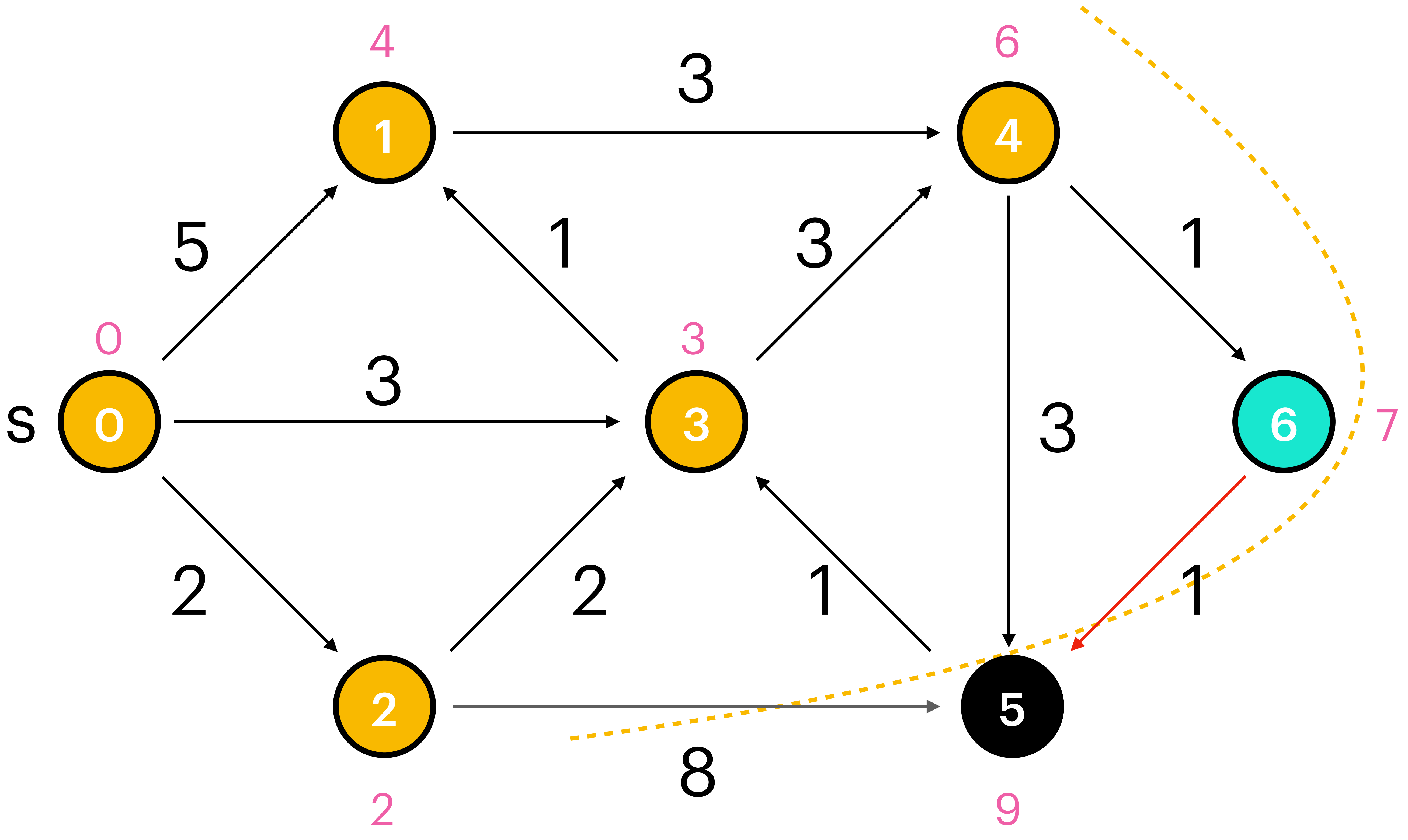
v* = 6

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	9	7

“Heap” :

5
9



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $\quad v^* \leftarrow$ extract-min(H)
6: $\quad S \leftarrow S \cup \{v^*\}$
7: $\quad$ **for** $(v^*, v) \in E,\ v \notin S$ **do**
8: $\quad\quad d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
9: $\quad\quad$ decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 , 6 }

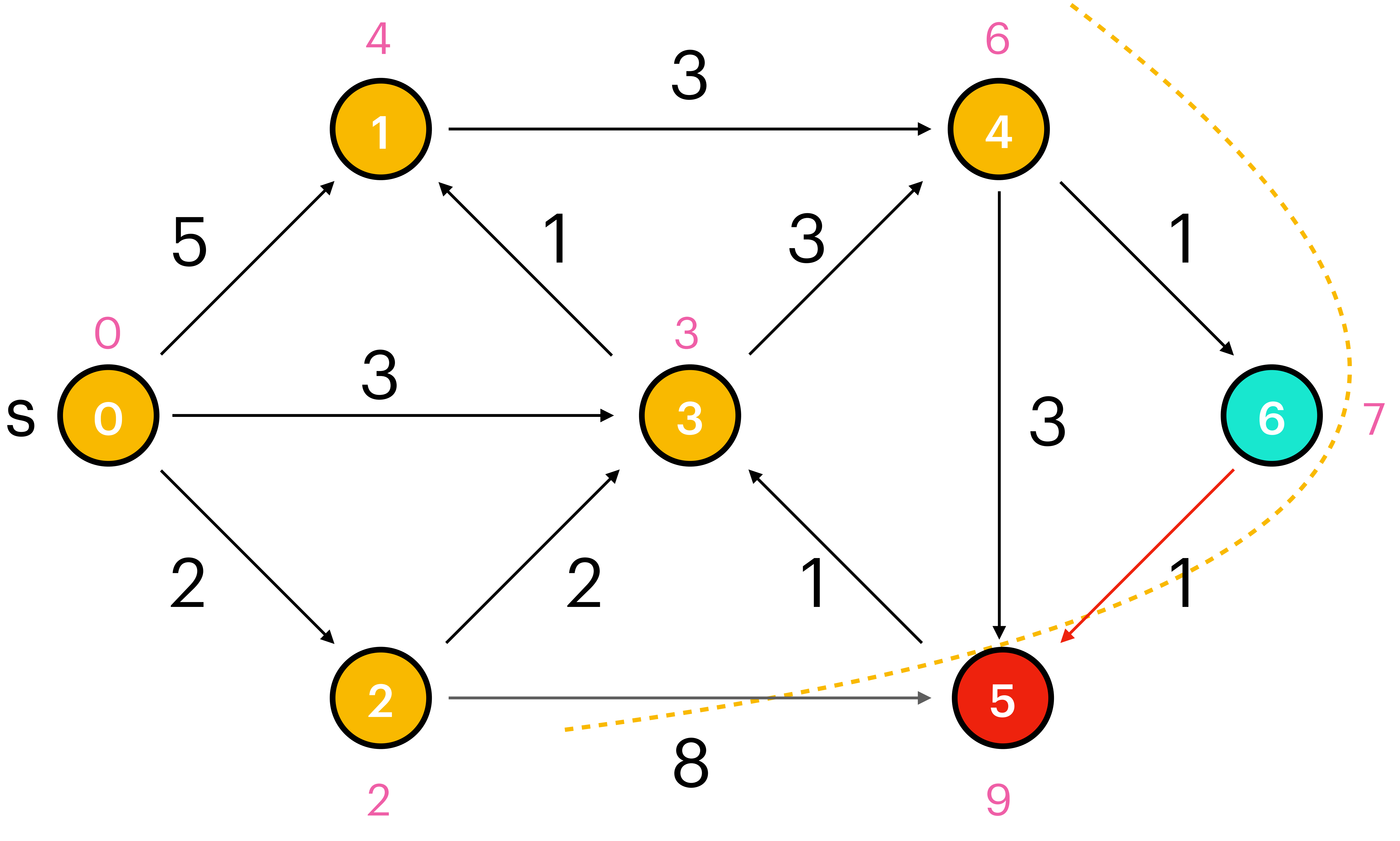
v* = 6

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	8	7

"Heap" :

5
8



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

```
d[s] ← 0;    d[v] ← ∞  ∀v ∈ V \ {s}
S ← ∅
H ← make-heap(V); decrease-key(H, s, 0)
while S ≠ V do
    v* ← extract-min(H)
    S ← S ∪ {v*}
    for (v*, v) ∈ E,  v ∉ S do
        d[v] ← min{d[v], d[v*] + c(v*, v)}
        decrease-key(H, v, d[v])
```

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 , 6 }

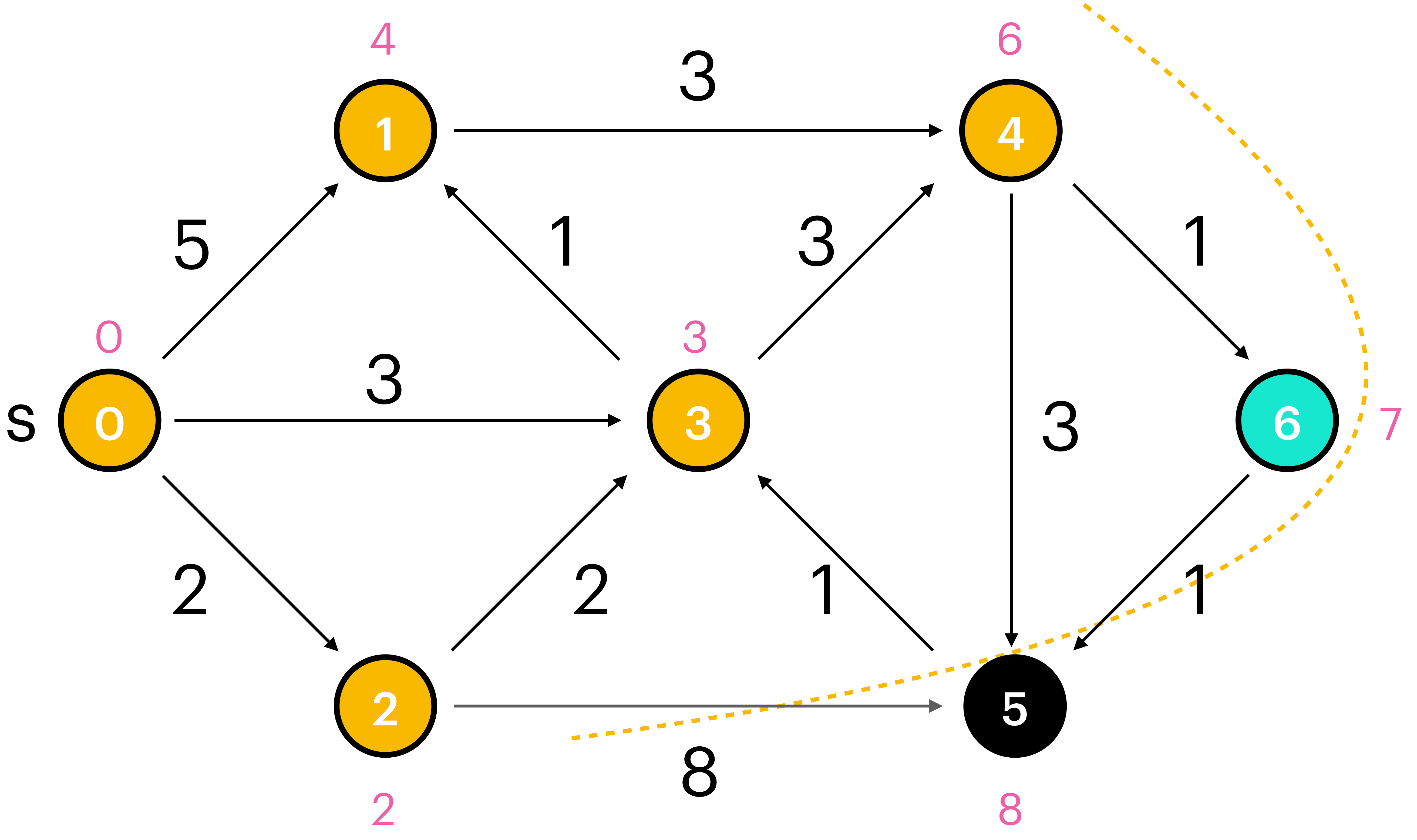
$v^* = 6$

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	8	7

"Heap" :

5
8



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \;\; \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\; v \notin S$ **do**
    $d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
    decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

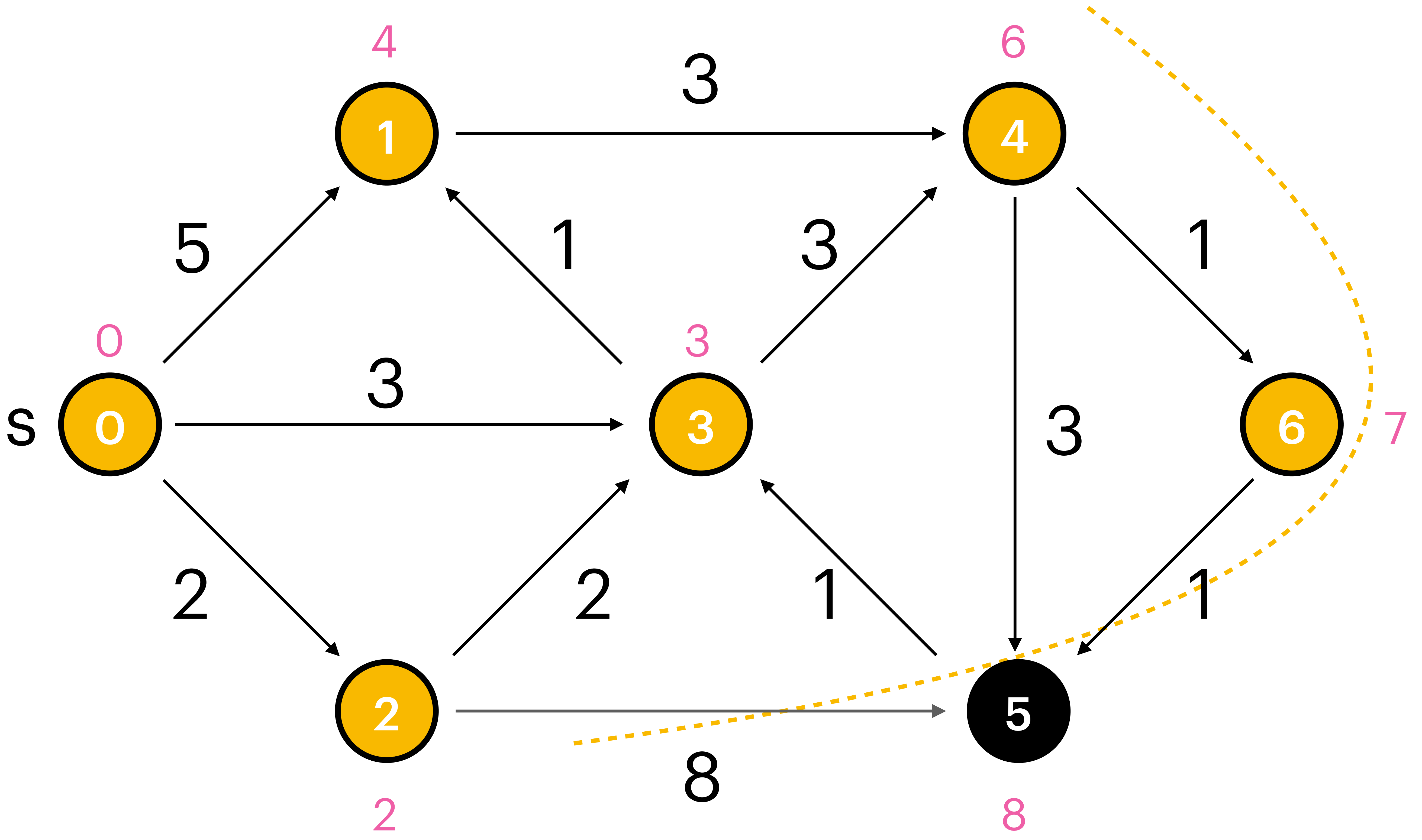
S : { 0 , 2 , 3 , 1 , 4 , 6 }

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	8	7

"Heap" :

5
8



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

```
1: d[s] ← 0;   d[v] ← ∞  ∀v ∈ V \ {s}
2: S ← ∅
3: H ← make-heap(V); decrease-key(H, s, 0)
4: while S ≠ V do
5:     v* ← extract-min(H)
6:     S ← S ∪ {v*}
7:     for (v*, v) ∈ E, v ∉ S do
8:         d[v] ← min{d[v], d[v*] + c(v*, v)}
9:         decrease-key(H, v, d[v])
```

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

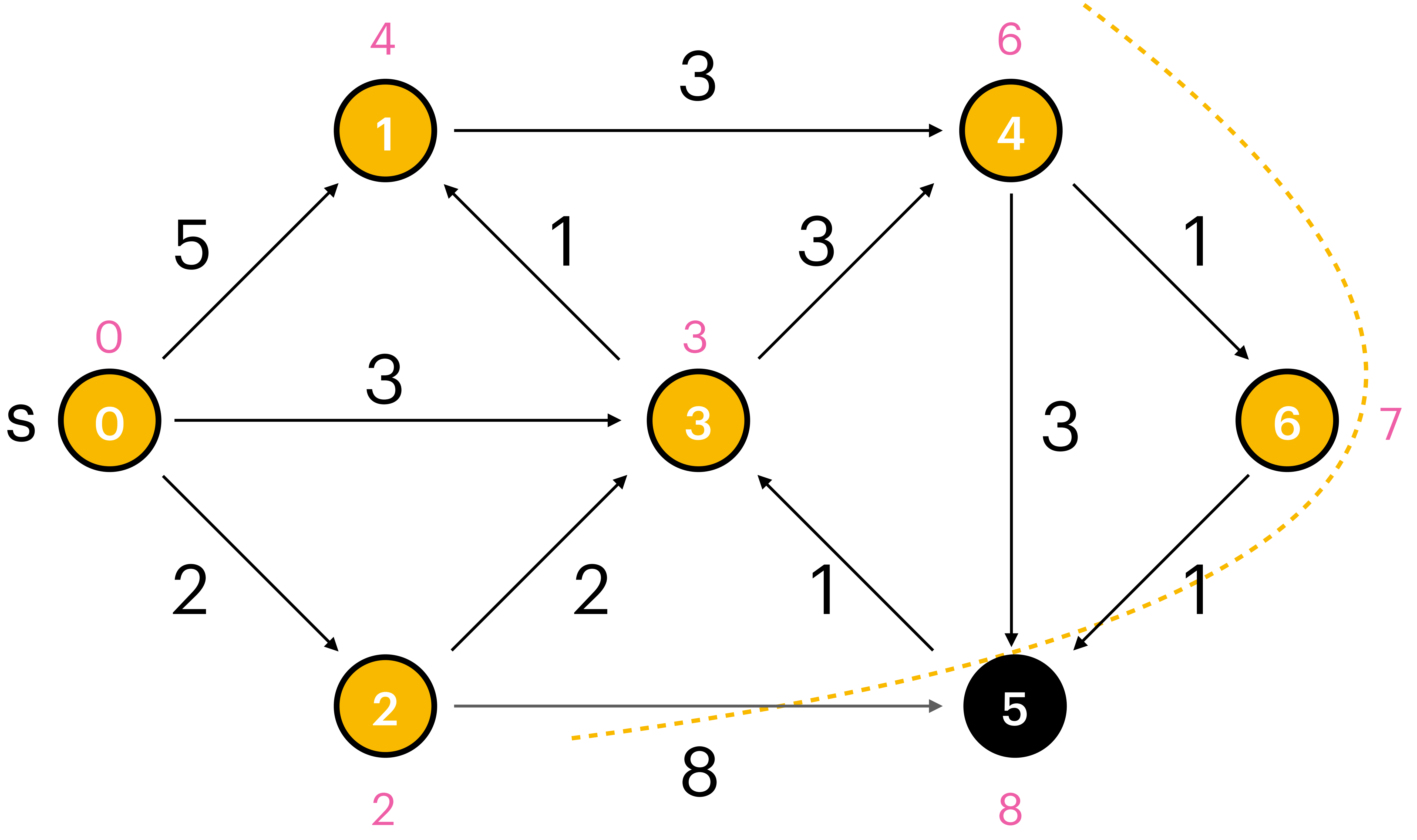
S : { 0 , 2 , 3 , 1 , 4 , 6 }

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	8	7

"Heap" :

5
8



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

```
1: d[s] ← 0;    d[v] ← ∞  ∀v ∈ V \ {s}
2: S ← ∅
3: H ← make-heap(V); decrease-key(H, s, 0)
4: while S ≠ V do
5:     v* ← extract-min(H)
6:     S ← S ∪ {v*}
7:     for (v*, v) ∈ E,  v ∉ S do
8:         d[v] ← min{d[v], d[v*] + c(v*, v)}
9:         decrease-key(H, v, d[v])
```

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

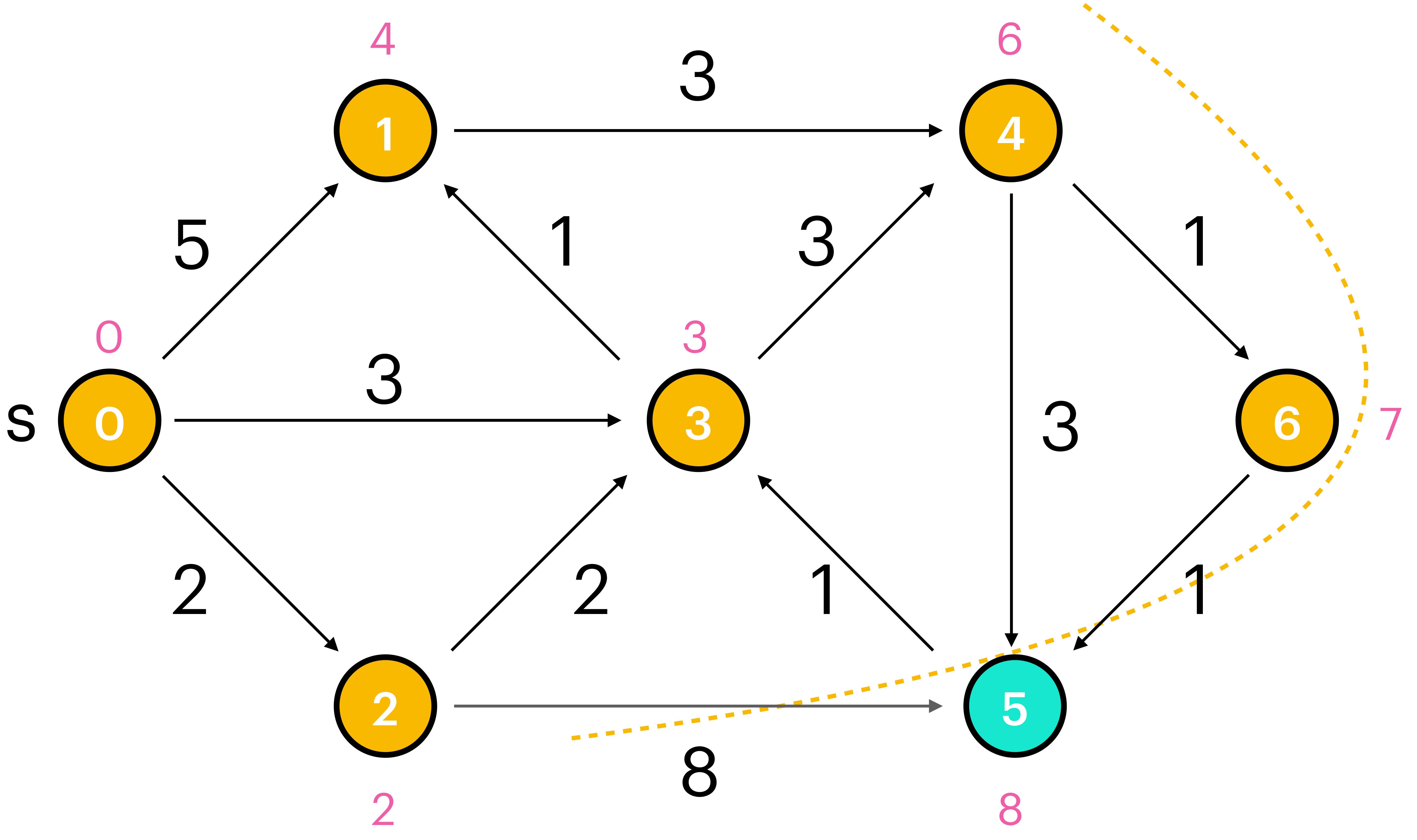
S : { 0 , 2 , 3 , 1 , 4 , 6 }

v* = 5

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	8	7

"Heap" :



Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

$\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
$S \leftarrow \varnothing$
$H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
while $S \neq V$ **do**
  $v^* \leftarrow$ extract-min(H)
  $S \leftarrow S \cup \{v^*\}$
  for $(v^*, v) \in E,\ v \notin S$ **do**
    $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
    decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

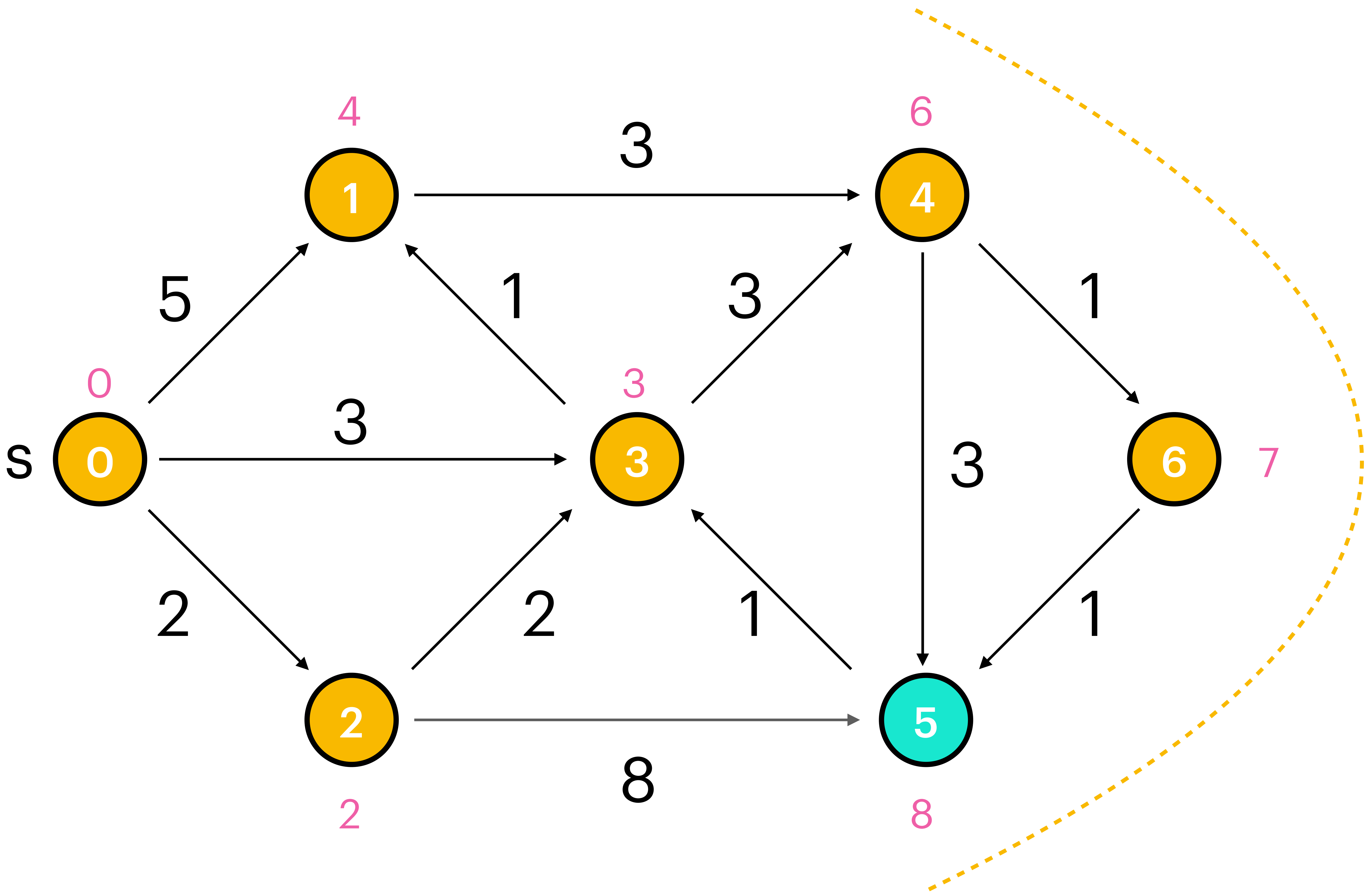
S : { 0 , 2 , 3 , 1 , 4 , 6 , 5 }

$v^* = 5$

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	8	7

"Heap" :



Shortest Paths

Dijkstra’s Algorithm

Algorithm 6 Dijkstra(s)

1: $d[s] \leftarrow 0$; $\quad d[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $\quad v^* \leftarrow$ extract-min(H)
6: $\quad S \leftarrow S \cup \{v^*\}$
7: $\quad$ **for** $(v^*, v) \in E,\ v \notin S$ **do**
8: $\quad\quad d[v] \leftarrow \min\{d[v], d[v^*] + c(v^*, v)\}$
9: $\quad\quad$ decrease-key($H, v, d[v]$)

make-heap(V) :

Create a min heap of the vertices

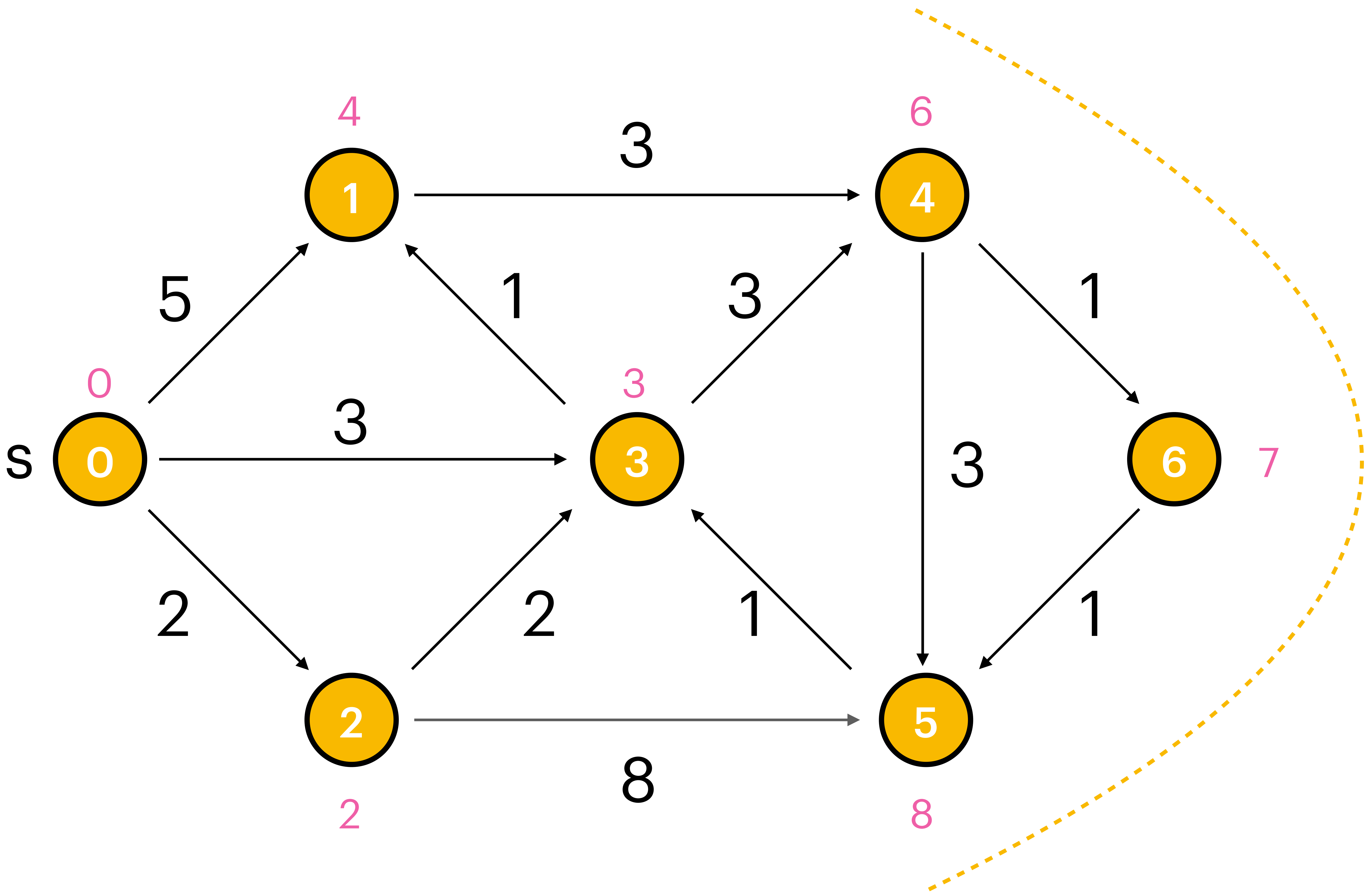
extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

S : { 0 , 2 , 3 , 1 , 4 , 6 , 5 }



d[] :

0	1	2	3	4	5	6
0	4	2	3	6	8	7

“Heap” :

Shortest Paths

Dijkstra's Algorithm

Algorithm 6 Dijkstra(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \ \forall v \in V \setminus \{s\}$
2: $S \leftarrow \varnothing$
3: $H \leftarrow$ make-heap(V); decrease-key($H, s, 0$)
4: **while** $S \neq V$ **do**
5: $\quad v^* \leftarrow$ extract-min(H)
6: $\quad S \leftarrow S \cup \{v^*\}$
7: $\quad$ **for** $(v^*, v) \in E,\ v \notin S$ **do**
8: $\quad\quad \mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[v^*] + c(v^*, v)\}$
9: $\quad\quad$ decrease-key($H, v, \mathrm{d}[v]$)

make-heap(V) :

Create a min heap of the vertices

extract-min(H) :

Extract (= remove and assign) the node with the minimum distance from the heap

decrease-key(H, v, k) :

Update the distance of v in heap H to the key k

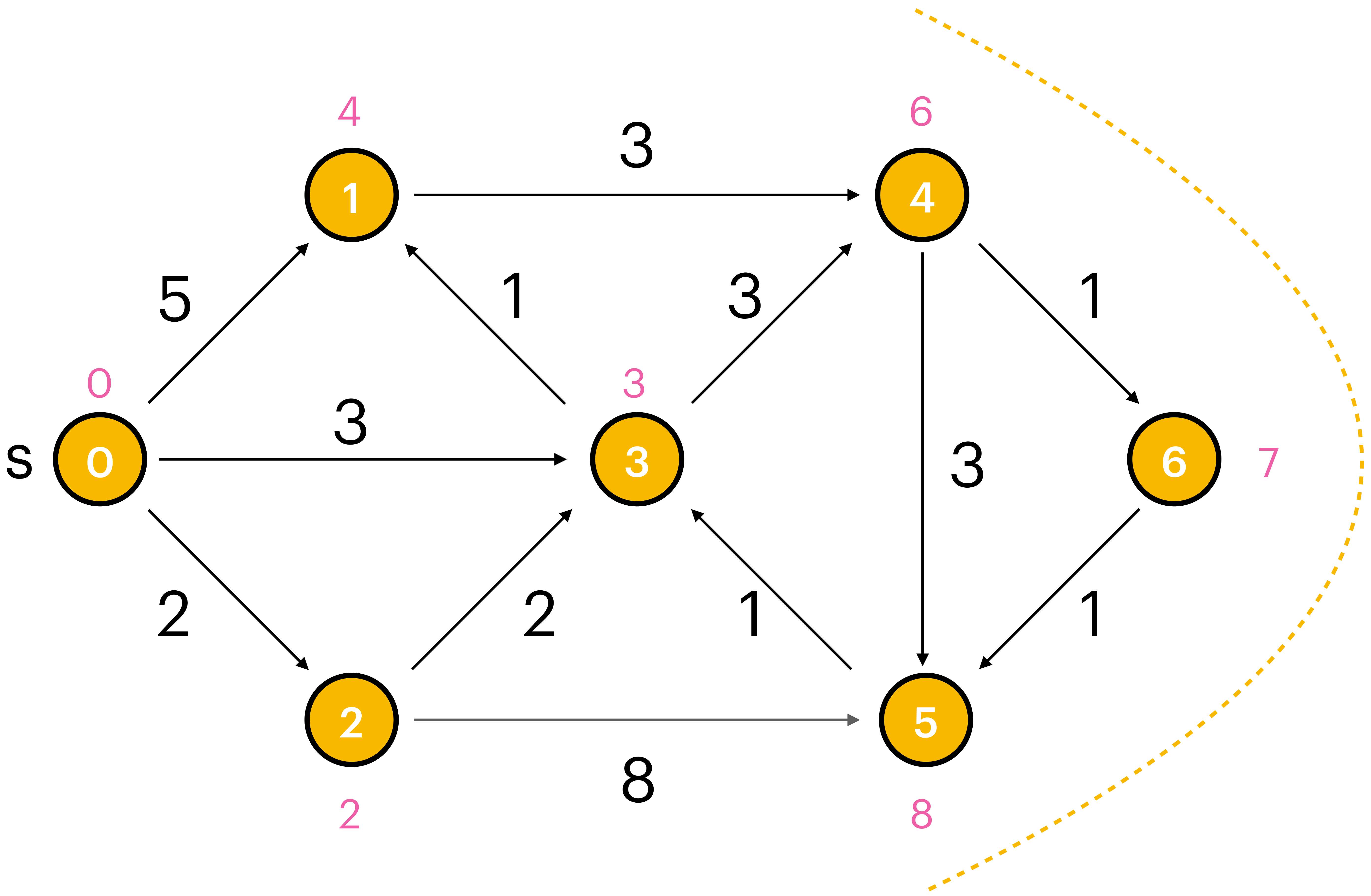
S : { 0 , 2 , 3 , 1 , 4 , 6 , 5 }

S = V

d[] :

0	1	2	3	4	5	6
0	4	2	3	6	8	7

"Heap" :

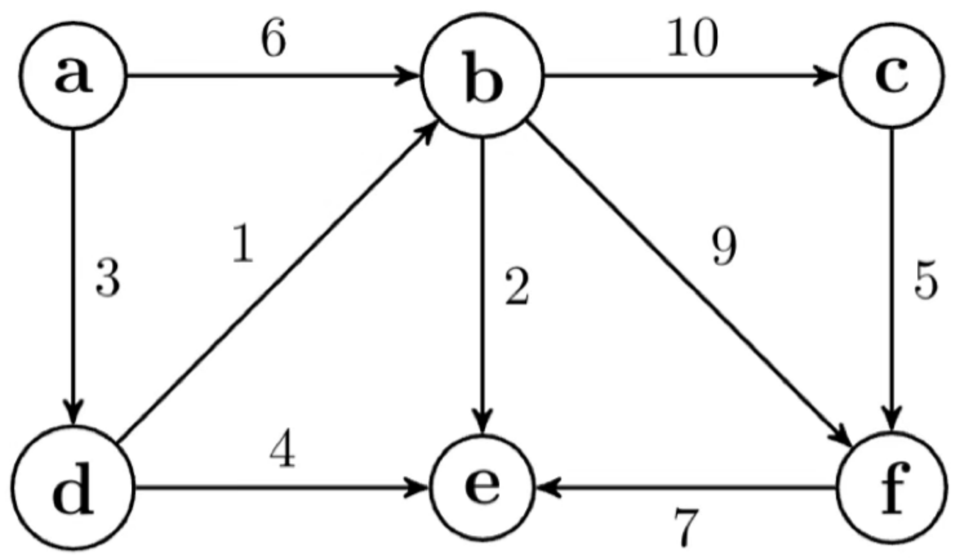


Dijkstra

Exam Questions

/ 2 P

f) *Shortest Path Tree:* Consider the following graph:



i) Highlight the edges that are part of the shortest-path tree rooted at vertex a (i.e., the output of Dijkstra's algorithm if we were to start from vertex a).

ii) Does the above graph have a topological ordering? If yes, write down one topological ordering. If no, give an argument.

Dijkstra

Exam Tipps

- Don’t rush to the solution
  - Update the distances just like dijkstra’s algo

- Don’t mix up with MST ! It’s not the same !

Let’s take a break

Shortest Paths

Bellman Ford

weighted, positive and (possibly) negative edge weights ,
(possibly) negative closed walks
$c(e) \in \mathbb{R}$

Runtime : O (|V| *|E|)

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \; \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3:     **for** $(u, v) \in E$ **do**
4:         $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

---

relax the edges for n-1 times

Shortest Paths

Bellman Ford

weighted, positive and (possibly) negative edge weights ,
(possibly) negative closed walks
$c(e) \in \mathbb{R}$

Runtime : O (|V| *|E|)

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \; \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \dots n-1\}$ **do** — relax the edges for n-1 times
3: **for** $(u, v) \in E$ **do**
4: $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

Why n-1 iterations ?

A shortest path in a graph without cycles will have at most n−1 edges

(since a path cannot visit any vertex more than once in a simple graph)

How to detect negative closed walks :

Do one extra iteration

If any distance improves, it indicates the existence of a directed closed walk with negative total weight

Shortest Paths

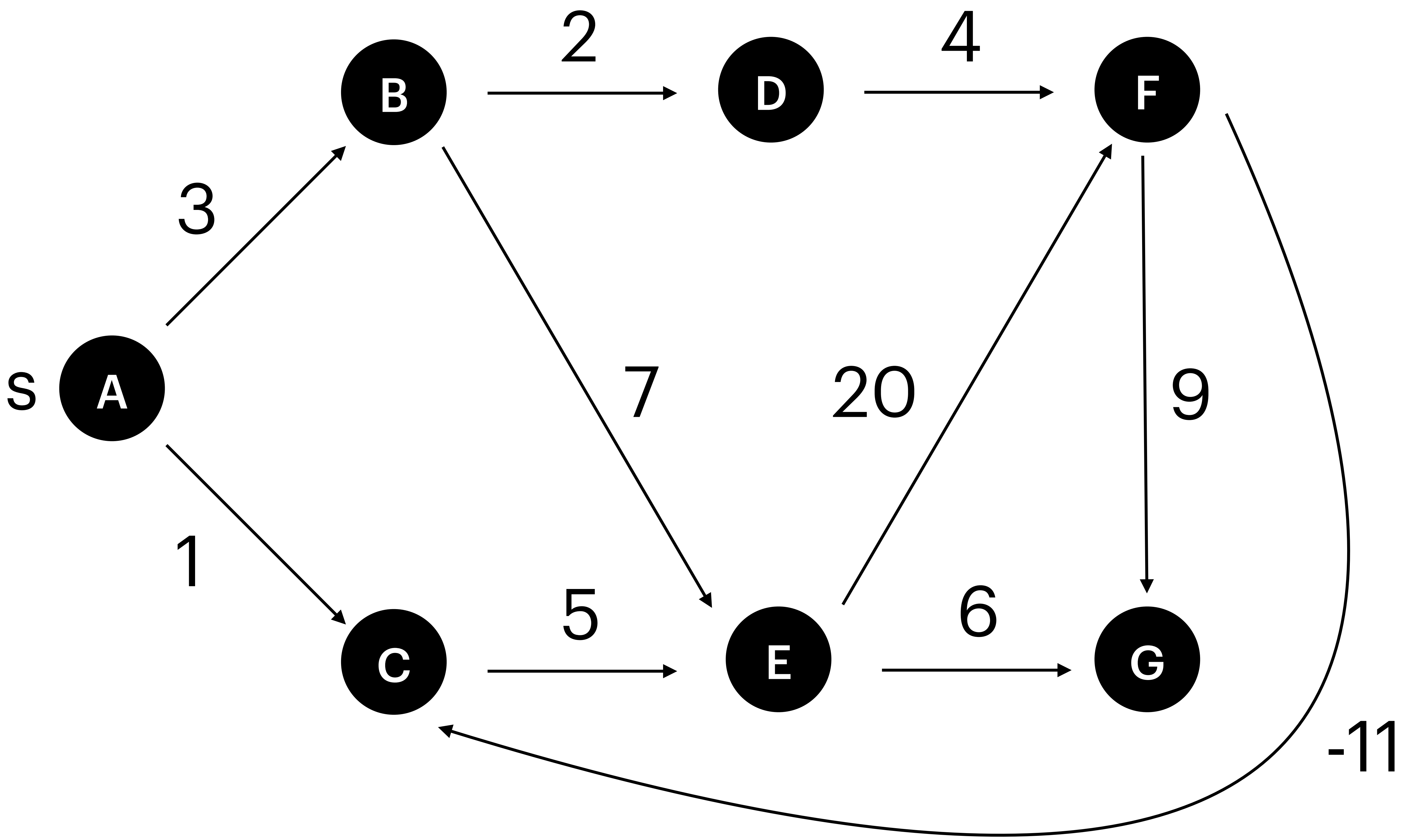
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3: **for** $(u, v) \in E$ **do**
4: $d[v] \leftarrow \min\{d[v], d[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0							
1							
2							
3							
4							
5							
6							



Shortest Paths

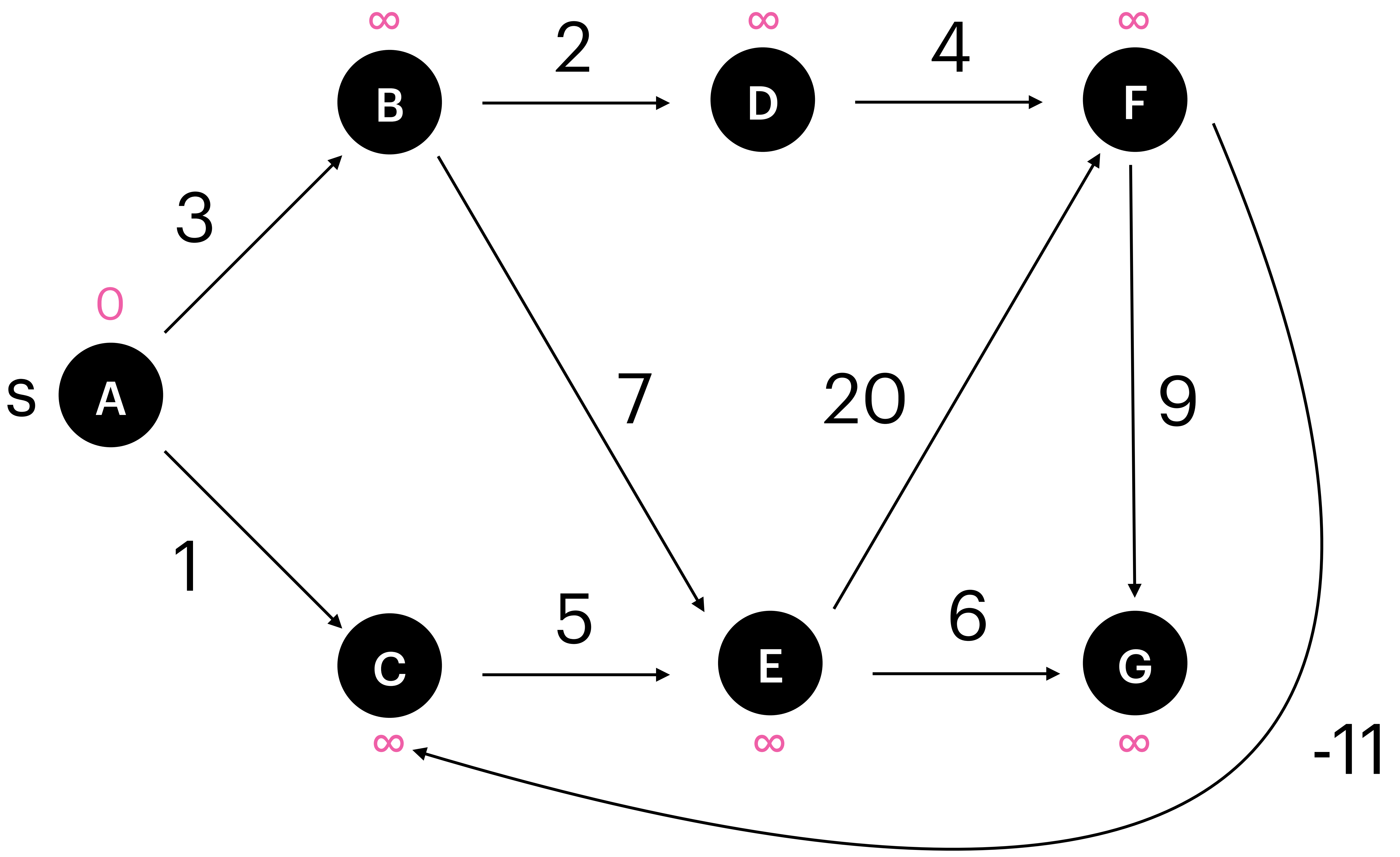
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3:     **for** $(u, v) \in E$ **do**
4:         $d[v] \leftarrow \min\{d[v], d[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1							
2							
3							
4							
5							
6							



Shortest Paths

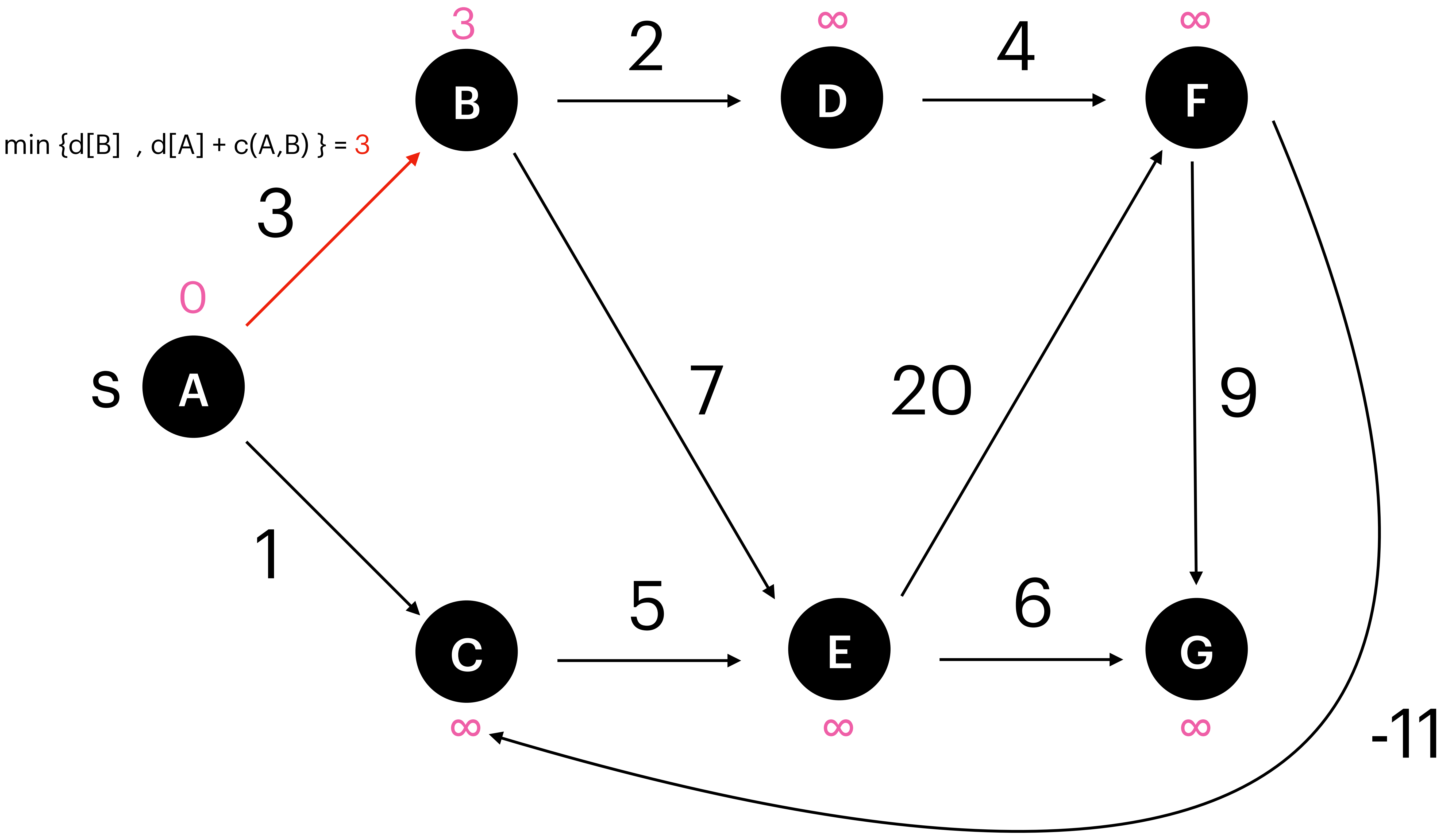
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \; \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3: **for** $(u, v) \in E$ **do**
4: $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3					
2							
3							
4							
5							
6							



Shortest Paths

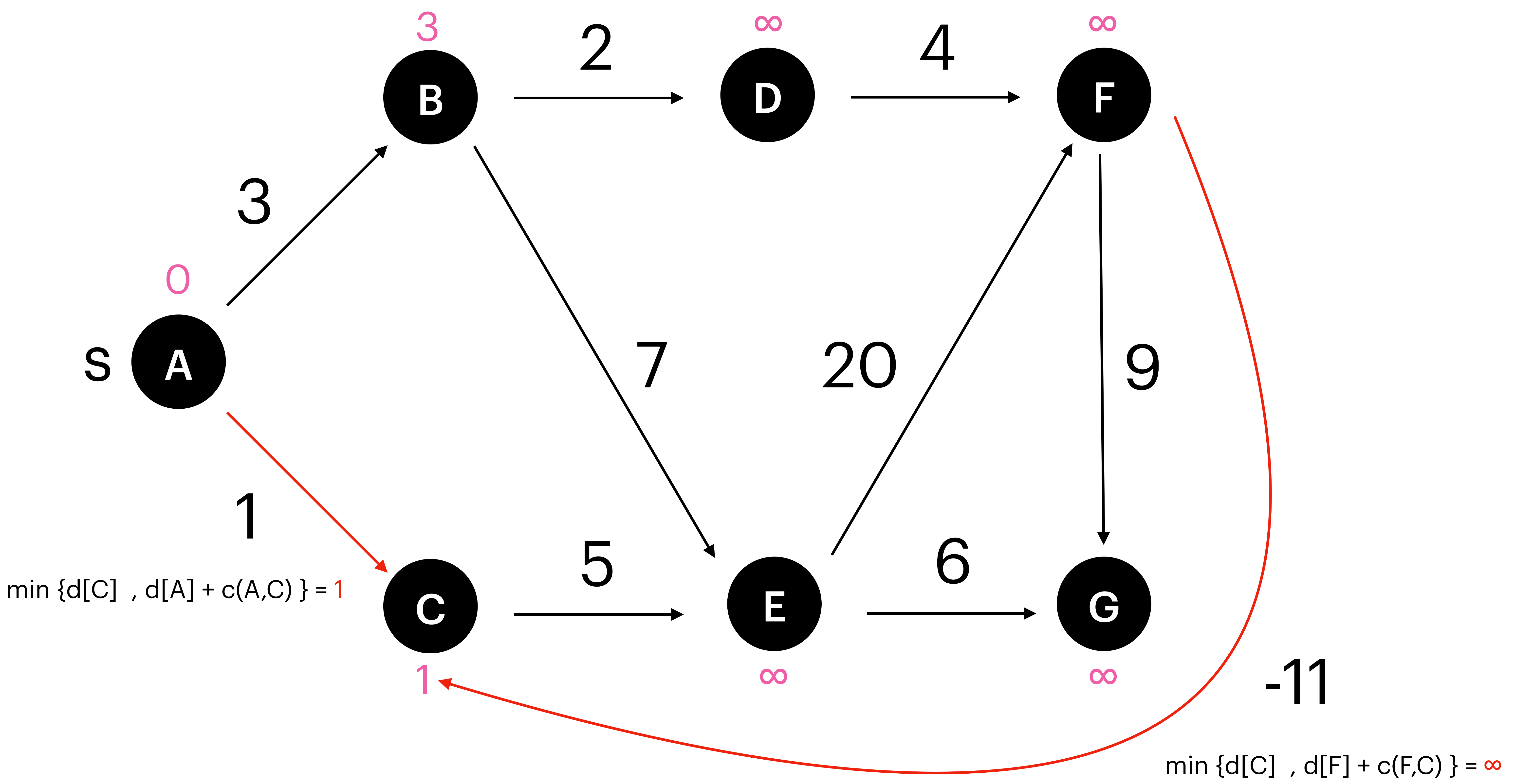
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty\ \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3:     **for** $(u, v) \in E$ **do**
4:         $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1				
2							
3							
4							
5							
6							



Shortest Paths

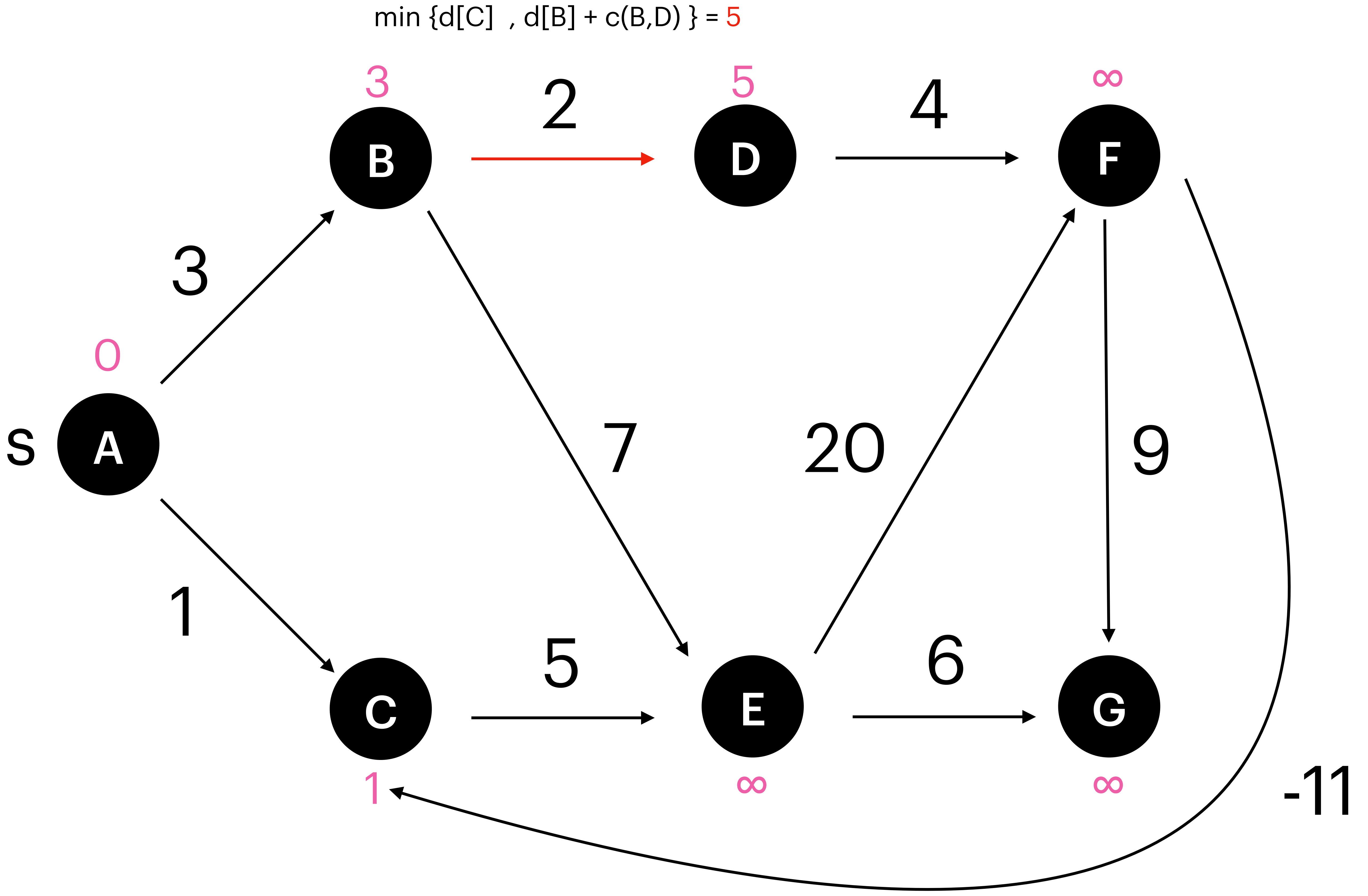
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3: **for** $(u, v) \in E$ **do**
4: $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1	5			
2							
3							
4							
5							
6							



Shortest Paths

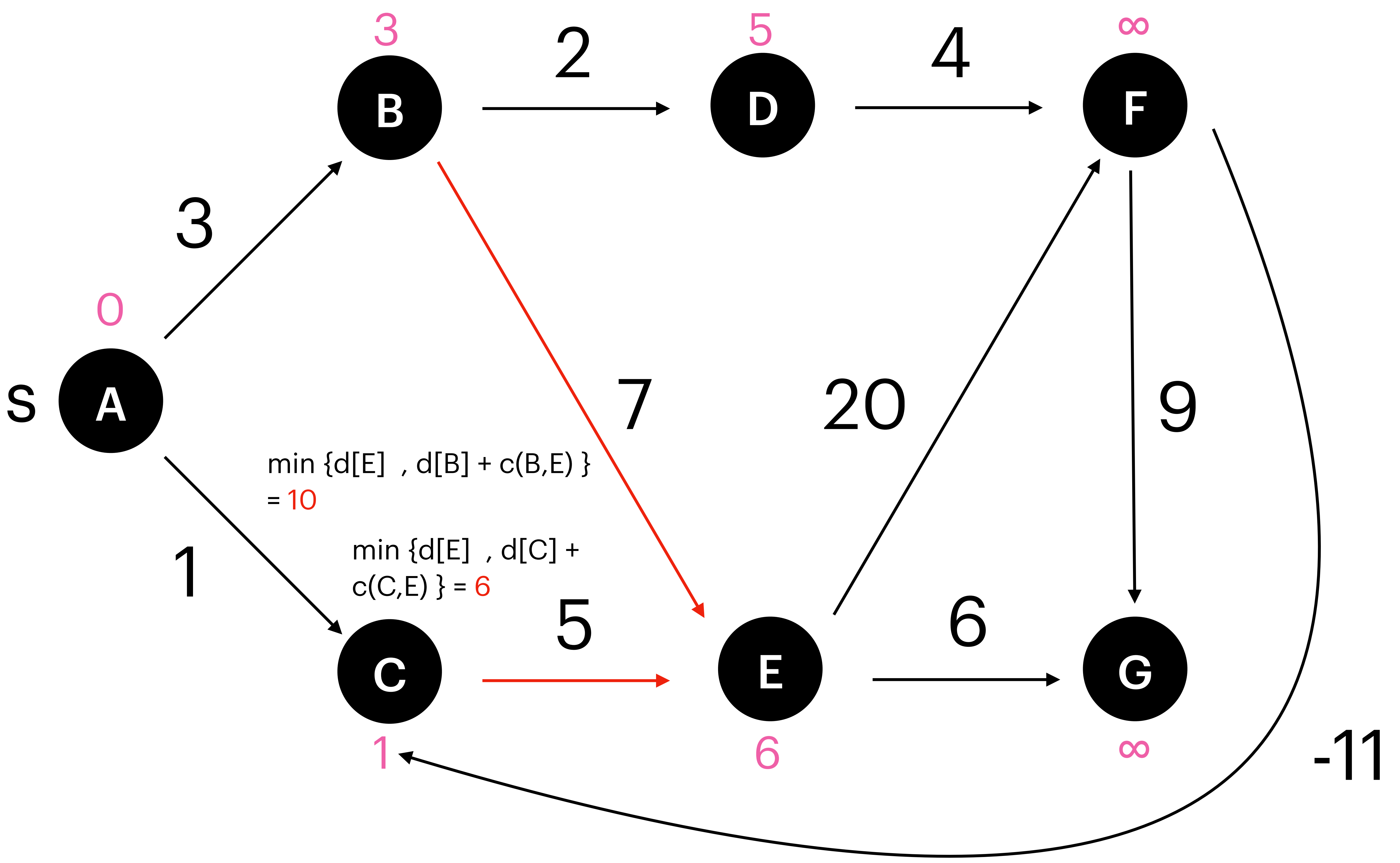
Bellman Ford

Algorithm 7 Bellman-Ford(s)

$d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \; \forall v \in V \setminus \{s\}$
for $i \in \{1, \dots n-1\}$ **do**
  for $(u, v) \in E$ **do**
    $d[v] \leftarrow \min\{d[v], d[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1	5	6		
2							
3							
4							
5							
6							



Shortest Paths

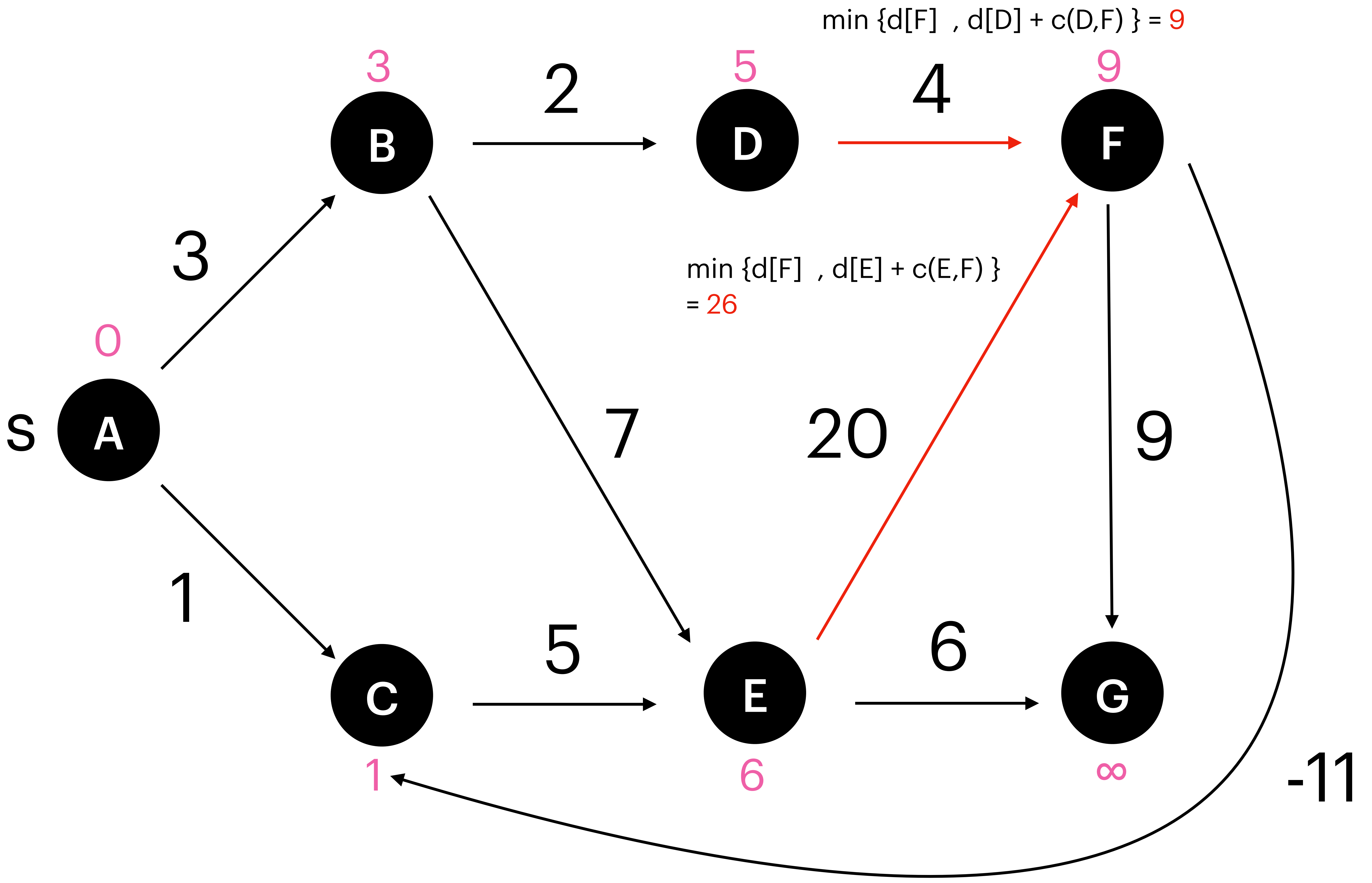
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3:     **for** $(u,v) \in E$ **do**
4:         $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u,v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1	5	6	9	
2							
3							
4							
5							
6							



Shortest Paths

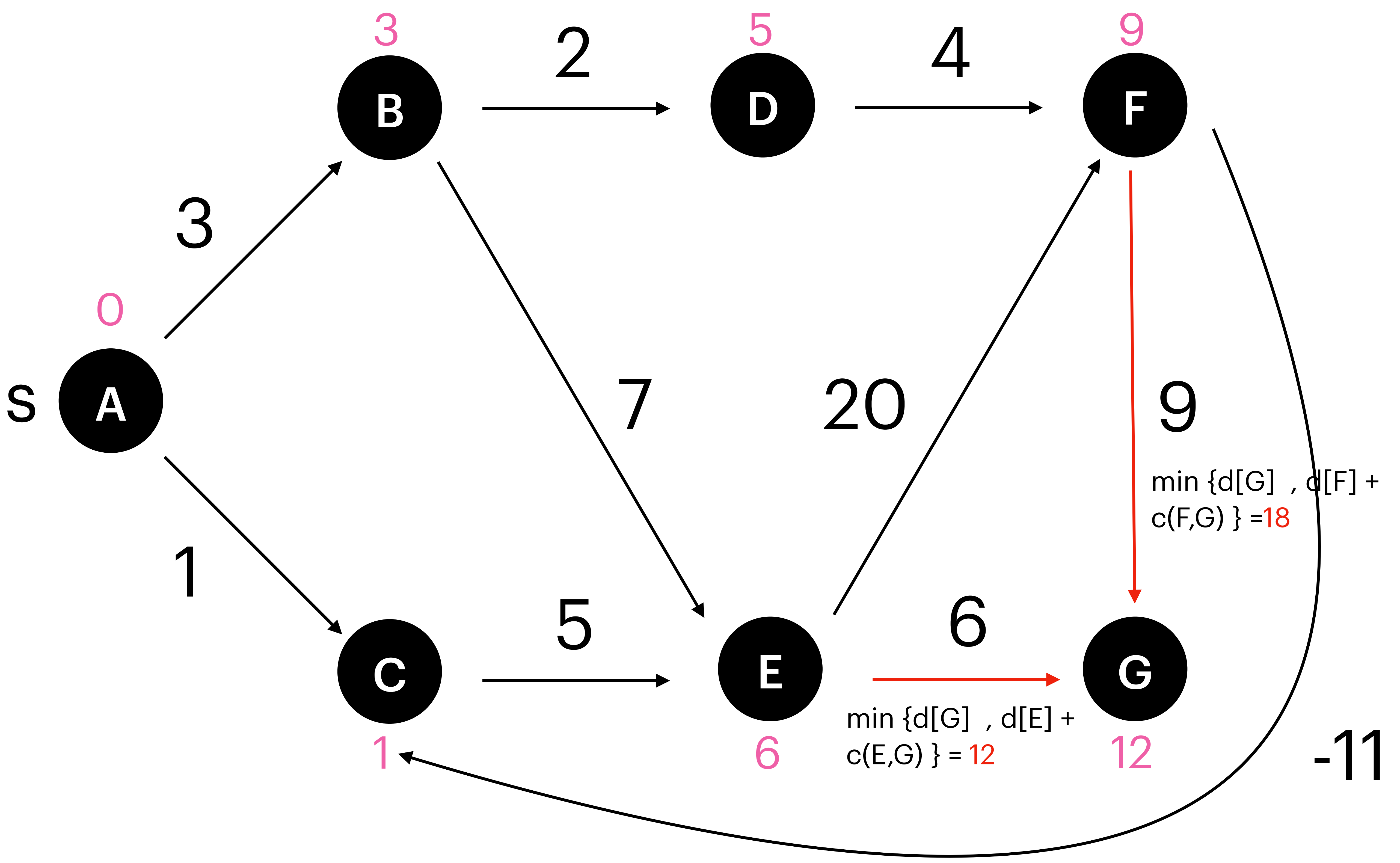
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \dots n-1\}$ **do**
3: **for** $(u, v) \in E$ **do**
4: $d[v] \leftarrow \min\{d[v], d[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1	5	6	9	12
2							
3							
4							
5							
6							



Shortest Paths

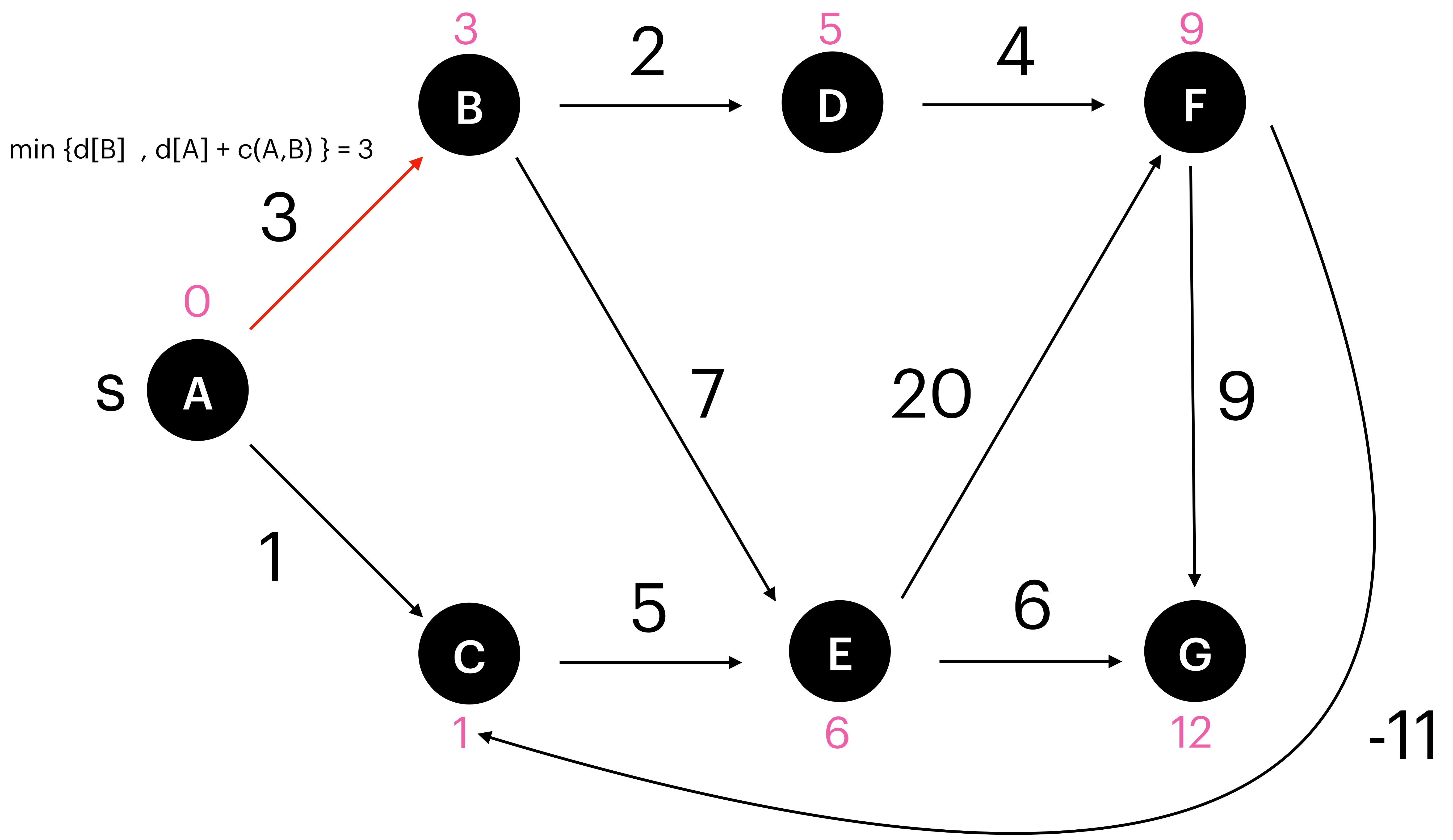
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3: **for** $(u, v) \in E$ **do**
4: $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1	5	6	9	12
2	0	3	1	5	6	9	12
3							
4							
5							
6							



Shortest Paths

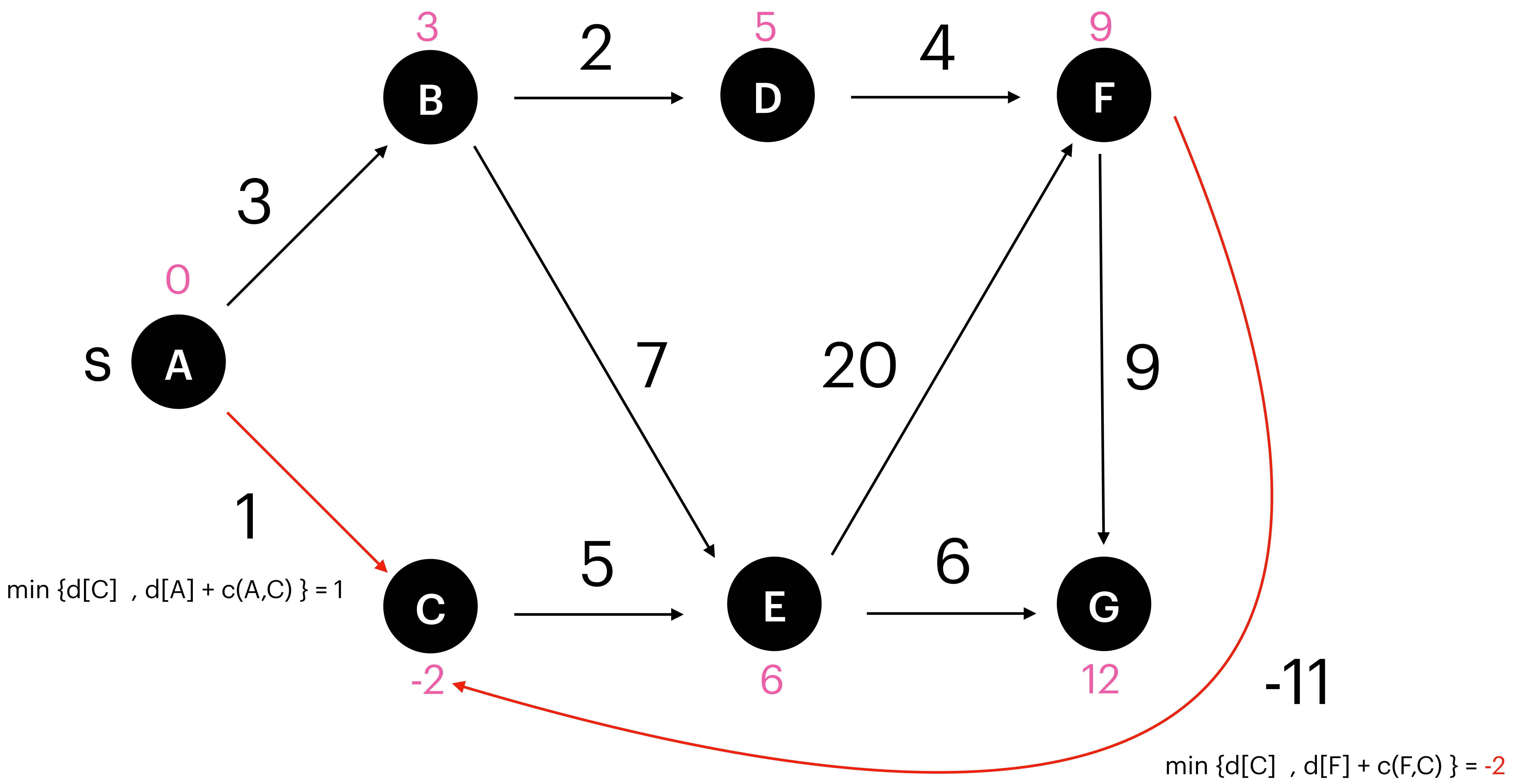
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $d[s] \leftarrow 0; \quad d[v] \leftarrow \infty \; \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3: **for** $(u,v) \in E$ **do**
4: $d[v] \leftarrow \min\{d[v], d[u] + c(u,v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1	5	6	9	12
2	0	3	-2	5	6	9	12
3							
4							
5							
6							



Shortest Paths

Bellman Ford

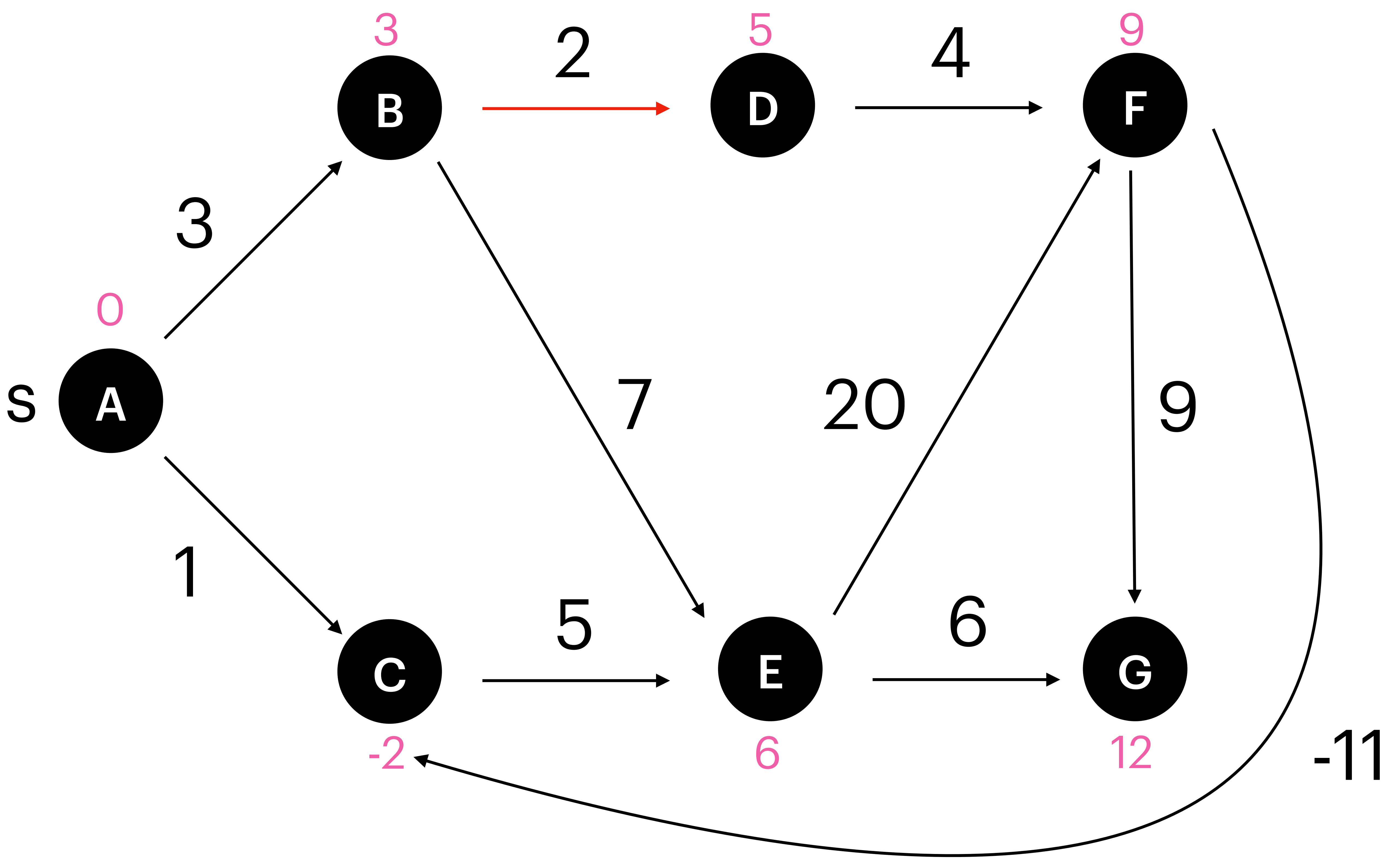
Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \; \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3: **for** $(u, v) \in E$ **do**
4: $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1	5	6	9	12
2	0	3	-2	5	6	9	12
3							
4							
5							
6							

min {d[C] , d[B] + c(B,D) } = 5



Shortest Paths

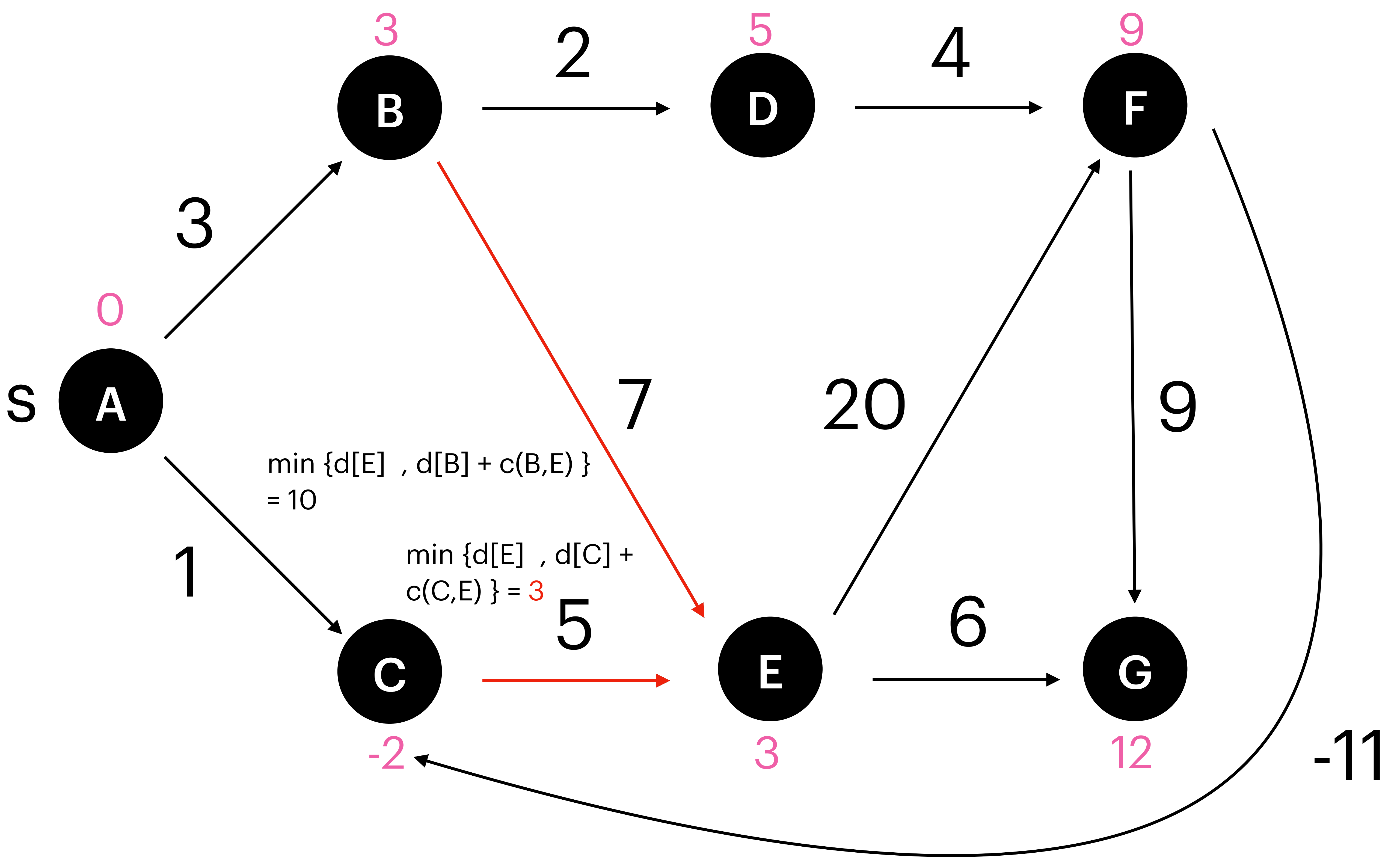
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \; \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3: **for** $(u, v) \in E$ **do**
4: $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1	5	6	9	12
2	0	3	-2	5	3	9	12
3							
4							
5							
6							



Shortest Paths

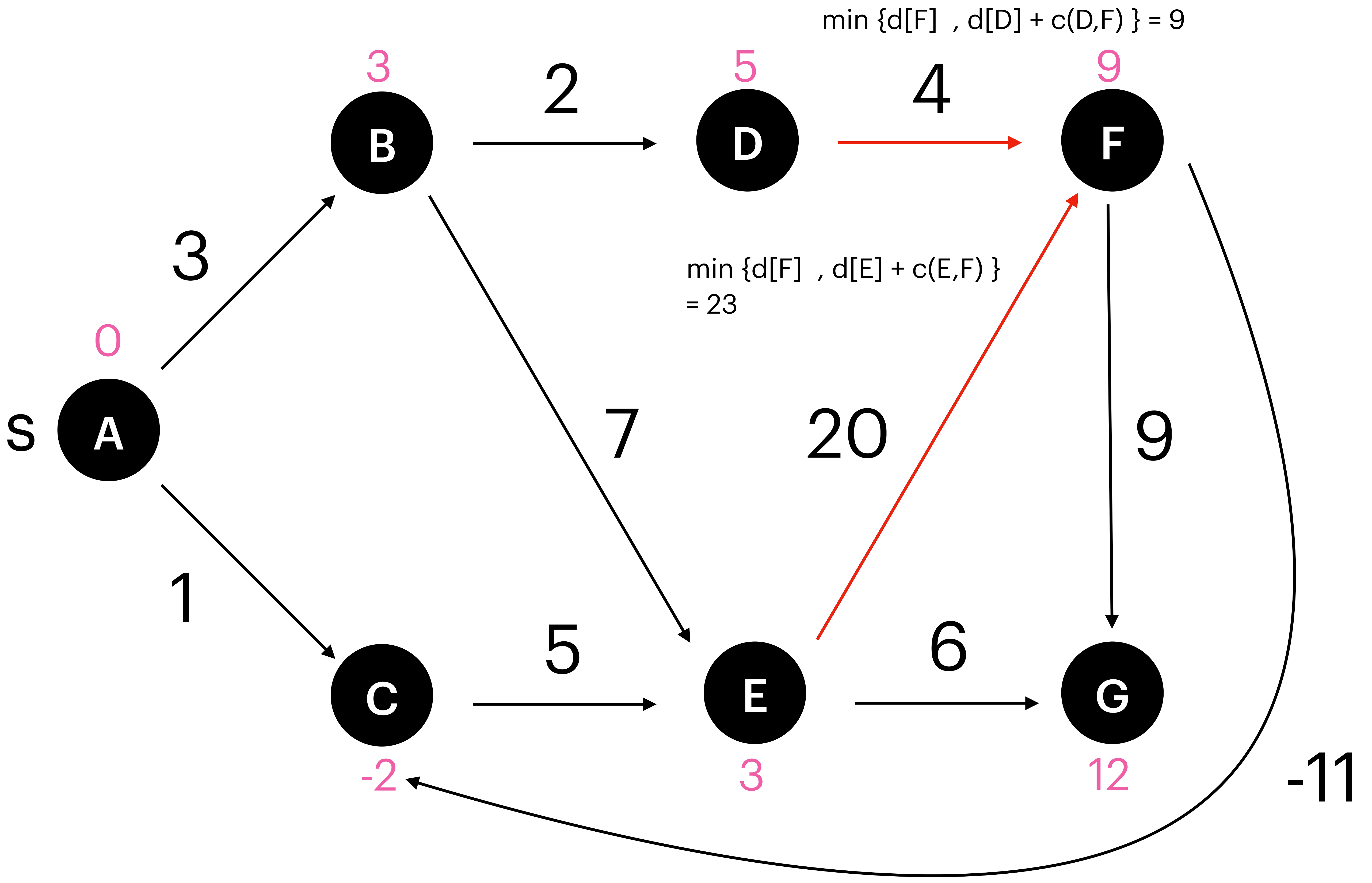
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3: **for** $(u, v) \in E$ **do**
4: $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1	5	6	9	12
2	0	3	-2	5	3	9	12
3							
4							
5							
6							



Shortest Paths

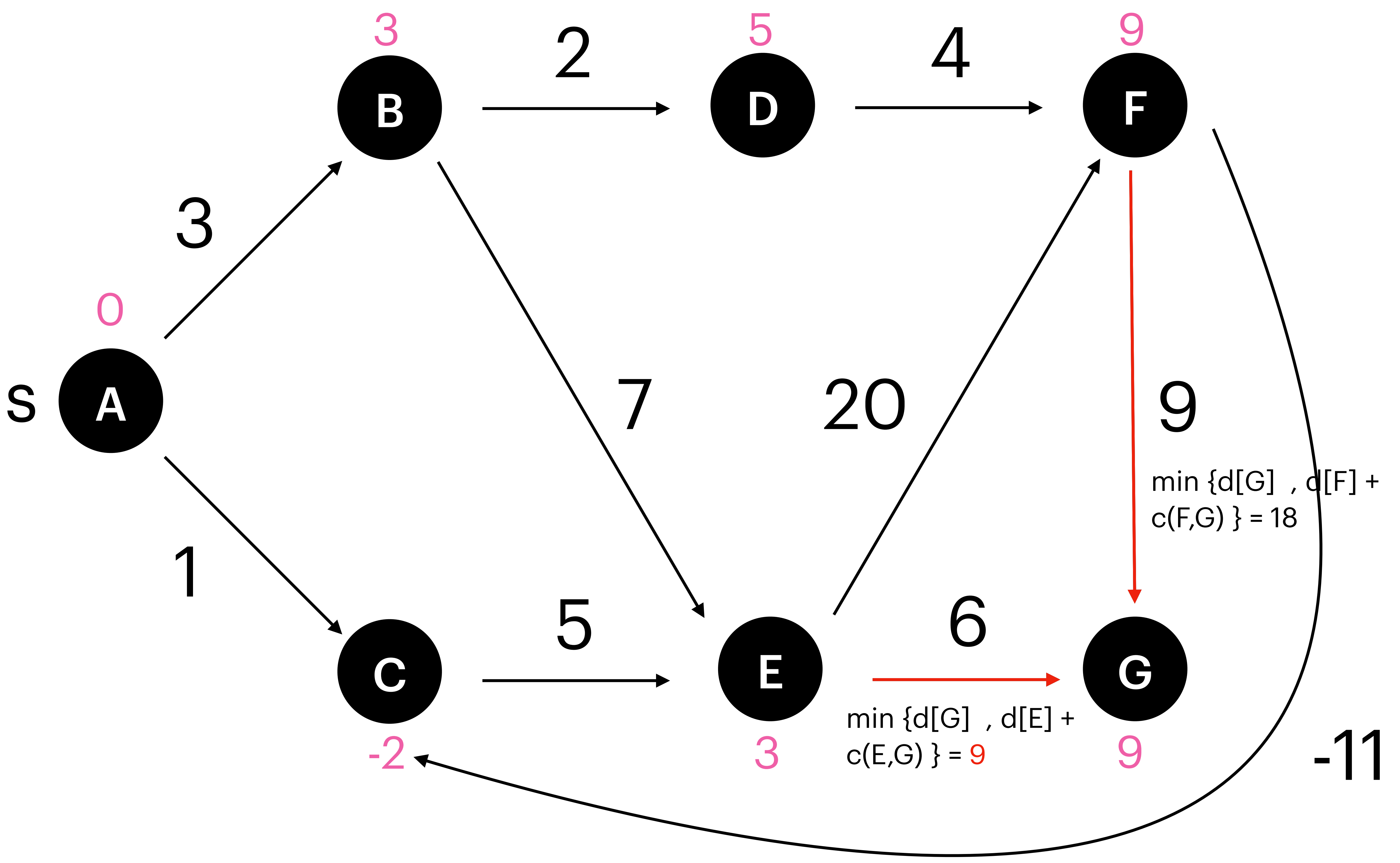
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \; \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3:     **for** $(u, v) \in E$ **do**
4:         $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1	5	6	9	12
2	0	3	-2	5	3	9	9
3							
4							
5							
6							



Shortest Paths

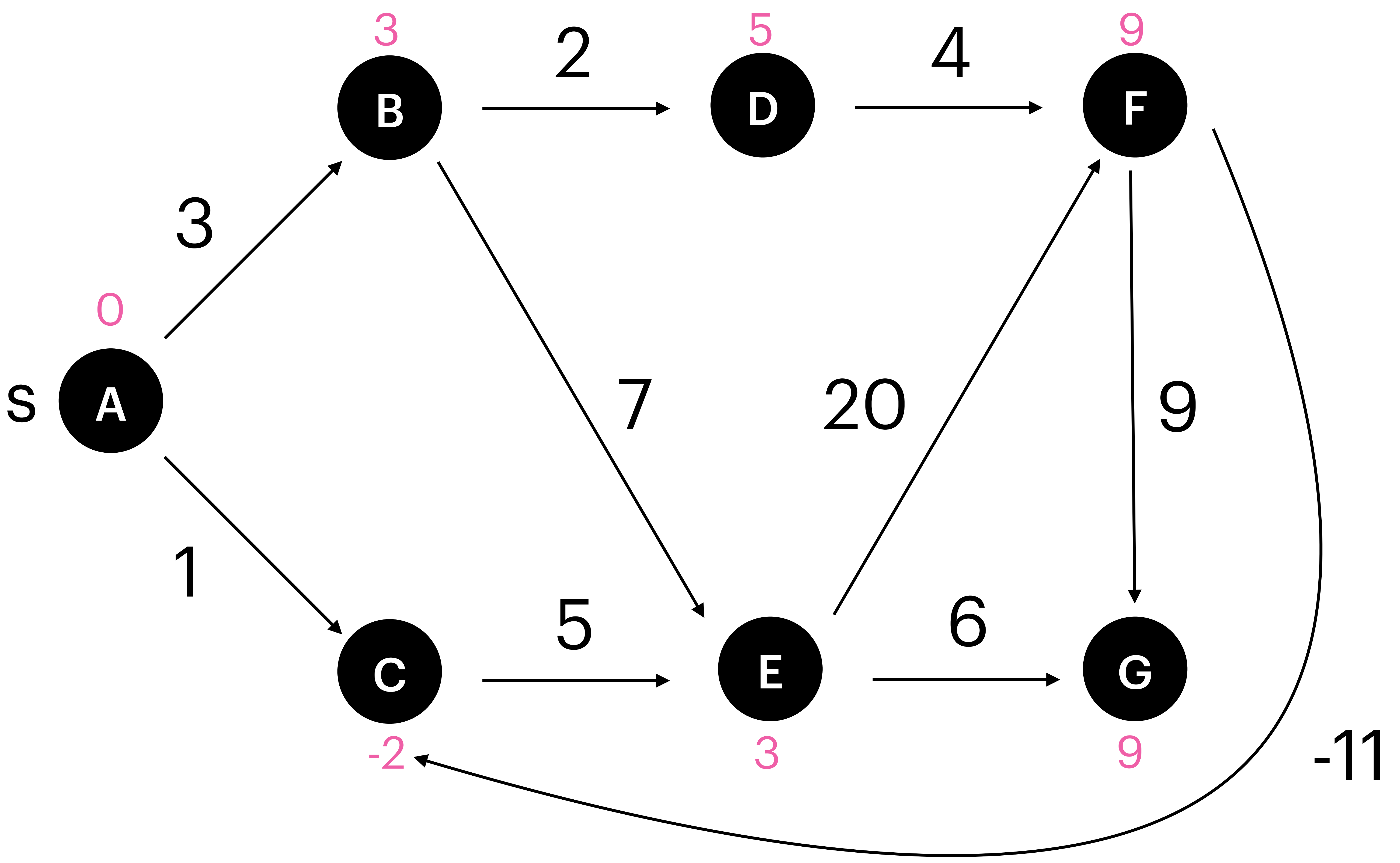
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \; \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3: **for** $(u, v) \in E$ **do**
4: $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	3	1	5	6	9	12
2	0	3	-2	5	3	9	9
3							
4							
5							
6							



Shortest Paths

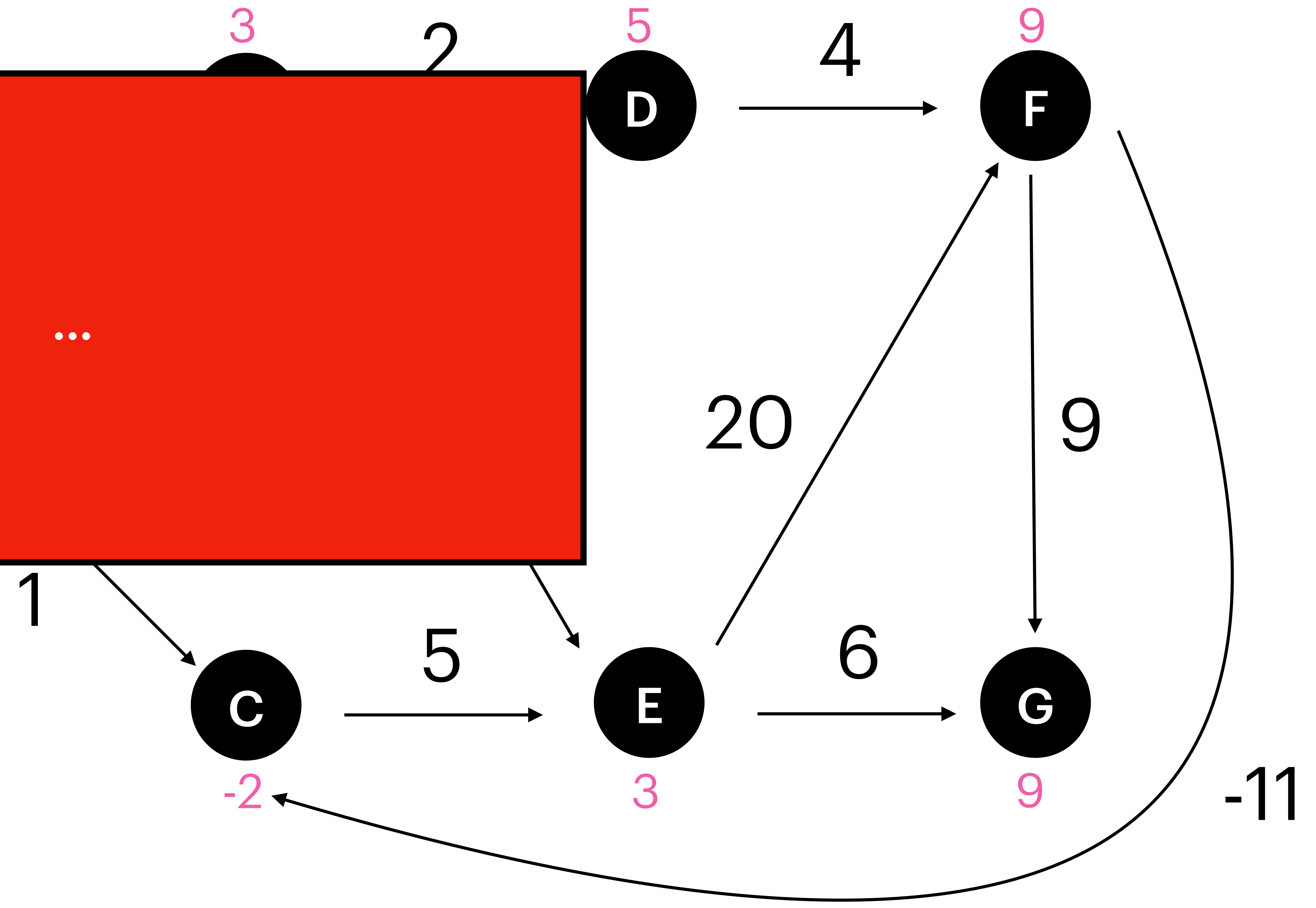
Bellman Ford

Algorithm 7 Bellman-Ford(s)

1: $\mathrm{d}[s] \leftarrow 0; \quad \mathrm{d}[v] \leftarrow \infty \ \forall v \in V \setminus \{s\}$
2: **for** $i \in \{1, \ldots n-1\}$ **do**
3: **for** $(u, v) \in E$ **do**
4: $\mathrm{d}[v] \leftarrow \min\{\mathrm{d}[v], \mathrm{d}[u] + c(u, v)\}$

d[] :

i	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	
1	0	3	1	5	6	9	
2	0	3	-2	5	3	9	
3							
4							
5							
6							



Shortest Paths

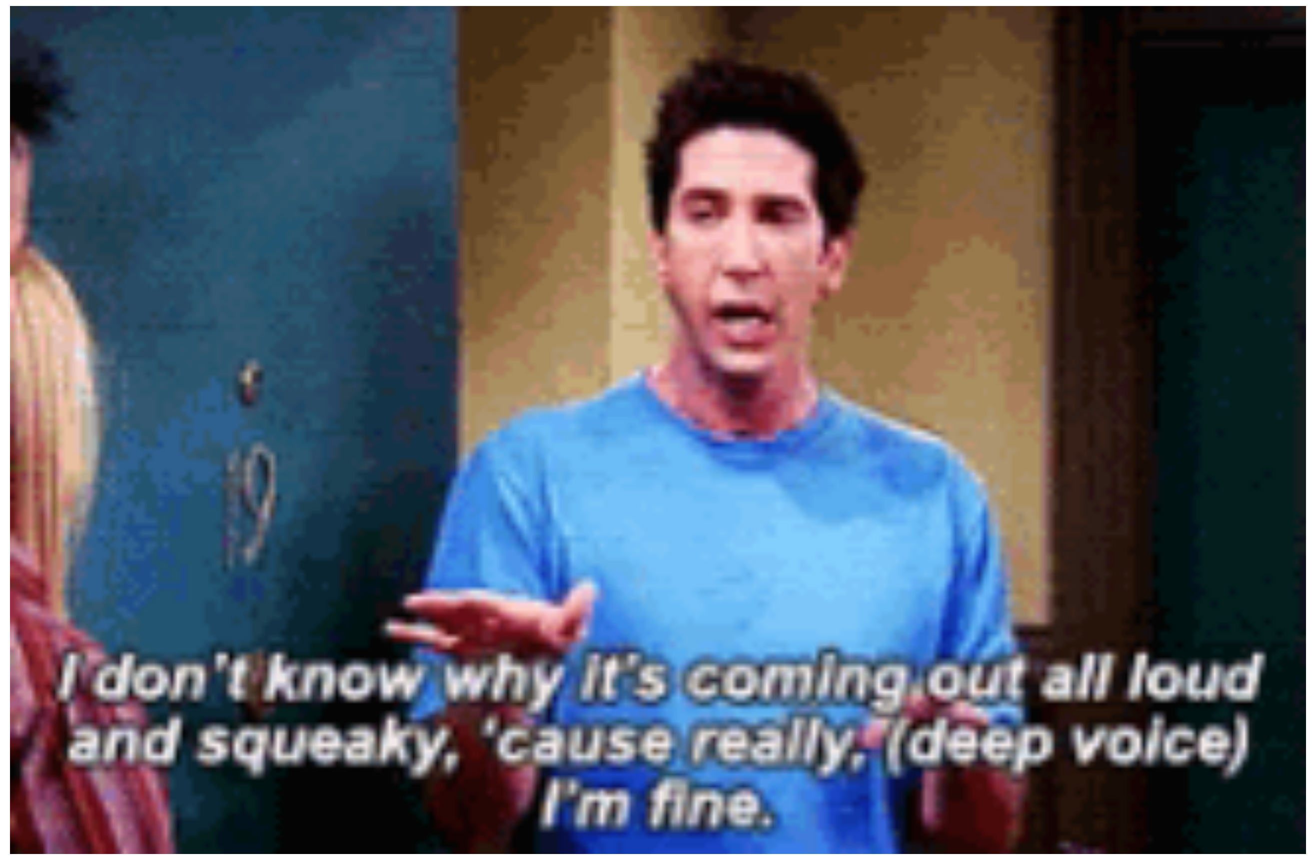
one - to - all

G (directed/undirected)	Algorithm	Runtime
unweighted , all nodes with the same positive weight	BFS usage	O (|V| + |E|)
weighted , positive edge weights $c(e) \geq 0$	Dijsktra	O ((|V| + |E|) * log n)
weighted, positive and (possibly) negative edge weights $c(e) \in \mathbb{R}$	Belmann-Ford	O (|V| * |E|)
G has no cycles	topological sorting + DP	O (|V| + |E|)

Graph Modelling

Exam Question

Code Expert - Graph Sets



Next Week ...

MST

Questions

Feedbacks , Recommendations

Nil Ozer