

그래프 2

최백준 choi@startlink.io

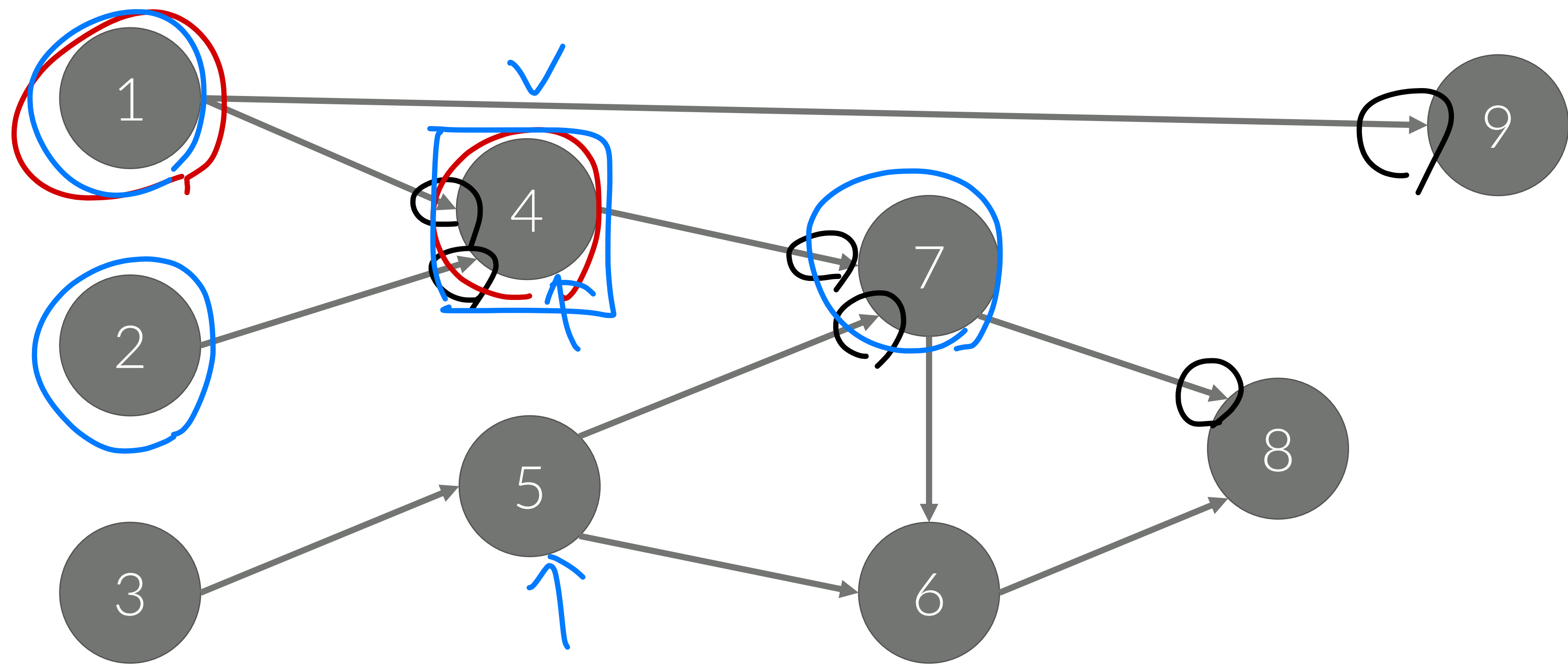
DAG

DAG

Directed Acyclic Graph

선형 순서

- 사이클이 없는 방향 있는 그래프를 DAG라고 한다.



위상 정렬

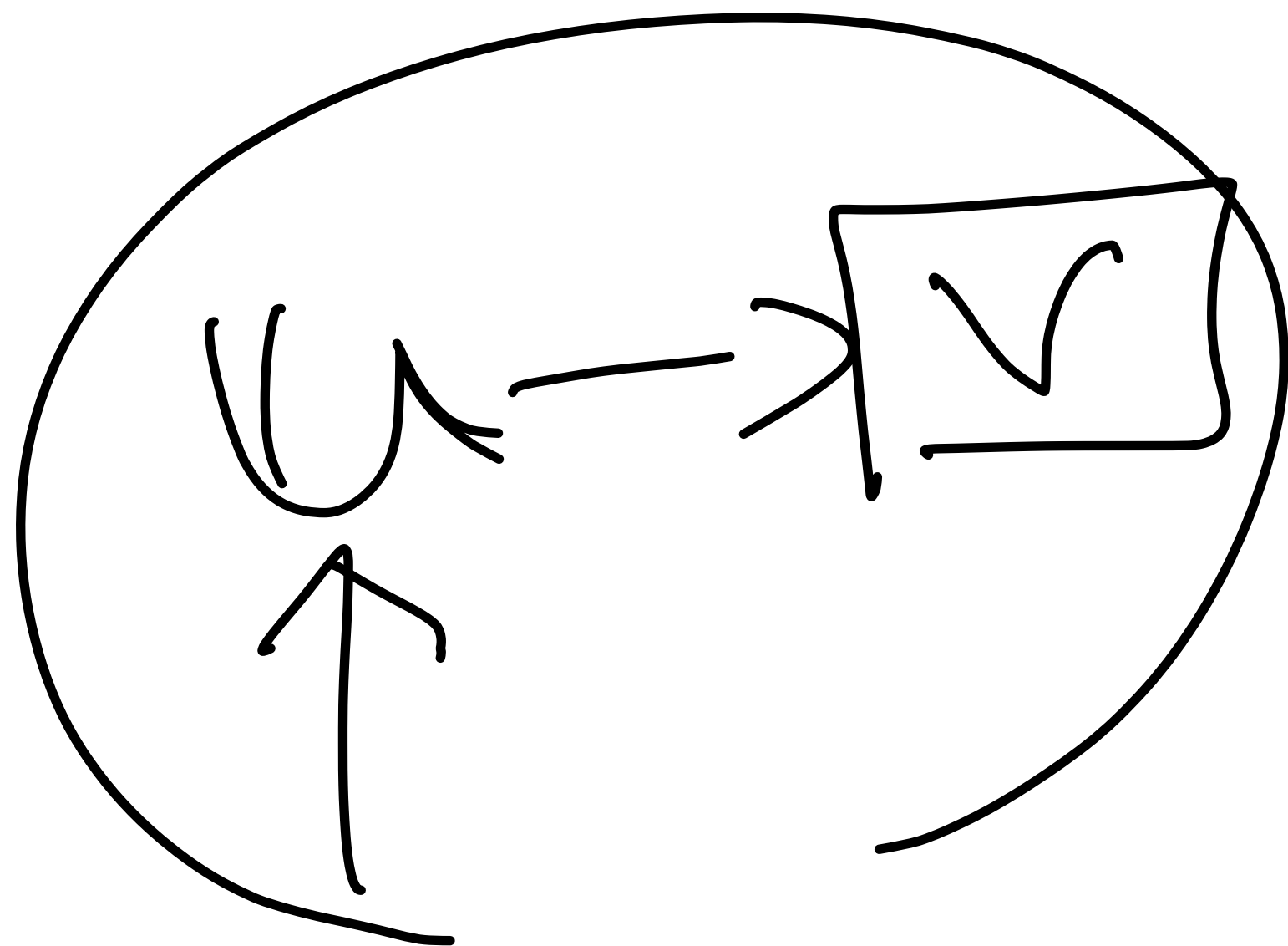
위상 정렬

Topological Sort

5

꼭 이 순서로

- 그래프의 간선 $u \rightarrow v$ 가 u 가 v 보다 먼저라는 의미일 때 정점의 순서를 찾는 알고리즘이다.



신발 \rightarrow 양말

양말 \rightarrow 신발

신발, 양말

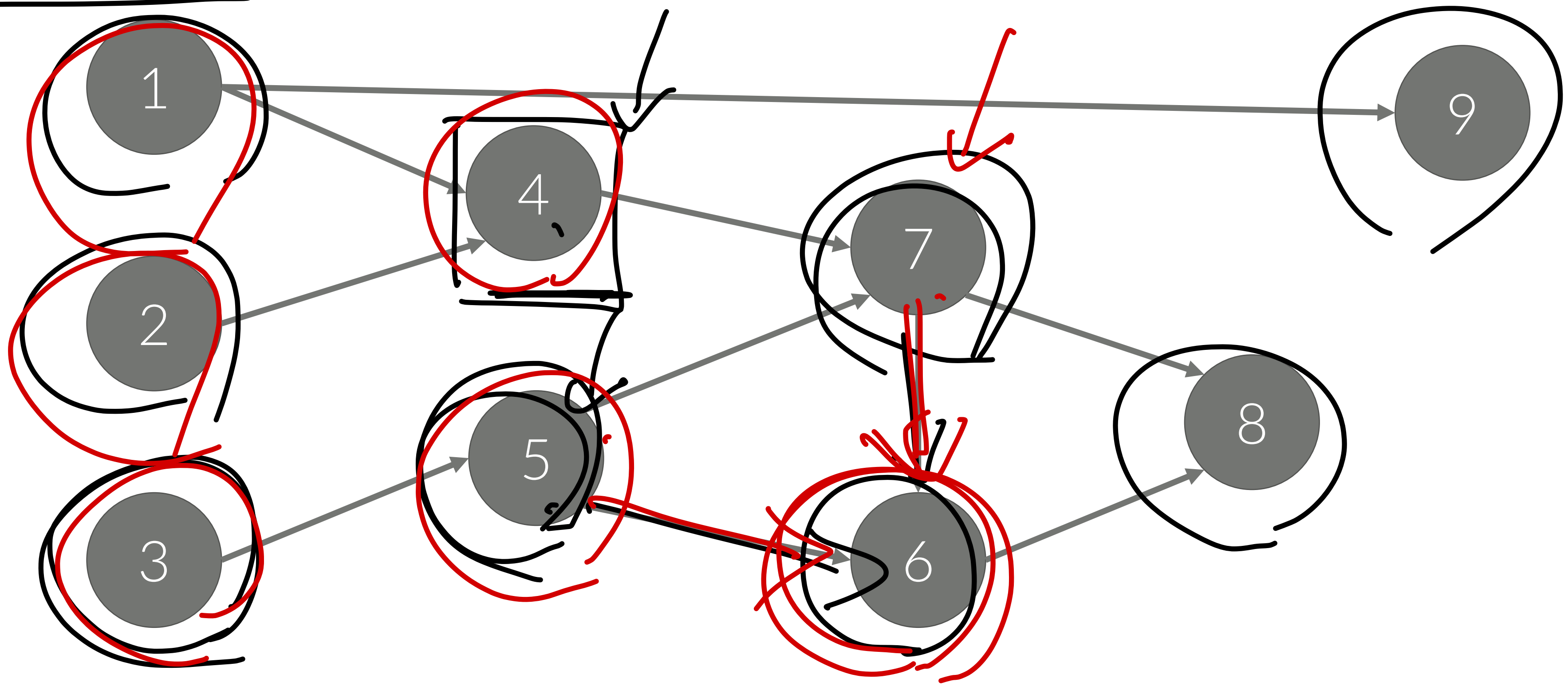
위상 정렬

Topological Sort

~~[1, 2, 3, 4, 5, 6, 7, 8, 9]~~

• 아래 그래프의 위상 정렬 결과는 여러가지가 있다.

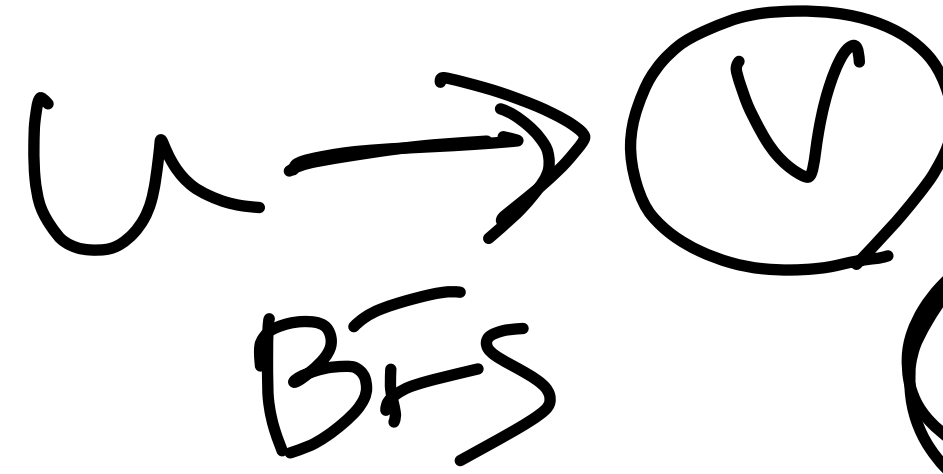
• [1, 2, 3, 4, 5, 7, 6, 8, 9], [1, 2, 9, 3, 5, 4, 7, 6, 8], ...



위상 정렬

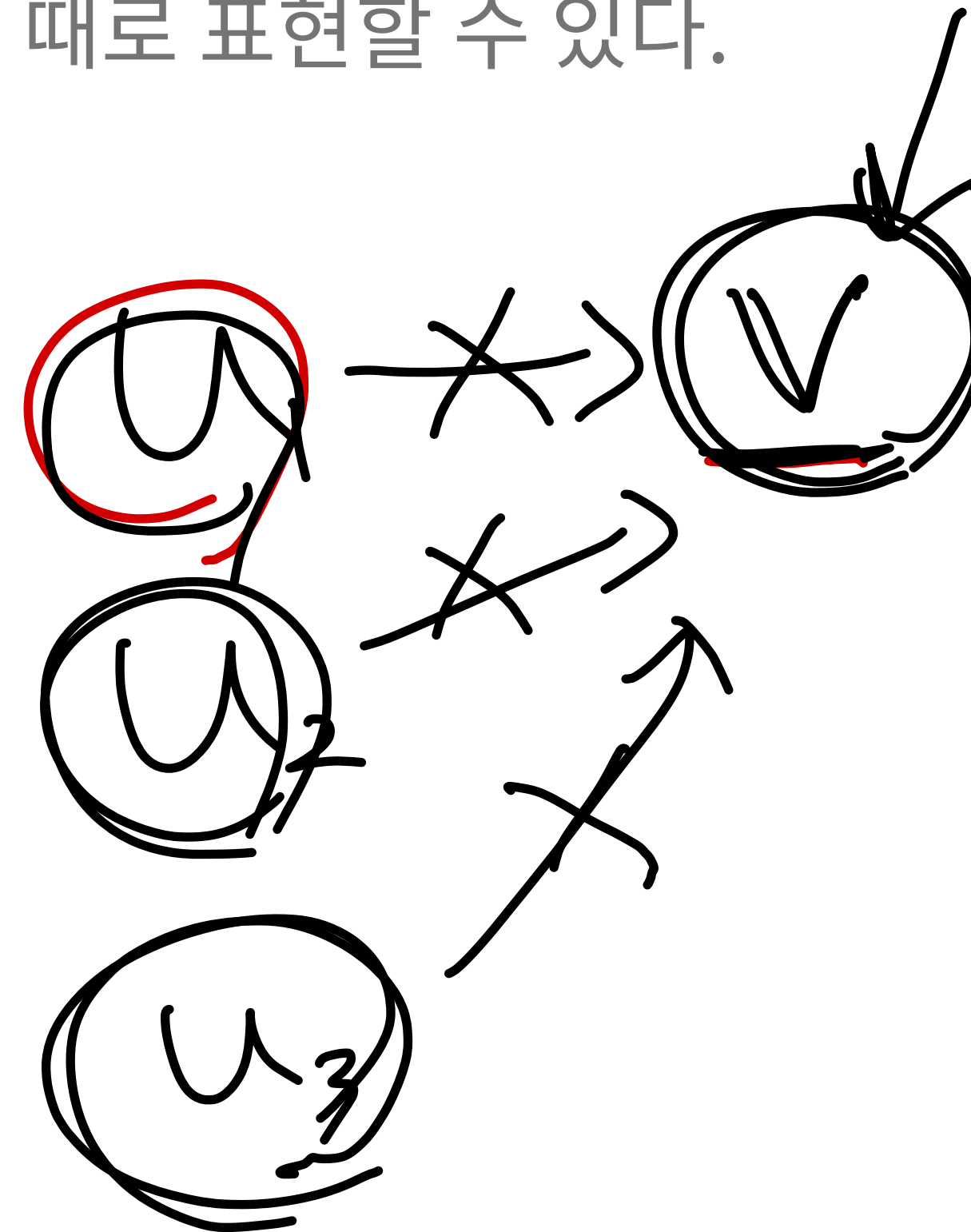
Topological Sort

- 그래프의 간선 $u \rightarrow v$ 가 u 가 v 보다 먼저라는 의미이기 때문에
- 어떤 정점 v 의 순서에 추가되는 것은 $u \rightarrow v$ 의 u 가 모두 순서에 추가되었을 때이다.
- 순서에 추가되는 정점은 in-degree가 0일 때로 표현할 수 있다.
- BFS를 응용해서 구현할 수 있다.



~~방문 X~~

$\text{in-degree} = 0$



위상 정렬

Topological Sort

- BFS 알고리즘을 이용해 위상 정렬을 구현할 수 있고,
- 어떤 정점이 큐에 추가되는 것은 in-degree가 0이 되는 순간이다.



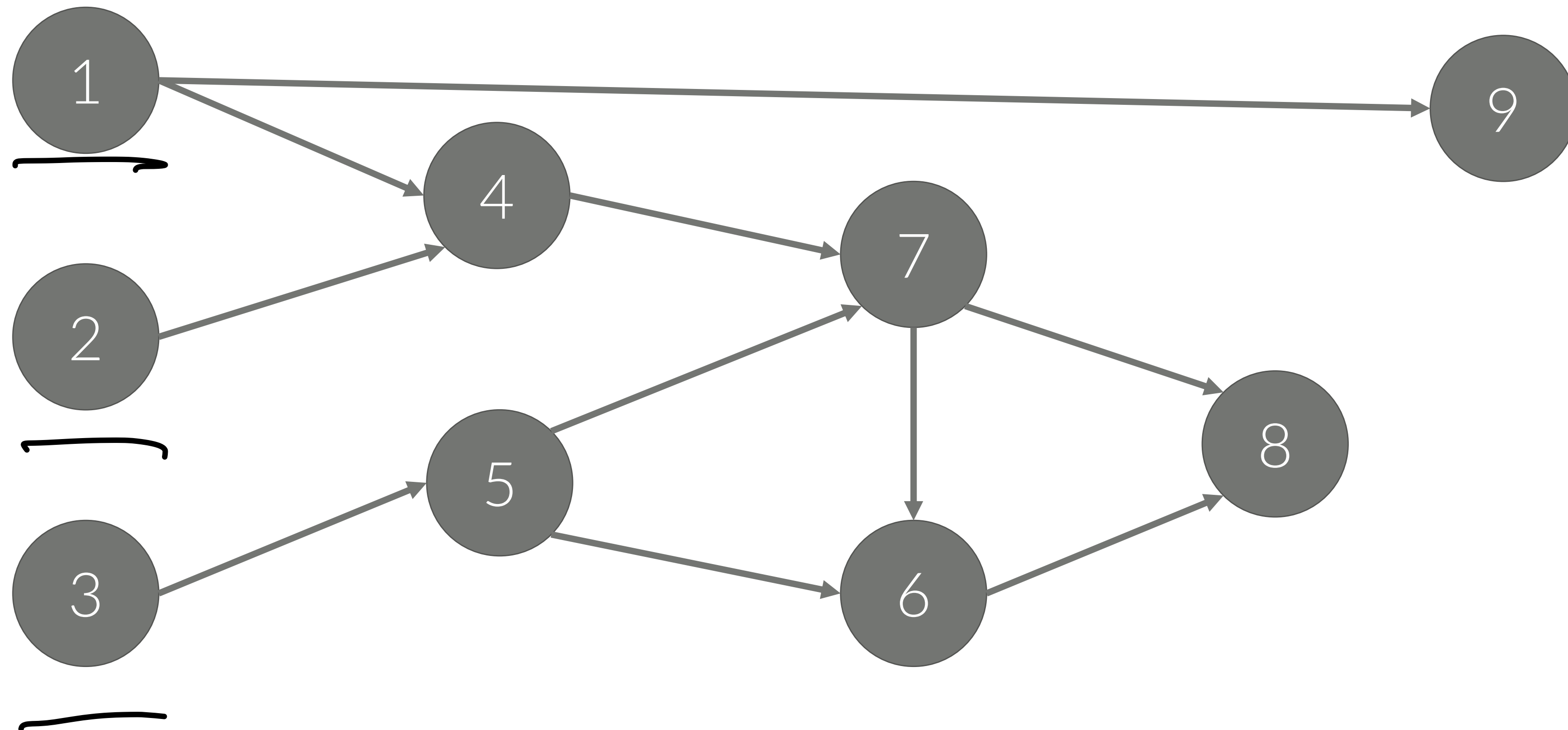
위상 정렬

Topological Sort

- 큐에 가장 들어있는 것은
- 들어오는 간선의 개수가 0인 것이다.

• 순서:

• 큐: 123



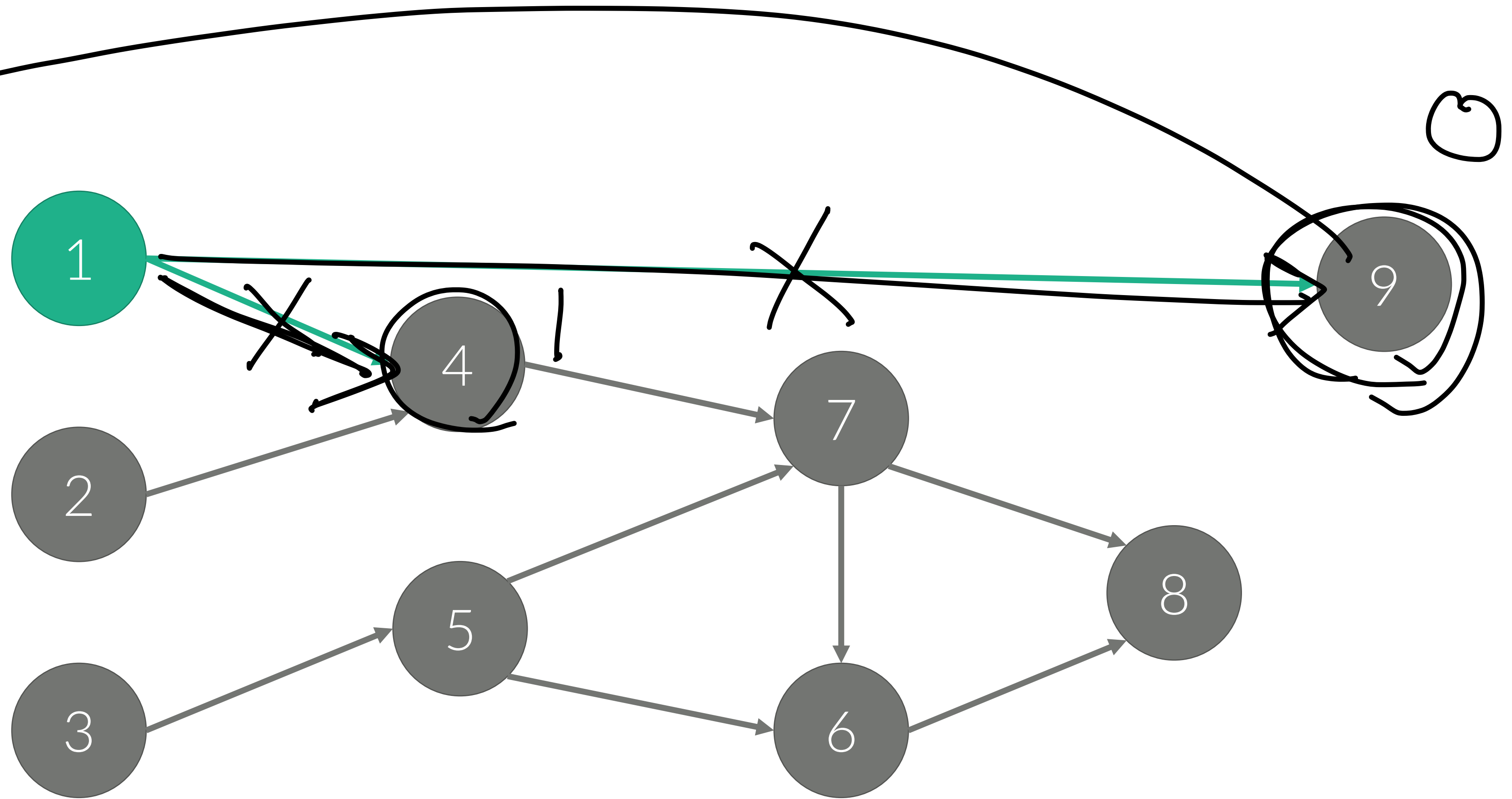
위상 정렬

Topological Sort

10

- 순서: 1

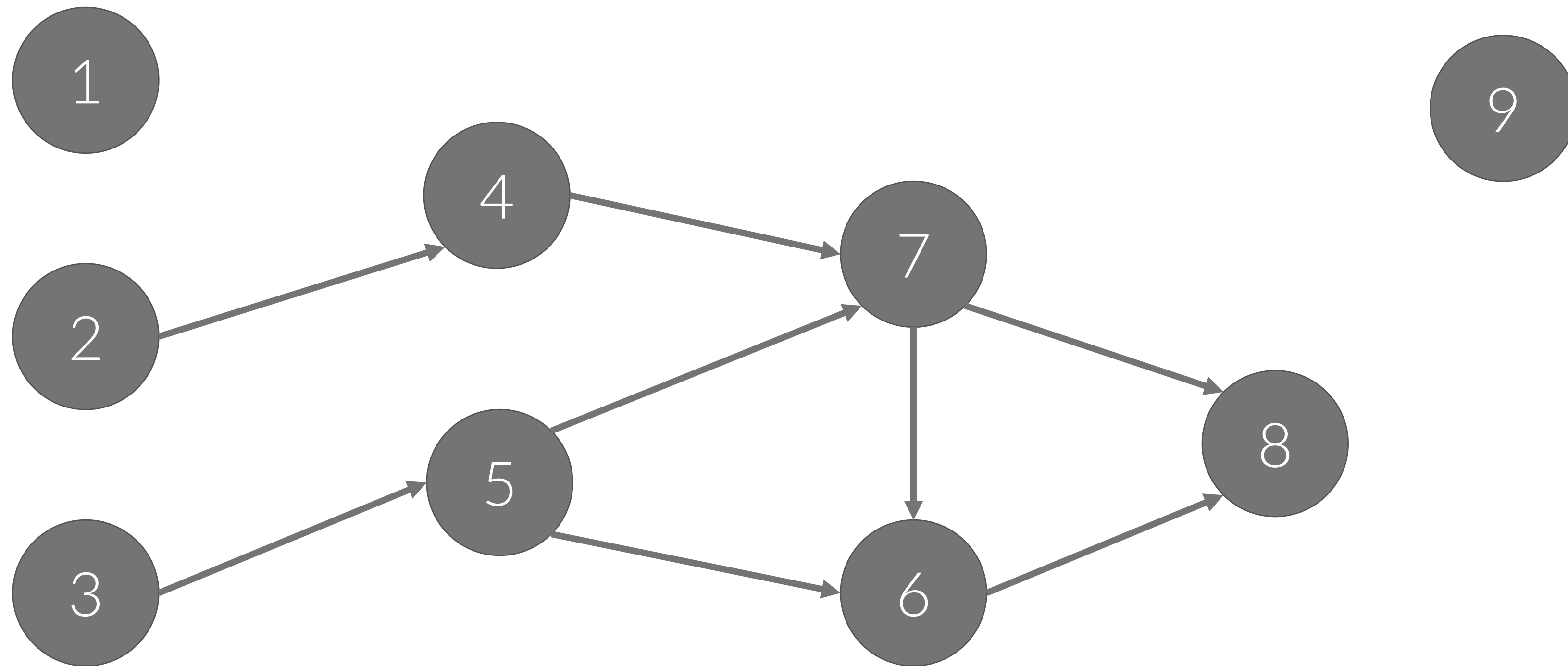
- 큐: 2 3



위상 정렬

Topological Sort

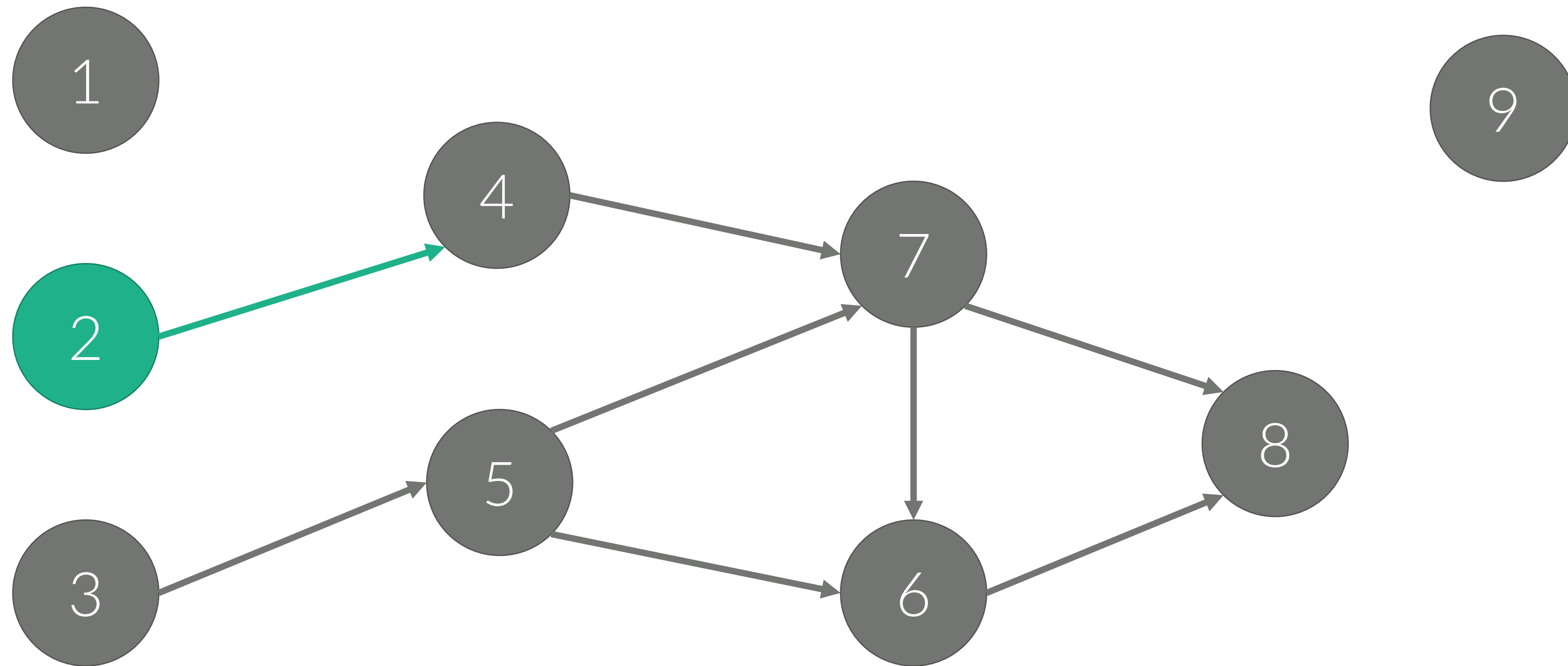
- 순서: 1
- 큐: 2 3 9



위상 정렬

Topological Sort

- 순서: 1 2
- 큐: 3 9



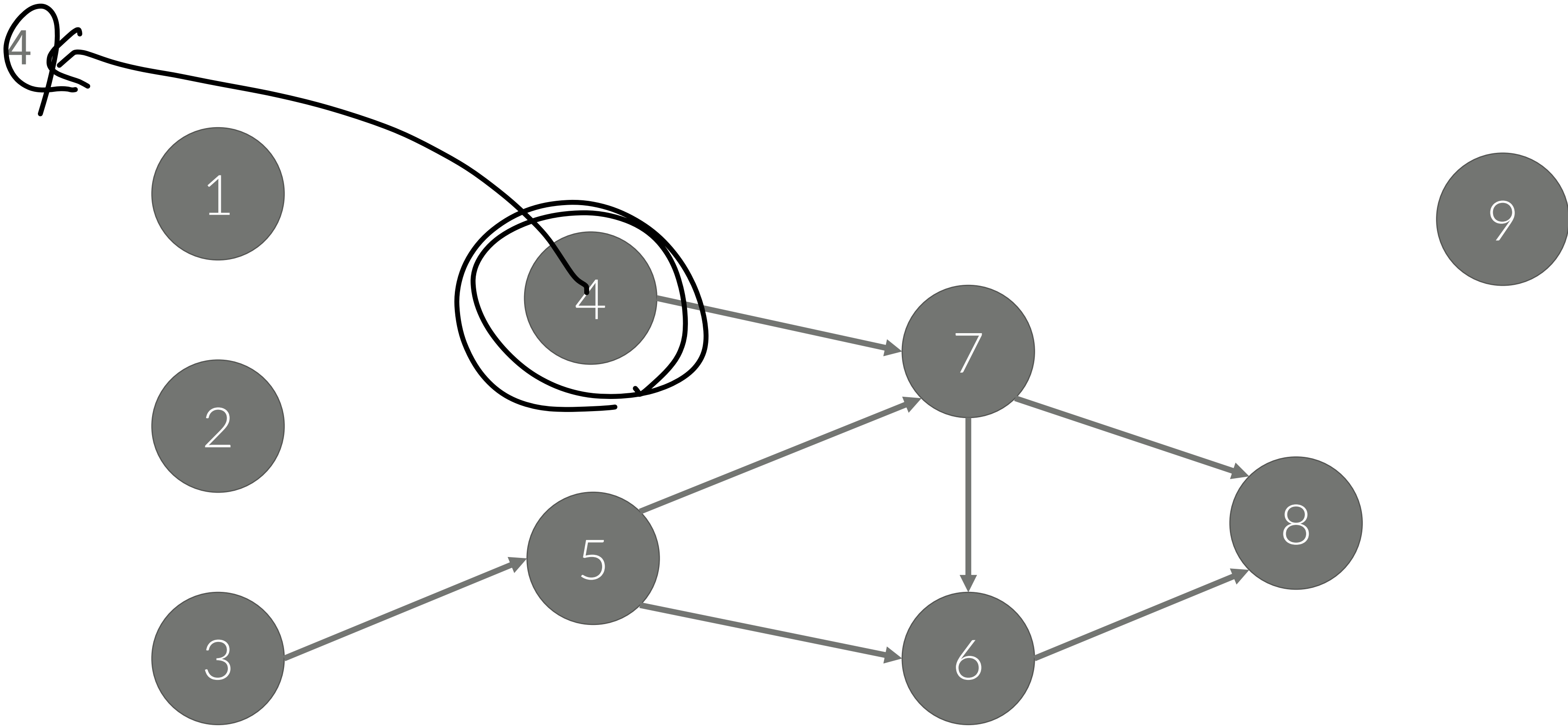
위상 정렬

Topological Sort

13

- 순서: 1 2

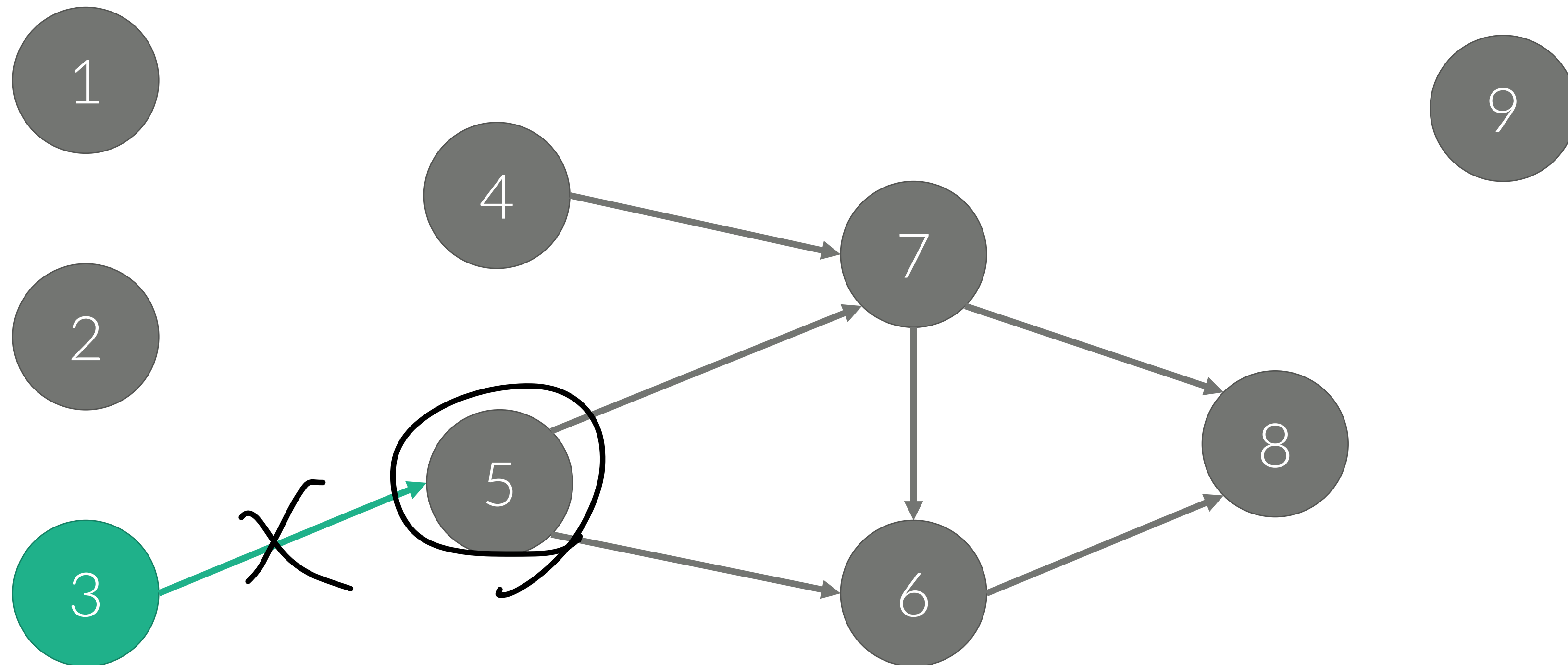
- 큐: 3 9 4



위상 정렬

Topological Sort

- 순서: 1 2 3
- 큐: 9 4



위상 정렬

Topological Sort

• 순서: 1 2 3

• 큐: 9 4 5

Queue
BFS

1

2

3

4

5

7

6

8

9

In-degree 0

우선순위 큐 (작은 값)
Min-Heap

위상 정렬

Topological Sort

- 순서: 1 2 3 9
- 큐: 4 5

1

2

3

4

5

7

6

8

9

위상 정렬

Topological Sort

- 순서: 1 2 3 9
- 큐: 4 5

1

2

3

4

5

7

6

8

9

위상 정렬

Topological Sort

- 순서: 1 2 3 9 4
- 큐: 5

1

2

3

4

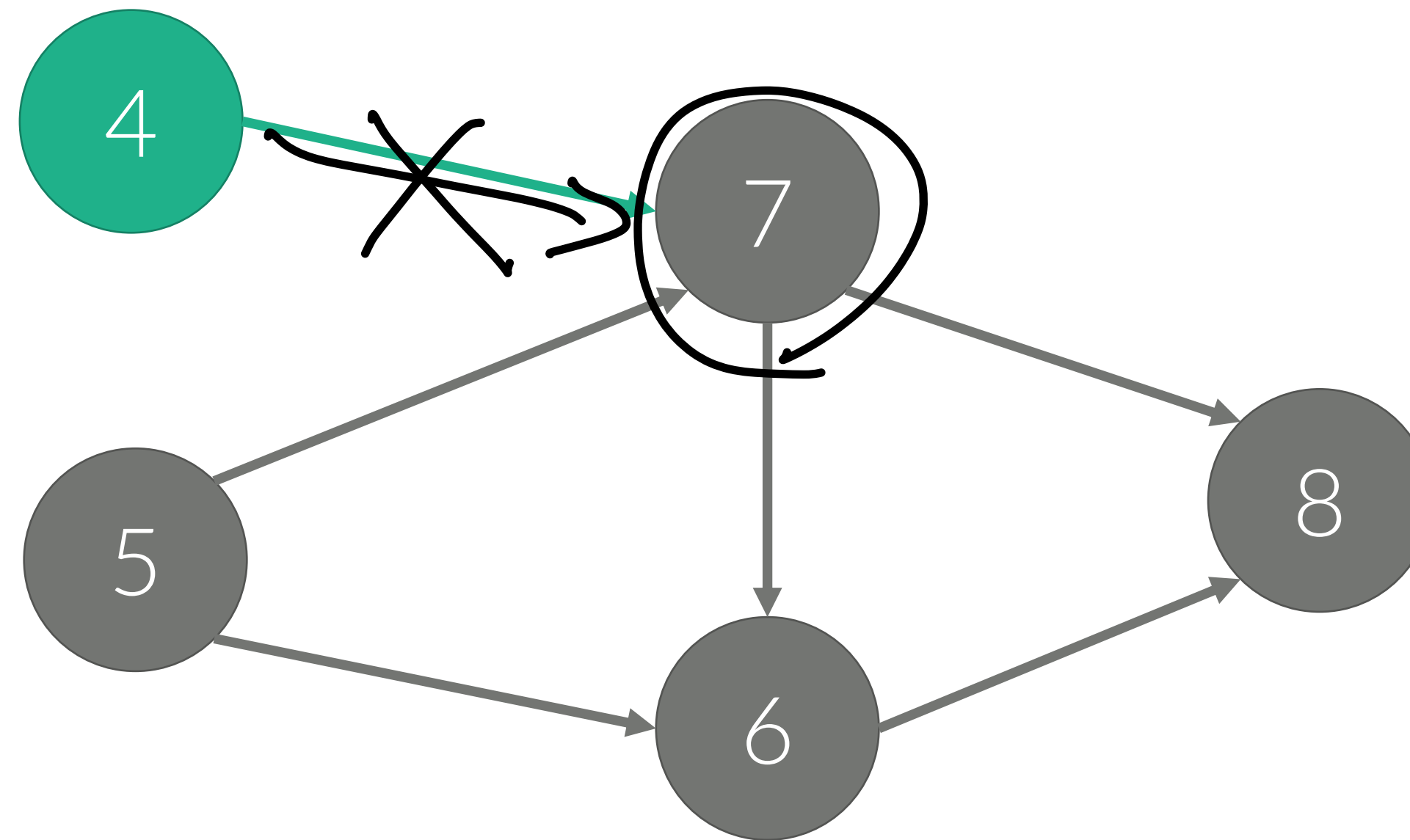
5

7

6

8

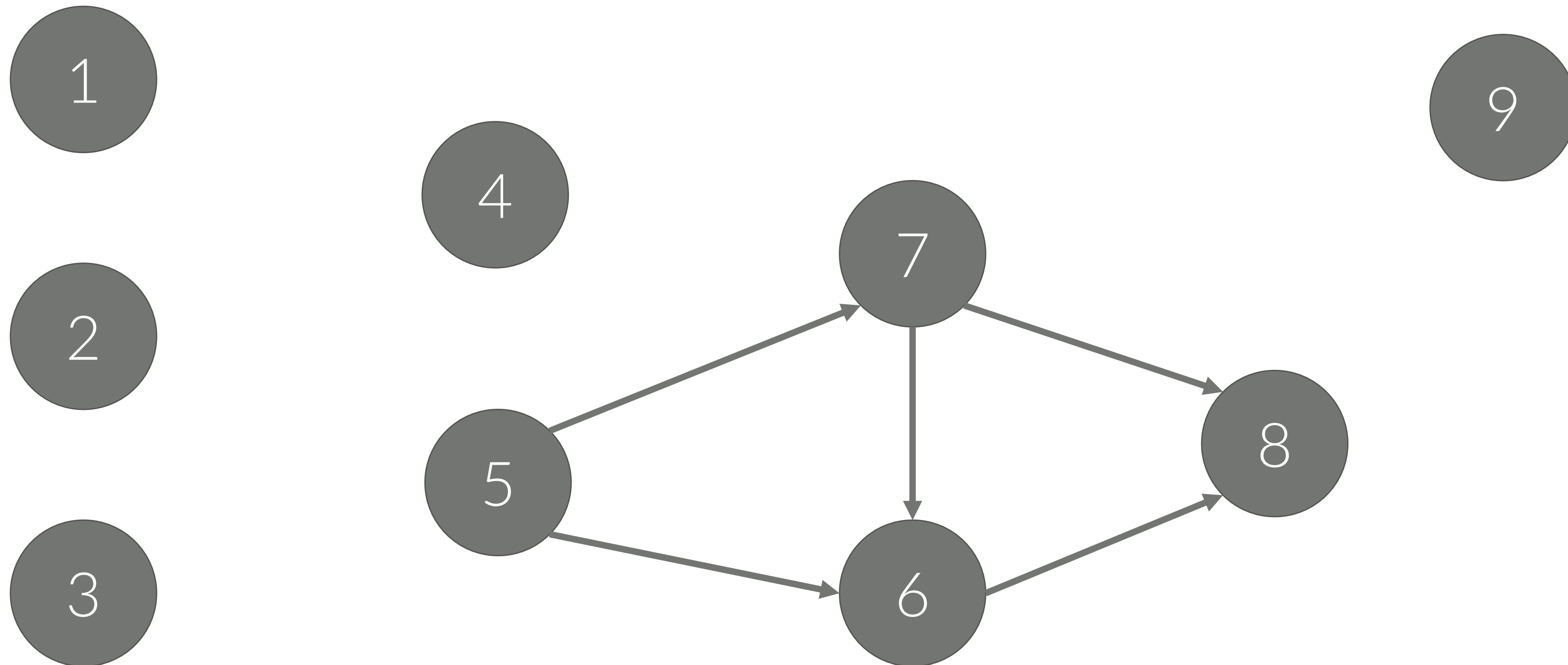
9



위상 정렬

Topological Sort

- 순서: 1 2 3 9 4
- 큐: 5



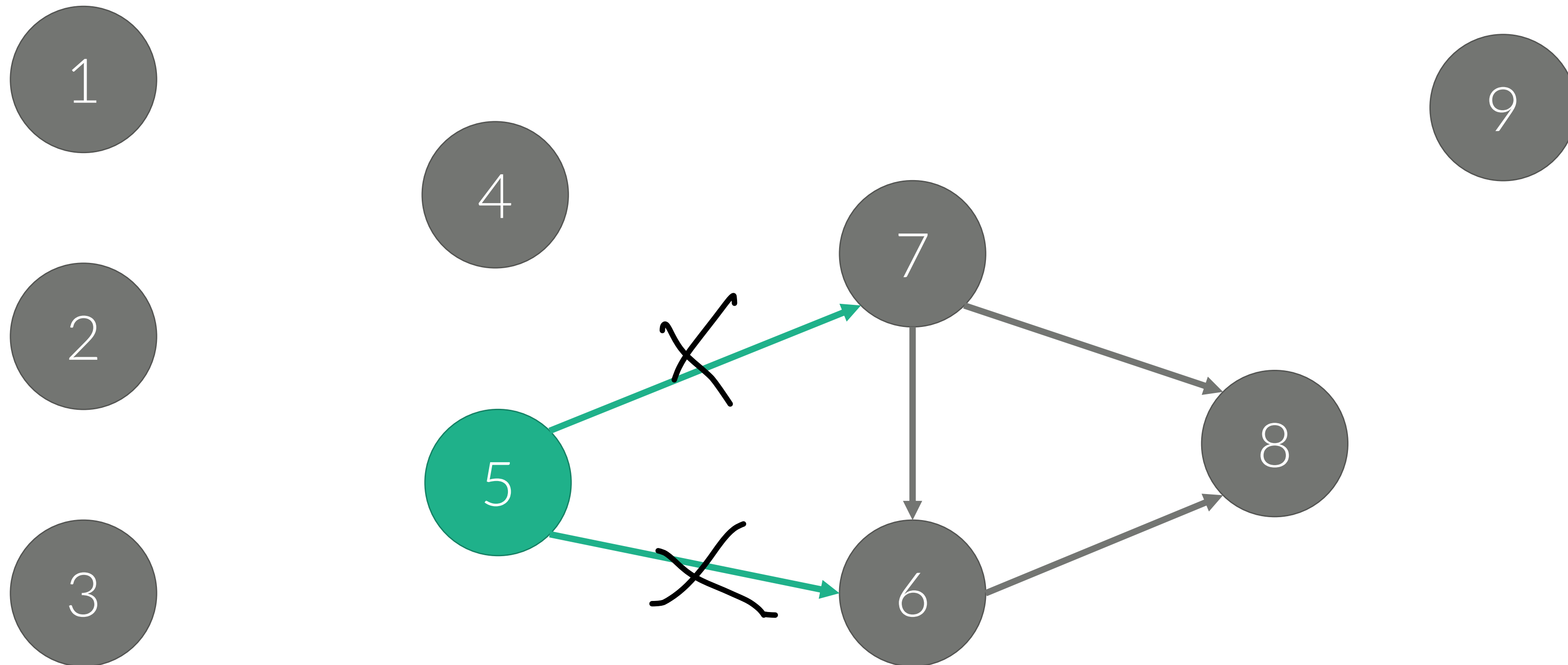
위상 정렬

Topological Sort

20

- 순서: 1 2 3 9 4 5

- 큐:

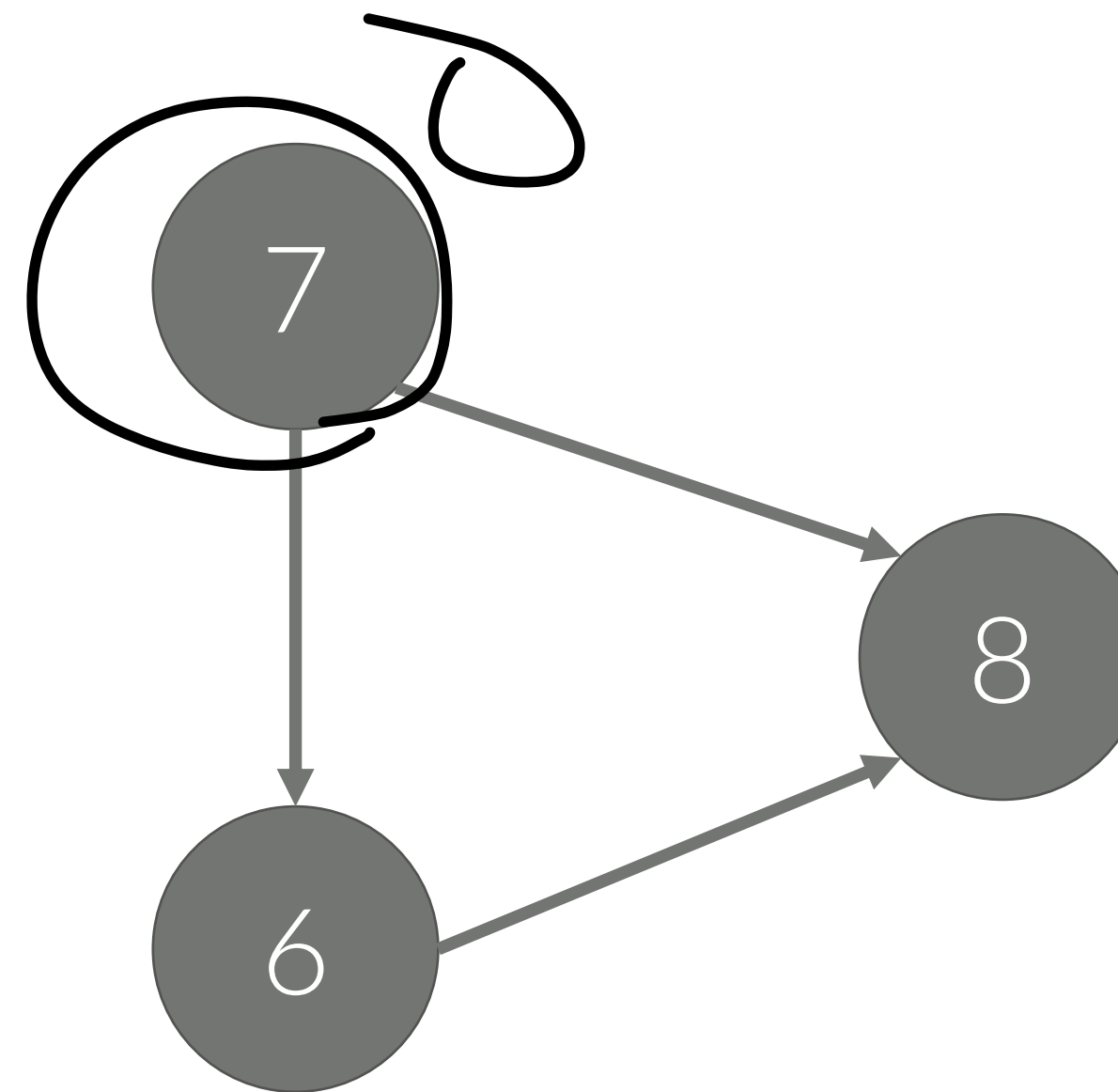


위상 정렬

Topological Sort

- 순서: 1 2 3 9 4 5

- 큐: 7



위상 정렬

Topological Sort

22

- 순서: 1 2 3 9 4 5 7
- 큐:

1

2

3

4

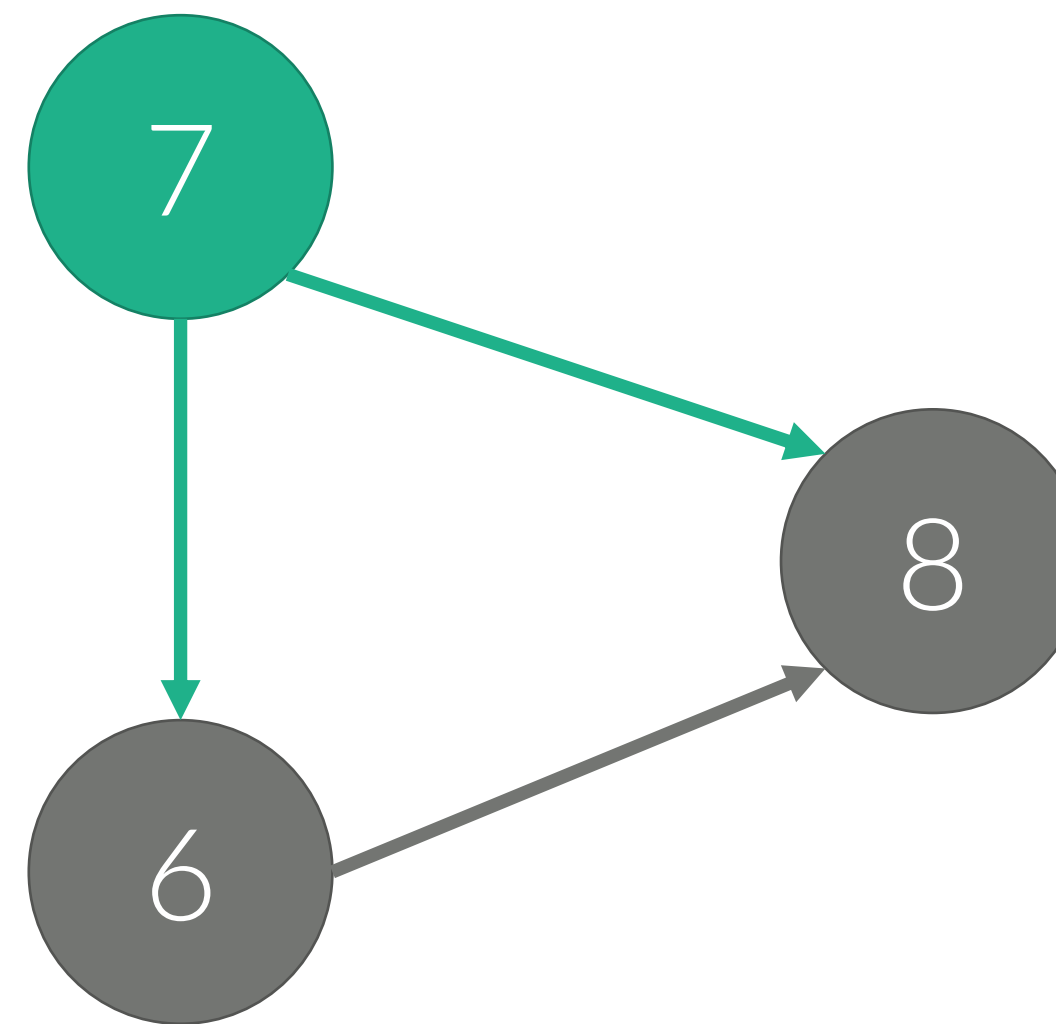
5

7

6

8

9

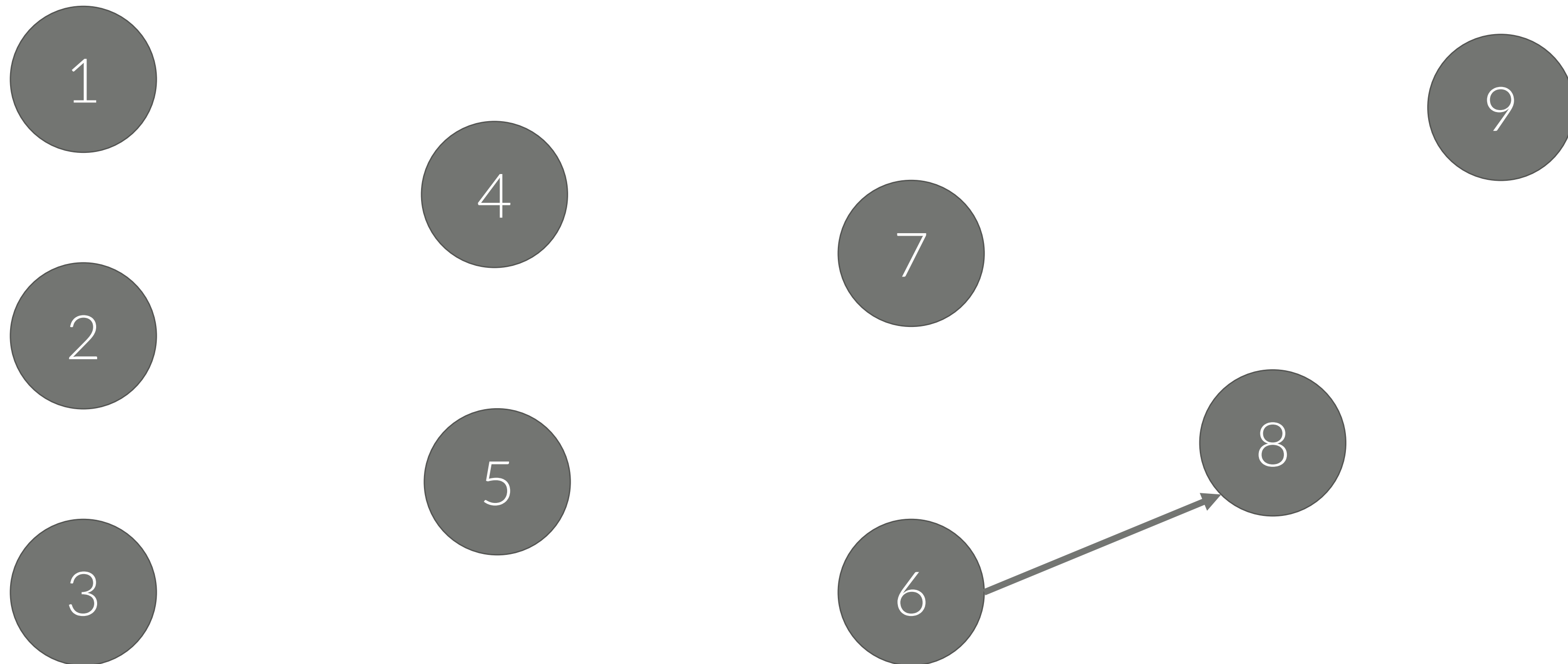


위상 정렬

Topological Sort

23

- 순서: 1 2 3 9 4 5 7
- 큐: 6

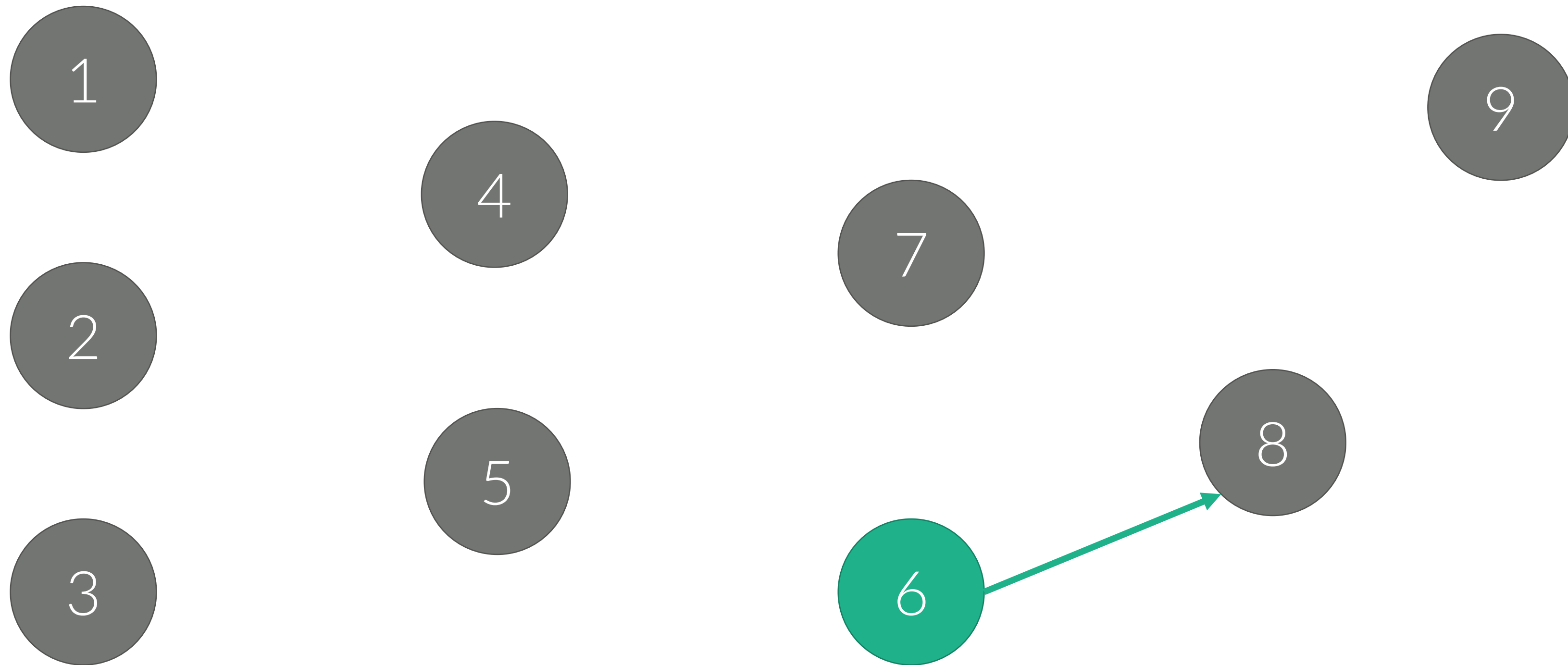


위상 정렬

Topological Sort

24

- 순서: 1 2 3 9 4 5 7 6
- 큐:



위상 정렬

Topological Sort

25

- 순서: 1 2 3 9 4 5 7 6
- 큐: 8

1

2

3

4

5

7

6

8

9

위상 정렬

Topological Sort

26

- 순서: 1 2 3 9 4 5 7 6 8

- 큐:

1

2

3

4

5

7

6

8

9

위상 정렬

Topological Sort

27

- 순서: 1 2 3 9 4 5 7 6 8

- 큐:

1

2

3

4

5

7

6

8

9

위상 정렬

Topological Sort

- 가장 처음

```
queue<int> q;  
for (int i=1; i<=n; i++) {  
    if (ind[i] == 0) {  
        q.push(i);  
    }  
}
```

$ind[i] = i$ 의 In-degree

위상 정렬

Topological Sort

```
while (!q.empty()) {
```

```
    int x = q.front();  
    q.pop();
```

```
    printf("%d ", x);
```

```
    for (int i=0; i<a[x].size(); i++) {
```

```
        int y = a[x][i];
```

```
        ind[y] -= 1;
```

```
        if (ind[y] == 0) {
```

```
            q.push(y);
```

```
        }
```

```
    }
```

```
}
```

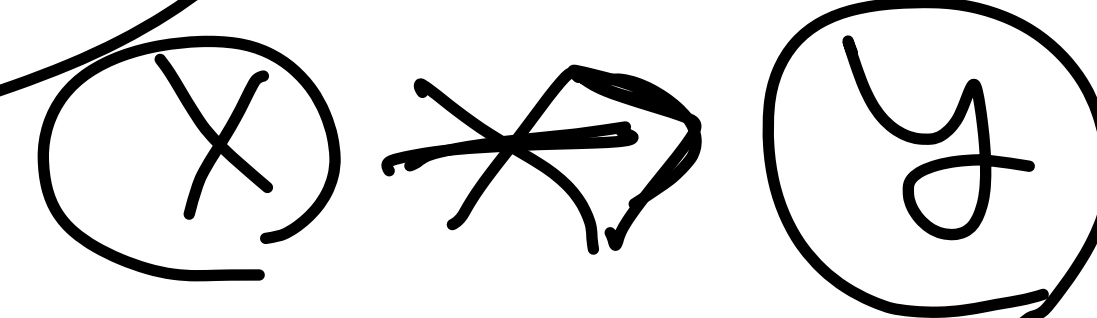
BFS

$O(E)$

pp

모든 간선을 한번씩 검사

~~$O(N+E)$~~



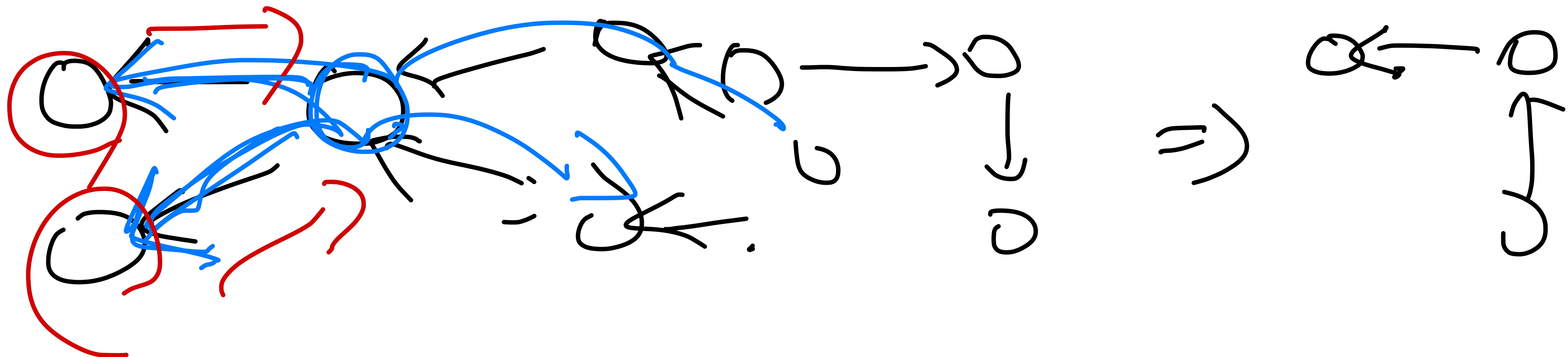
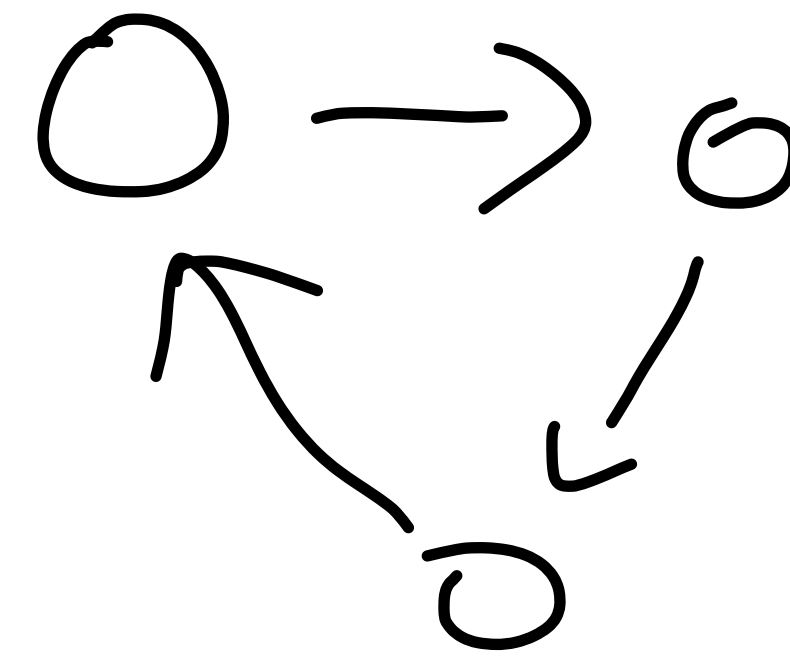
```
if (check[y] == false) {  
    check[y] = true;  
    q.push(y);  
}
```

위상 정렬

Topological Sort

30

- 위상 정렬은 DAG에서만 할 수 있고, DAG는 사이클이 없다.
- 그래프의 간선을 모두 뒤집어놓고 DFS를 수행하고
- 정점이 스택에서 빠져나오는 순서를 기록하면 위상 정렬의 순서와 같아지게 된다.
- 스택에서 빠져나온다는 것은 더 이상 방문할 수 있는 정점이 없다는 것을 의미하고
- 방문할 수 있는 정점이 없다는 것은 이제 이 일을 수행할 수 있다는 것과 같은 의미이다.

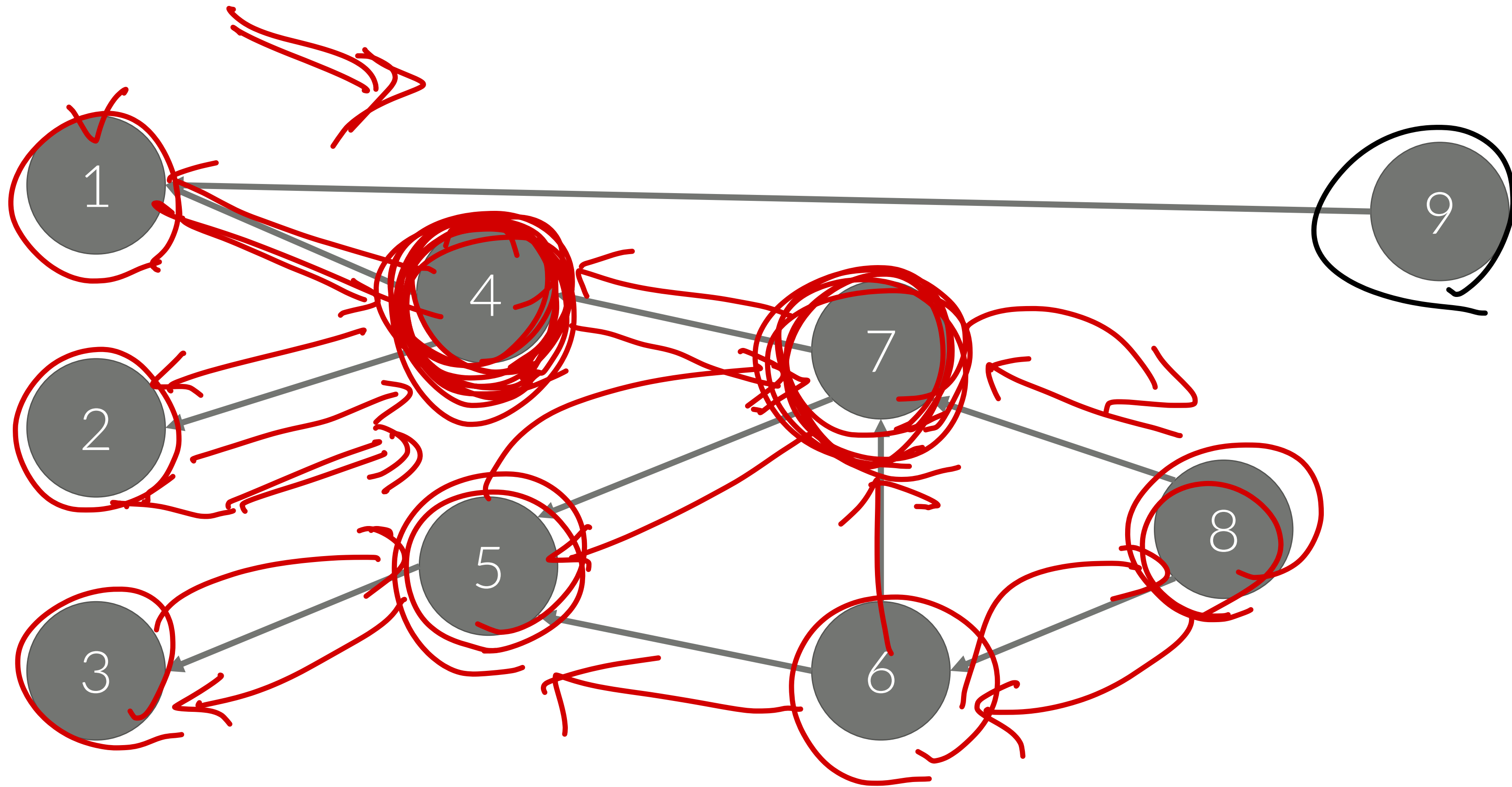


위상 정렬

Topological Sort

31

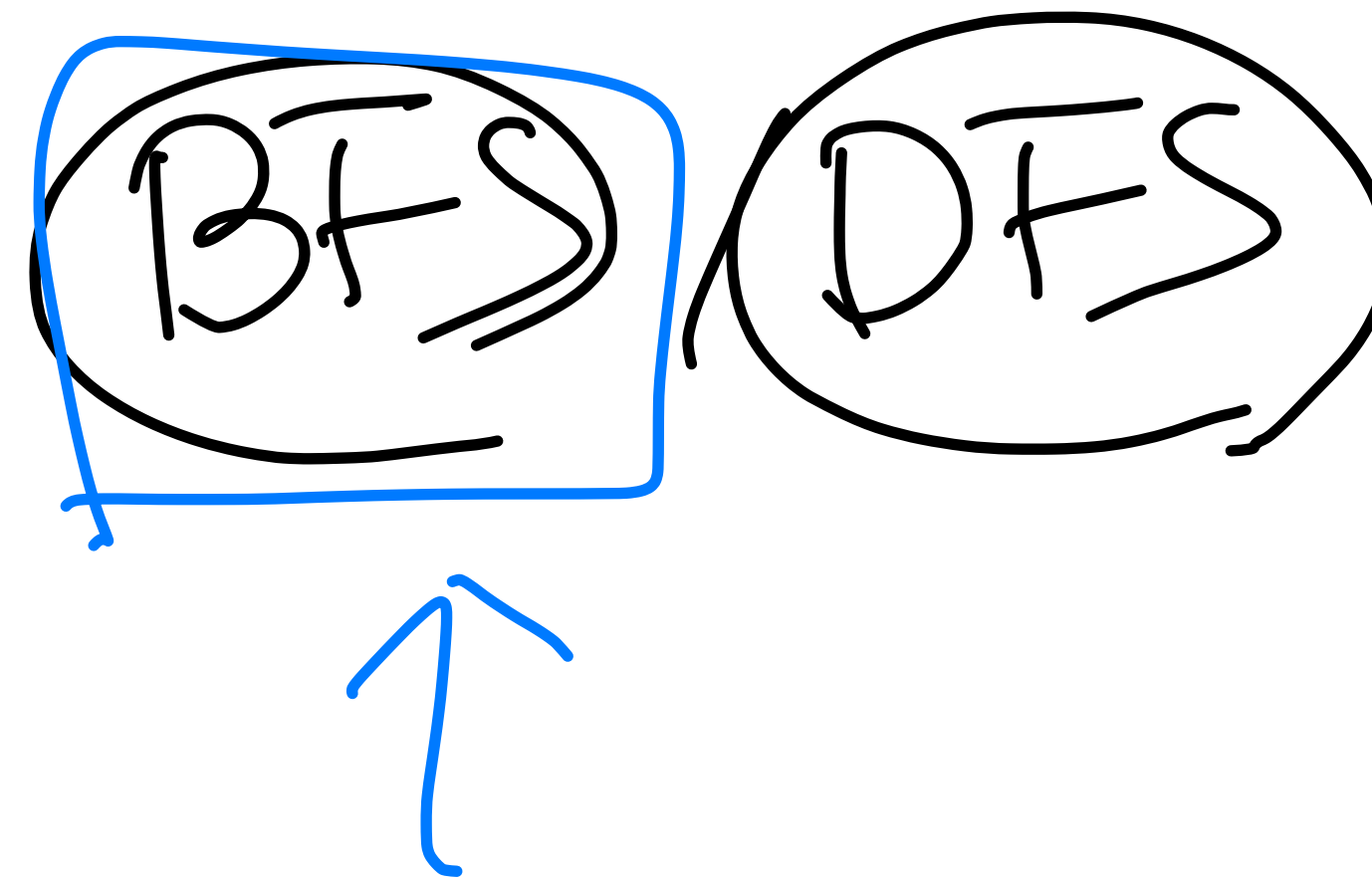
1 2 4 3 5 7 6 8 9



위상 정렬

Topological Sort

```
void go(int x) {  
    check[x] = true;  
    for (int y : a[x]) {  
        if (check[y] == false) {  
            go(y);  
        }  
    }  
    cout << x << ' ';  
}
```

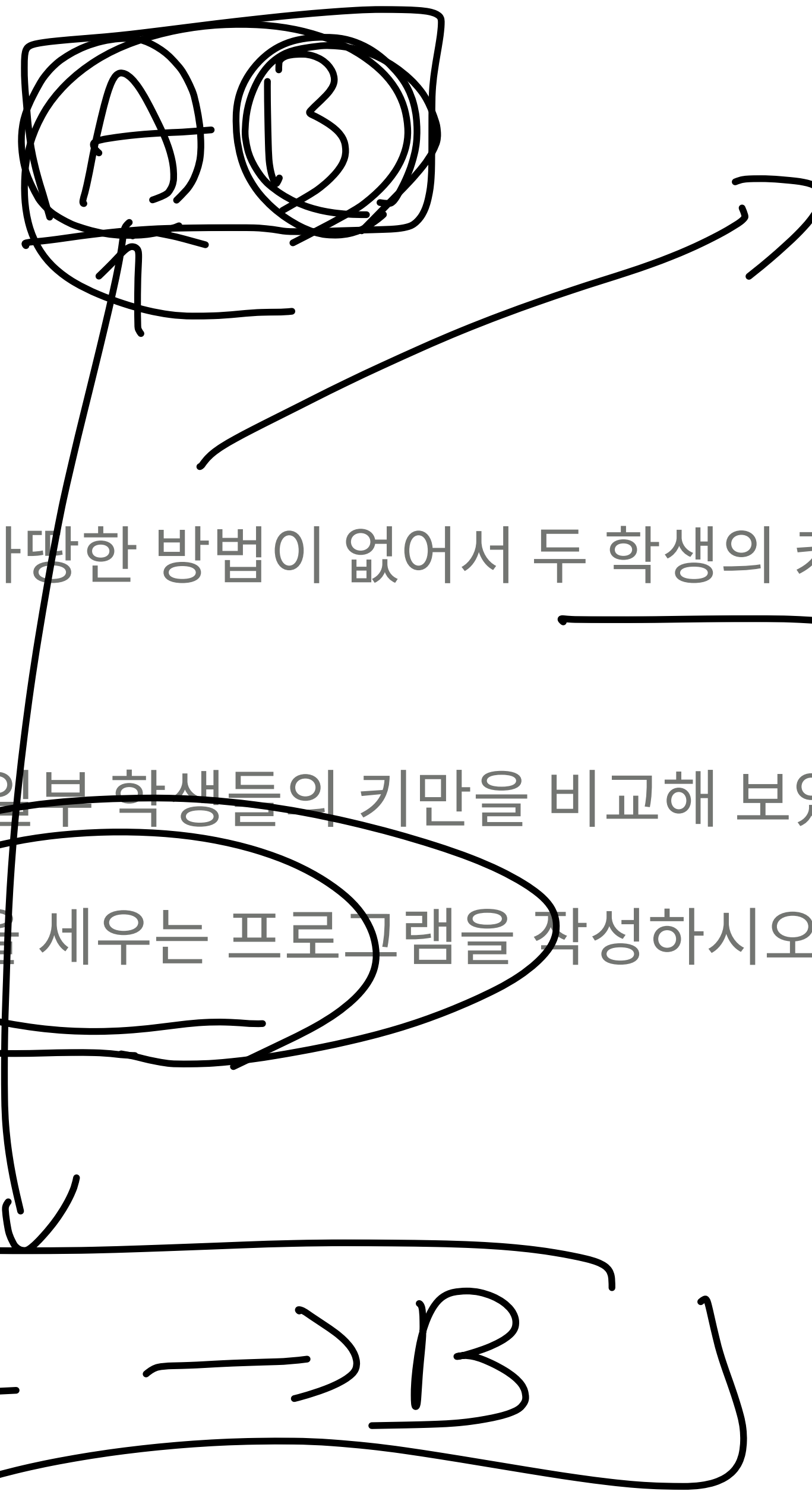


줄 세우기

<https://www.acmicpc.net/problem/2252>

33

- N명의 학생들을 키 순서대로 줄을 세우려고 한다
- 각 학생의 키를 직접 재서 정렬하면 간단하겠지만, 마땅한 방법이 없어서 두 학생의 키를 비교하는 방법을 사용하기로 하였다
- 그나마도 모든 학생들을 다 비교해 본 것이 아니고, 일부 학생들의 키만을 비교해 보았다.
- 일부 학생들의 키를 비교한 결과가 주어졌을 때, 줄을 세우는 프로그램을 작성하시오.



줄 세우기

<https://www.acmicpc.net/problem/2252>

- DFS 소스: <http://codeplus.codes/f97220b5d23541699cb143496c901863>
- BFS 소스: <http://codeplus.codes/00307b954a134908afa8a60f5d1f9a44>

문제집

위상정렬

① 3

35

<https://www.acmicpc.net/problem/1766>

- 1번부터 N번까지 문제를 풀려고 한다. 문제 번호는 난이도 순이다.
- 먼저 푸는 것이 좋은 문제가 있다. 아래 3가지 조건에 따라 문제 풀 순서를 정하기로 했다.

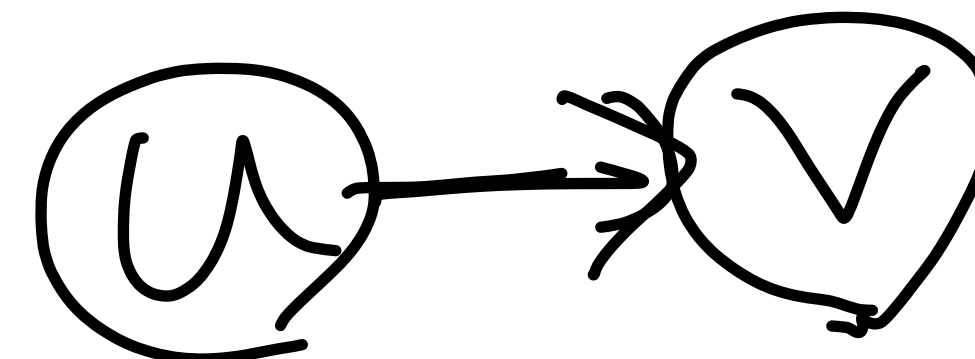
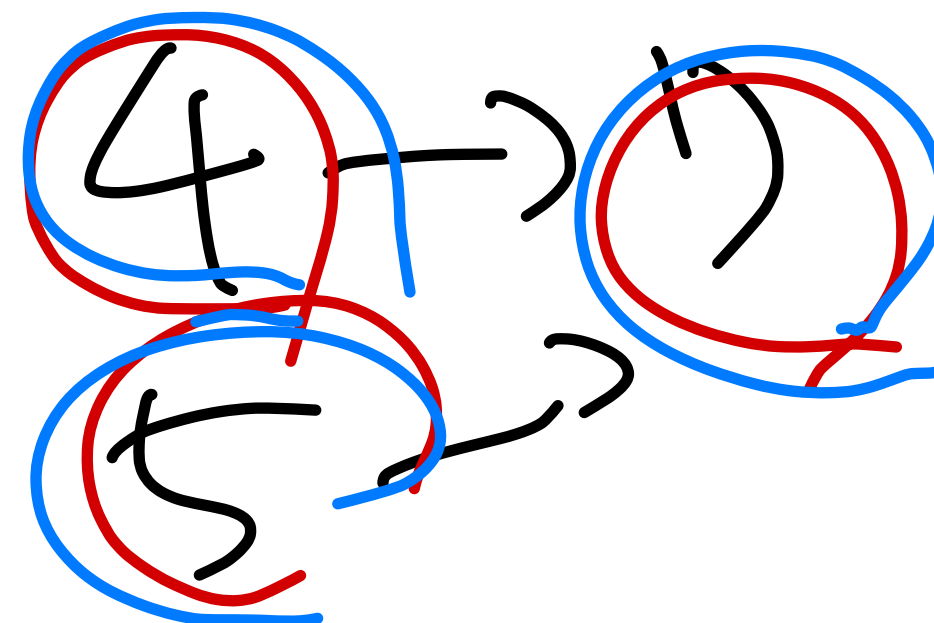


→ 1. N개의 문제는 모두 풀어야 한다.

→ 2. 먼저 푸는 것이 좋은 문제가 있는 문제는, 먼저 푸는 것이 좋은 문제를 반드시 먼저 풀어야 한다.

3. 가능하면 쉬운 문제부터 풀어야 한다.

- 문제를 푸는 순서를 결정해주는 문제



문제집

<https://www.acmicpc.net/problem/1766>

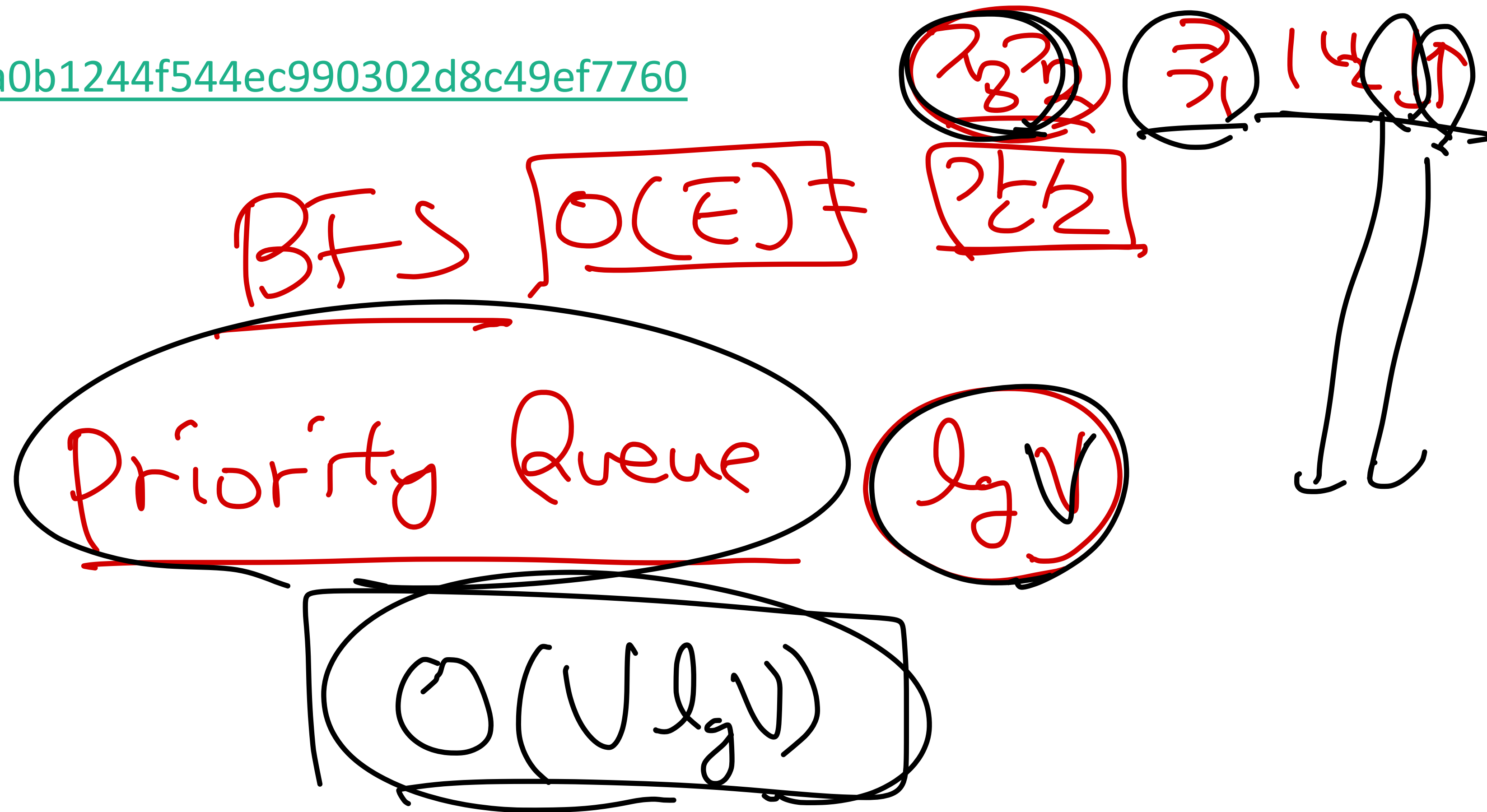
- 3번 조건 때문에, 위상 정렬 순서 아무거나 구하면 안된다.
- 사전 순으로 가장 앞서는 것을 구해야 한다.
- 따라서, 큐 대신 우선순위 큐를 사용해 가장 작은 수를 큐에서 pop하는 방식으로 구현할 수 있다.

문제집

37

<https://www.acmicpc.net/problem/1766>

- 소스: <http://codeplus.codes/91a0b1244f544ec990302d8c49ef7760>



작업

<https://www.acmicpc.net/problem/2056>

- 작업의 선행관계가 주어졌을 때, 모두 마치는 가장 빠른 시간
- 위상 정렬을 응용해서 풀 수 있다

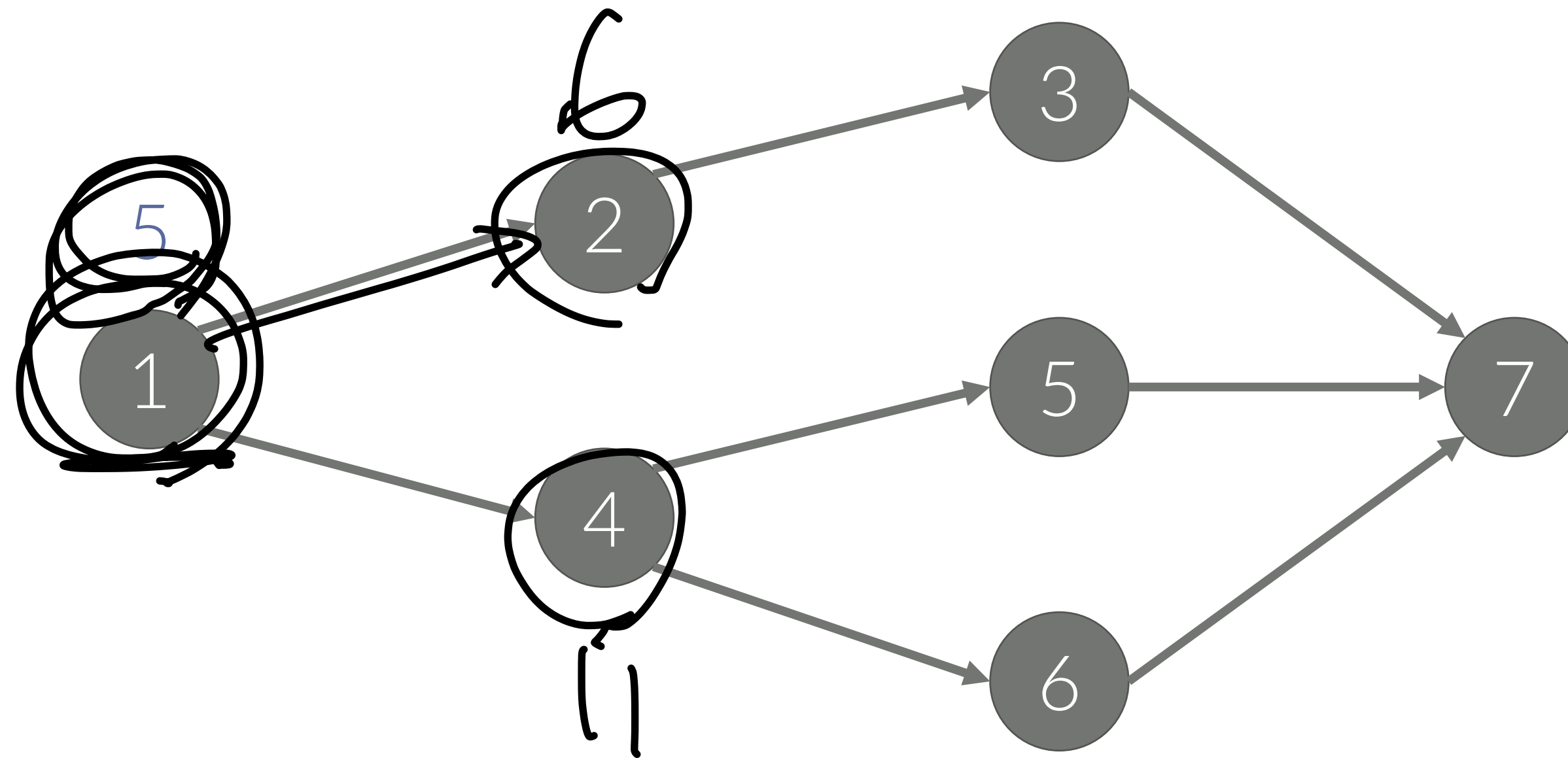
작업

39

<https://www.acmicpc.net/problem/2056>

- 큐: 1
- 현재 작업:

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4



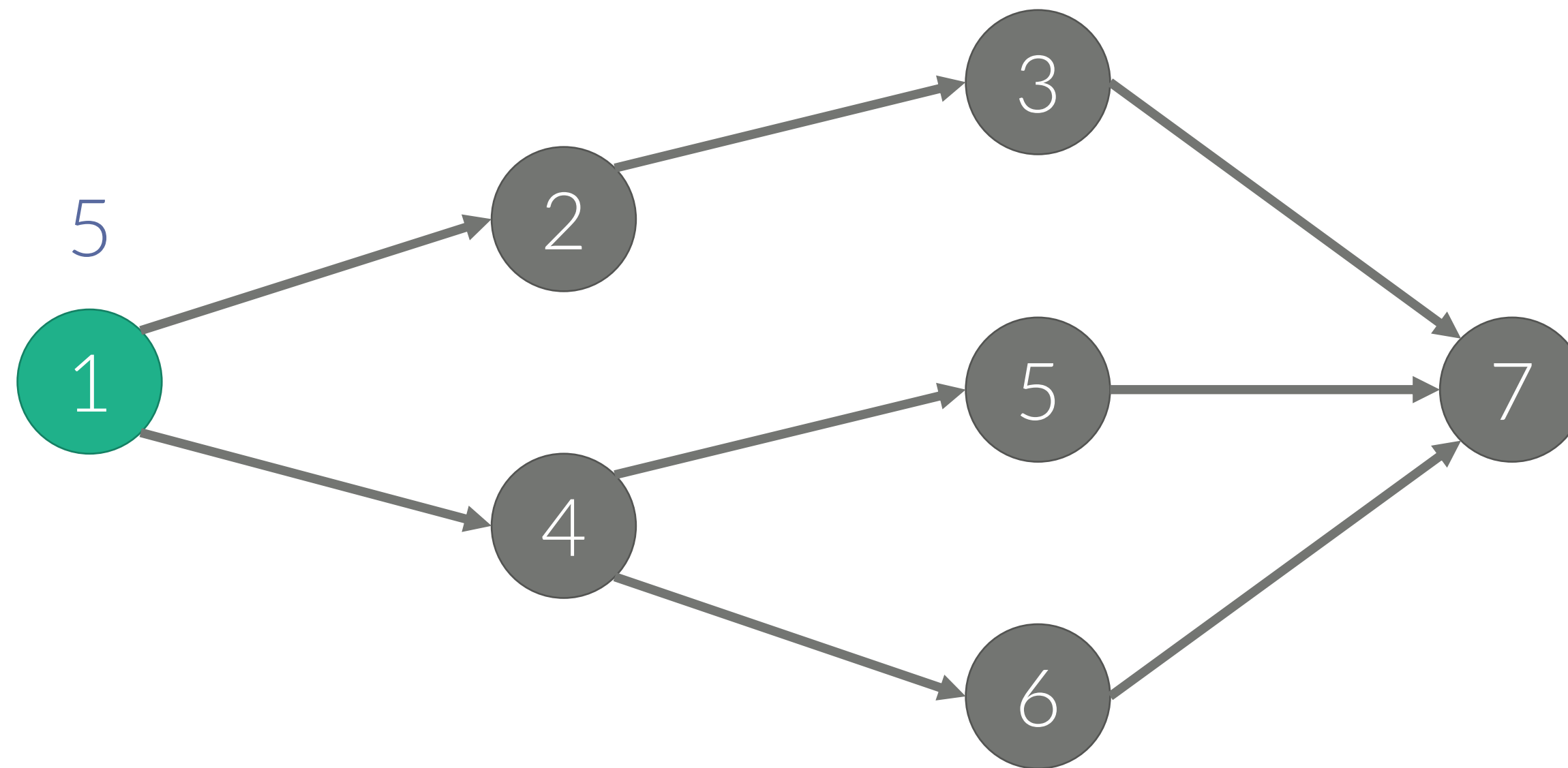
작업

40

<https://www.acmicpc.net/problem/2056>

- 큐:
- 현재 작업: 1

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4

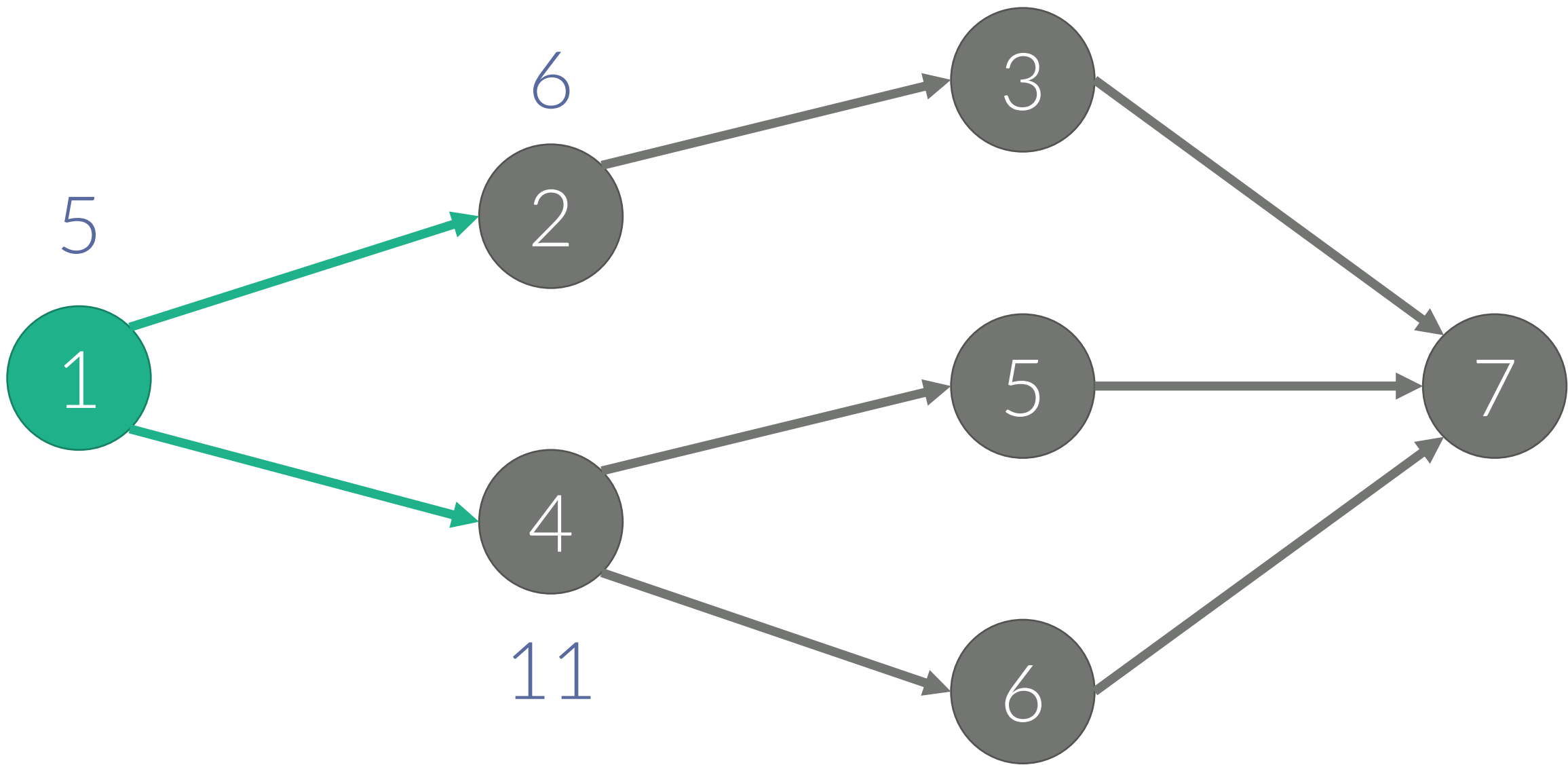


작업

<https://www.acmicpc.net/problem/2056>

- 큐: 2 4
- 현재 작업: 1

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4



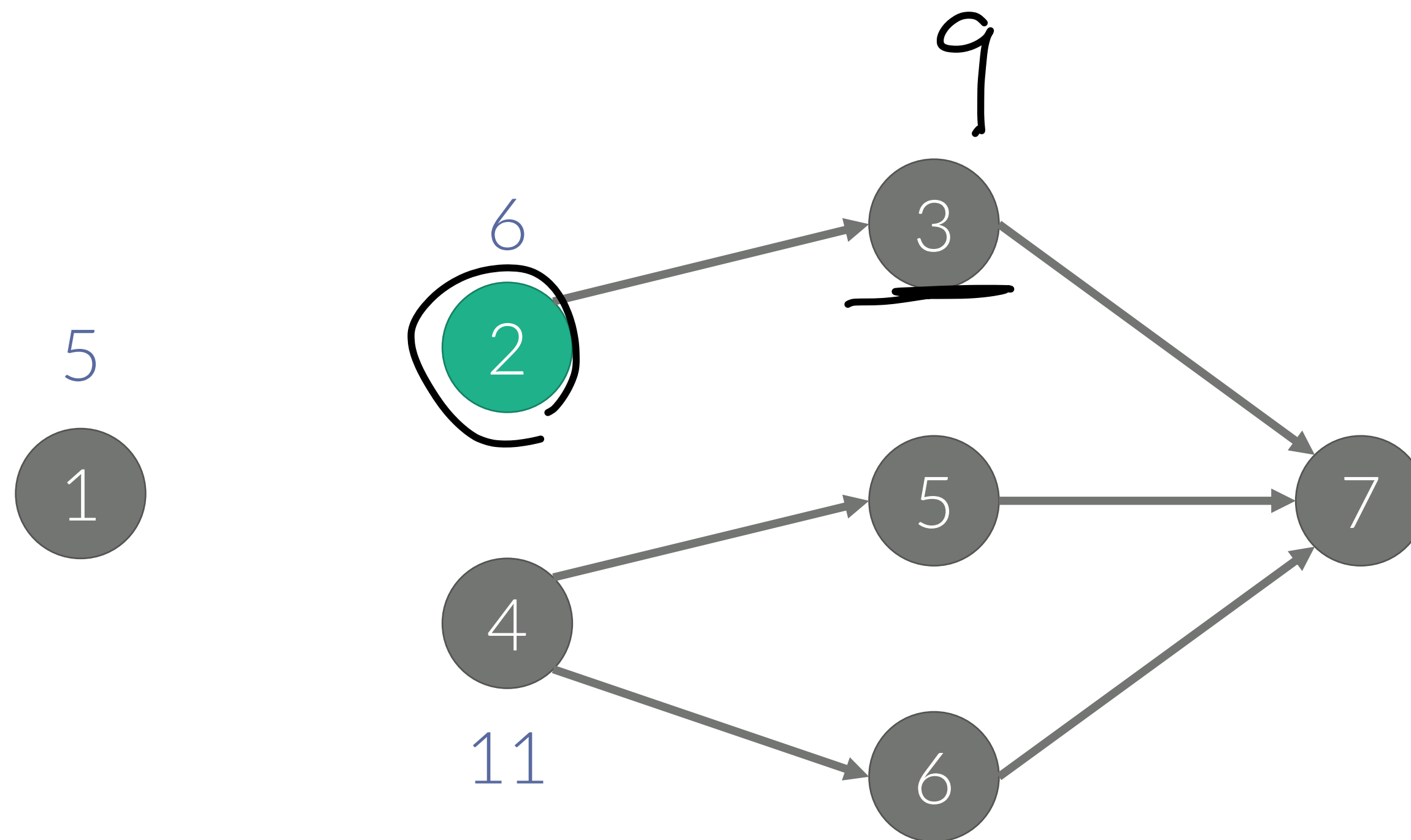
작업

42

<https://www.acmicpc.net/problem/2056>

- 큐: 4
- 현재 작업: 2

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4

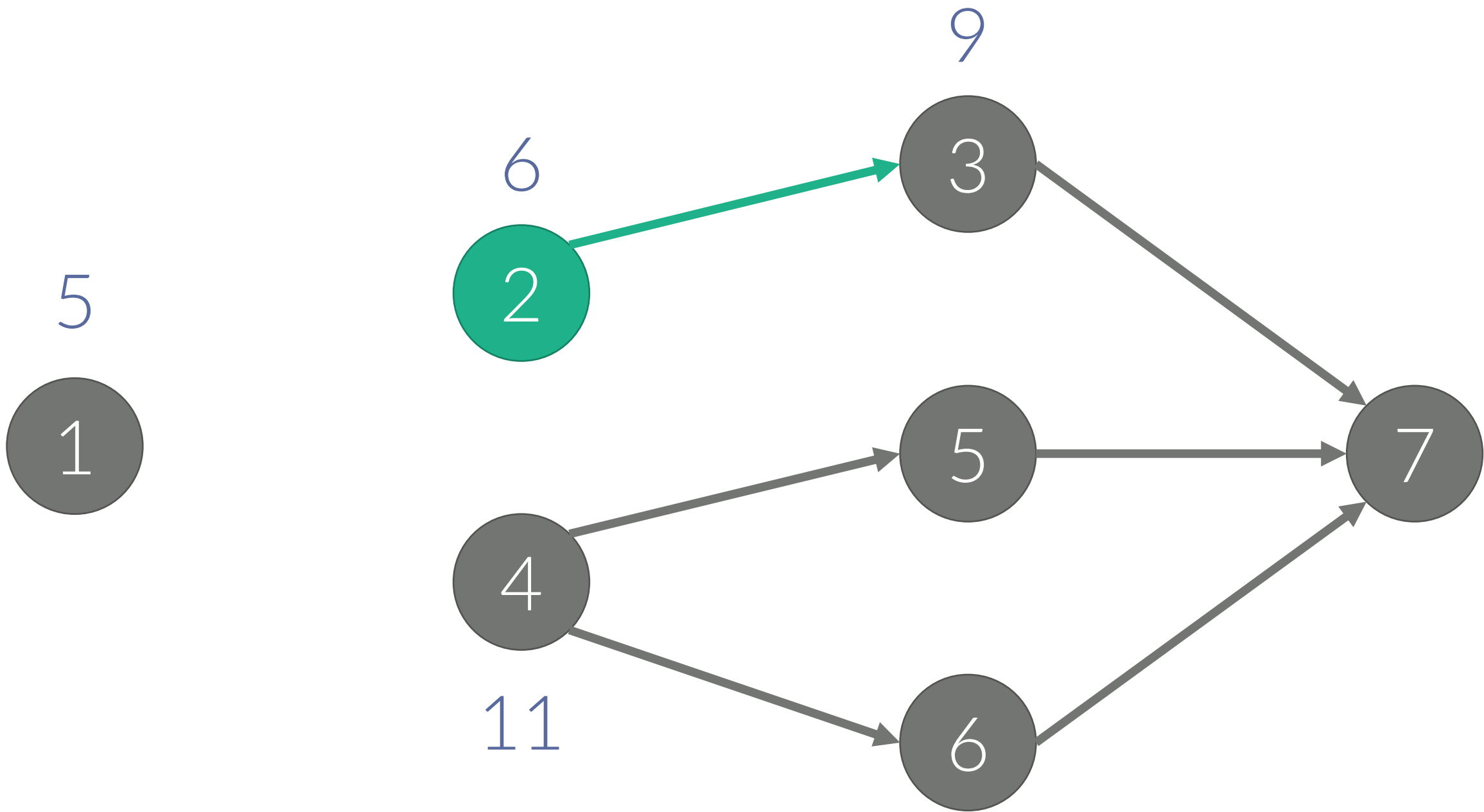


작업

<https://www.acmicpc.net/problem/2056>

- 큐: 4
- 현재 작업: 2

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4



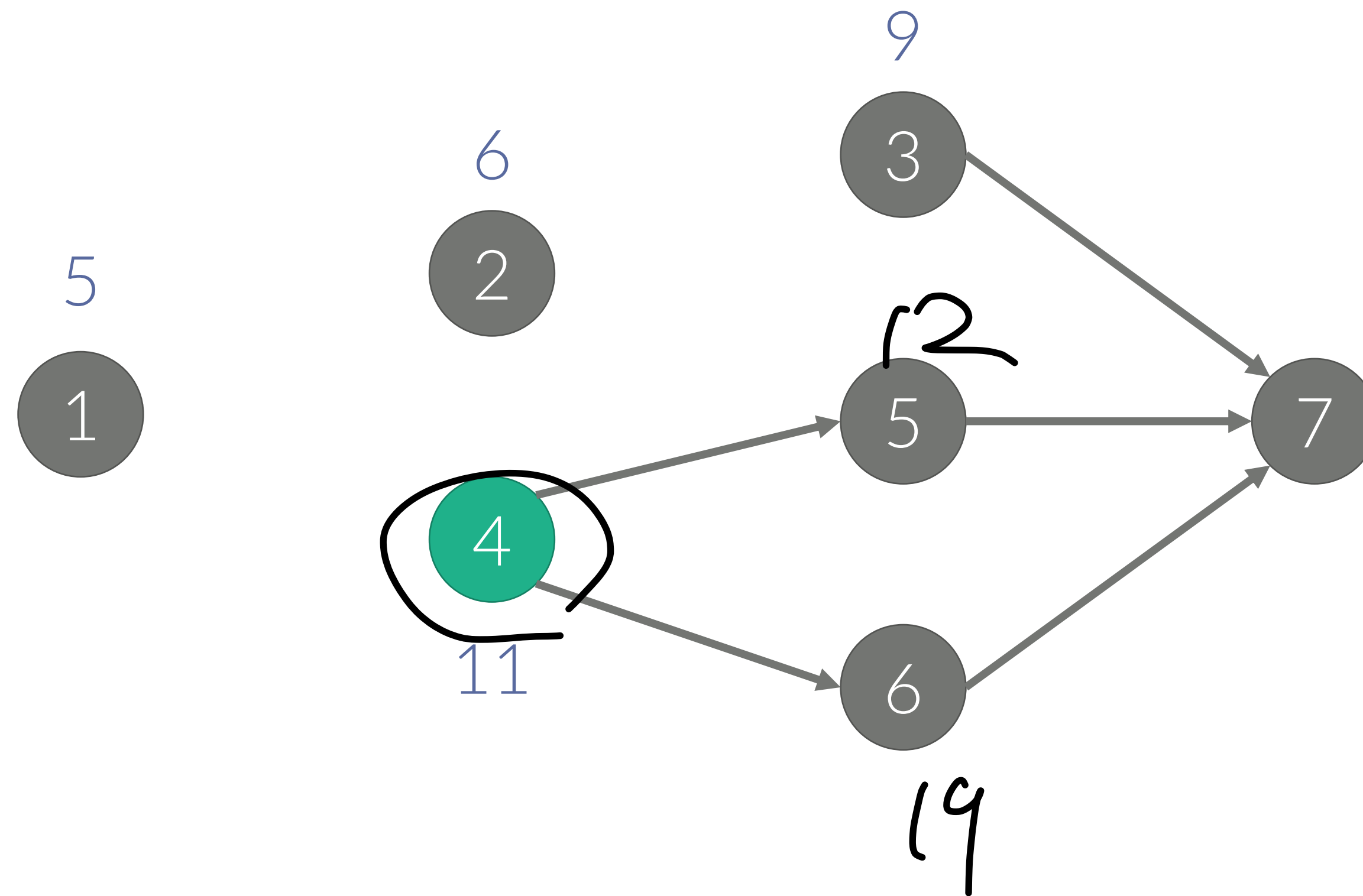
작업

44

<https://www.acmicpc.net/problem/2056>

- 큐: 3
- 현재 작업: 4

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4

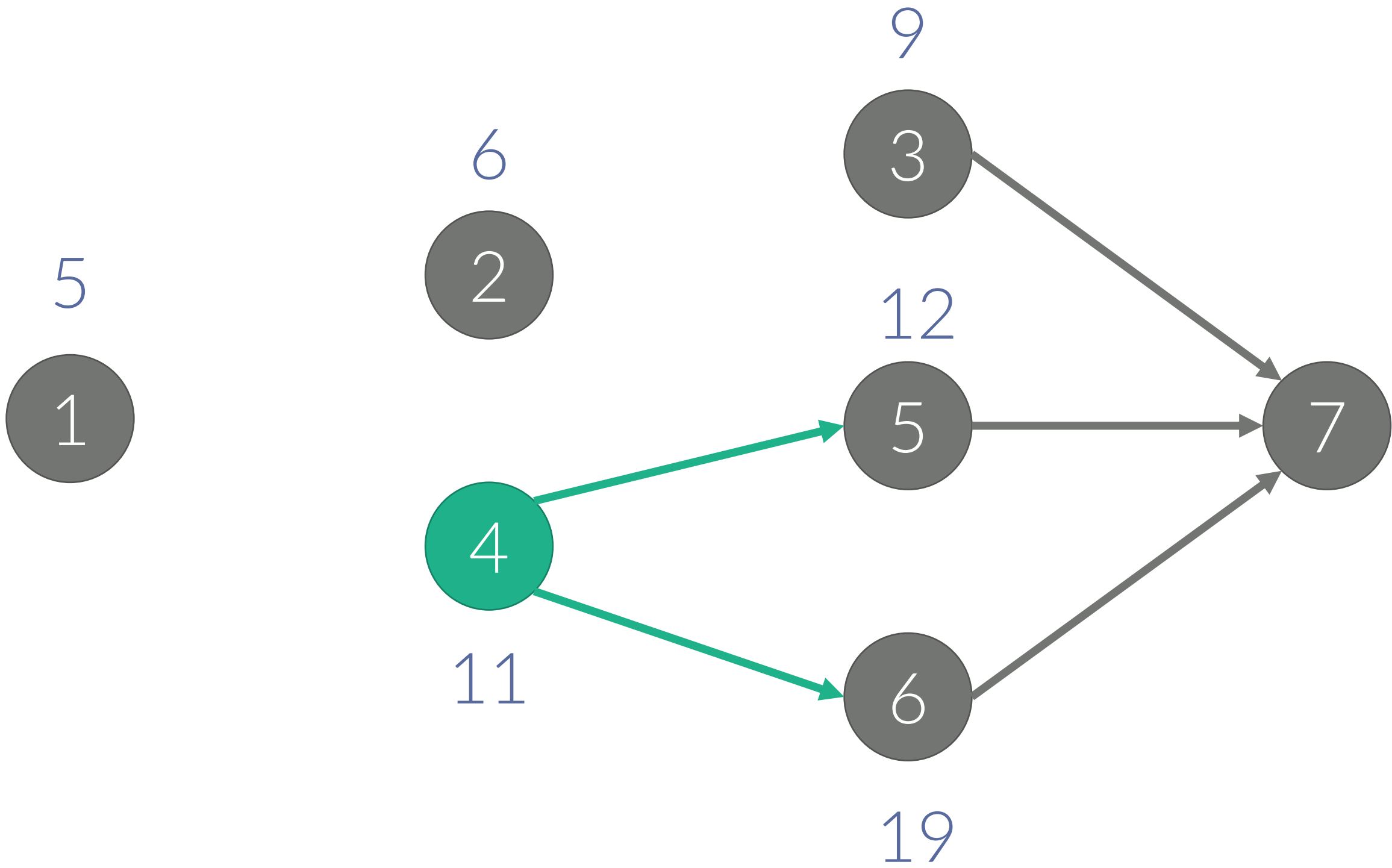


작업

<https://www.acmicpc.net/problem/2056>

- 큐: 3
- 현재 작업: 4

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4



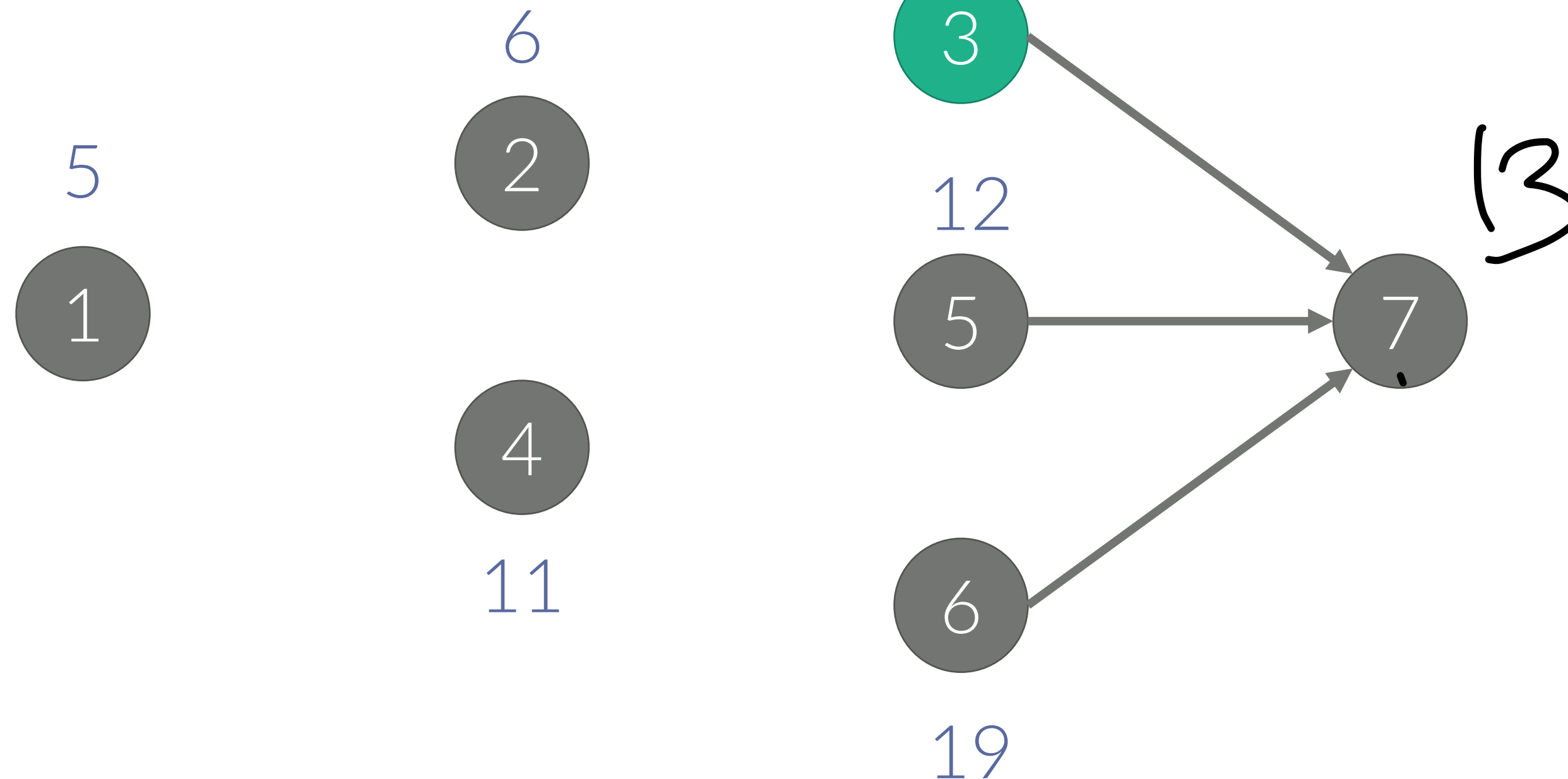
작업

46

<https://www.acmicpc.net/problem/2056>

- 큐: 5 6
- 현재 작업: 3

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4

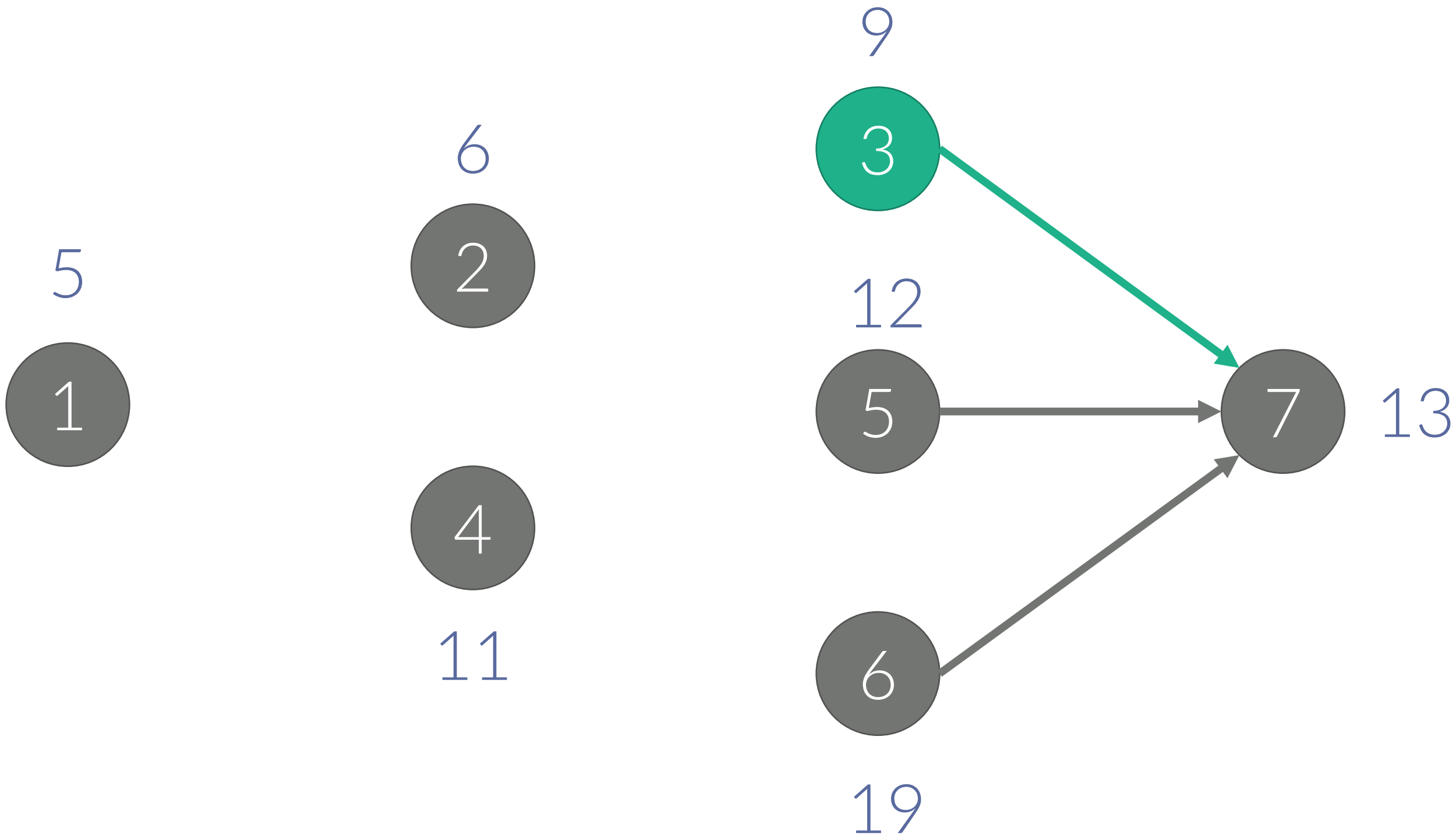


작업

<https://www.acmicpc.net/problem/2056>

- 큐: 5 6
- 현재 작업: 3

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4

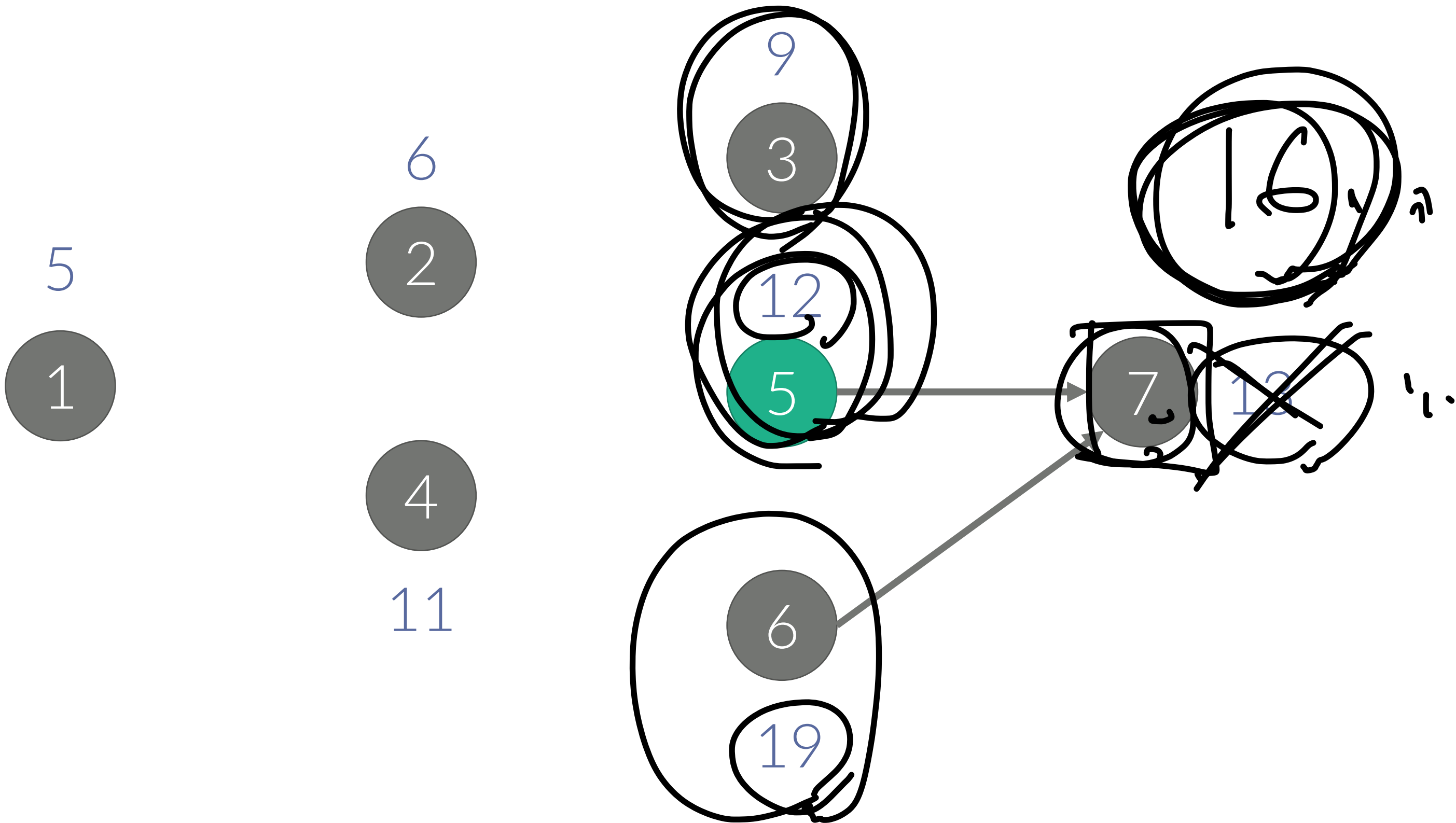


작업

<https://www.acmicpc.net/problem/2056>

- 큐: 6
- 현재 작업: 5

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4

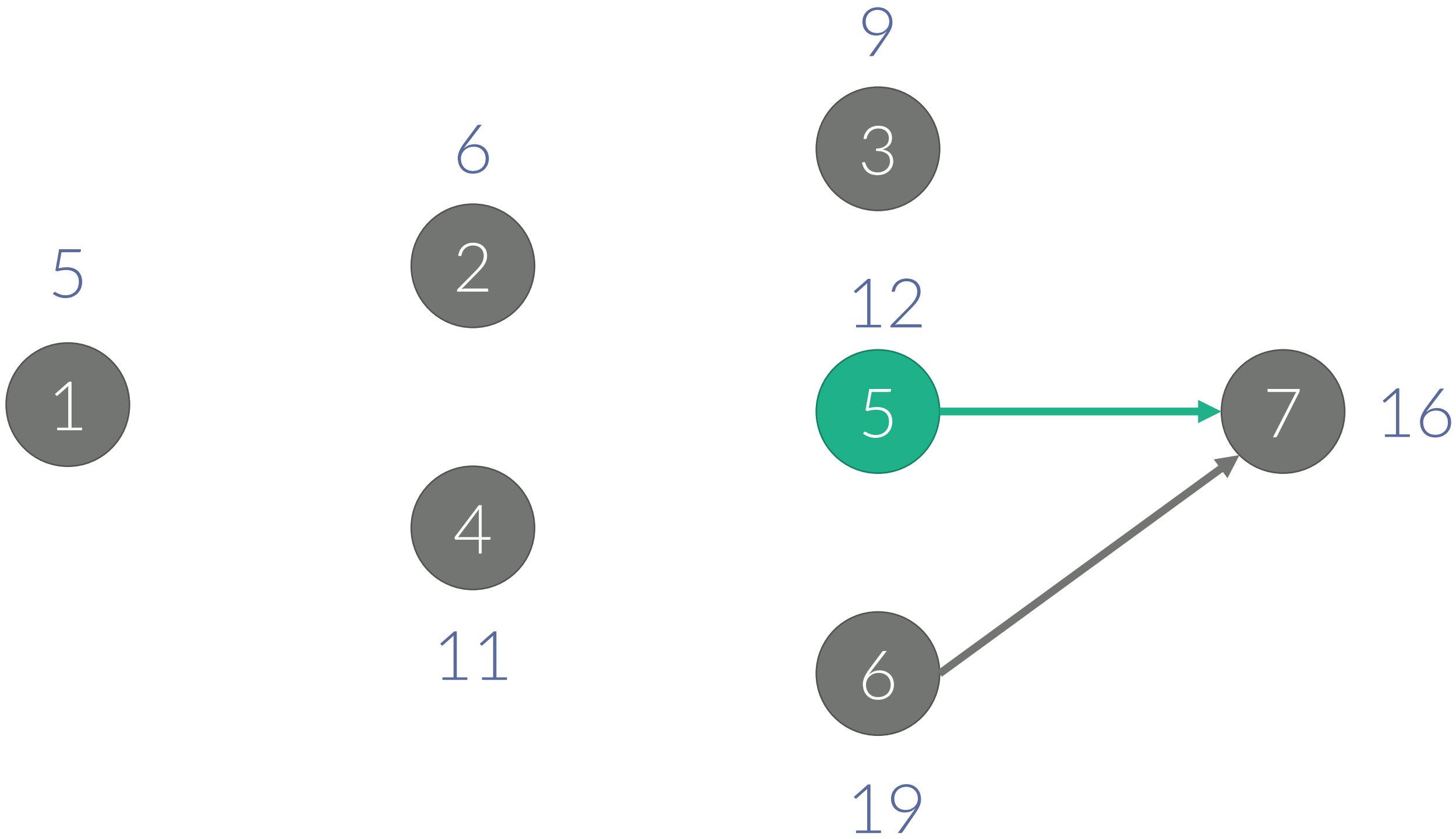


작업

<https://www.acmicpc.net/problem/2056>

- 큐: 6
- 현재 작업: 5

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4

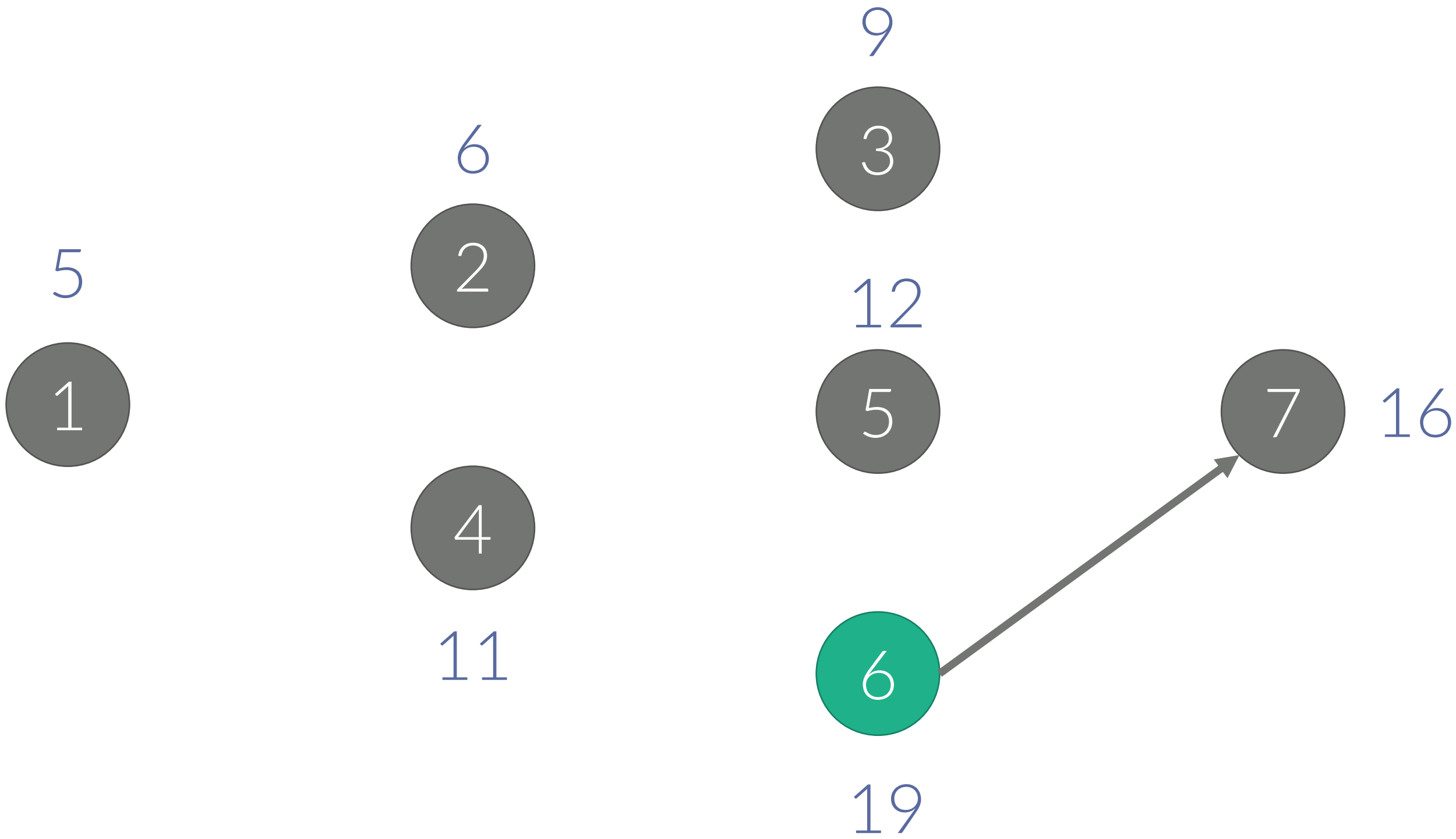


작업

<https://www.acmicpc.net/problem/2056>

- 큐:
- 현재 작업: 6

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4



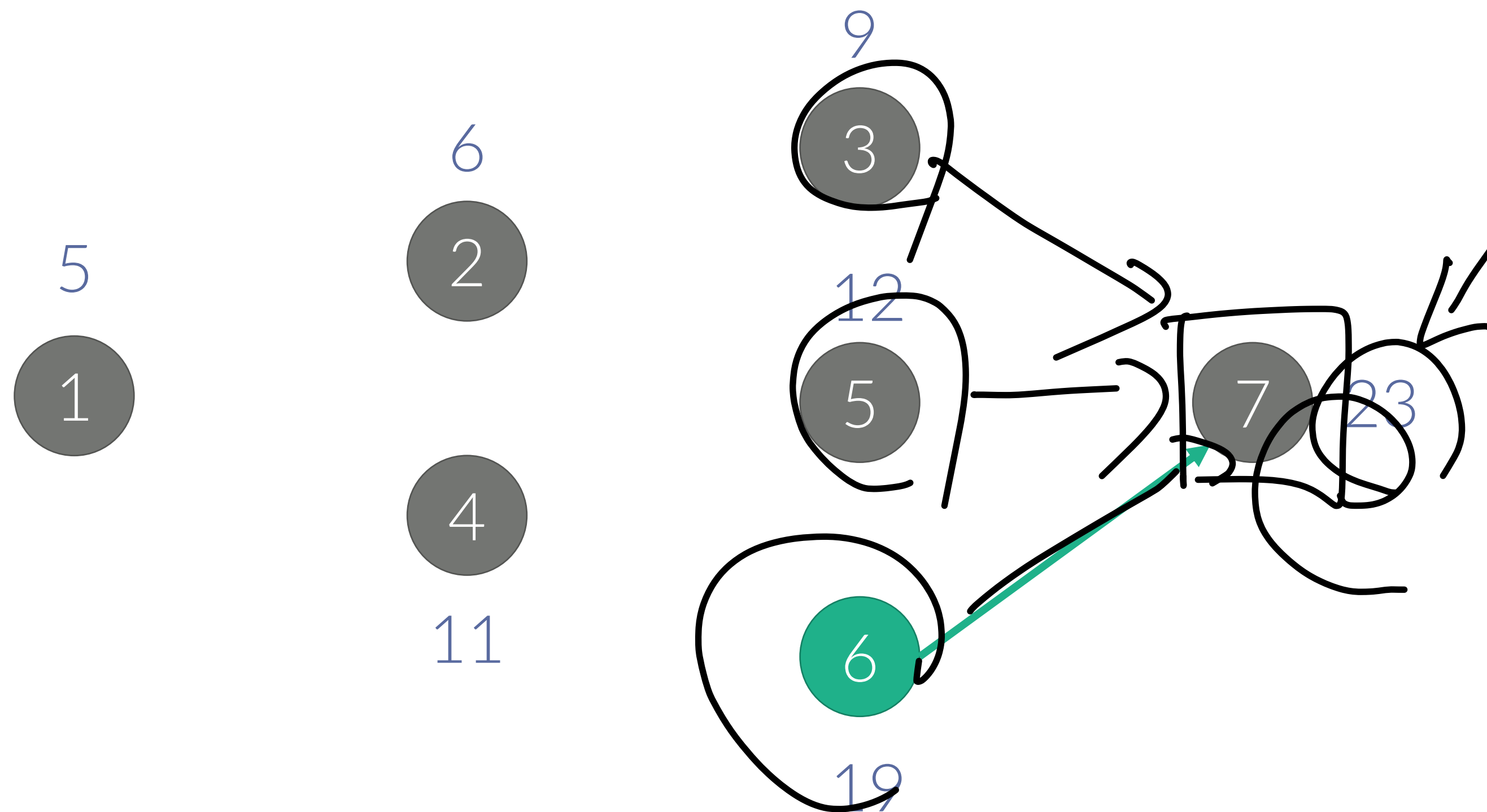
작업

51

<https://www.acmicpc.net/problem/2056>

- 큐:
- 현재 작업: 6

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4

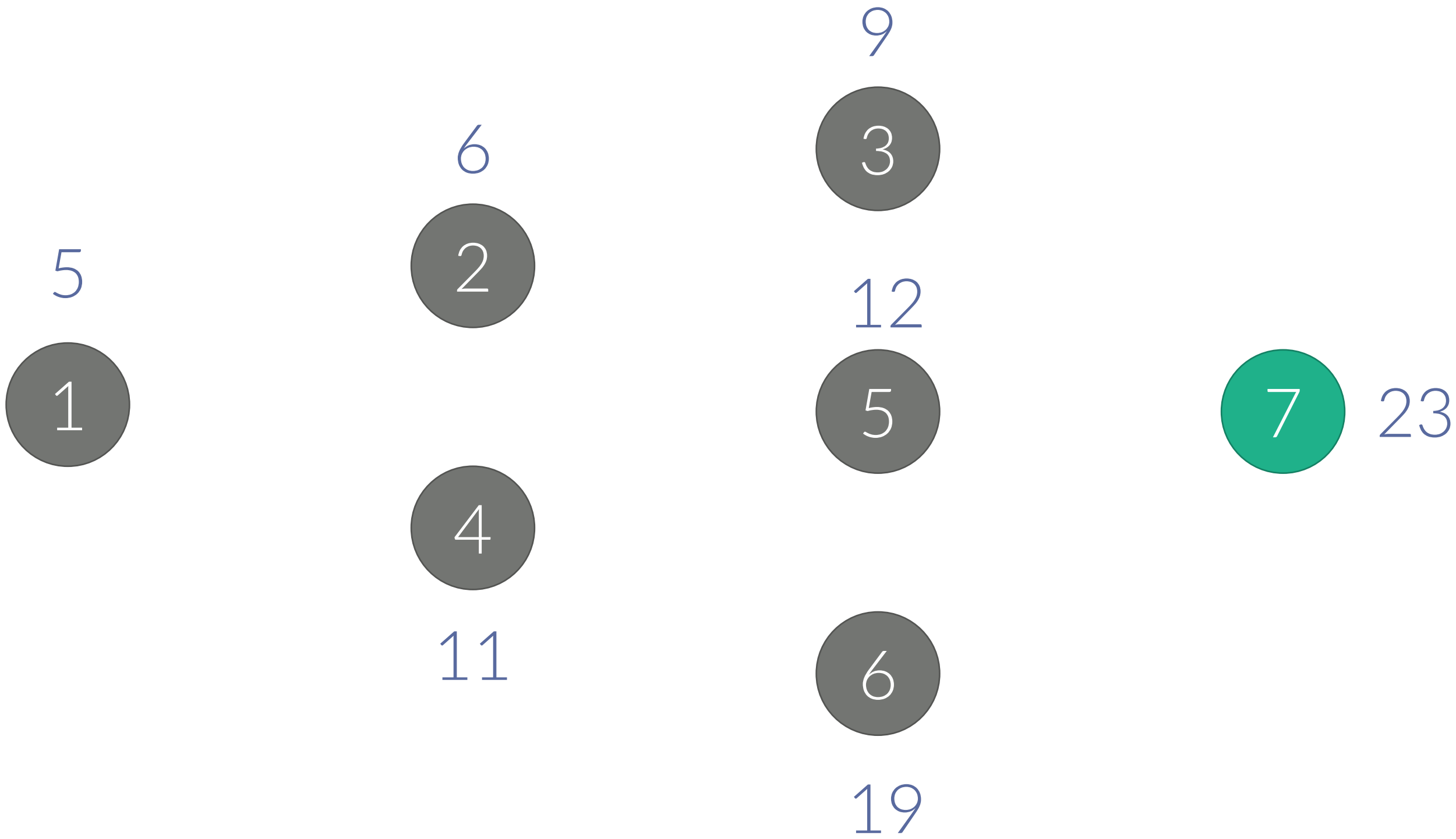


작업

<https://www.acmicpc.net/problem/2056>

- 큐:
- 현재 작업: 7

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4

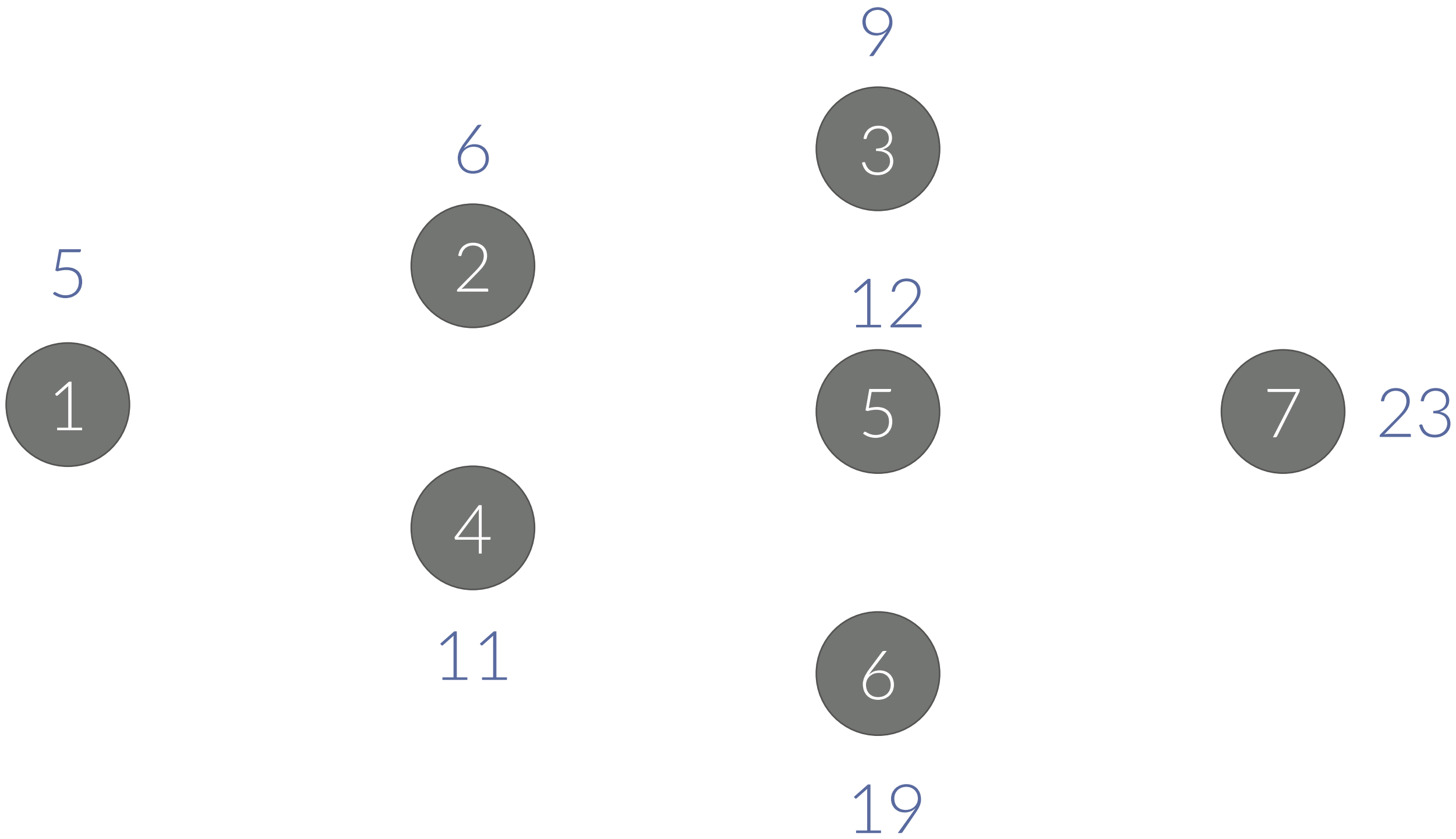


작업

<https://www.acmicpc.net/problem/2056>

- 큐:
- 현재 작업:

i	1	2	3	4	5	6	7
시간	5	1	3	6	1	8	4



작업

<https://www.acmicpc.net/problem/2056>

- 소스: <http://codeplus.codes/2b818650b1b64e169cf509702f940ae7>

$123 \rightarrow \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}$
 2321

MST

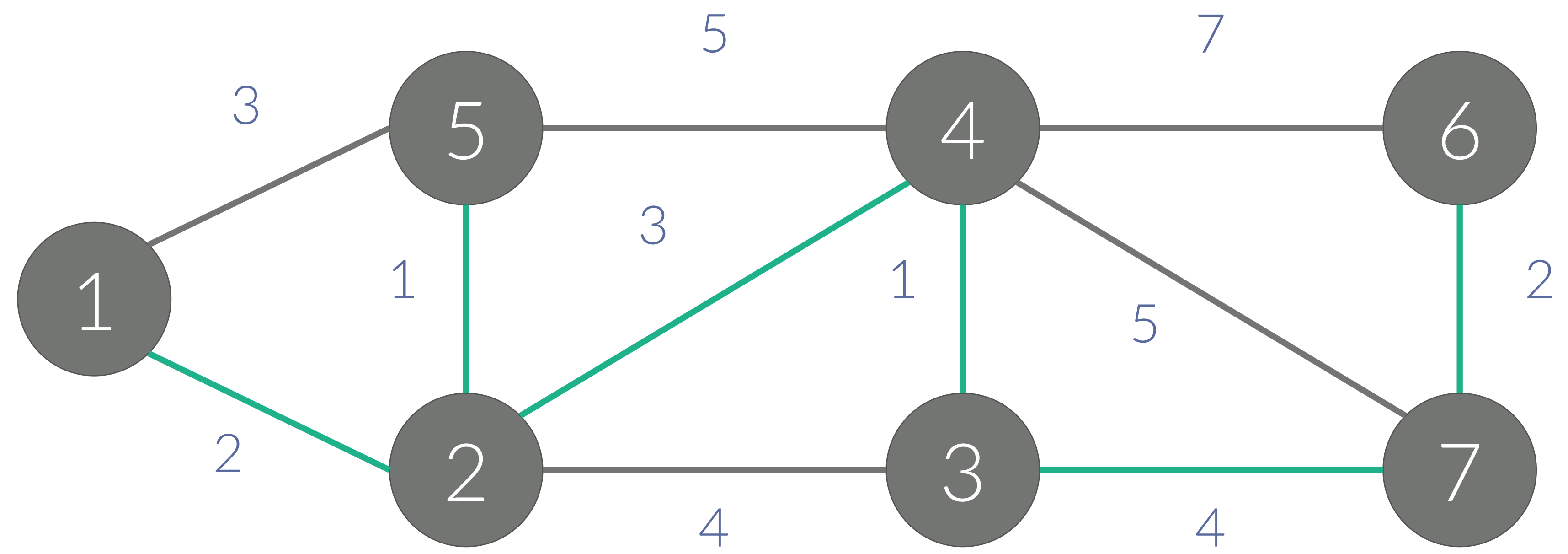


최소 스패닝 트리

그래프 → 트리

Minimum Spanning Tree

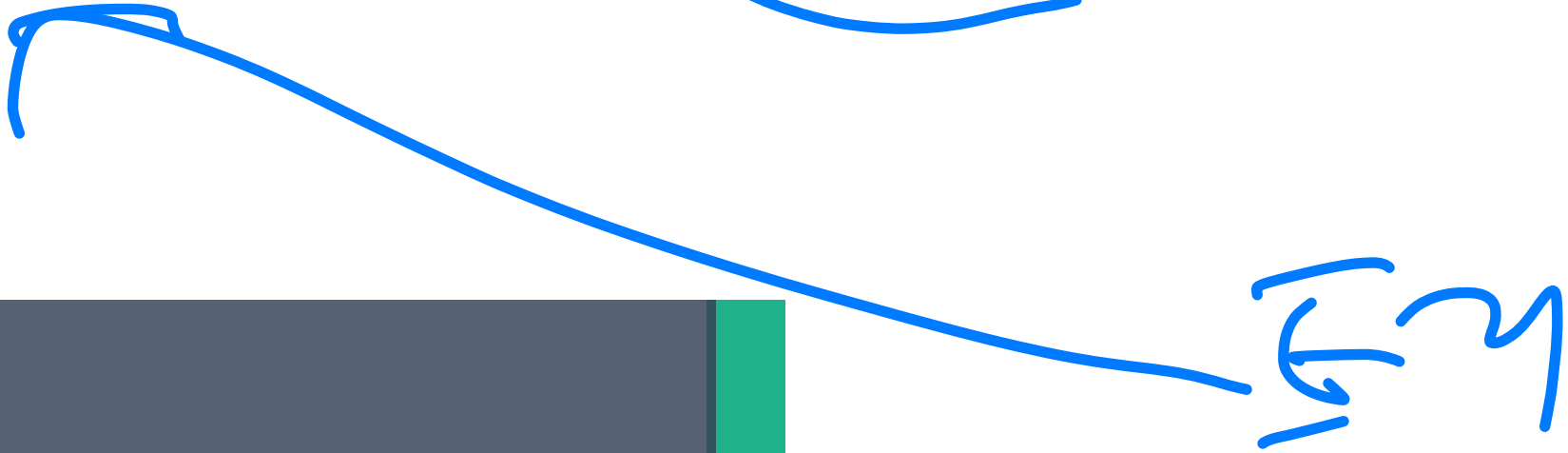
- 스패닝 트리: 그래프에서 일부 간선을 선택해서 만든 트리
- 최소 스패닝 트리: 스패닝 트리 중에 선택한 간선의 가중치의 합이 최소인 트리



4012

Prim

Kruskal

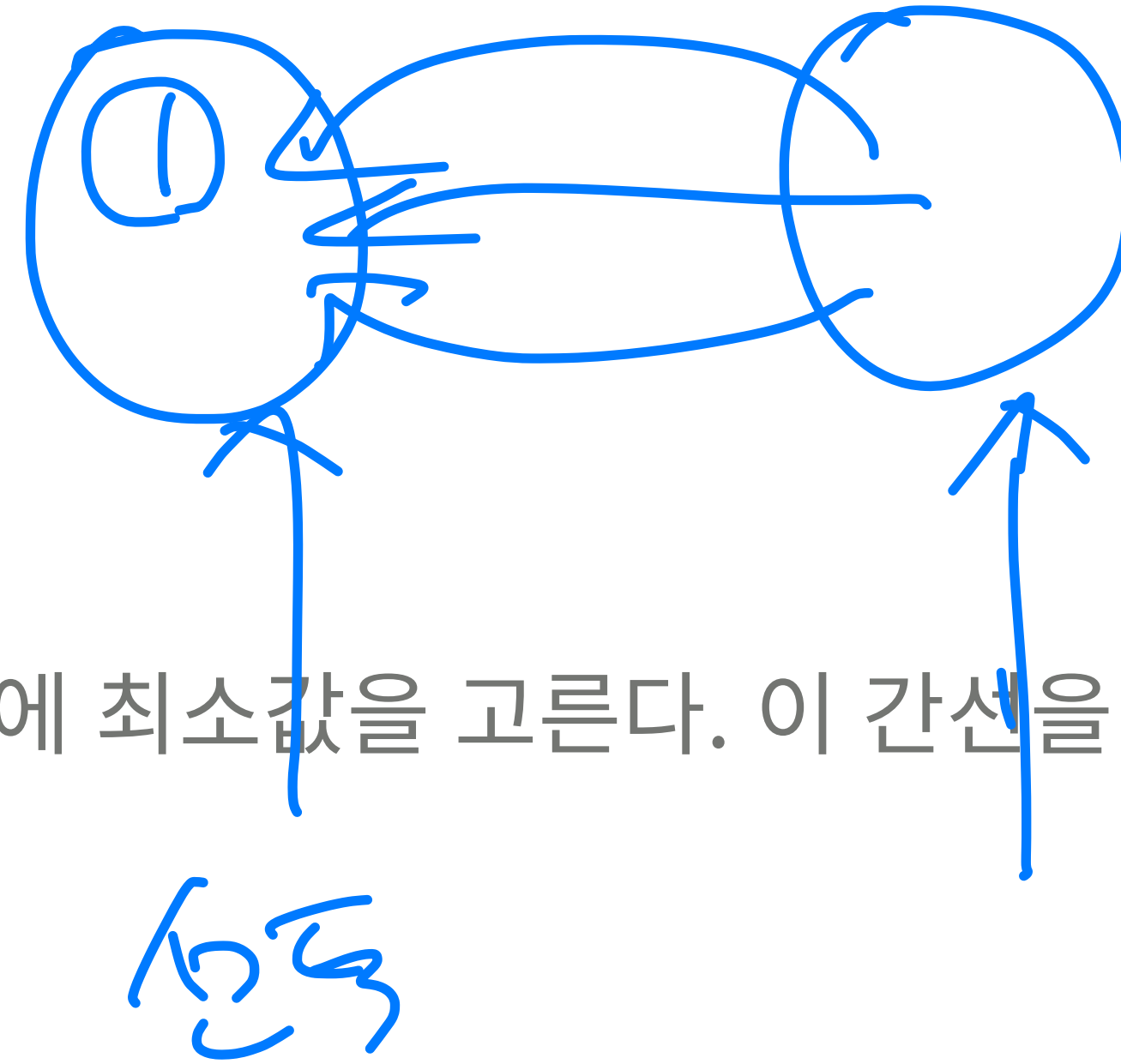


프림

프림

Prim

1. 그래프에서 아무 정점이나 선택한다.
2. 선택한 정점과 선택하지 않은 정점을 연결하는 간선중에 최소값을 고른다. 이 간선을 (u, v) 라고 한다. (u = 선택, v = 선택하지 않음)
3. 선택한 간선을 MST에 추가하고, v 를 선택한다.
4. 모든 정점선택하지 않았다면, 2번 단계로 돌아간다.



프림

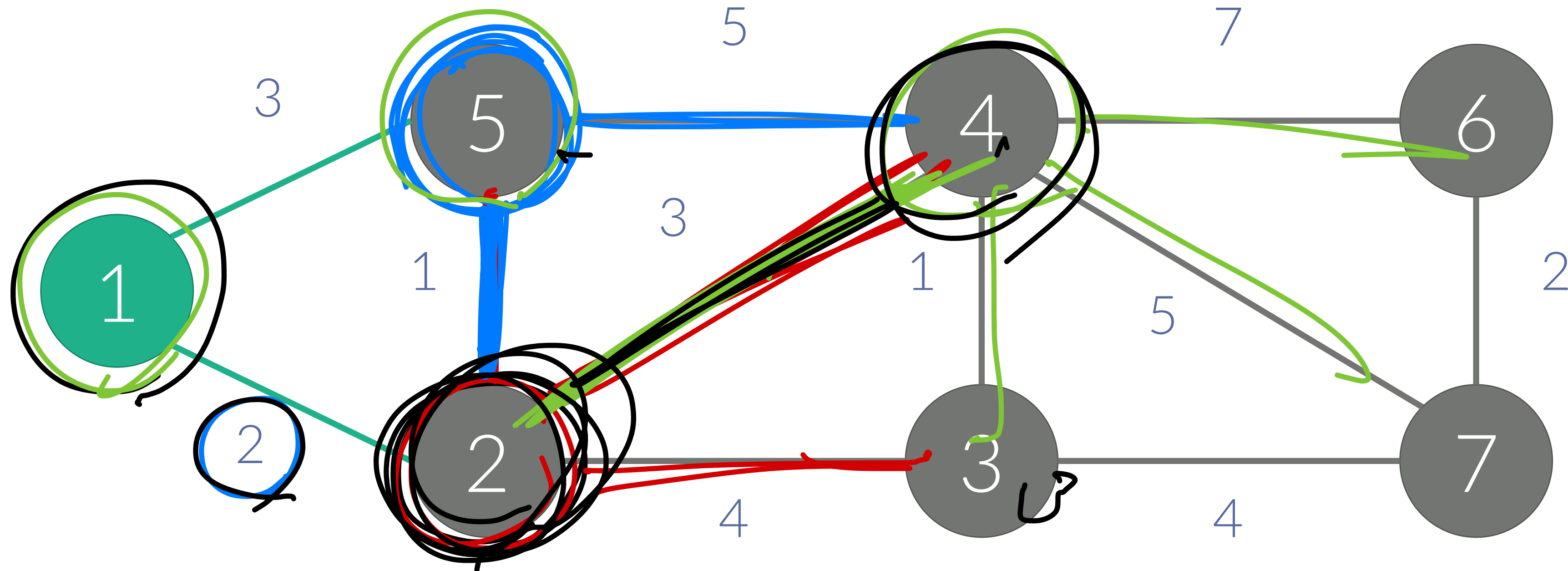
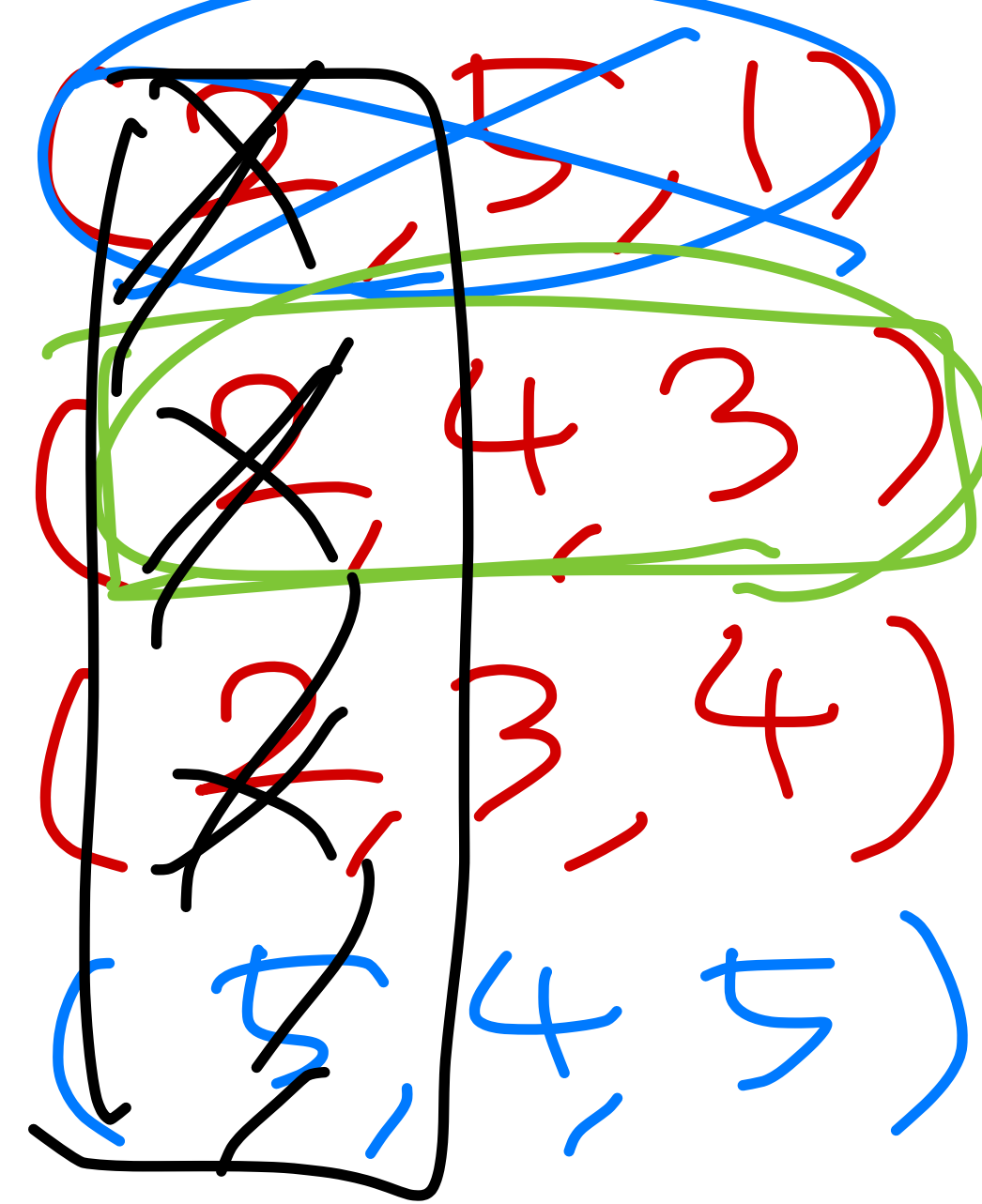
Prim

- 선택: 1

f t c

기

f t

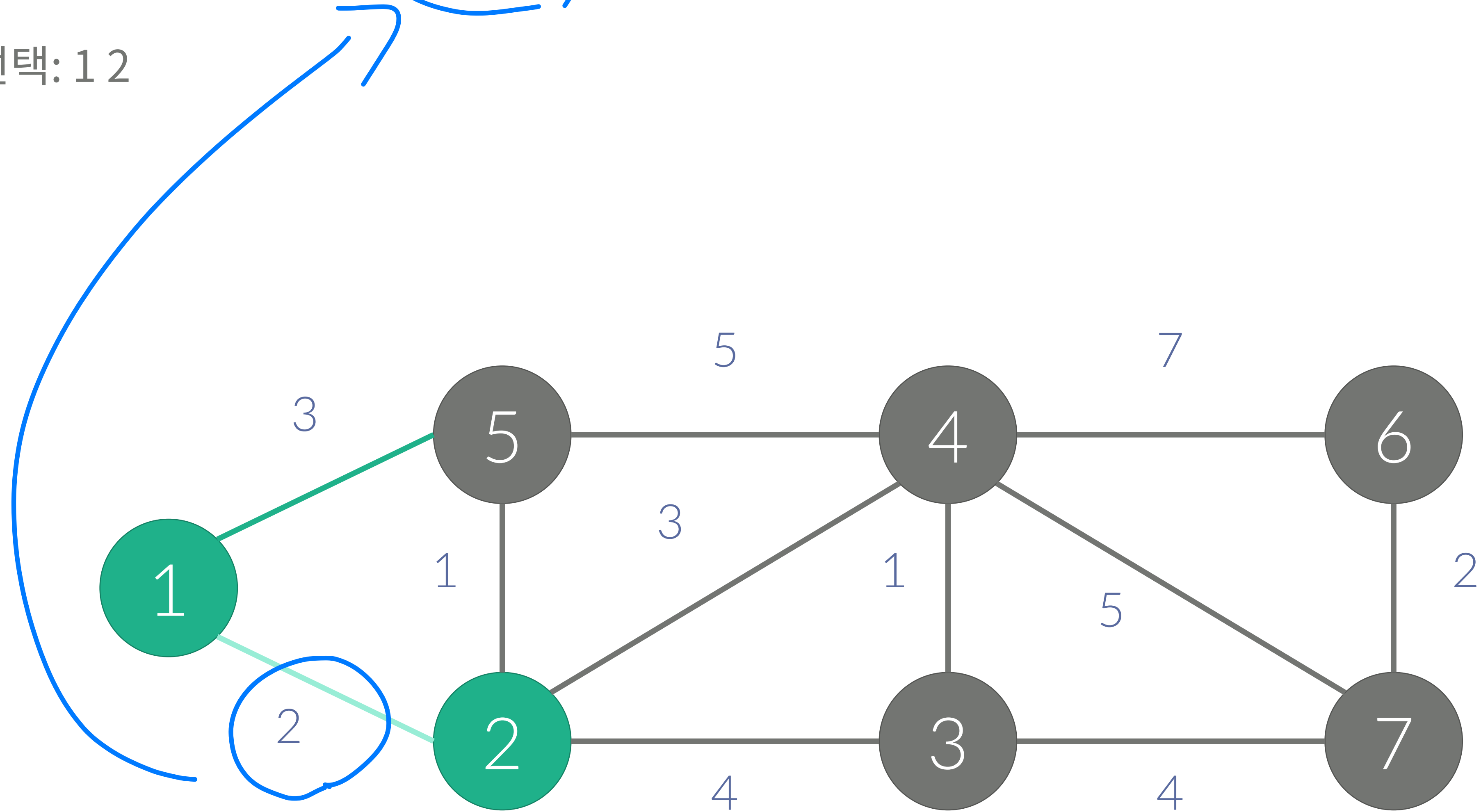


프림

Prim

- 선택: 1 2

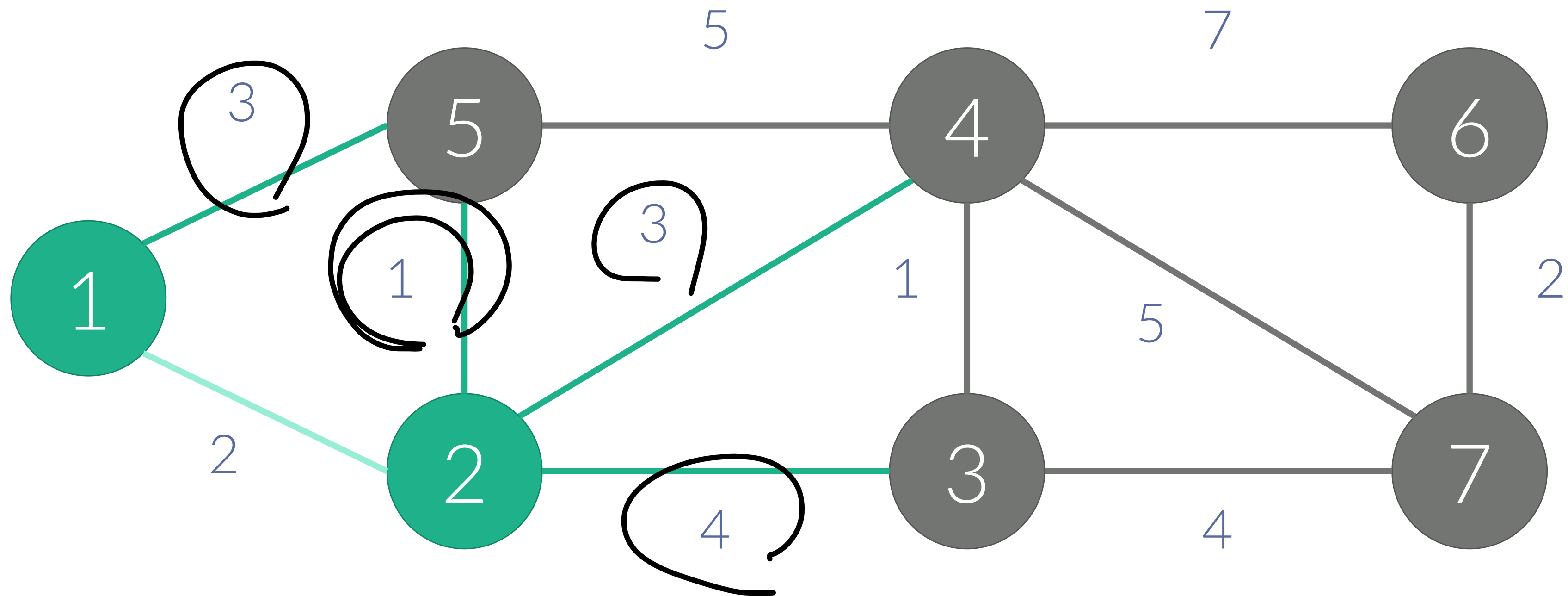
MST



프림

Prim

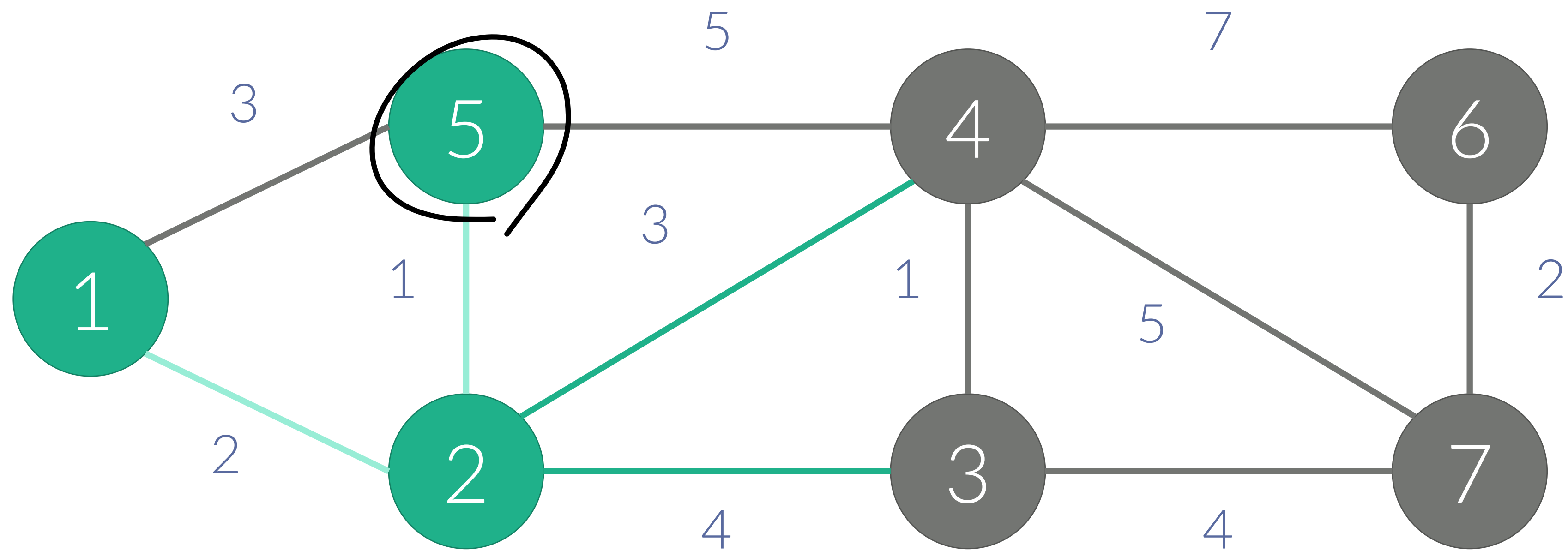
- 선택: 1 2



프림

Prim

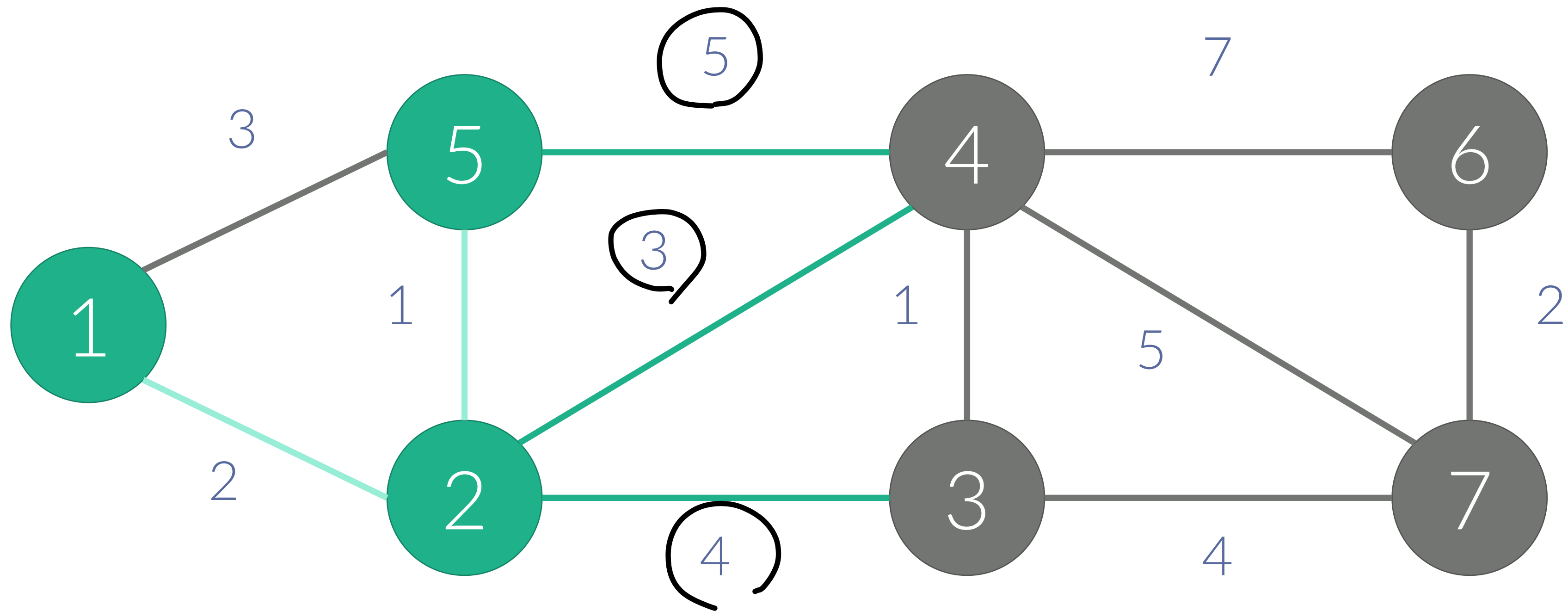
- 선택: 1 2 5



프림

Prim

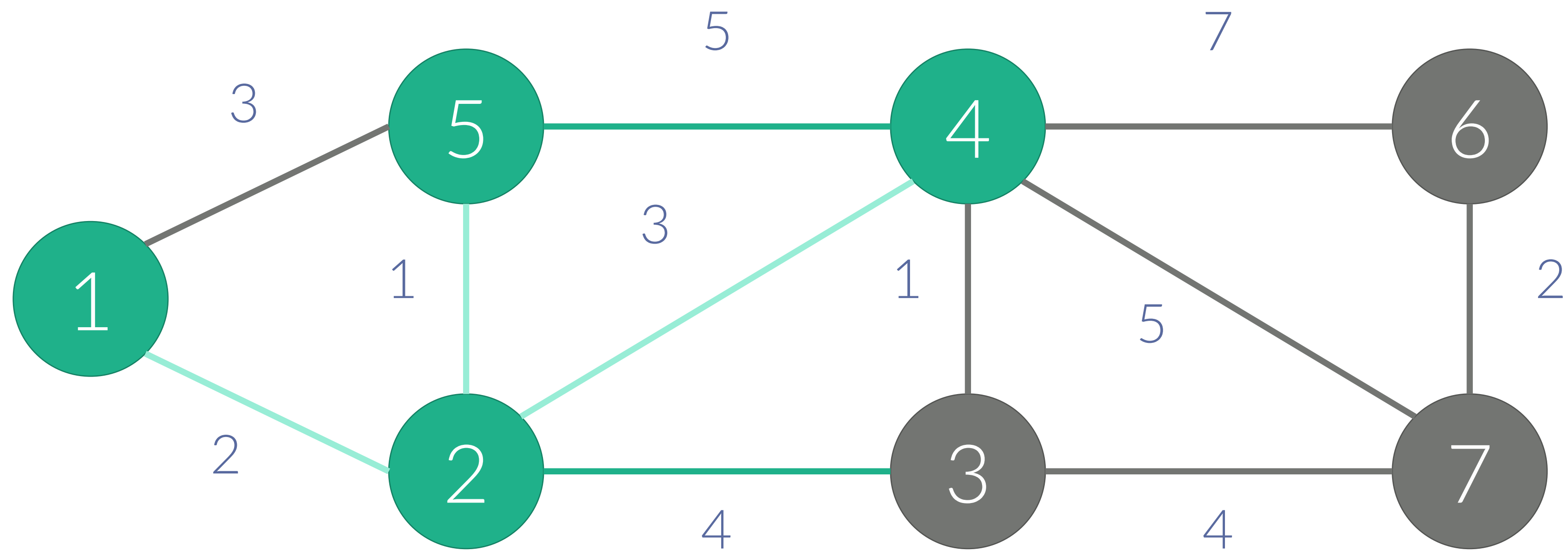
- 선택: 1 2 5



프림

Prim

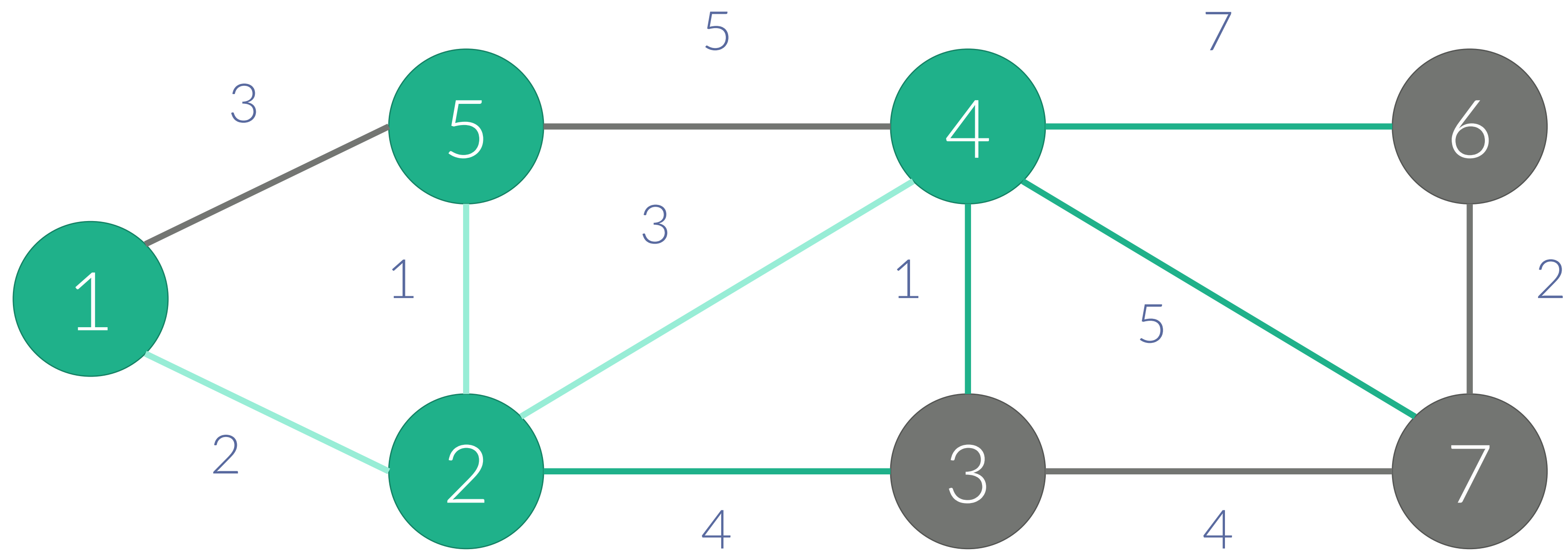
- 선택: 1 2 5 4



프림

Prim

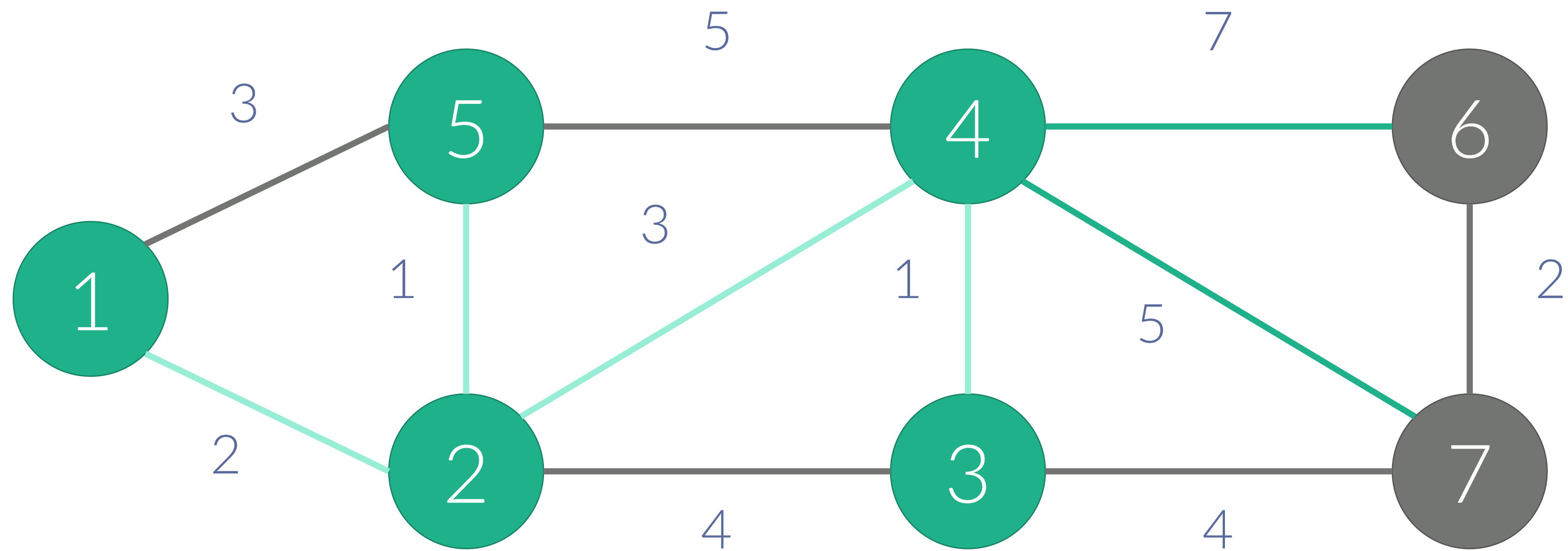
- 선택: 1 2 5 4



프림

Prim

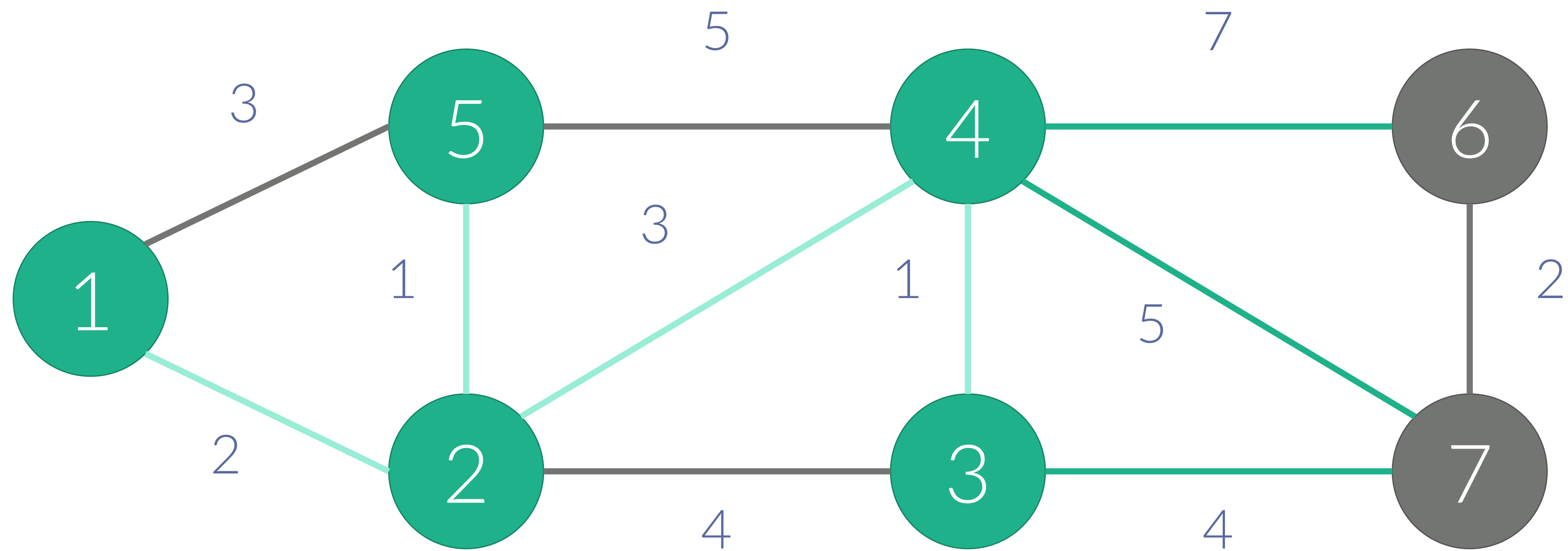
- 선택: 1 2 5 4 3



프림

Prim

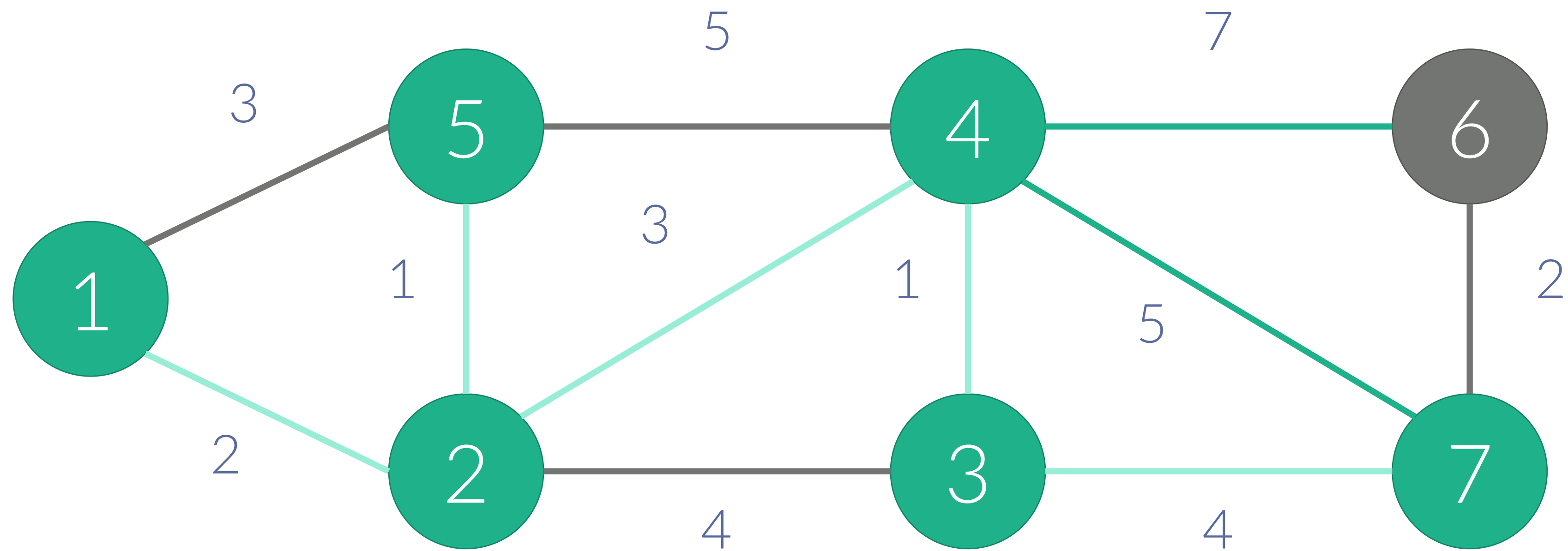
- 선택: 1 2 5 4 3



프림

Prim

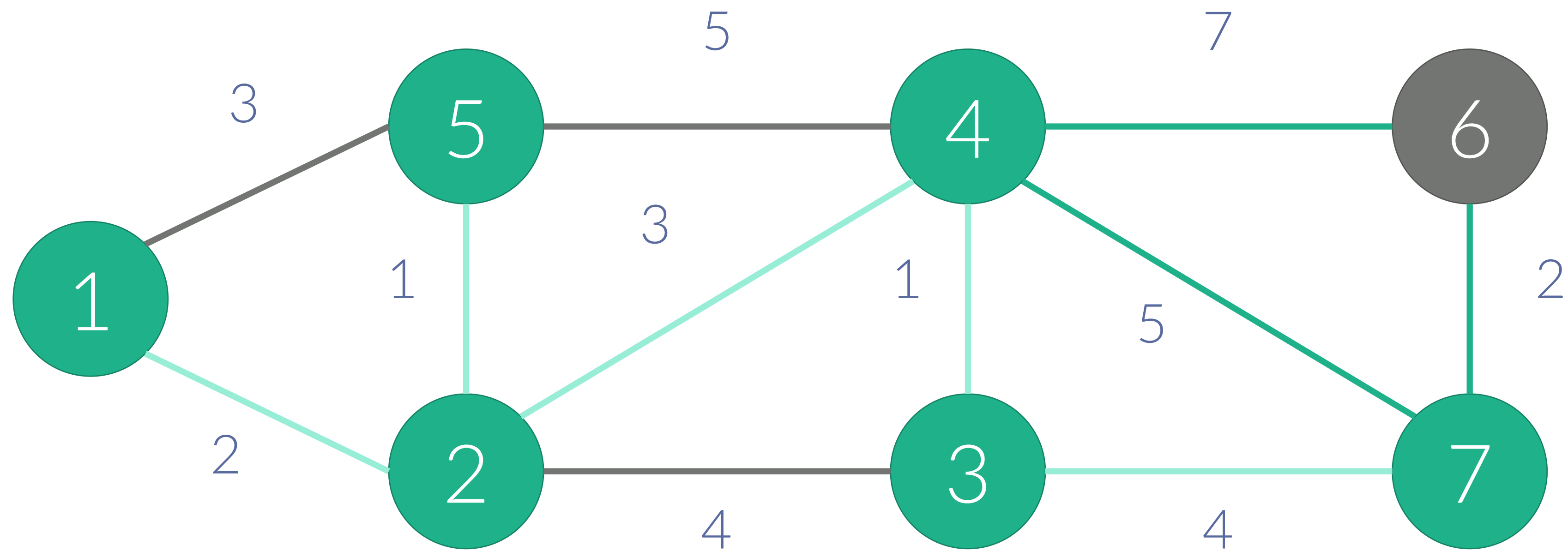
- 선택: 1 2 5 4 3 7



프림

Prim

- 선택: 1 2 5 4 3 7

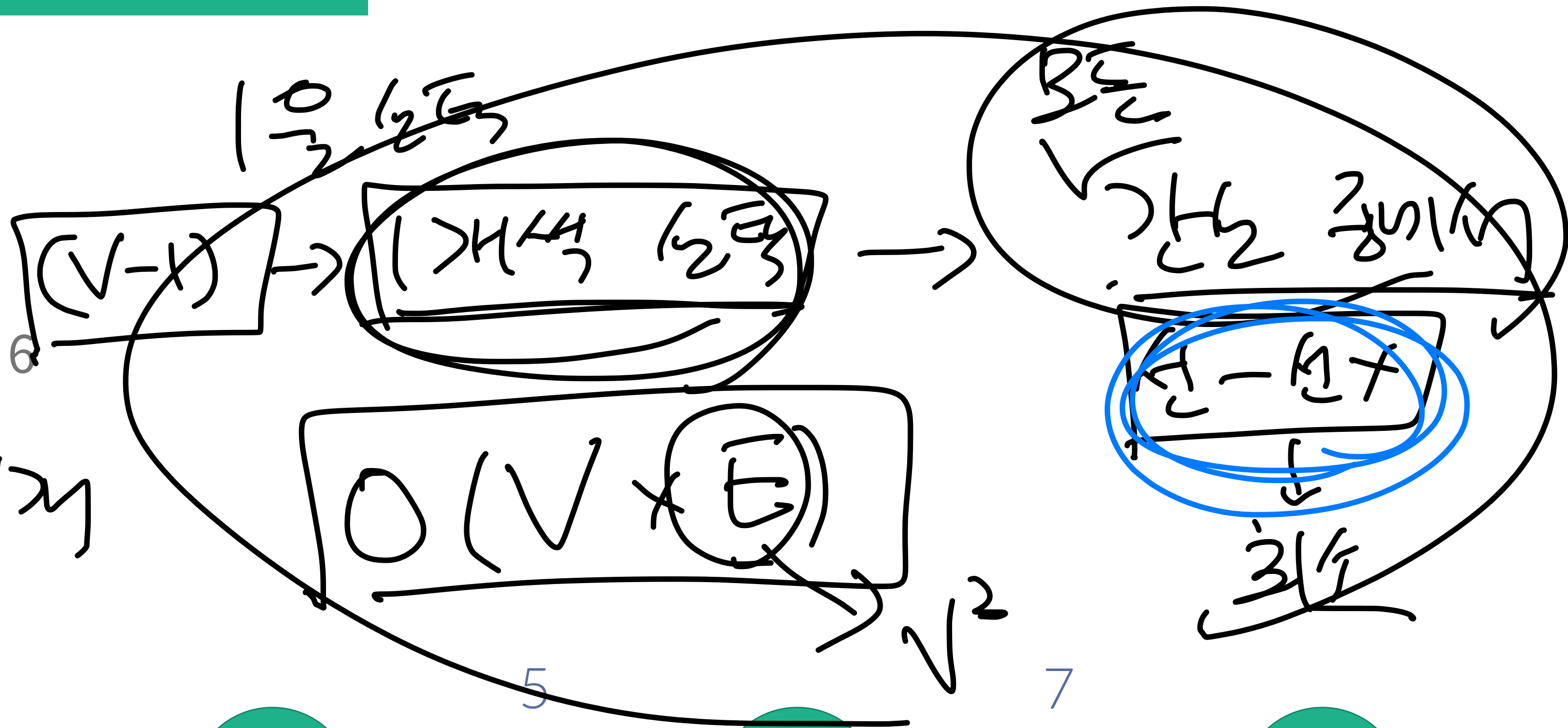
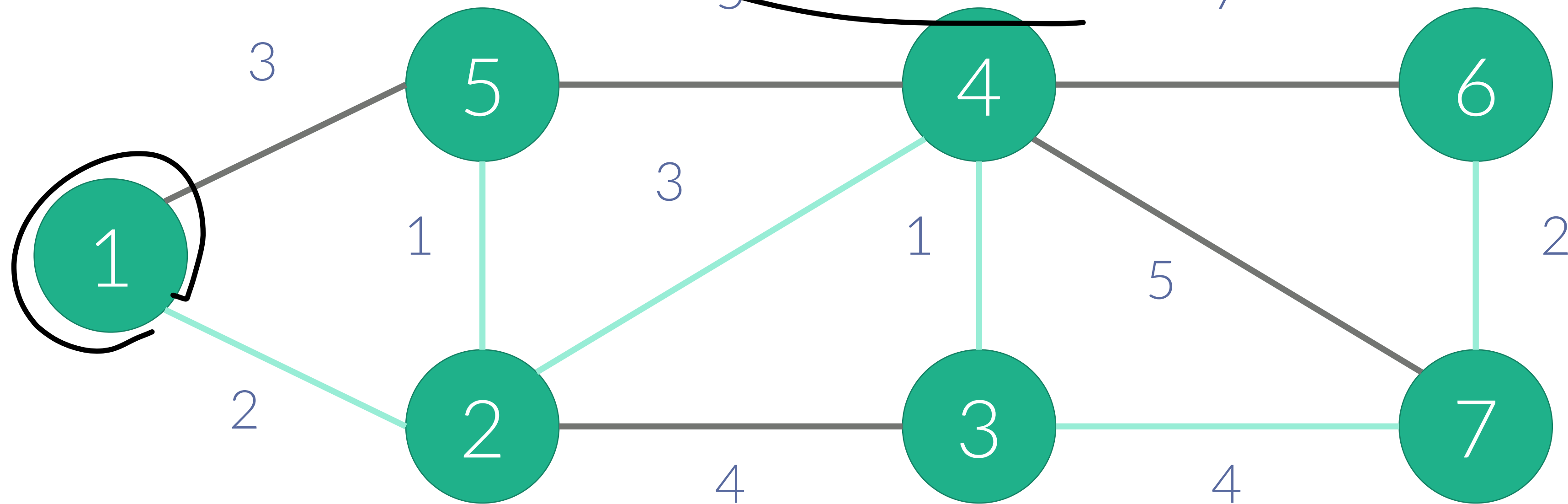


프림

Prim

- 선택: 1 2 5 4 3 7 6

정점: $\sqrt{2}$



프림

Prim

1. 그래프에서 아무 정점이나 선택한다.
2. 선택한 정점과 선택하지 않은 정점을 연결하는 간선중에 최소값을 고른다. 이 간선을 (u, v) 라고 한다. ($u =$ 선택, $v =$ 선택하지 않음)
3. 선택한 간선을 MST에 추가하고, v 를 선택한다.
4. 모든 정점을 선택하지 않았다면, 2번 단계로 돌아간다.

$$O(V^2)$$

$$O(E \log V)$$

- 각각의 정점을 선택하고 모든 간선을 살펴봐야 한다.

- 시간 복잡도: $O(V \times E)$ 최대 $O(V^3)$

프림

Prim

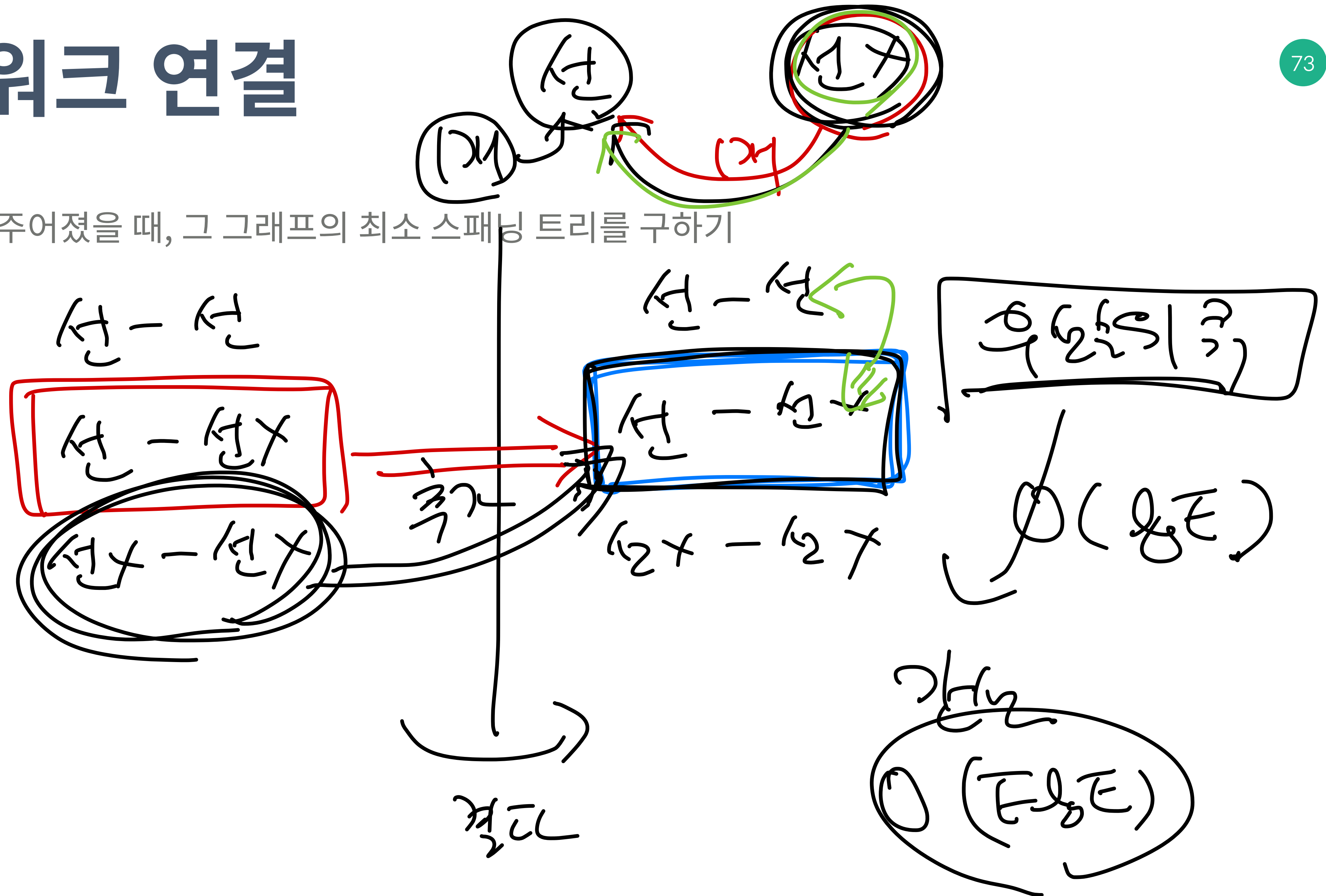
1. 그래프에서 아무 정점이나 선택한다.
 2. 선택한 정점과 선택하지 않은 정점을 연결하는 간선중에 최소값을 고른다. 이 간선을 (u, v) 라고 한다. ($u =$ 선택, $v =$ 선택하지 않음)
 3. 선택한 간선을 MST에 추가하고, v 를 선택한다.
 4. 모든 정점을 선택하지 않았다면, 2번 단계로 돌아간다.
- 최소값을 우선 순위 큐를 이용하면 $O(\lg E)$ 에 찾을 수 있다
 - 시간 복잡도: $O(E \lg E)$

네트워크 연결

Prim

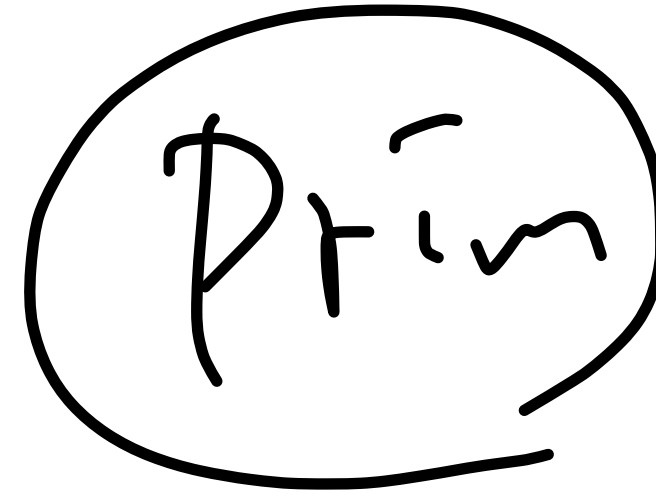
73

- 그래프가 주어졌을 때, 그 그래프의 최소 스패닝 트리를 구하기



네트워크 연결

<https://www.acmicpc.net/problem/1922>



- 소스: <http://codeplus.codes/706412a831e640bd89a88445a2cce561>

크루스칼

Kruskal

Union-Find

~~16/12~~

크루스칼

Kruskal

76

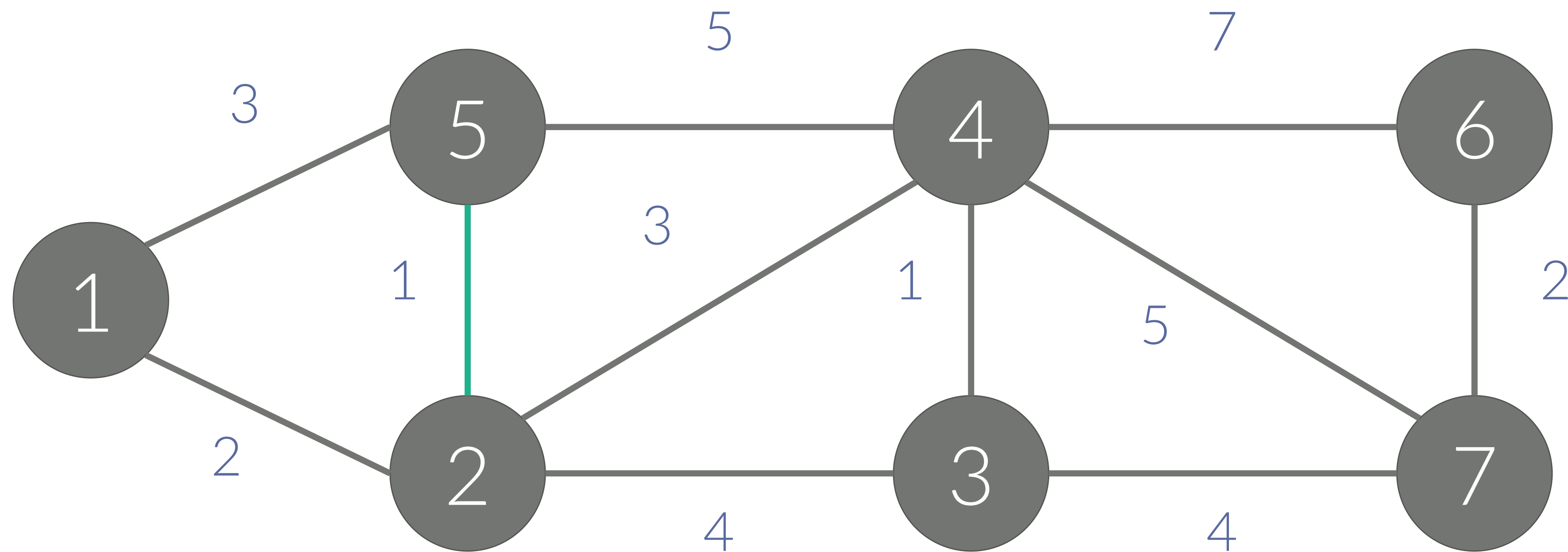
- 가중치가 작은 Edge부터 순서대로 살펴본다.
- Edge e 가 (u, v, c) 일 때
- u 와 v 가 다른 집합이면 e 를 MST에 추가한다

크루스칼

Kruskal

가중치 ↑ 정렬

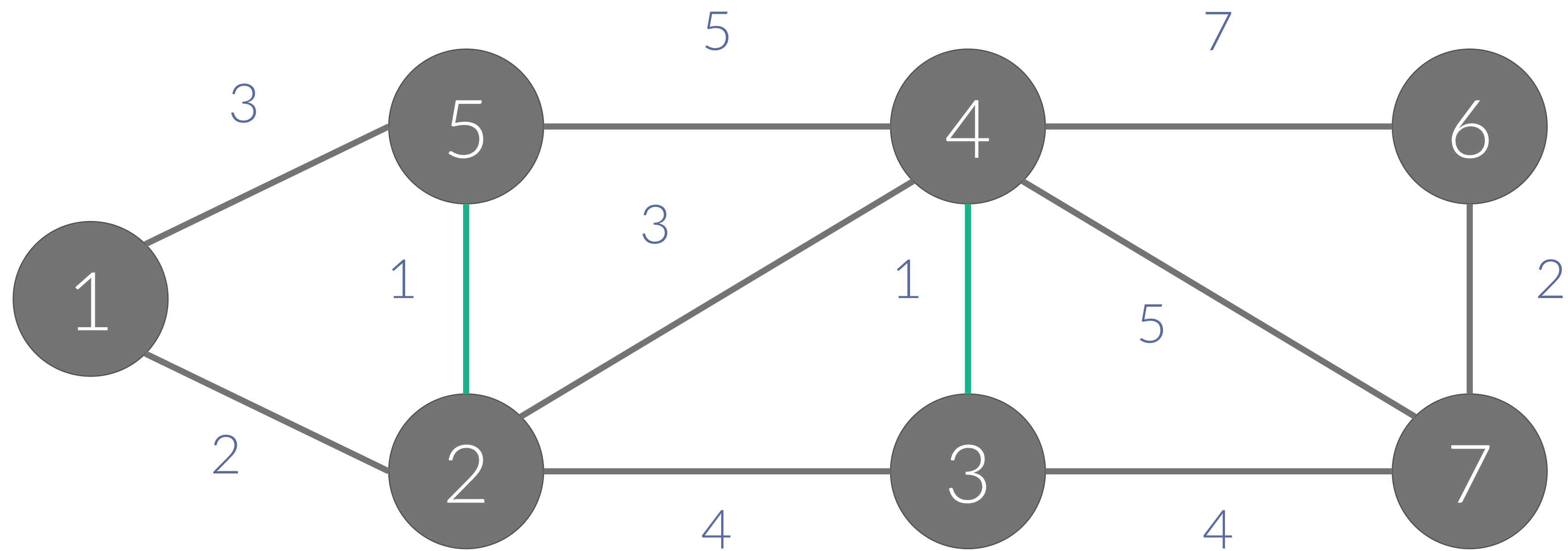
77



크루스칼

Kruskal

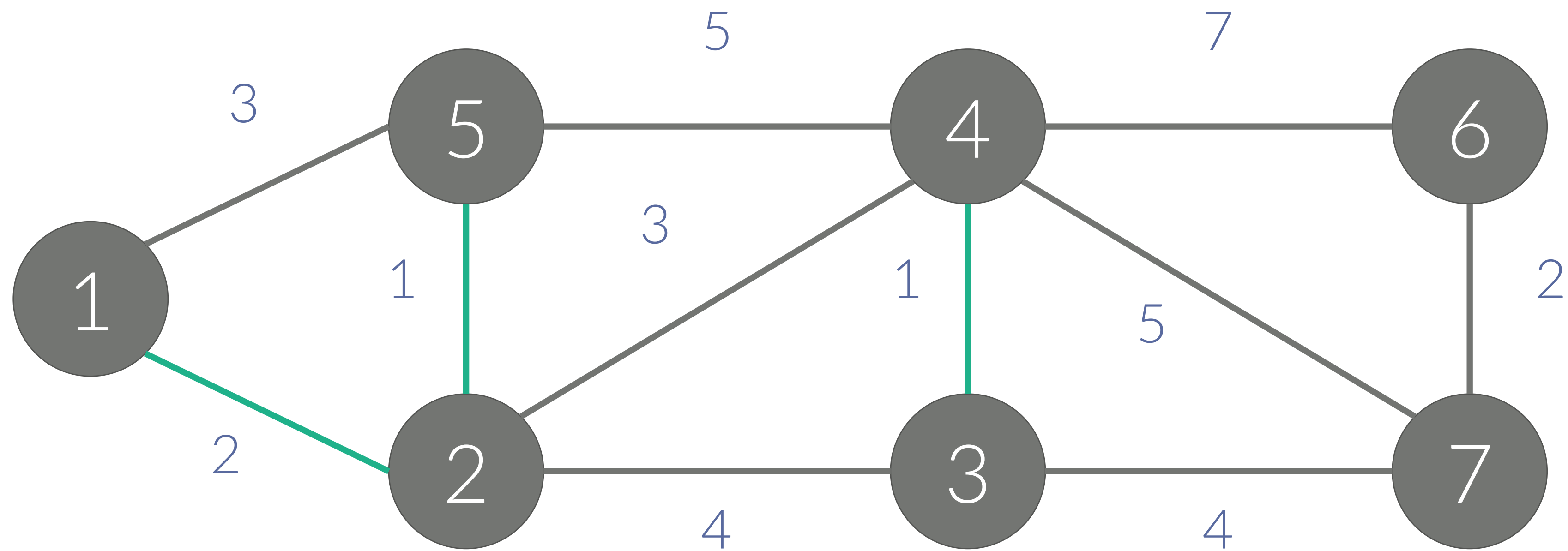
78



크루스칼

Kruskal

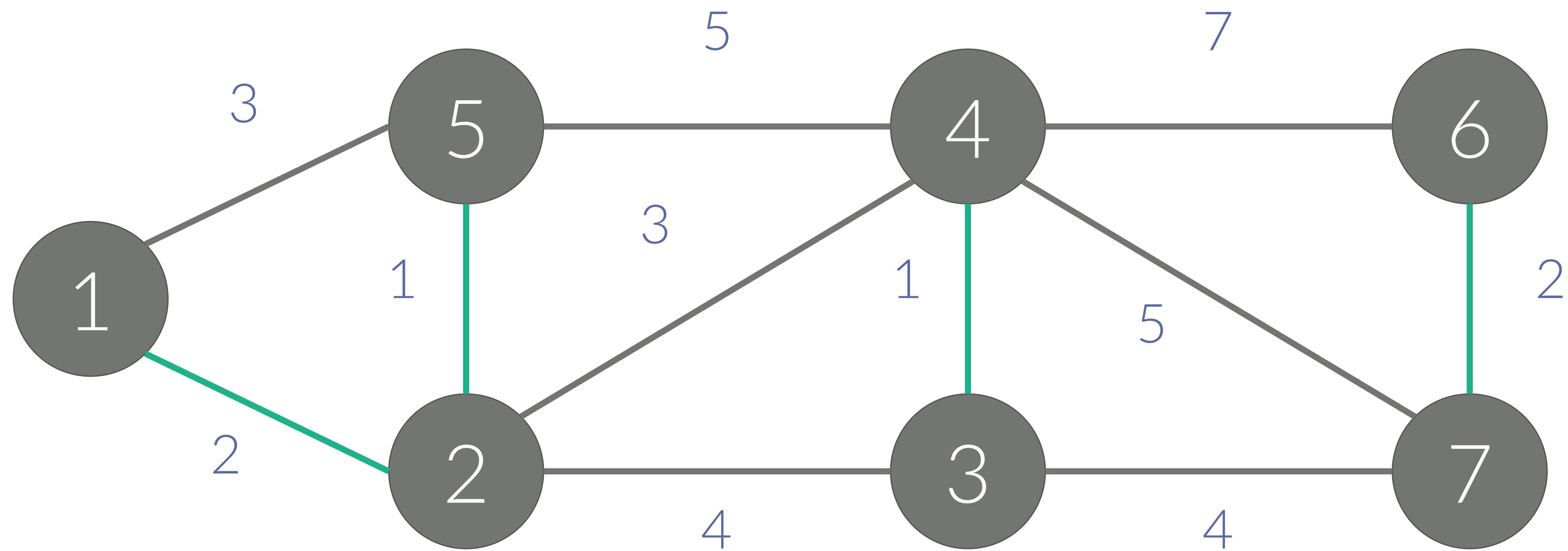
79



크루스칼

Kruskal

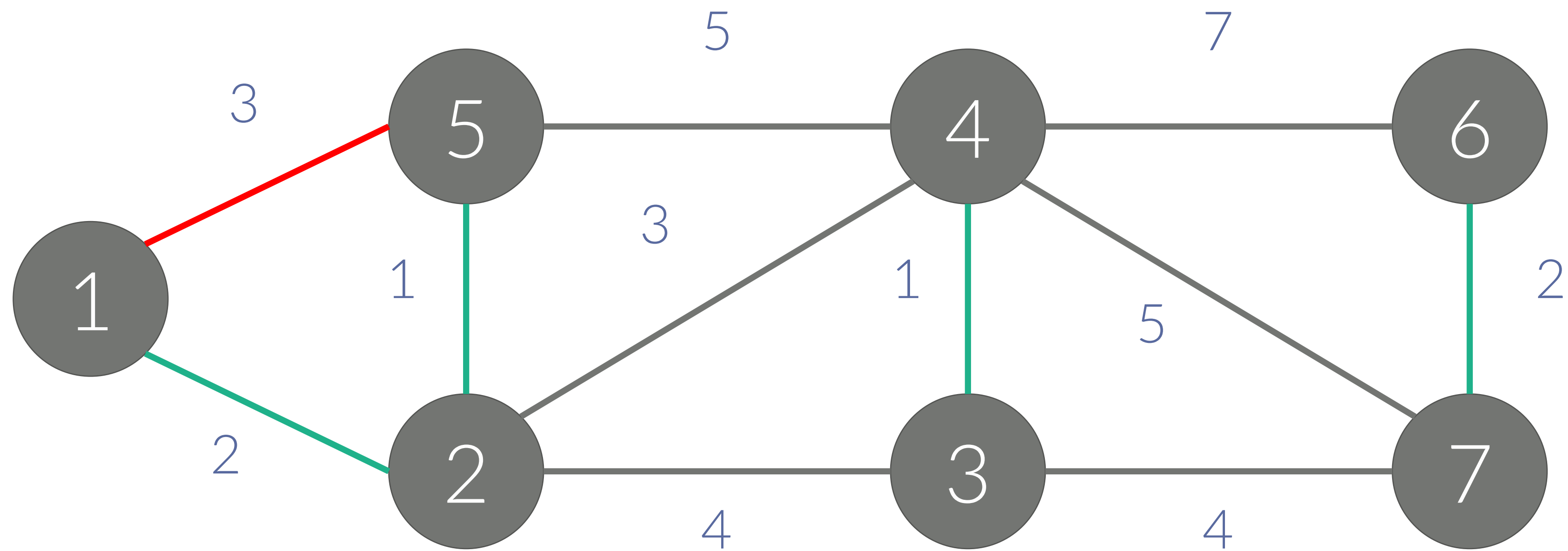
80



크루스칼

Kruskal

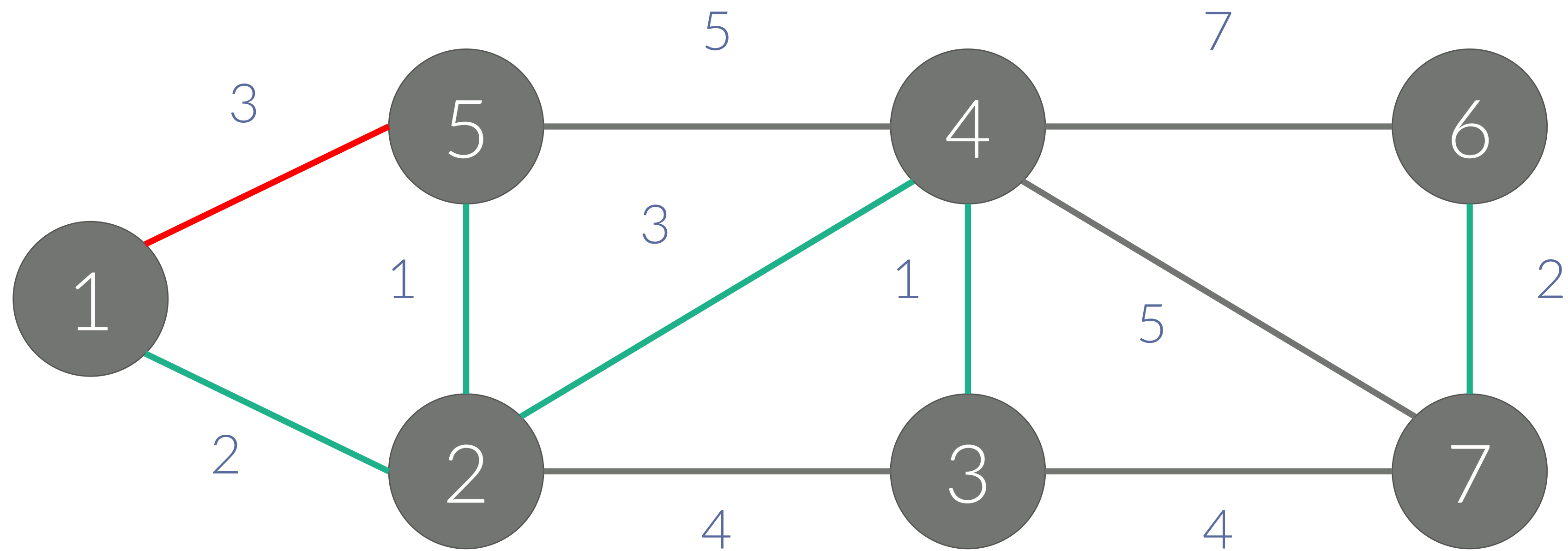
81



크루스칼

Kruskal

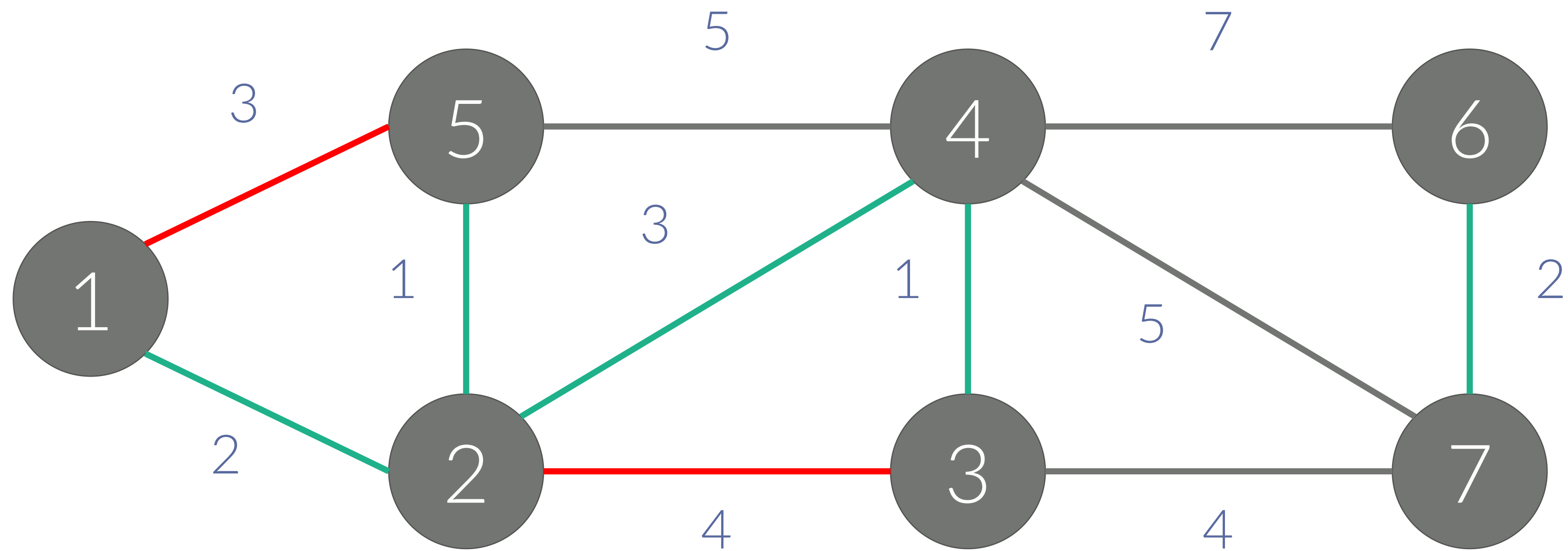
82



크루스칼

Kruskal

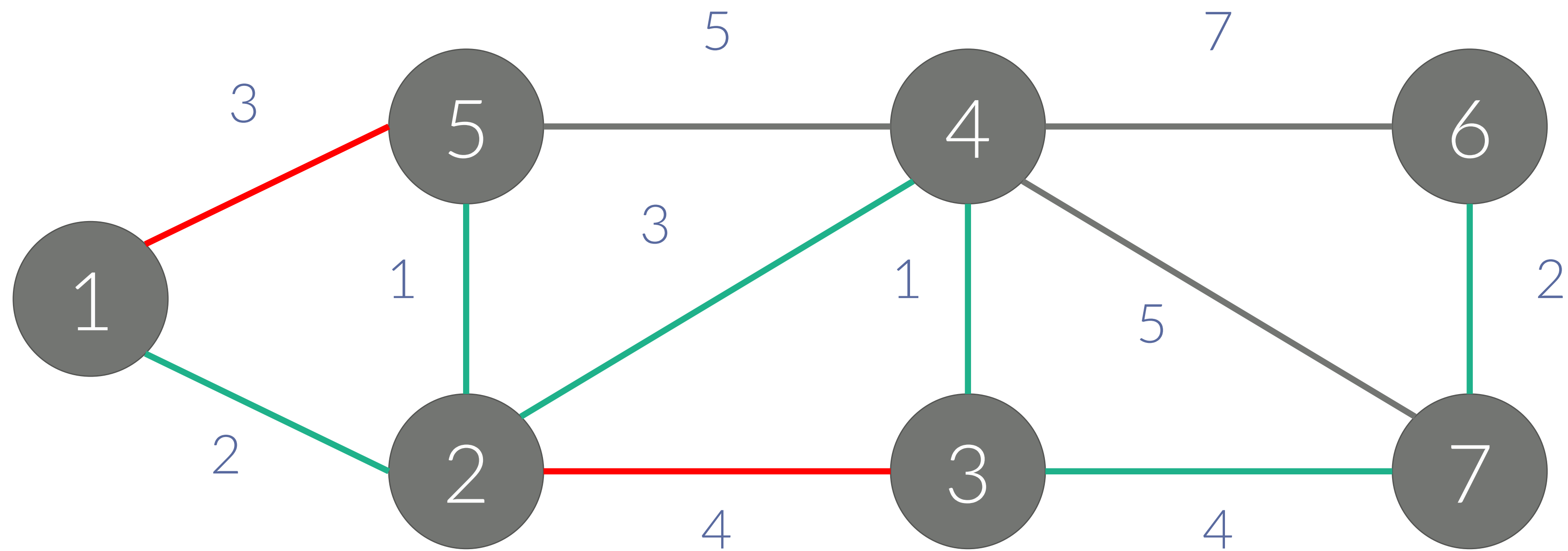
83



크루스칼

Kruskal

84



최소 스패닝 트리

<https://www.acmicpc.net/problem/1197>

- 그래프가 주어졌을 때, 그 그래프의 최소 스패닝 트리를 구하기

MST

최소 스패닝 트리

<https://www.acmicpc.net/problem/1197>

$O(E \log E)$

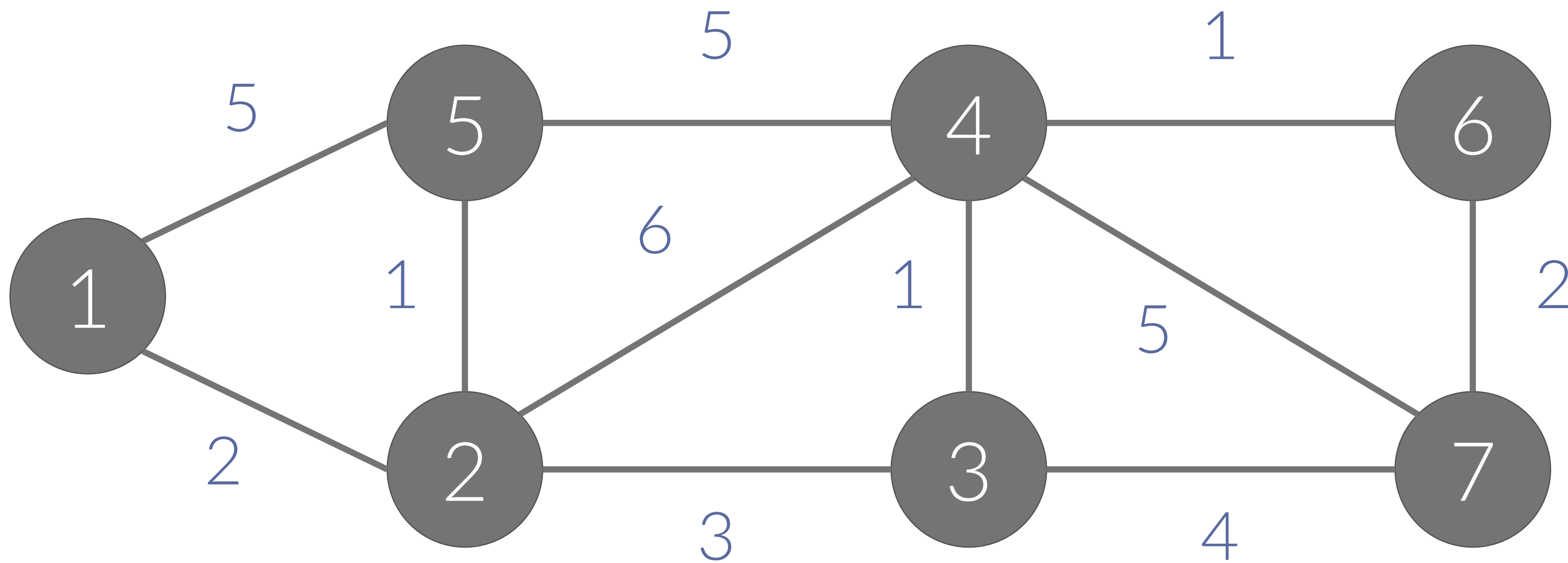
- 소스: <http://codeplus.codes/0b581f44383d42e081eaa14e524e36ad>

최단 경로

최단 경로

Single Source Shortest Path

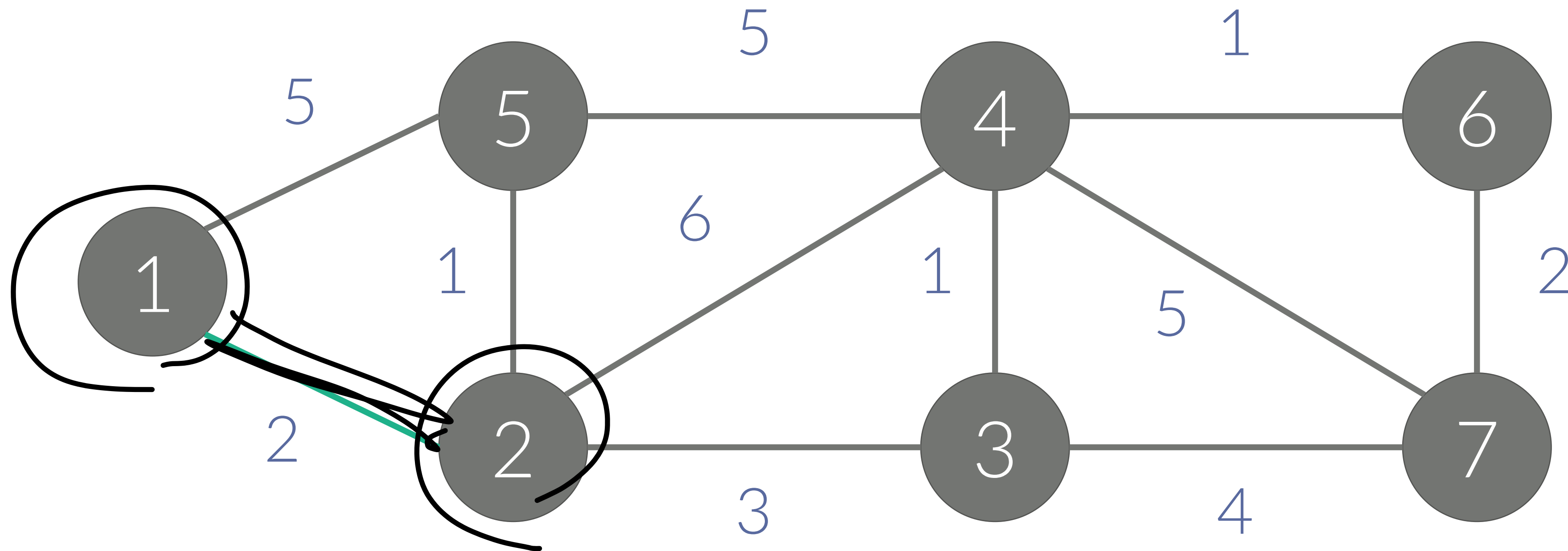
- 시작점이 1개일 때, 다른 모든 곳으로 가는 최단 경로 구하기



최단 경로

Single Source Shortest Path

- 1에서 2까지 최단 경로

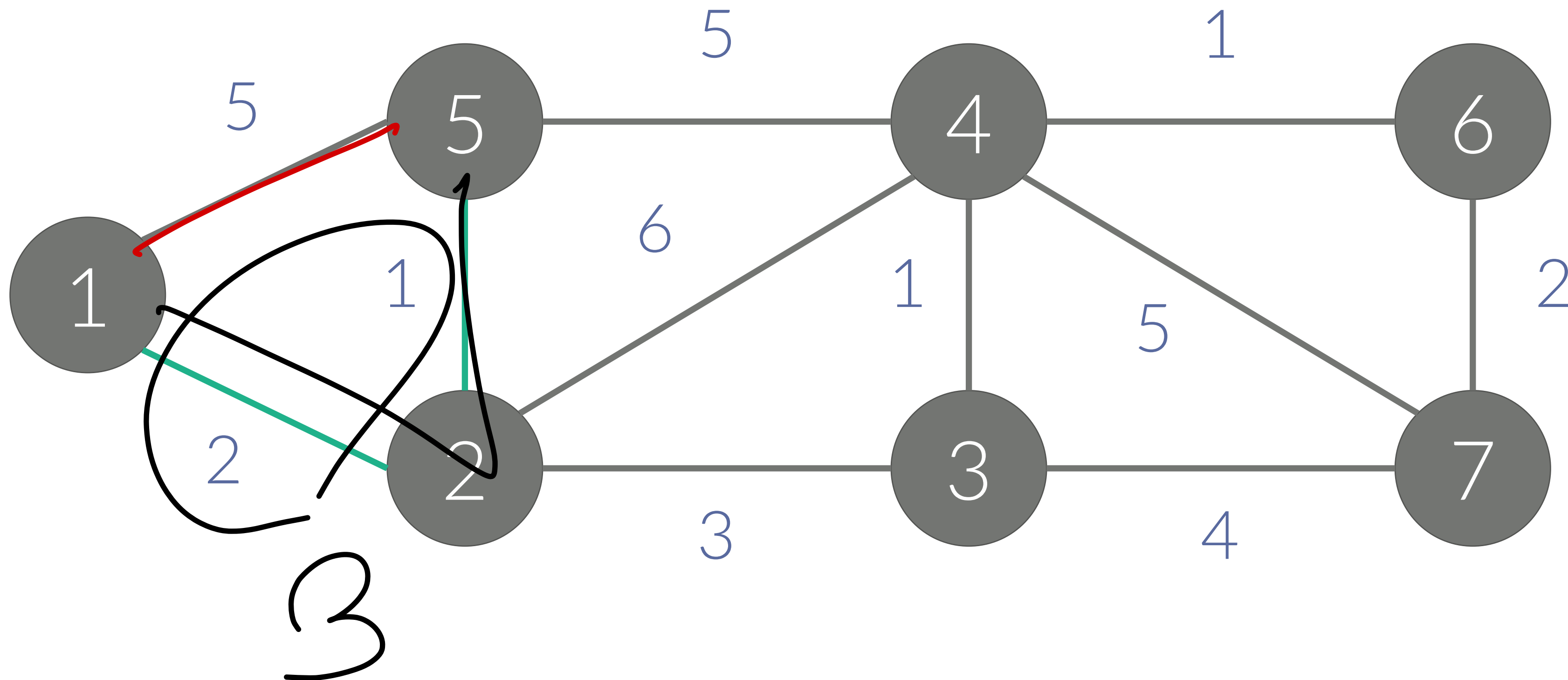


최단 경로

90

Single Source Shortest Path

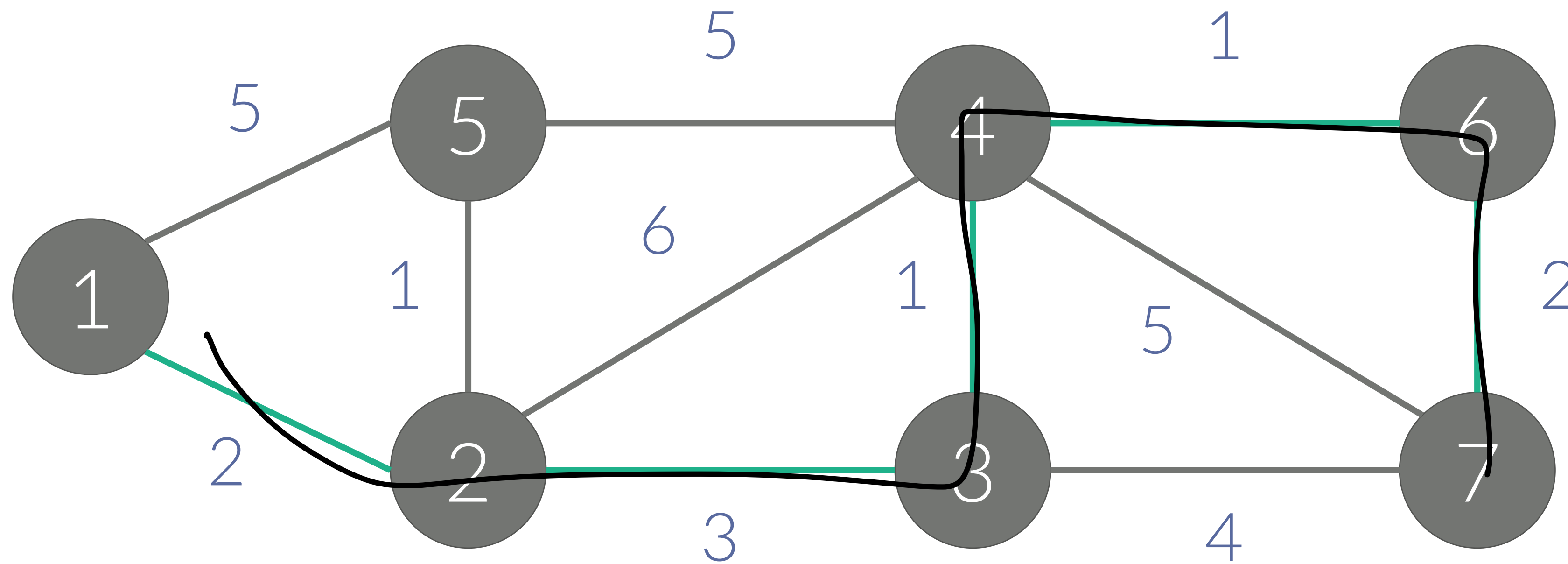
- 1에서 5까지 최단 경로



최단 경로

Single Source Shortest Path

- 1에서 7까지 최단 경로



최단 경로

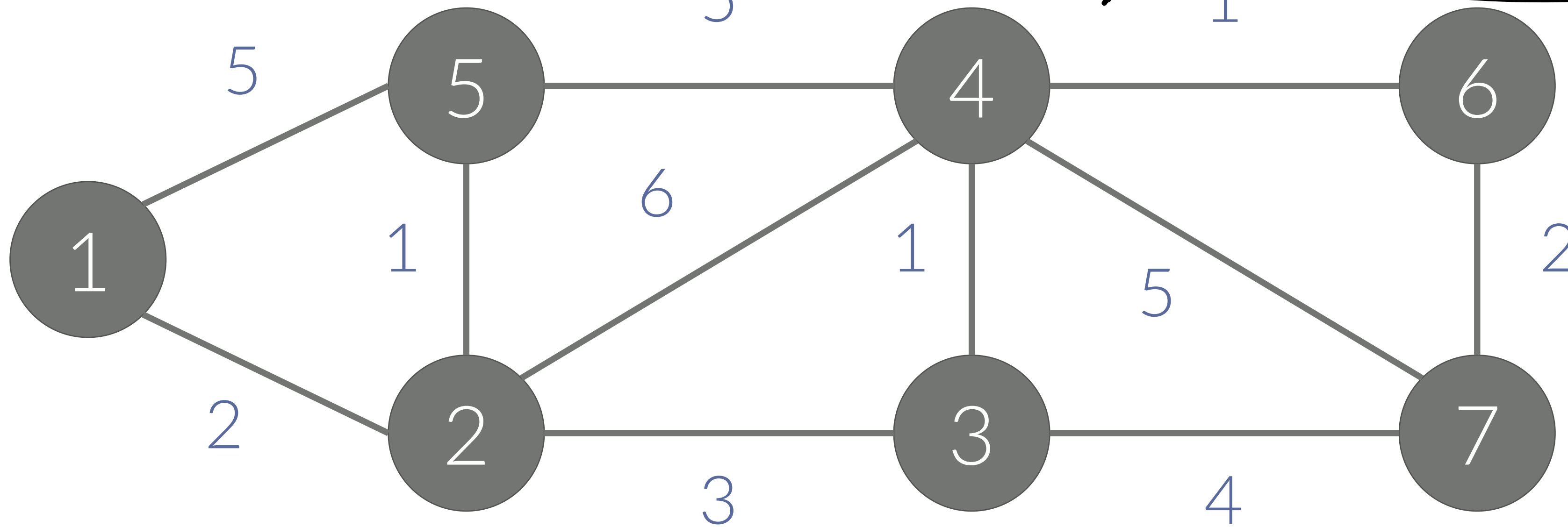
경로: N 개

Single Source Shortest Path

A → B로 가는 최단 경로는 최대 $N-1$ 개의 간선으로 이루어져 있다

A → $N-2$ 개의 경로 → B

→ 0 → 5 → 2 → 3 → 7 → 1



$dist[i]$

= 시작점 $\rightarrow i$ 최단경로



가장 작은

벨만포드

~~$dist[to] > dist[from] + Cost$~~

$dist[to]$
 $= dist[from] + Cost$

$(N-1)회$

벨만포드

Bellman-Ford Algorithm

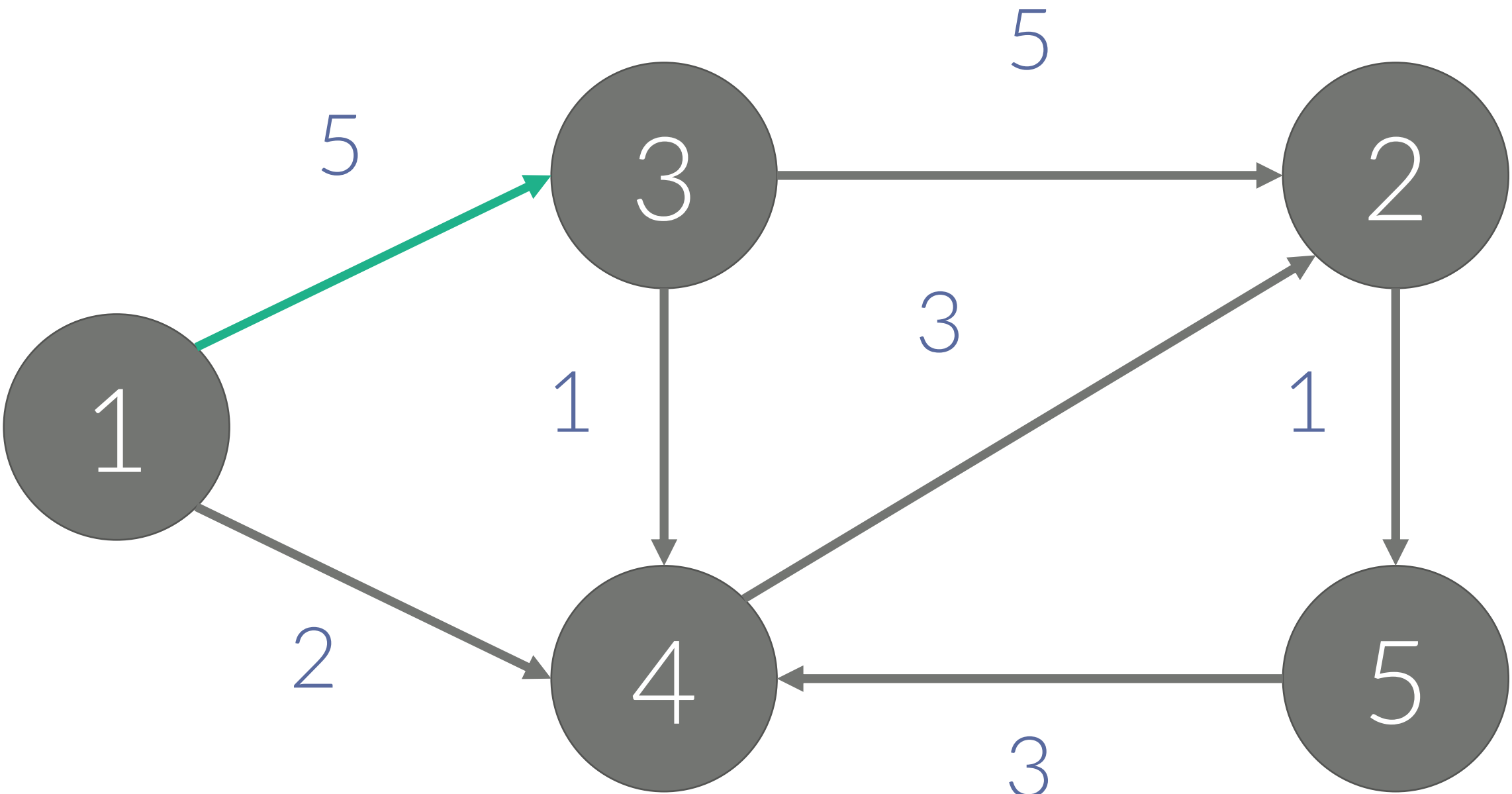
- $\text{dist}[i]$ = 시작점에서 i 로 가는 최단경로
1. 모든 간선 e (from , to , cost)에 대해서 다음을 검사한다.
 - $\text{dist}[\text{to}] = \min(\text{dist}[\text{to}], \text{dist}[\text{from}] + \text{cost})$
- 1번 과정을 총 $N-1$ 번 반복한다.

벨만포드

Bellman-Ford Algorithm

- 1단계

i	1	2	3	4	5
dist[i]	0	∞	5	∞	∞

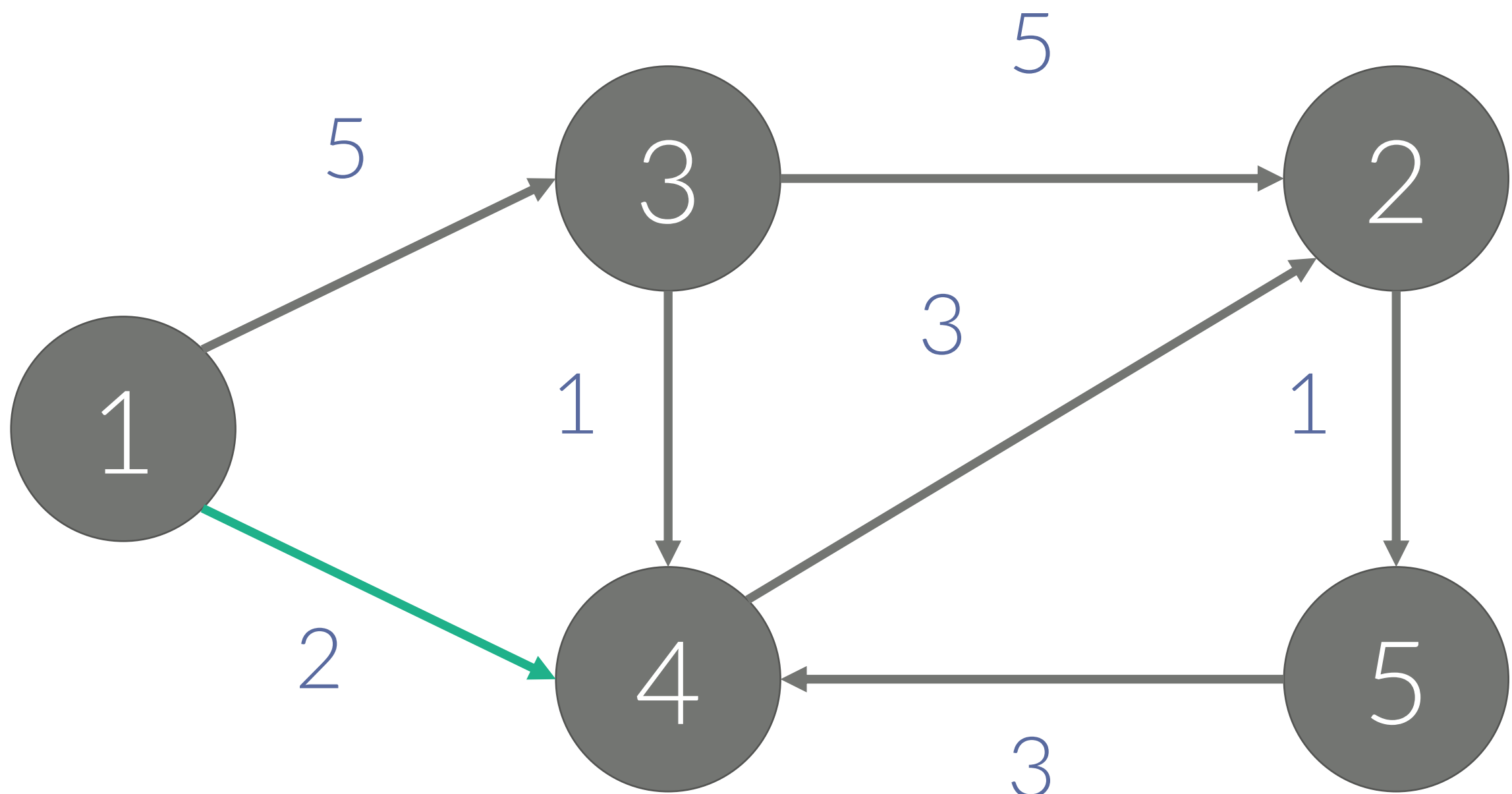


벨만포드

Bellman-Ford Algorithm

- 1단계

i	1	2	3	4	5
dist[i]	0	∞	5	2	∞

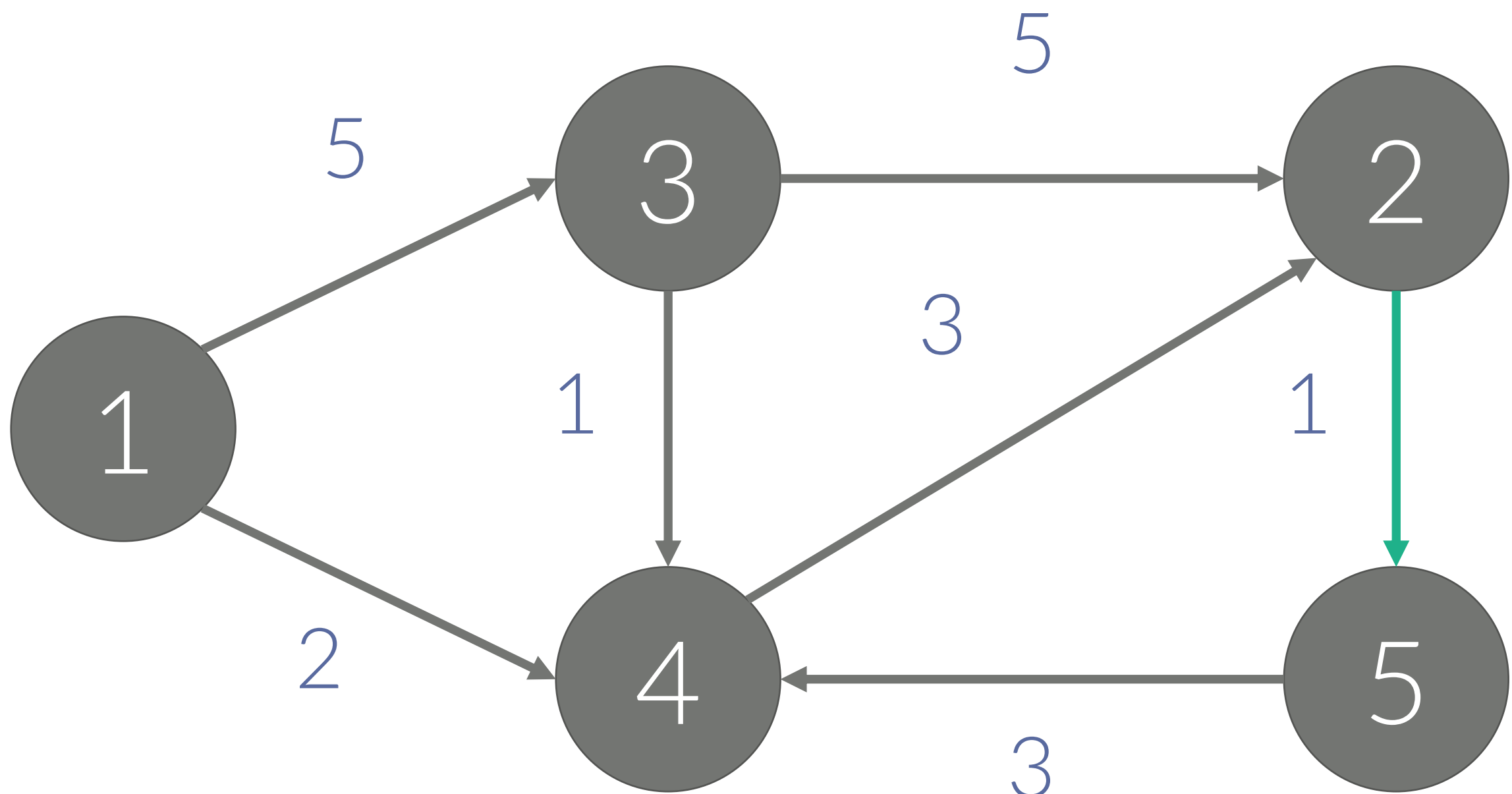


벨만포드

Bellman-Ford Algorithm

- 1단계

i	1	2	3	4	5
dist[i]	0	∞	5	2	∞

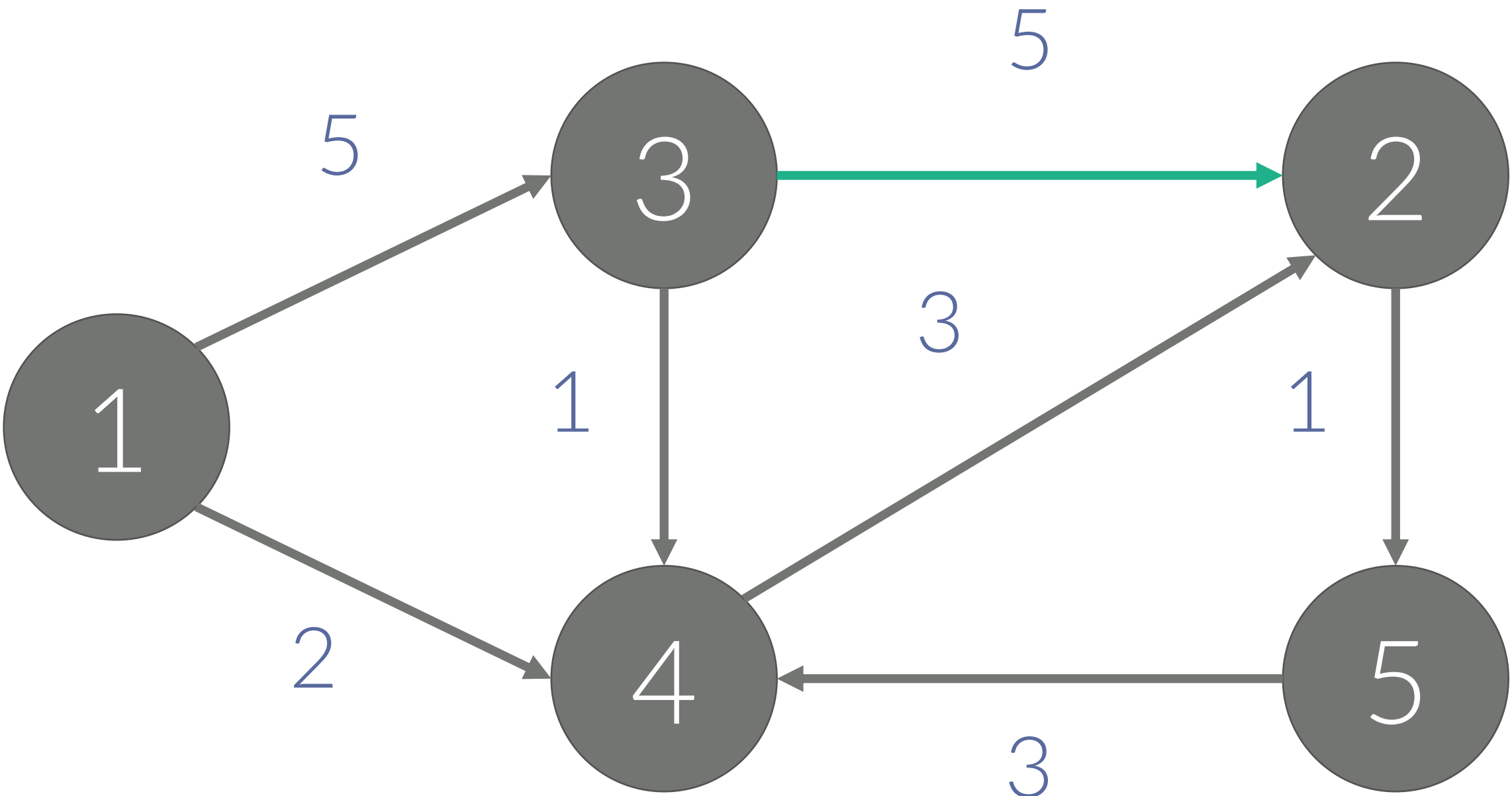


벨만포드

Bellman-Ford Algorithm

- 1단계

i	1	2	3	4	5
dist[i]	0	10	5	2	∞

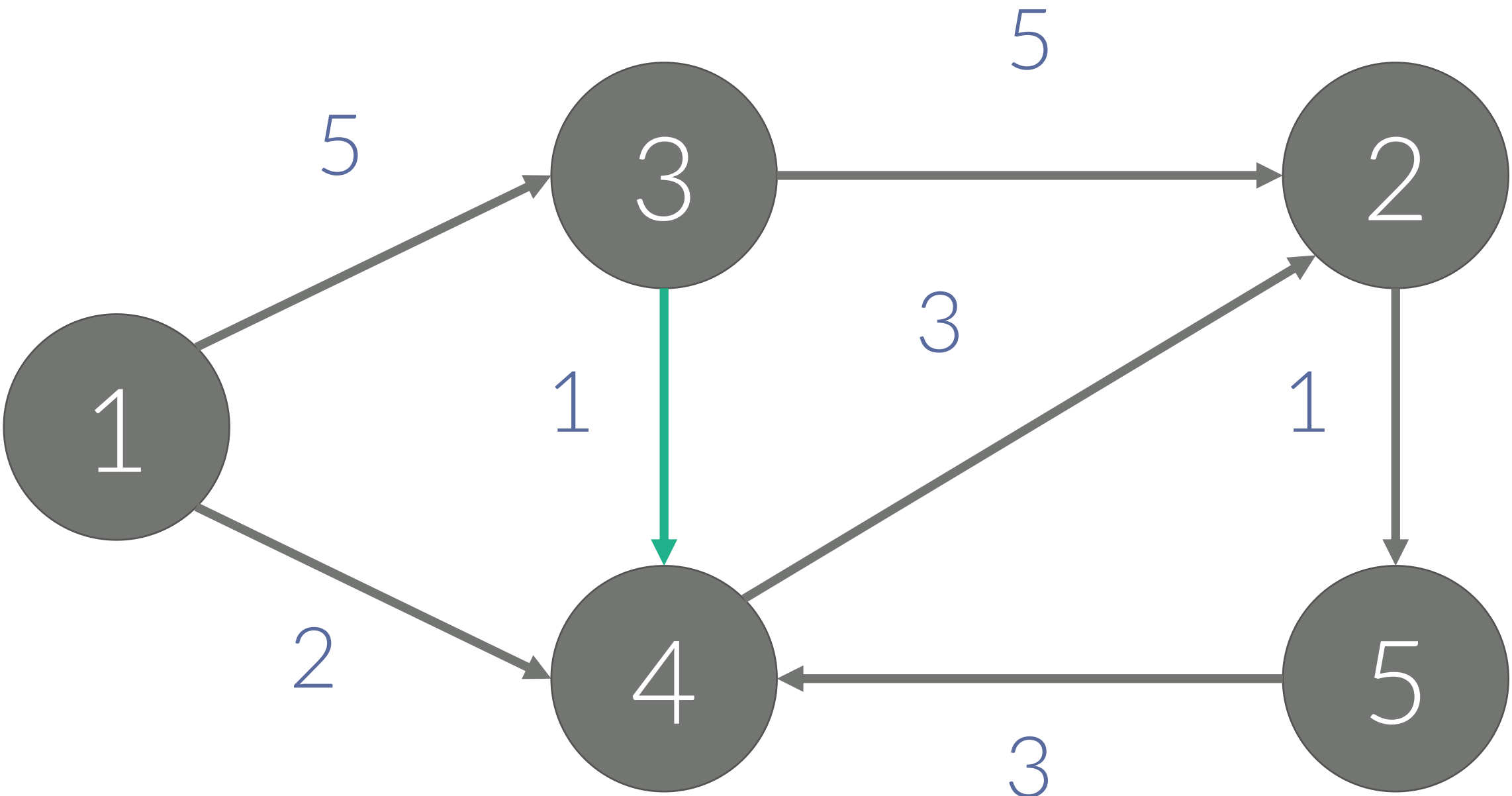


벨만포드

Bellman-Ford Algorithm

- 1단계

i	1	2	3	4	5
dist[i]	0	10	5	2	∞

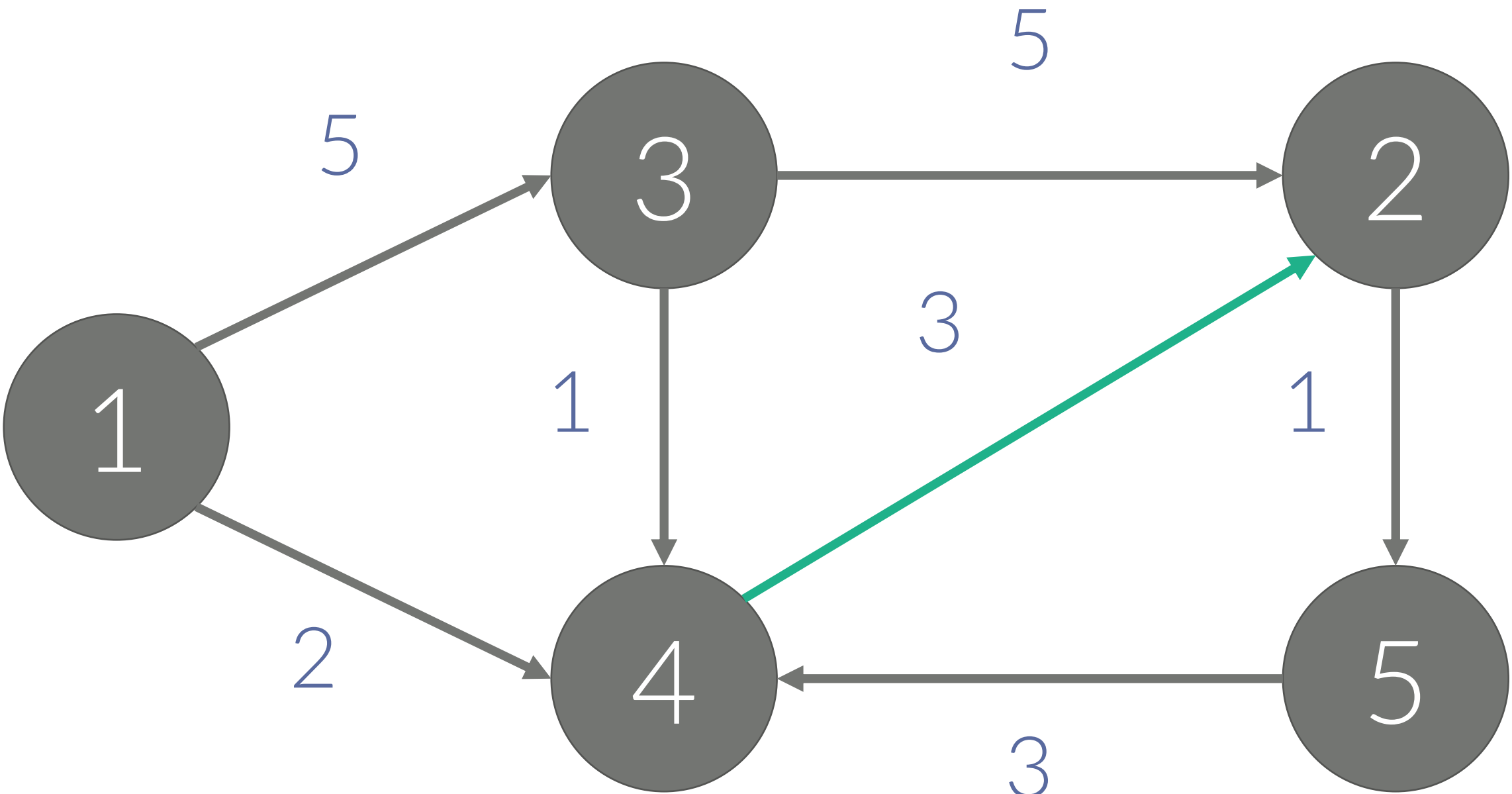


벨만포드

Bellman-Ford Algorithm

- 1단계

i	1	2	3	4	5
dist[i]	0	5	5	2	∞

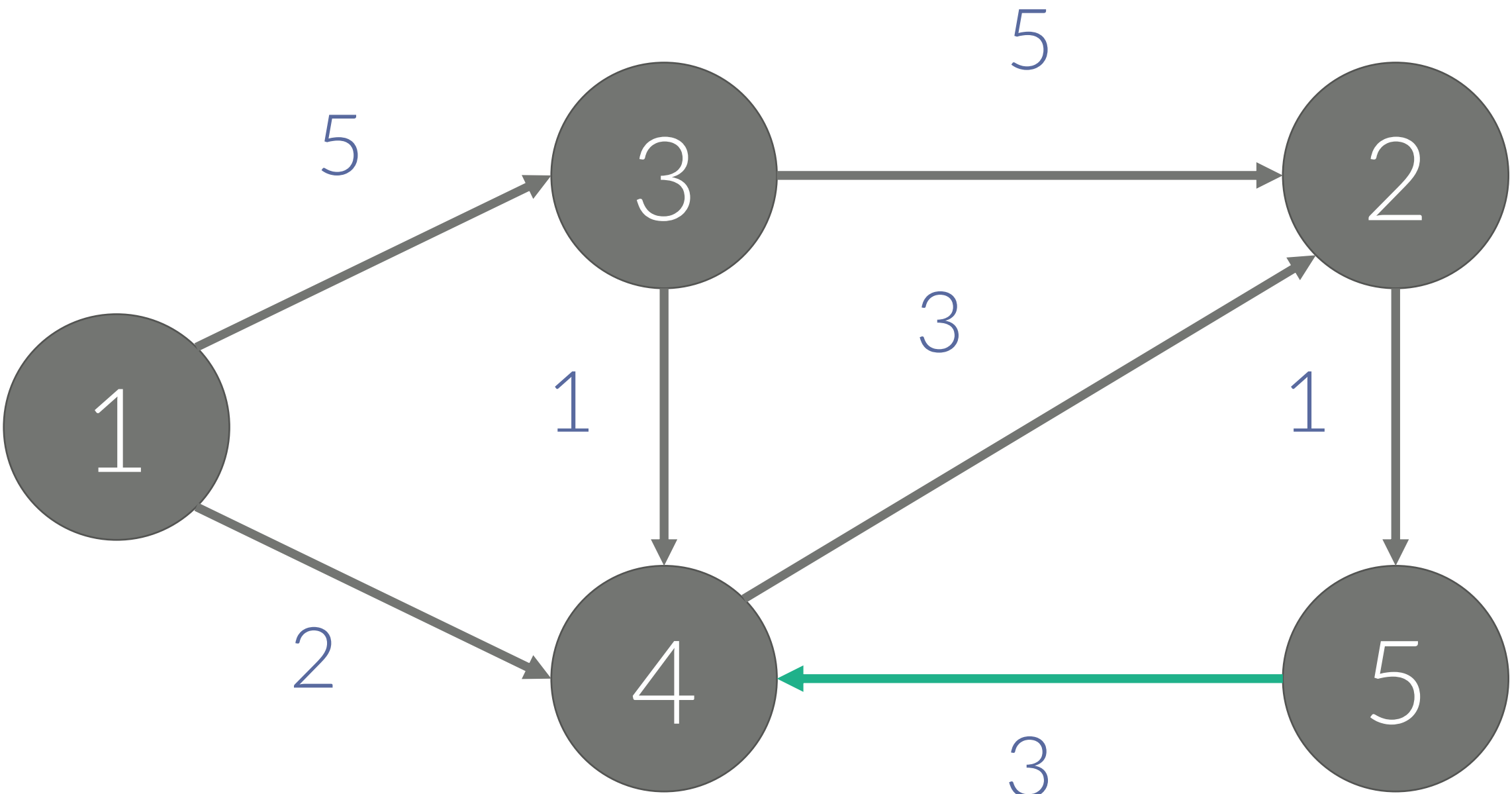


벨만포드

Bellman-Ford Algorithm

- 1단계

i	1	2	3	4	5
dist[i]	0	5	5	2	∞

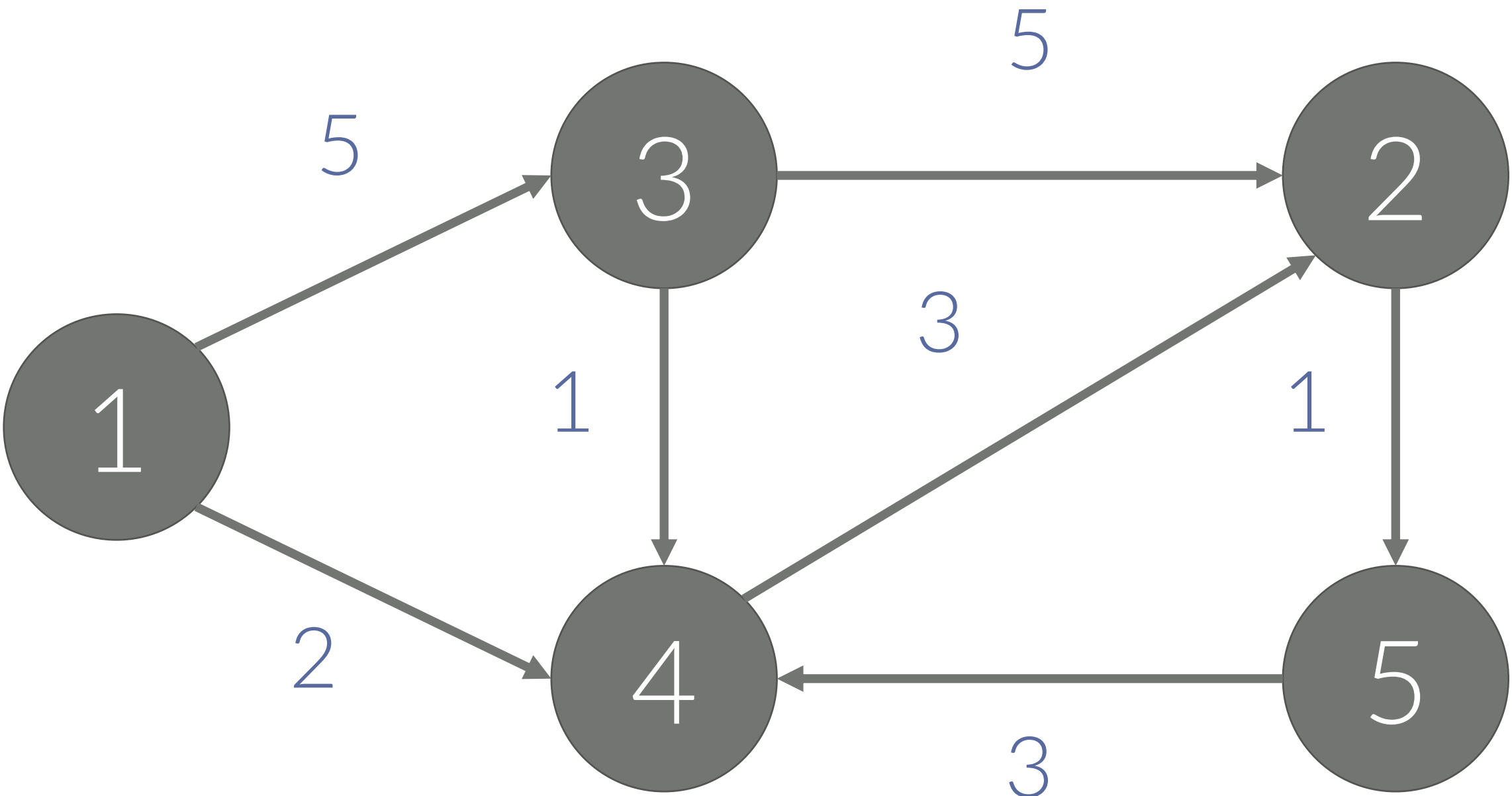


벨만포드

Bellman-Ford Algorithm

- 이런 단계를 N-1번 반복한다

i	1	2	3	4	5
dist[i]	0	5	5	2	∞

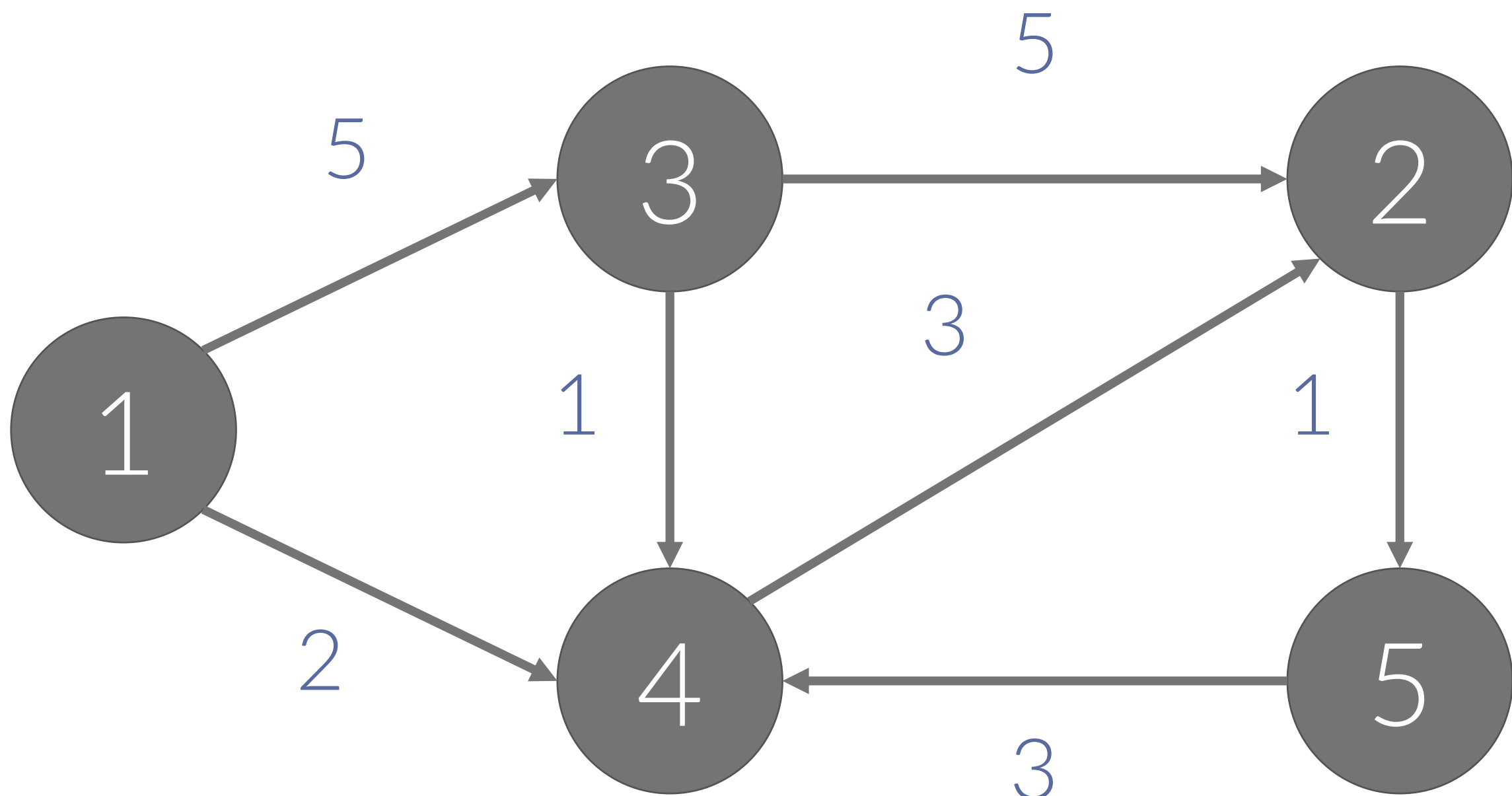


벨만포드

Bellman-Ford Algorithm

- 이런 단계를 N-1번 반복한다

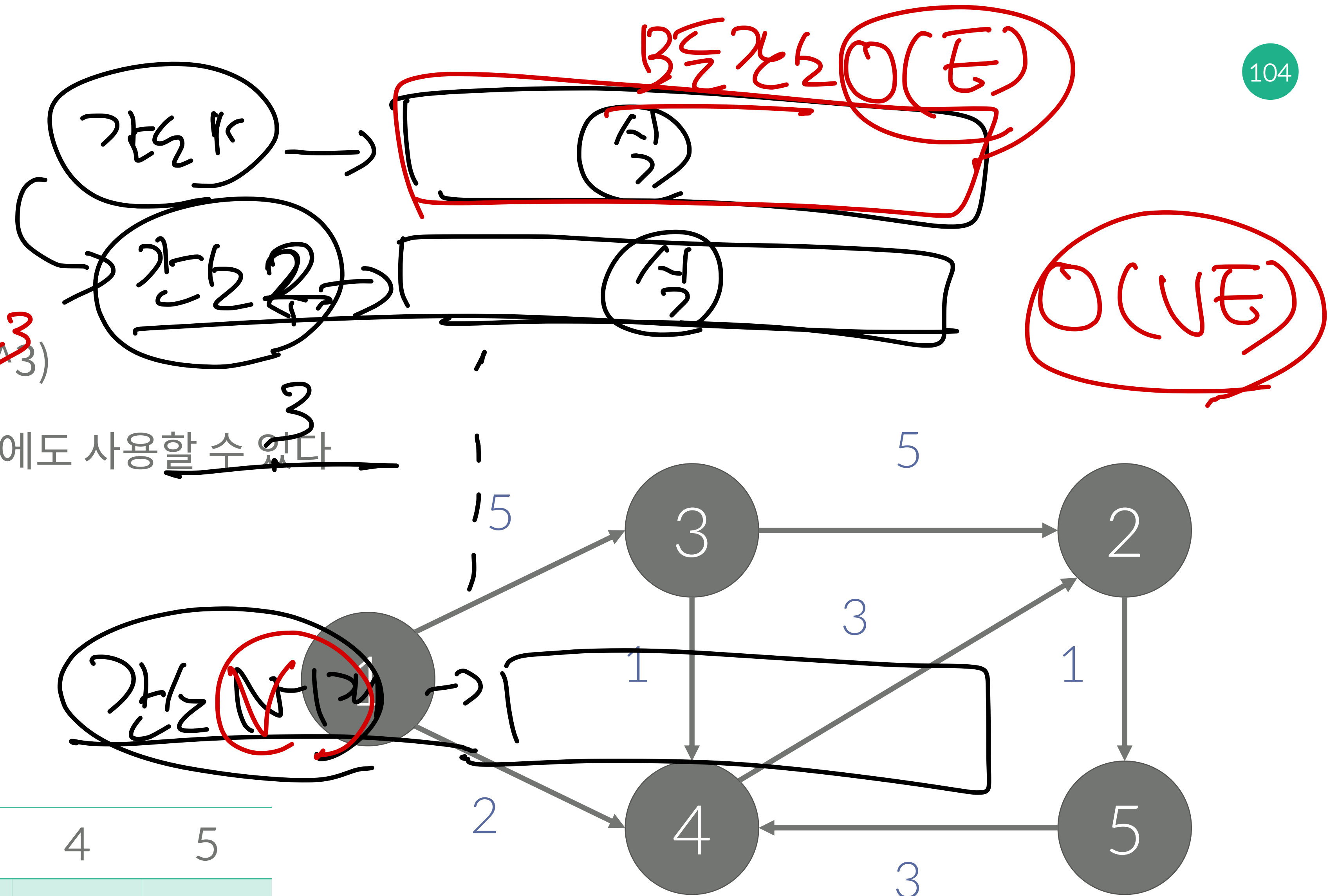
i	1	2	3	4	5
dist[i]	0	5	5	2	6



벨만포드

Bellman-Ford Algorithm

- 시간 복잡도: $O(VE)$
- $E \leq V^2$ 이기 때문에 $O(V^3)$
- 가중치가 음수가 있는 경우에도 사용할 수 있다



i	1	2	3	4	5
dist[i]	0	5	5	2	6

타임머신

105

<https://www.acmicpc.net/problem/11657>

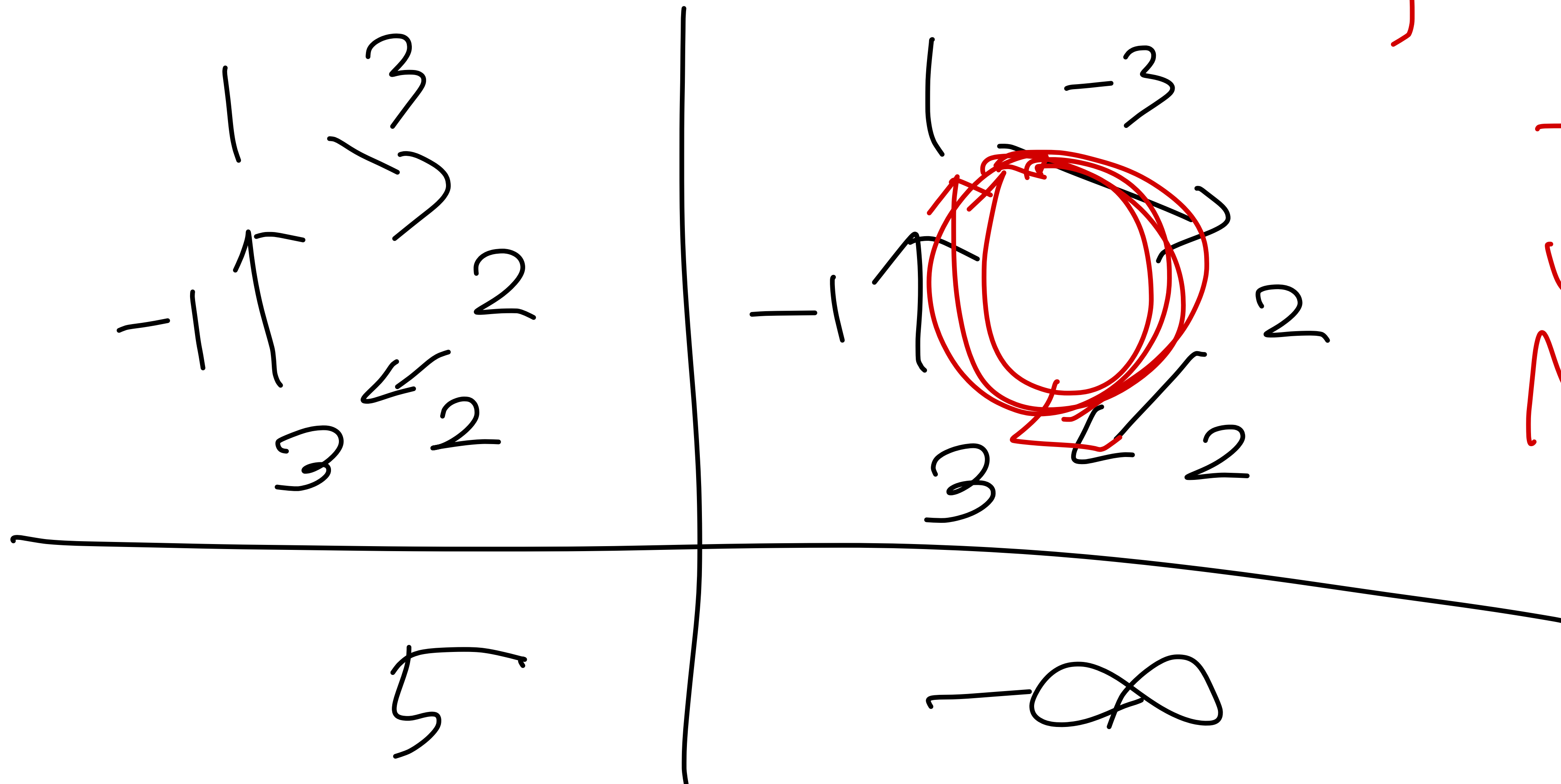
- N개의 도시가 있다
- 한 도시에서 출발하여 다른 도시에 도착하는 버스가 M개 있다
- 각 버스는 A, B, C로 나타낼 수 있는데, A는 시작도시, B는 도착도시, C는 버스를 타고 이동하는데 걸리는 시간이다
- 시간 C가 양수가 아닌 경우가 있다.
- $C = 0$ 인 경우는 순간 이동을 하는 경우, $C < 0$ 인 경우는 타임머신으로 시간을 되돌아가는 경우이다.
- 1번 도시에서 출발해서 나머지 도시로 가는 가장 빠른 시간을 구하는 프로그램을 작성하시오.

타임머신

<https://www.acmicpc.net/problem/11657>

106

- 음수가 있기 때문에, 벨만포드 알고리즘을 이용해서 최단거리를 구해야 한다.



타임머신

107

<https://www.acmicpc.net/problem/11657>

- 소스: <http://codeplus.codes/a6ccd548f6284f80a186b83cd41848c5>

단계 1 :

단계 2 :

⋮

단계 N-1 :

시간

시간 !

시간

시간 2012

⋮

시간

시간 N-1

시간

시간

시간 N

시간 N

웜홀

<https://www.acmicpc.net/problem/1865>

- N 개의 지점 사이에는 M 개의 도로와 W 개의 웜홀이 있다
- 어떤 지점에서 출발을 하여서 시간여행을 하기 시작하여 다시 출발을 하였던 위치로 돌아왔을 때, 출발을 하였을 때 보다 시간이 되돌아 가 있는 경우 찾기

웜홀

109

<https://www.acmicpc.net/problem/1865>

- 음수로 되어있는 사이클을 찾는 문제

<https://www.acmicpc.net/problem/1865>

- 최단 경로는 항상 최대 $N-1$ 개로 구성되어 있기 때문에
- N 번째 단계에서 최단 경로가 갱신되면, 음수로 되어있는 사이클이 존재하는 것이다

웜홀

111

<https://www.acmicpc.net/problem/1865>

- 소스: <http://codeplus.codes/c97c5258626548bf9360dbc3edcfceb1>

다익스트라

$\sqrt{3} | \tau_{n+2} |$

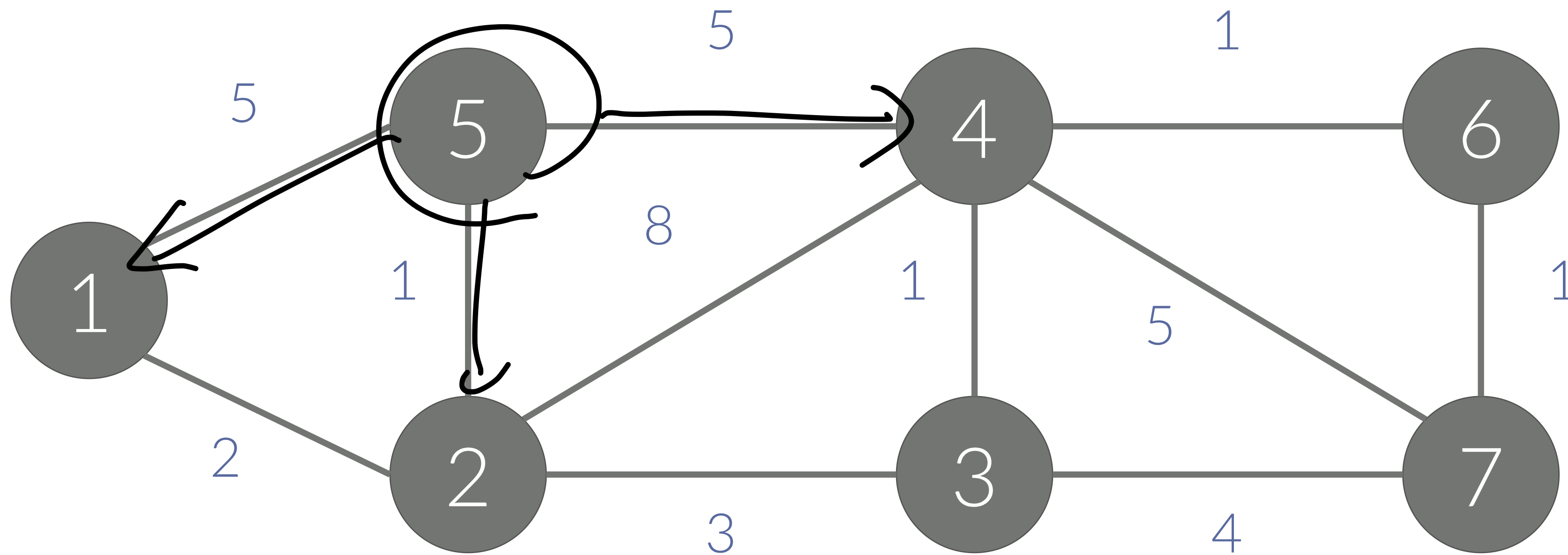
$\frac{V^2}{\epsilon_0} \approx \frac{E_{Q_2}^2}{\epsilon_0}$

다익스트라

113

Dijkstra Algorithm

- $\text{dist}[i]$ = 시작 \rightarrow i 의 최단 거리
- $\text{check}[i]$ = i 를 검사했으면 true, 아니면 false



다익스트라

Dijkstra Algorithm

114

1. 검사하지 않은 정점 중에서 dist의 값이 가장 작은 정점 v 를 선택한다.
 2. v 와 연결된 모든 정점을 검사한다.
 - 간선을 (from, to, cost)라고 했을 때
 - $\text{dist}[\text{to}] > \text{dist}[\text{from}] + \text{cost}$ 이면 갱신해준다.
- 1, 2단계를 모든 정점을 검사할 때까지 계속한다.

다익스트라

Dijkstra Algorithm

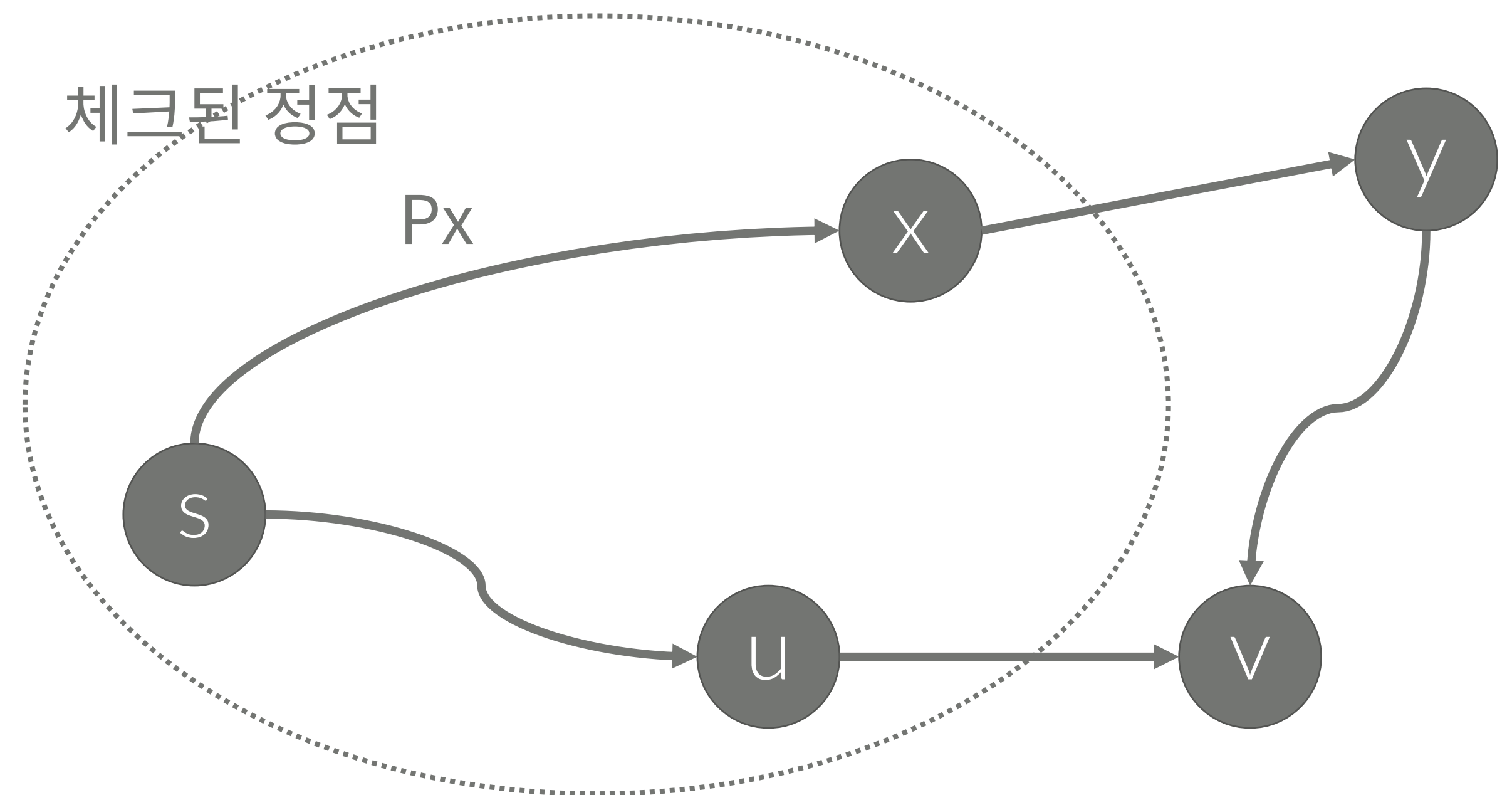
- 다익스트라의 증명
- $\text{dist}[v]$ = 다익스트라 알고리즘을 이용해서 구한 v 까지 최단거리
- $\text{shortest}[v]$ = v 까지 최단거리
- 체크된 정점의 개수가 1개인 경우에는 시작 정점(start)만 체크된 상태이다.
- 따라서, $\text{dist}[\text{start}] = \text{shortest}[\text{start}] = 0$ 이므로 최단 거리가 맞다.
- v 를 마지막으로 체크한 정점이라고 했을 때, $\text{dist}[x] = \text{shortest}[x]$ (x 는 체크된 정점) 을 증명
- 즉, 각 단계마다 $\text{dist}[v] = \text{shortest}[v]$ 만 증명하면 된다.

다익스트라

Dijkstra Algorithm

116

- 각 단계마다 $\text{dist}[v] = \text{shortest}[v]$ 를 증명 (v 는 마지막으로 체크된 정점)
- v 까지 최단 거리가 $u \rightarrow v$ 간선으로 업데이트 되었다고 하자.
- $\text{temp}[v] = s$ 에서 u 로 가는 최단 거리 ($\text{shortest}[u]$) + $u \rightarrow v$ 간선의 비용
- $s \rightarrow v$ 로 가는 경로 P 가 있다고 하자.
- 모든 P 의 거리는 $\text{temp}[v]$ 보다 크거나 같다.
- P 의 경로에서 마지막 체크된 정점을 x 라고 하자
- P 에서 x 까지 거리를 P_x 라고 하자.

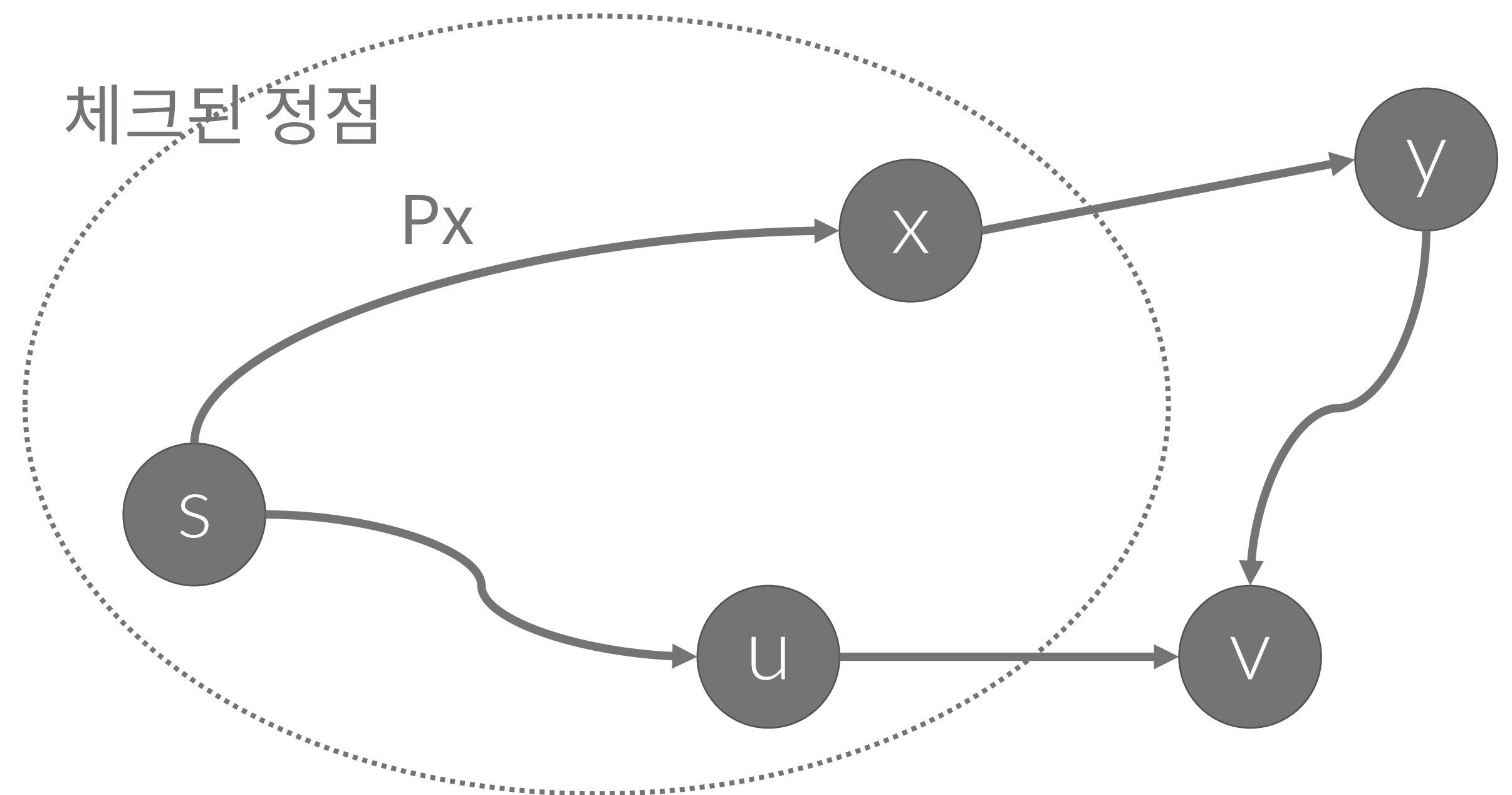


다익스트라

Dijkstra Algorithm

117

- $\text{temp}[v] \leq \text{temp}[y]$ (v 가 마지막으로 선택된 정점이기 때문)
- $\text{temp}[y] \leq \text{dist}[x] + |x \rightarrow y|$ (temp 의 정의 때문)
- $\text{dist}[x] + |x \rightarrow y| \leq |P_x| + |x \rightarrow y|$
- $|P_x| + |x \rightarrow y| \leq |P|$



다익스트라

Dijkstra Algorithm

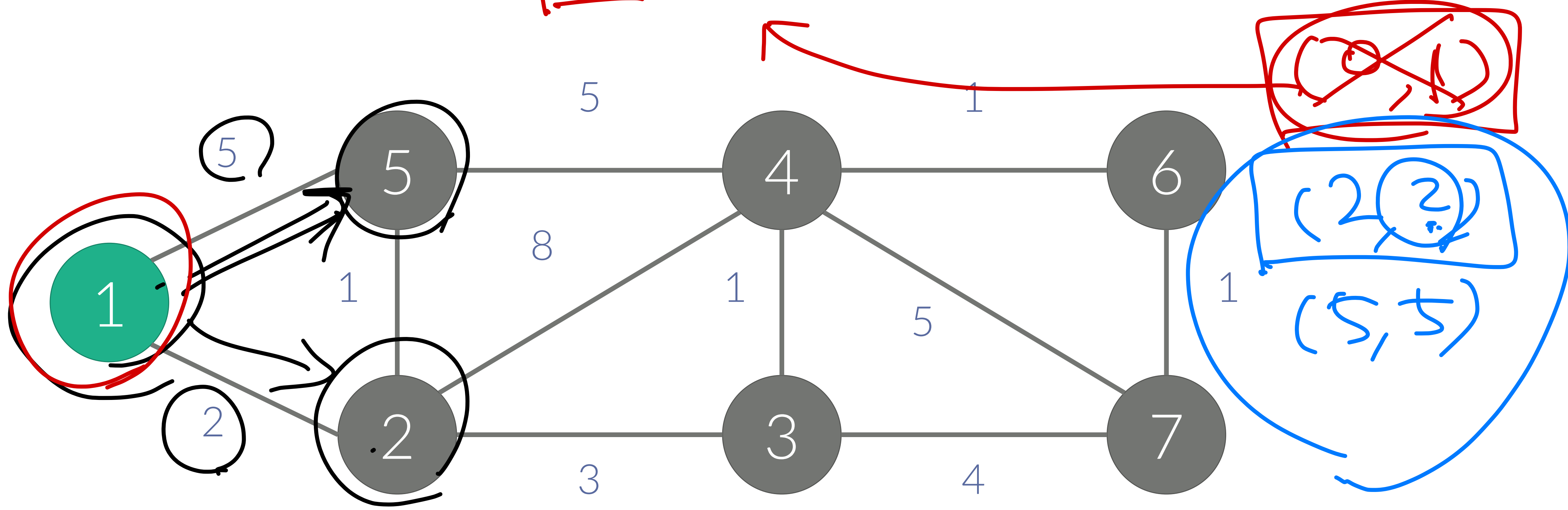
• 선택: 1

(dist[i], i)

시작점 1

dist ↓ Check X

i	1	2	3	4	5	6	7
D[i]	0	2	∞	∞	5	∞	∞
C[i]	1						



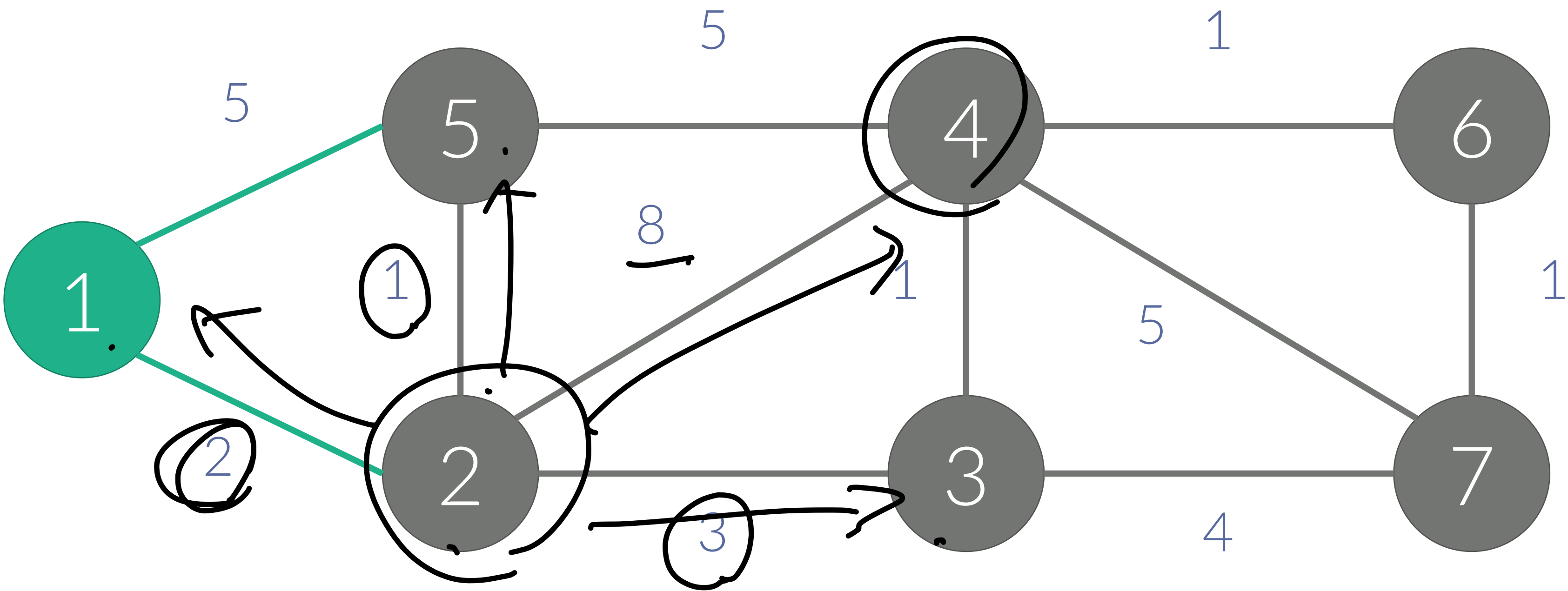
다익스트라

Dijkstra Algorithm

- 선택: 1

4 dist ↓ Check x

i	1	2	3	4	5	6	7
D[i]	0	2	5	10	3	∞	∞
C[i]	1						



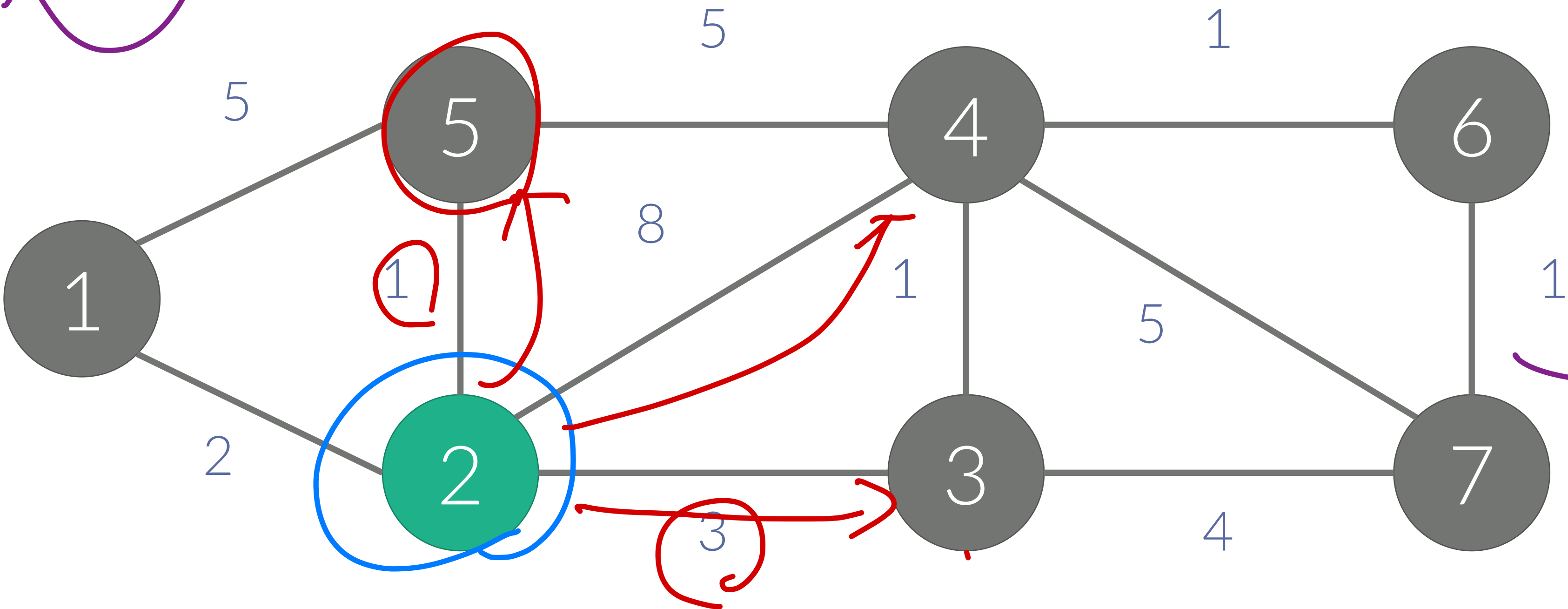
다익스트라

Dijkstra Algorithm

• 선택: 2

i	1	2	3	4	5	6	7
D[i]	0	2	5	10	3	∞	∞
C[i]	1	1					

(dist, 1)



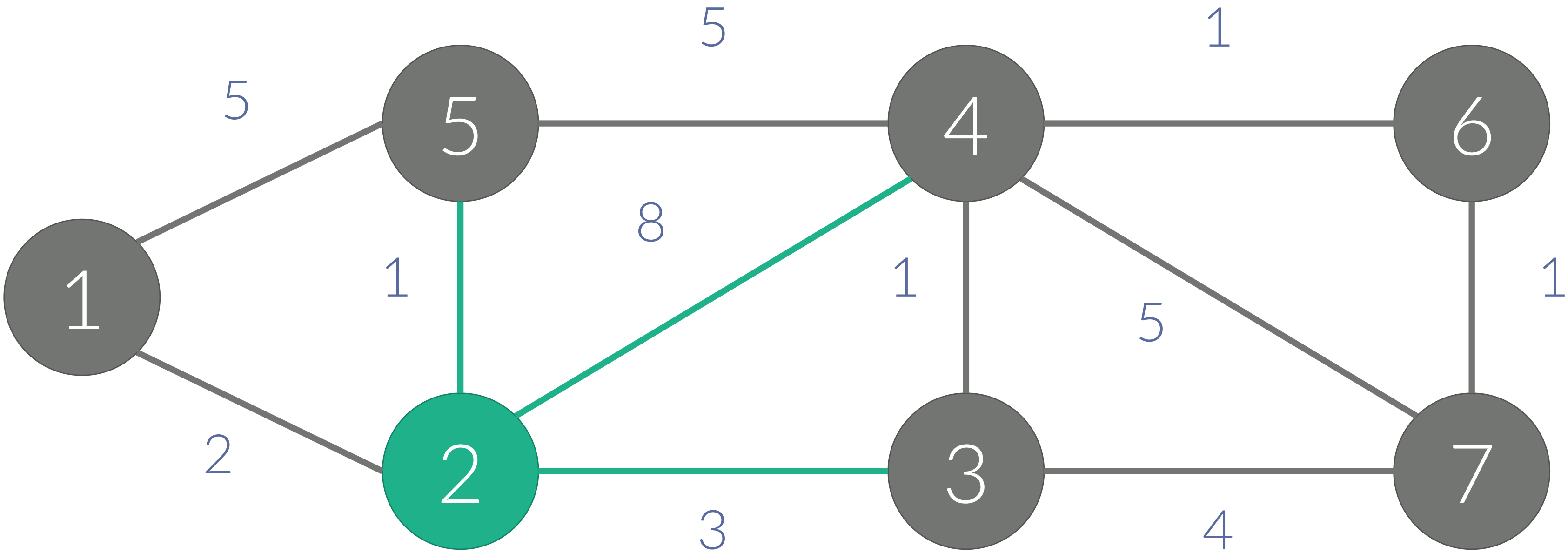
~~(2, 2)~~
(3, 5)
~~(3, 5)~~
(10, 4)
(5, 3)

다익스트라

Dijkstra Algorithm

- 선택: 2

i	1	2	3	4	5	6	7
D[i]	0	2	5	10	3	∞	∞
C[i]	1	1					

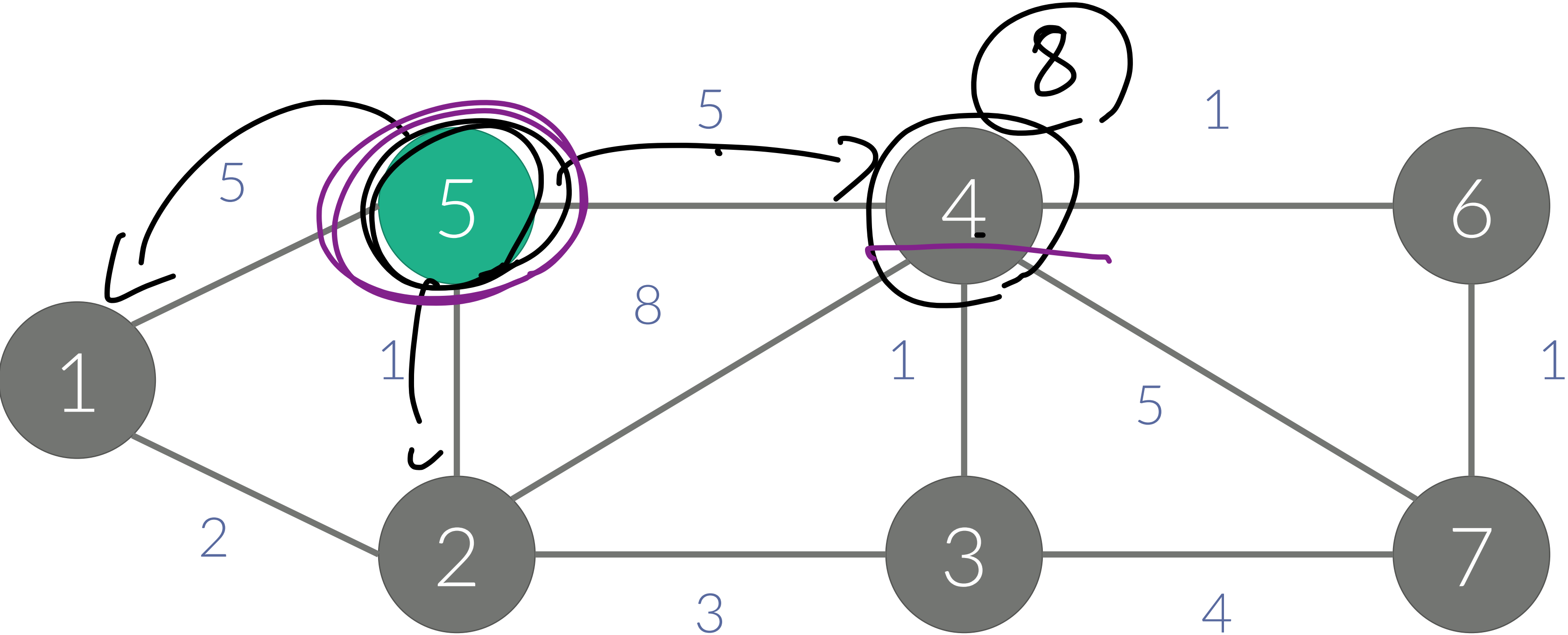


다익스트라

Dijkstra Algorithm

- 선택: 5

i	1	2	3	4	5	6	7
D[i]	0	2	5	8	3	∞	∞
C[i]	1	1			1		



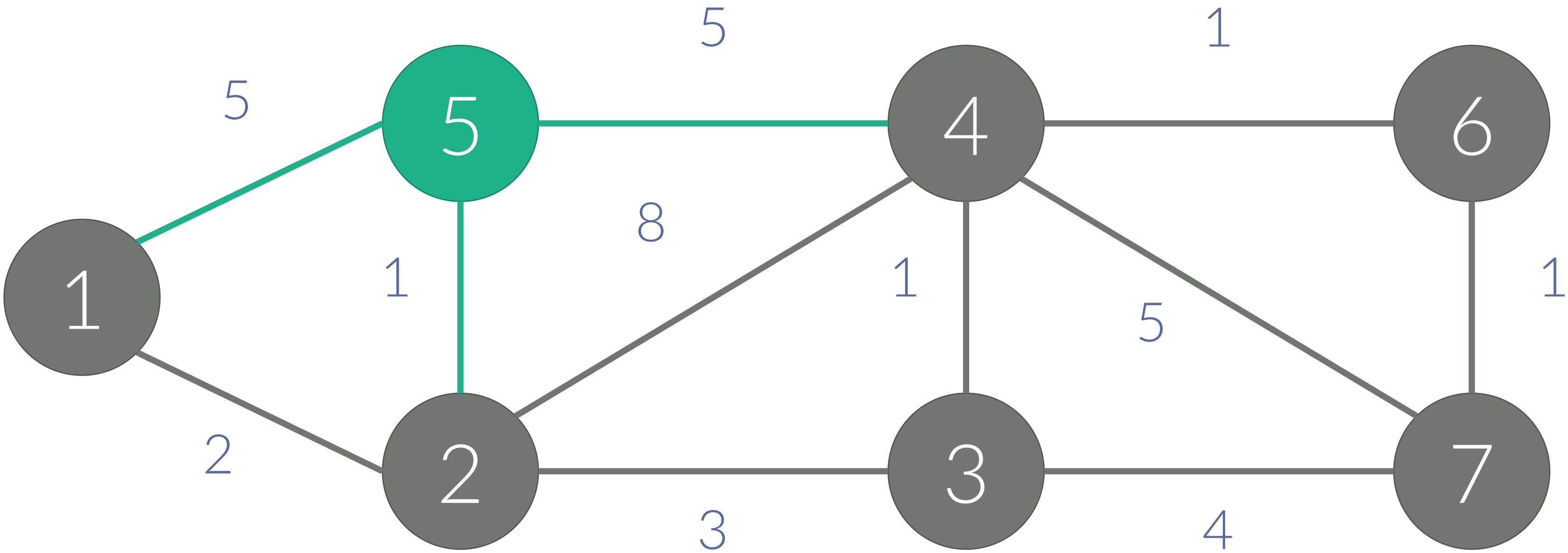
~~(5, 8)~~
(8, 4)
(10, 4)
(5, 3)

다익스트라

Dijkstra Algorithm

- 선택: 5

i	1	2	3	4	5	6	7
D[i]	0	2	5	8	3	∞	∞
C[i]	1	1			1		



다익스트라

Dijkstra Algorithm

• 선택: 3

(dist, 1)

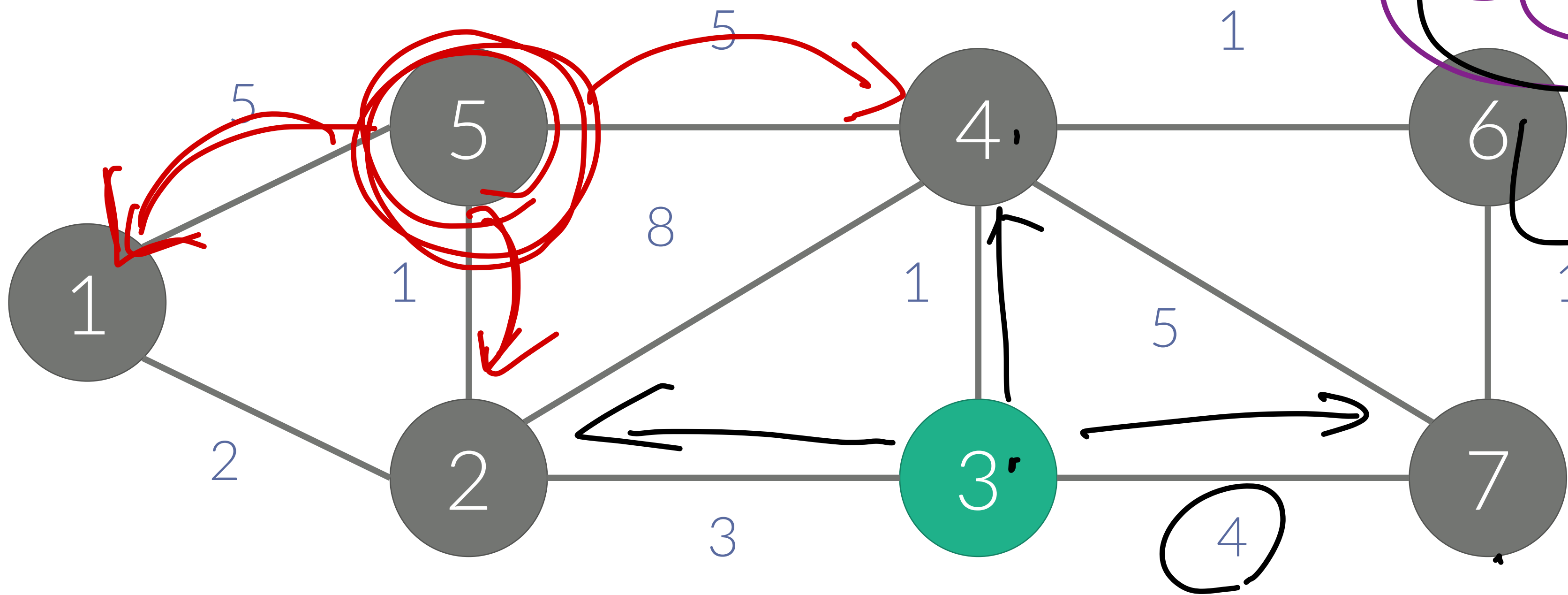
Min Heap $\rightarrow O(E \log E)$

124

i	1	2	3	4	5	6	7
D[i]	0	2	5	6	3	∞	9
C[i]	1	1	1	1	1	1	1

$E \log V$

BST

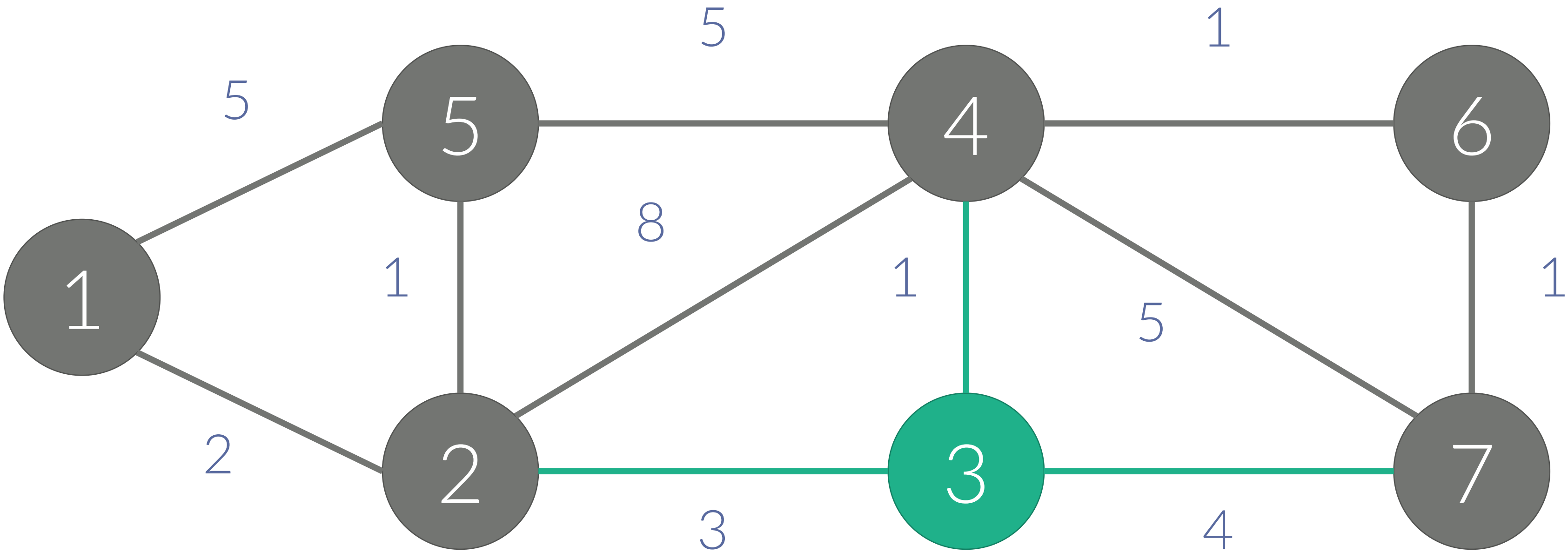


다익스트라

Dijkstra Algorithm

- 선택: 3

i	1	2	3	4	5	6	7
D[i]	0	2	5	6	3	∞	9
C[i]	1	1	1		1		

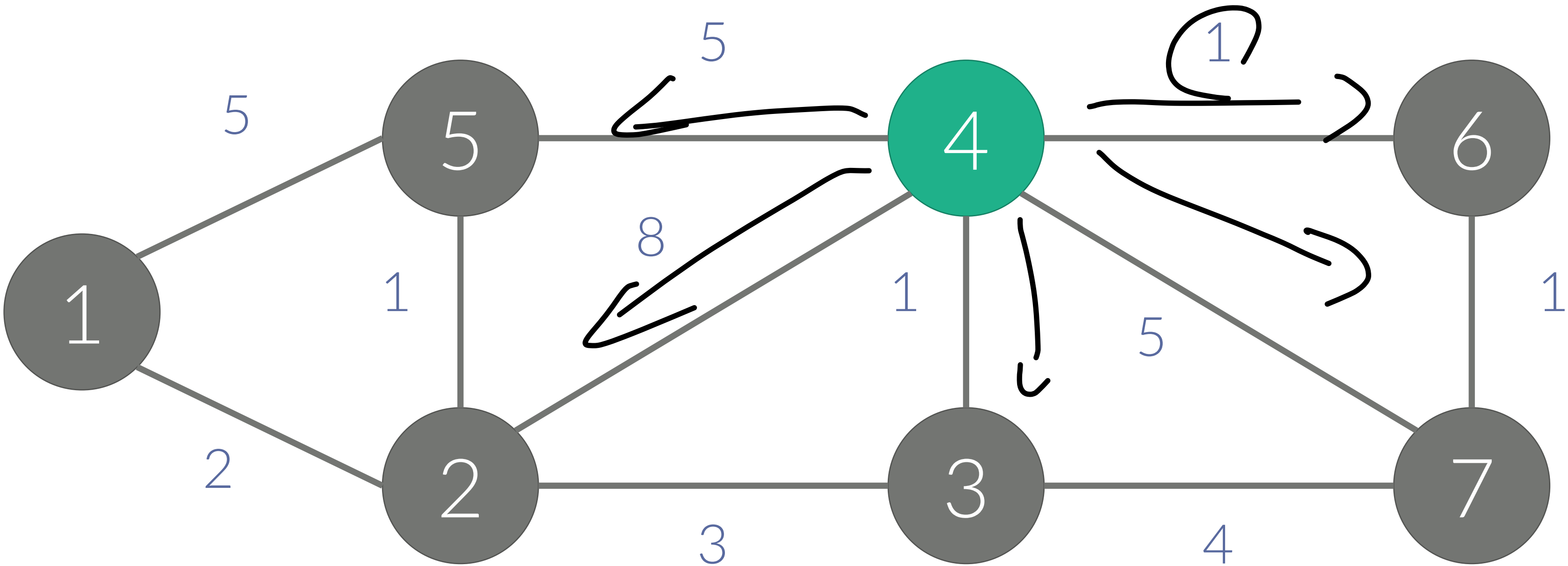


다익스트라

Dijkstra Algorithm

- 선택: 4

i	1	2	3	4	5	6	7
D[i]	0	2	5	6	3	∞	9
C[i]	1	1	1	1	1		

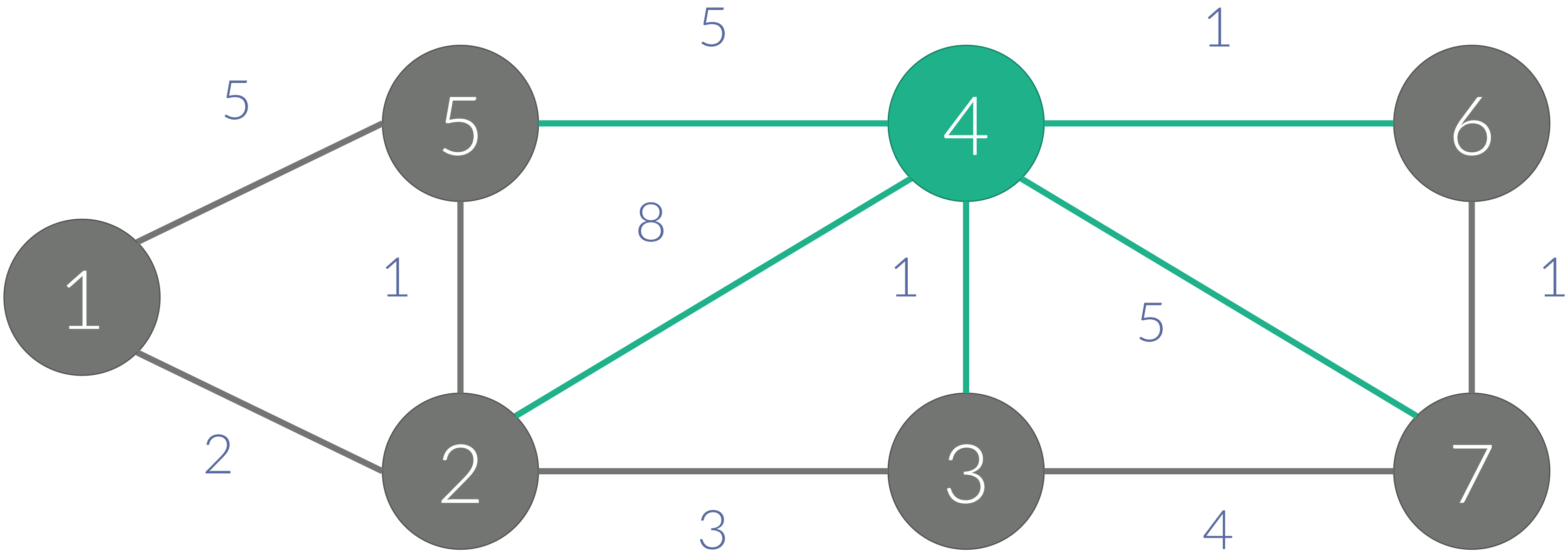


다익스트라

Dijkstra Algorithm

- 선택: 4

i	1	2	3	4	5	6	7
D[i]	0	2	5	6	3	7	9
C[i]	1	1	1	1	1		

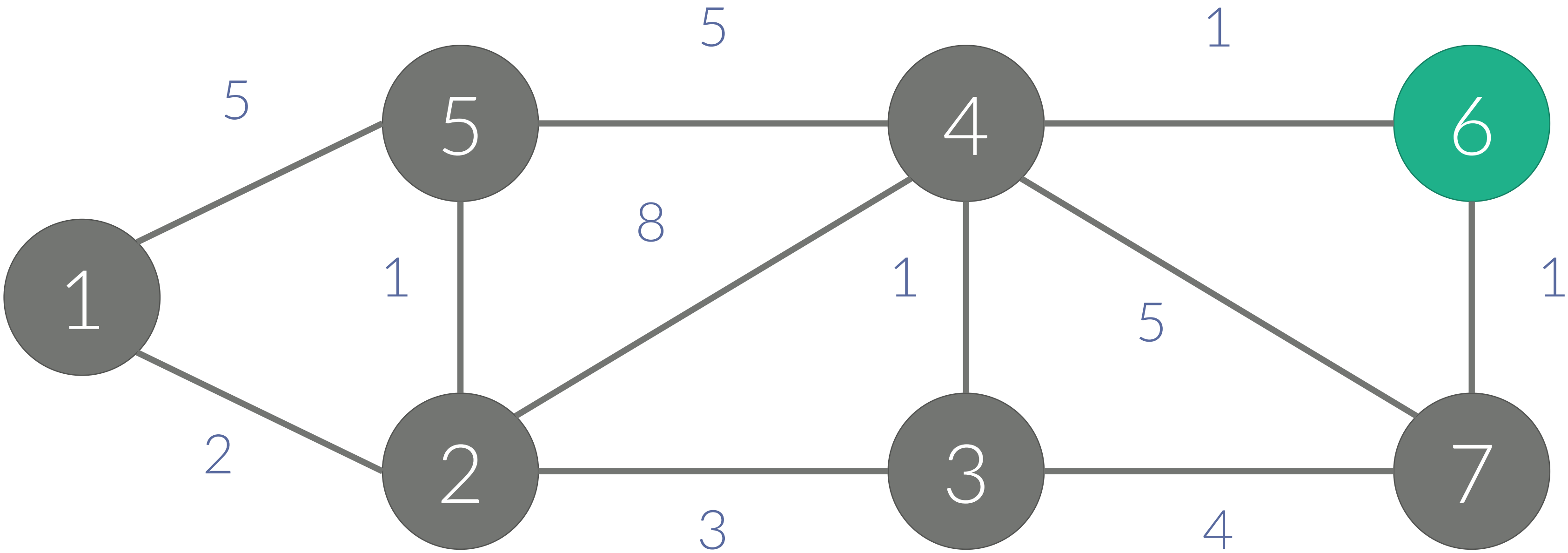


다익스트라

Dijkstra Algorithm

- 선택: 6

i	1	2	3	4	5	6	7
D[i]	0	2	5	6	3	7	8
C[i]	1	1	1	1	1	1	

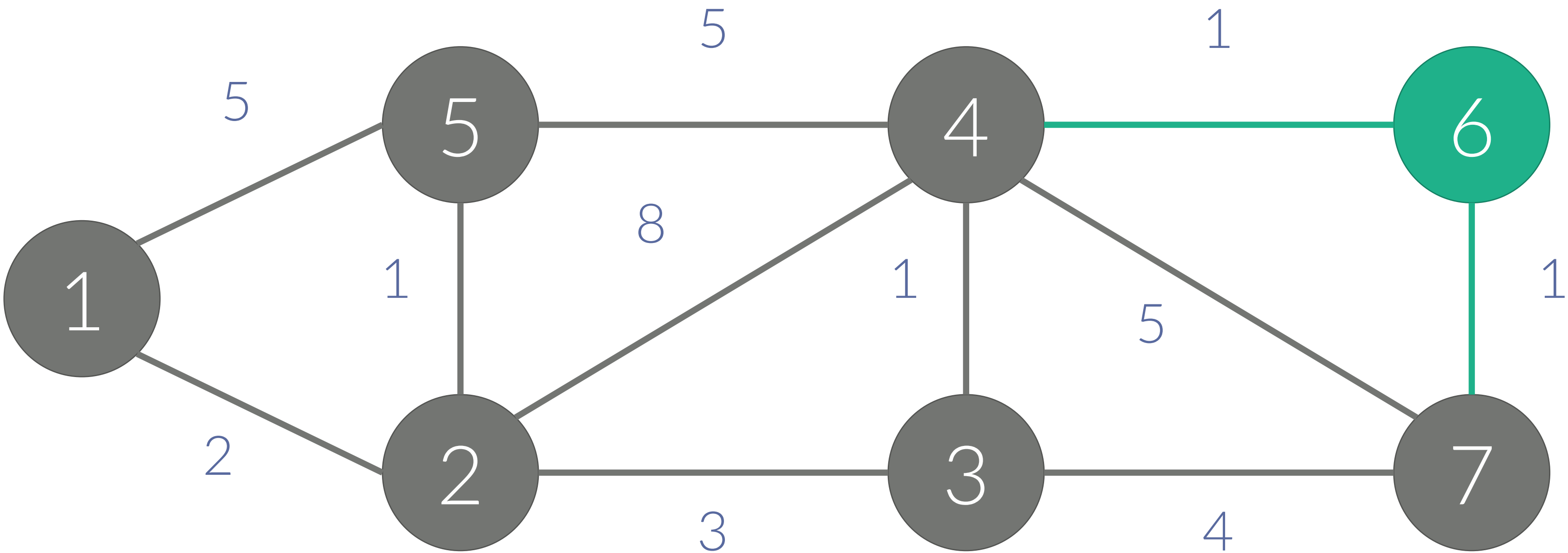


다익스트라

Dijkstra Algorithm

- 선택: 6

i	1	2	3	4	5	6	7
D[i]	0	2	5	6	3	7	8
C[i]	1	1	1	1	1	1	

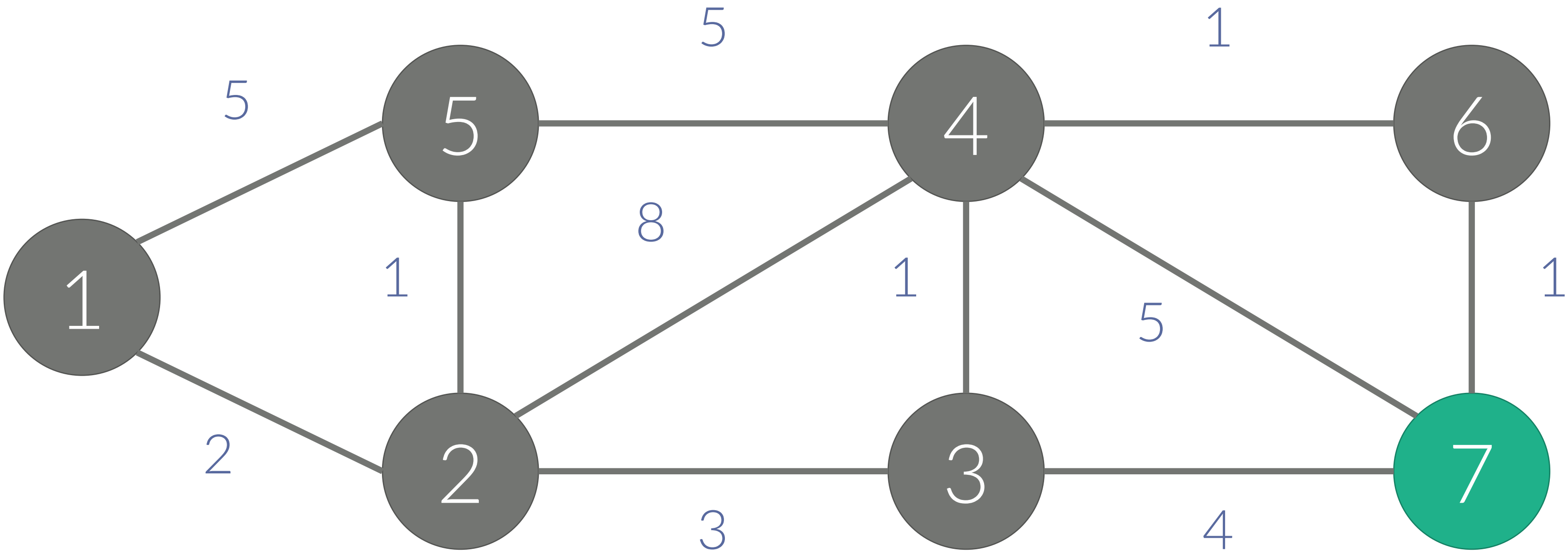


다익스트라

Dijkstra Algorithm

- 선택: 7

i	1	2	3	4	5	6	7
D[i]	0	2	5	6	3	7	8
C[i]	1	1	1	1	1	1	1



다익스트라

Dijkstra Algorithm

- 선택: 7

$O(V^2)$

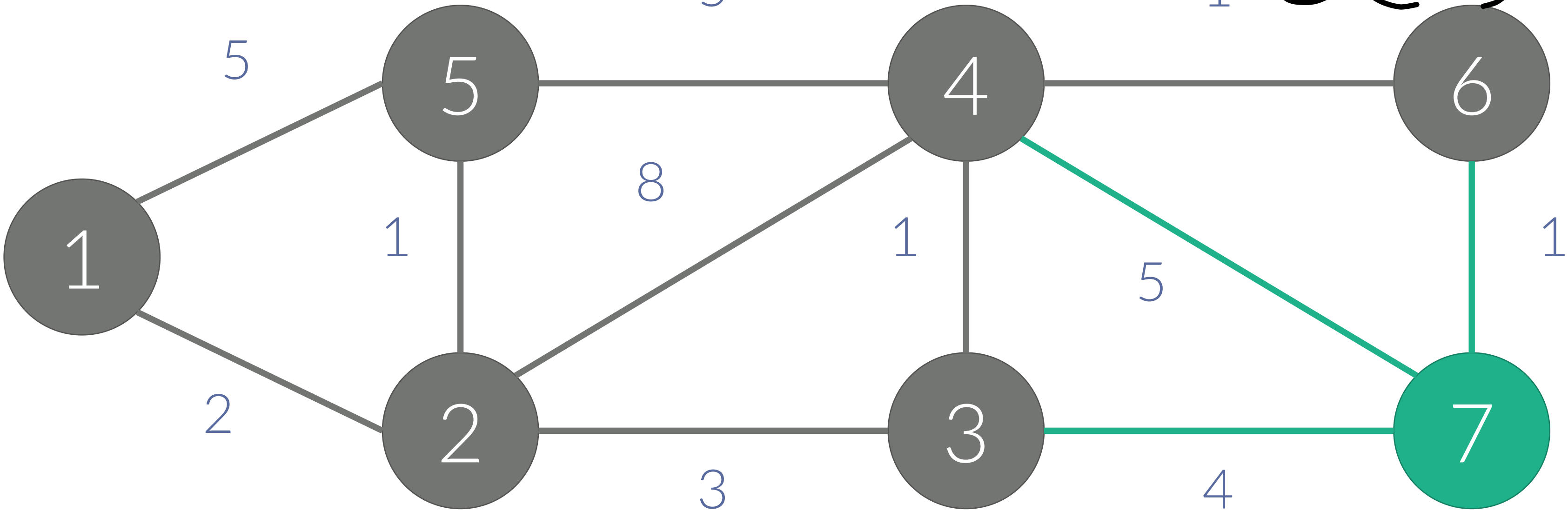
V

i	1	2	3	4	5	6	7
D[i]	0	2	5	6	3	7	8
C[i]	1	1	1	1	1	1	1

5

1

$O(V)$



선택/삭제

1E2

dist ↓ check

다익스트라

Dijkstra Algorithm

- 시간 복잡도
- 인접 행렬: $O(V^2)$
- 인접 리스트: $O(V^2)$

$E \log E$

최소비용 구하기

<https://www.acmicpc.net/problem/1916>

- A에서 B로 가는 최단 경로

최소비용 구하기

134

<https://www.acmicpc.net/problem/1916>

- 인접 행렬 소스: <http://codeplus.codes/61397d8211464aa68485384fd67ed82d>
- 인접 리스트 소스: <http://codeplus.codes/f965f0b9ac904208a7d0ee8dbf5252b4>

최소비용 구하기 2

135

<https://www.acmicpc.net/problem/11779>

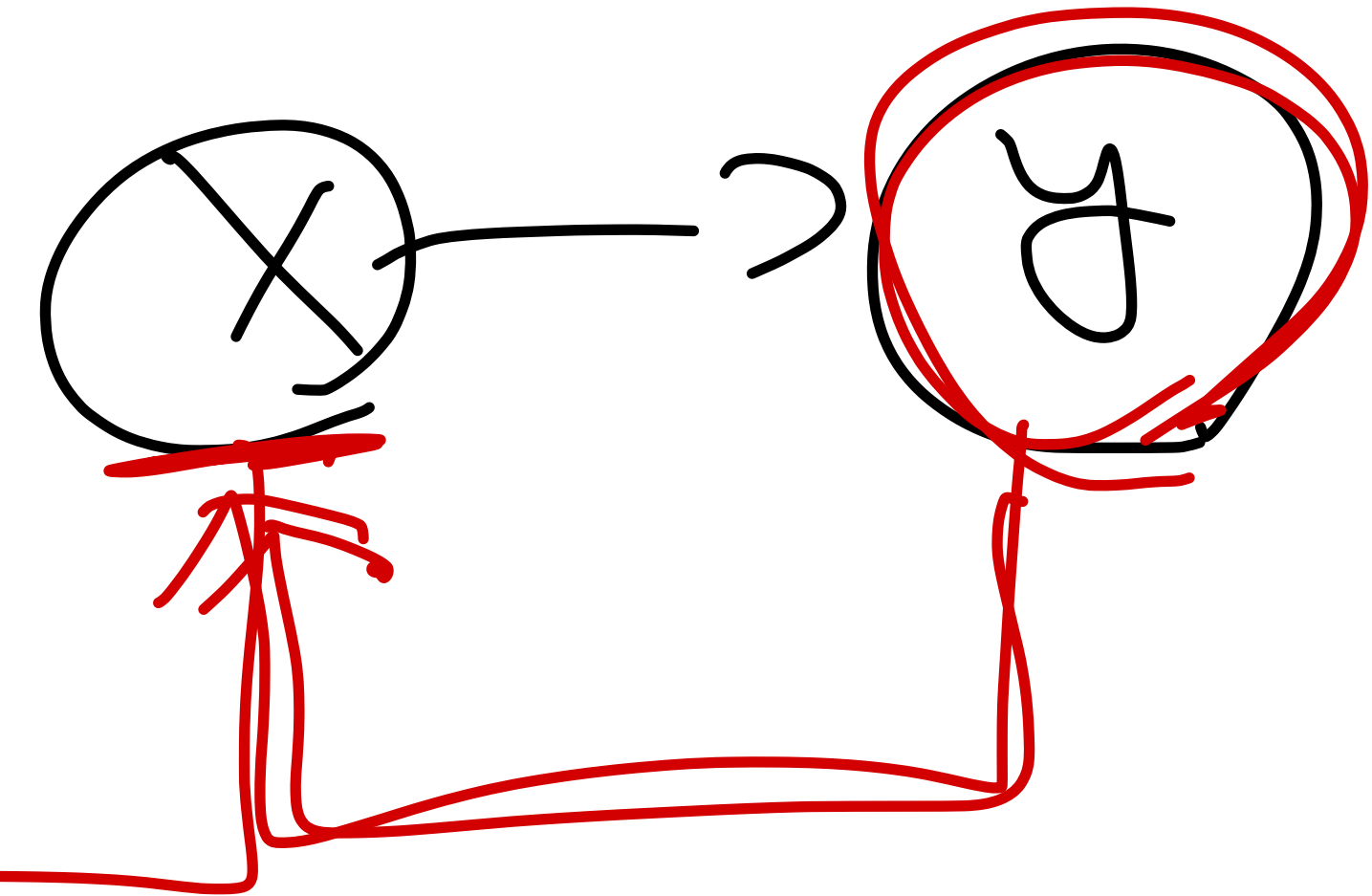
- A에서 B로 가는 최단 경로
- 최단 경로도 구해야 함

시작

- distance 값이 바뀔 때, 어디에서 왔는지를 저장해야 함

```
if (d[y] > d[x] + a[x][i].cost) {  
    d[y] = d[x] + a[x][i].cost;  
    v[y] = x;  
}
```

Dijkstra
BFS, DP



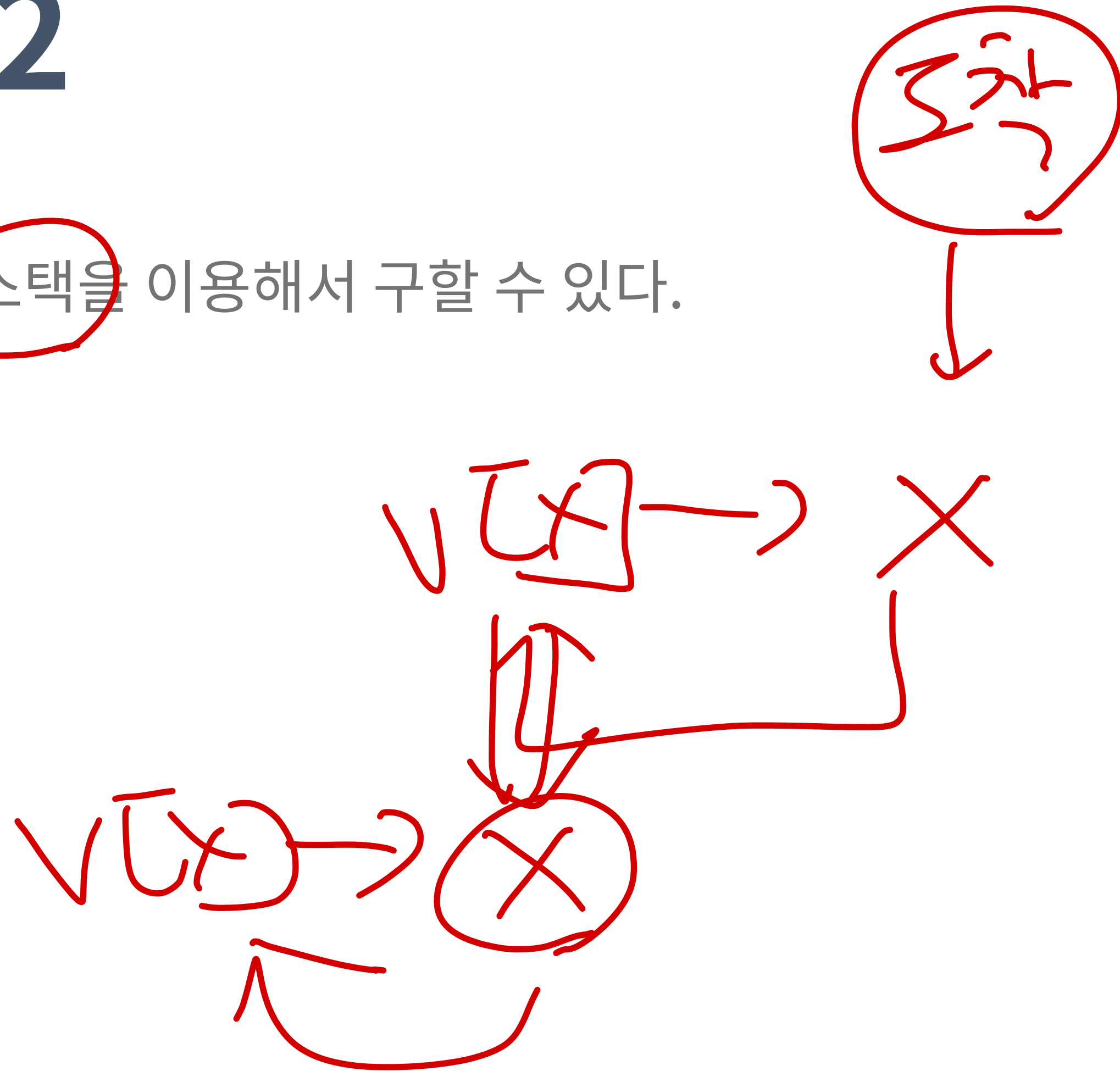
최소비용 구하기 2

136

<https://www.acmicpc.net/problem/11779>

- 다시 정답을 찾는 것은 재귀 호출이나 스택을 이용해서 구할 수 있다.
- 도착 -> 출발 과정을 거쳐서

```
stack<int> st;  
int x = end;  
while (x != -1) {  
    st.push(x);  
    x = v[x];  
}  
printf("%d\n", st.size());  
while (!st.empty()) {  
    printf("%d ", st.top());  
    st.pop();  
}
```



$\lambda_4(3)$

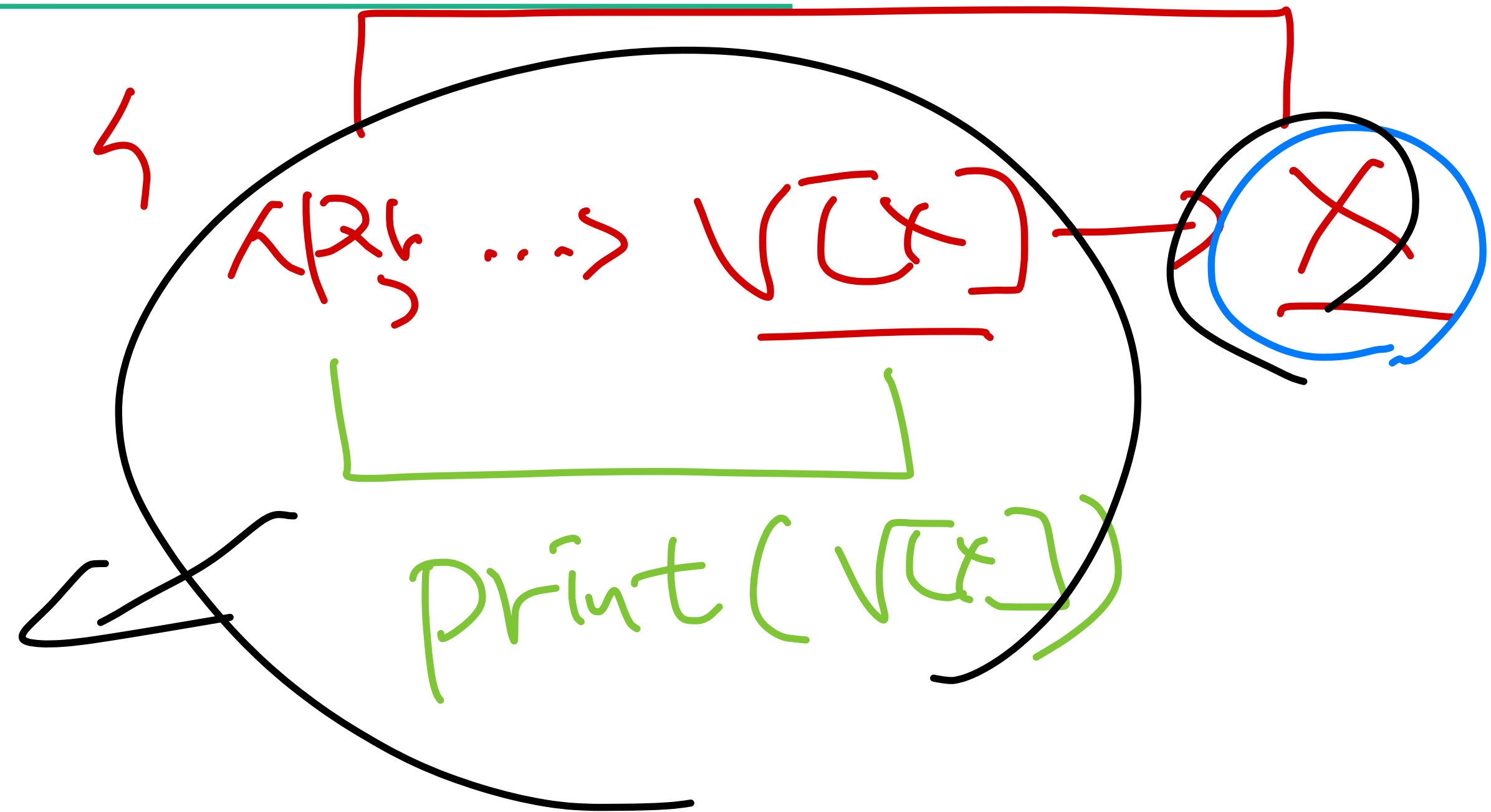
$Al_2S_3 \rightarrow X$ 정답

- 소스: <http://codeplus.codes/77ae143d3bde483ab280d3de8d4d8c1c>

$$\neg(f(\bigvee T(x)) \equiv \neg 1)$$

print(V[+])

Count $\ll x \ll \frac{1}{\epsilon}$



최단 경로

<https://www.acmicpc.net/problem/1753>

$$O(E \log E)$$

- 다익스트라를 이용해서 최단 경로를 구해야 하는데
- $1 \leq V \leq 20,000$ 이기 때문에
- $O(V^2)$ 이나 $O(VE)$ 는 시간이 너무 오래걸린다
- 인접행렬도 만들 수가 없다.
- 다익스트라에서 시간을 줄일 수 있는 부분은 어디일까?

다익스트라

Dijkstra Algorithm

139

1. 검사하지 않은 정점 중에서 dist의 값이 가장 작은 정점 v를 선택한다.

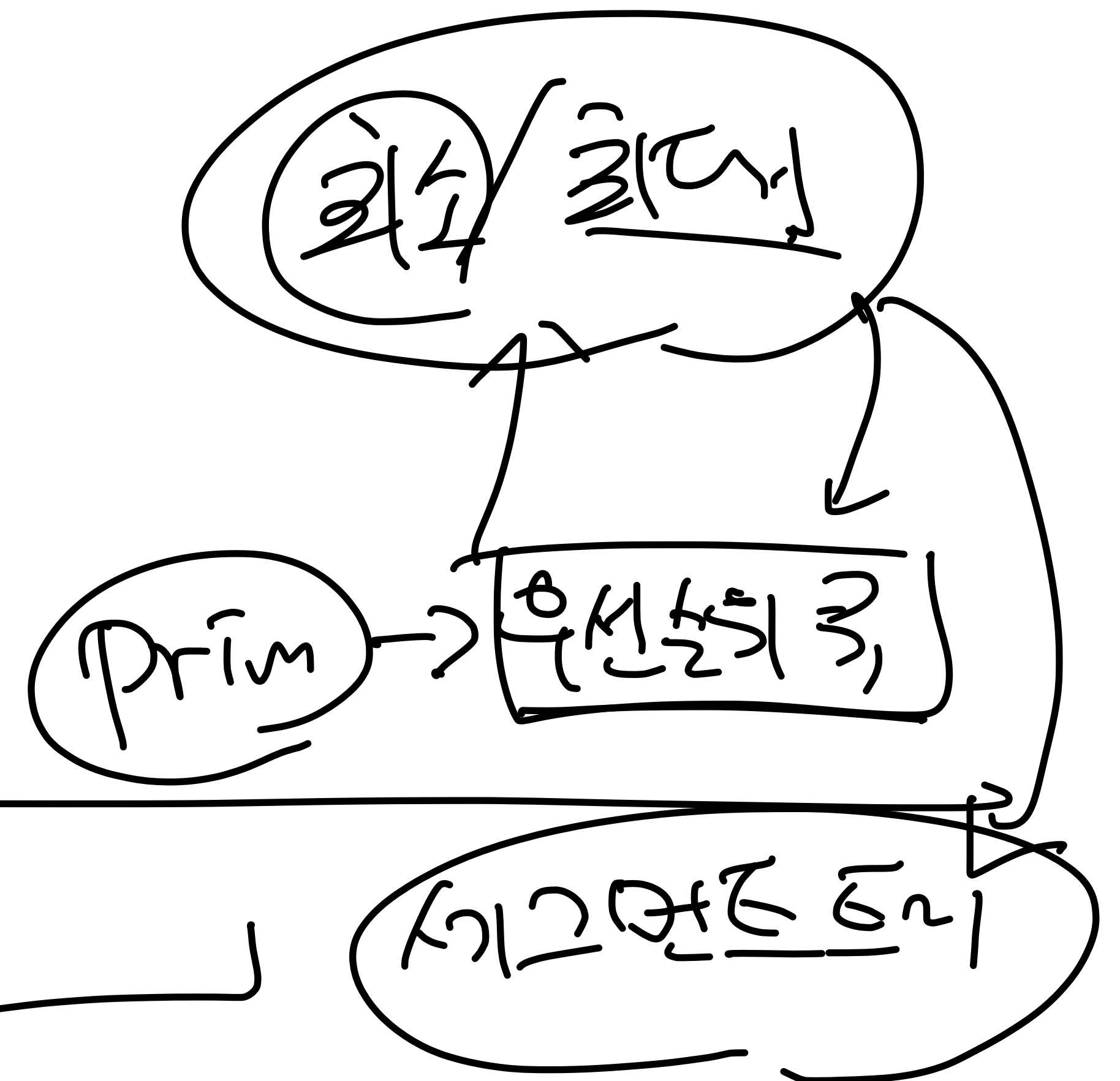
2. v와 연결된 모든 정점을 검사한다.

- 간선을 (from, to, cost)라고 했을 때
- $\text{dist}[\text{to}] > \text{dist}[\text{from}] + \text{cost}$ 이면 갱신해준다.

• 1, 2단계를 모든 정점을 검사할 때까지 계속한다.

최소

U-1번 U번



최단 경로

140

<https://www.acmicpc.net/problem/1753>

- 힙을 사용해서 최소값을 $O(\lg E)$ 만에 찾을 수 있다

최단 경로

<https://www.acmicpc.net/problem/1753>

- Heap을 이용한 소스: <http://codeplus.codes/264c3d4ee6574b3ea4fb8f8645294eda>
- Set을 이용한 소스: <http://codeplus.codes/053435a0b5434e6d960c2a1492faf5e4>

키/값 쌍이 1개

모든 쌍의 최단경로

xVHz

플로이드.

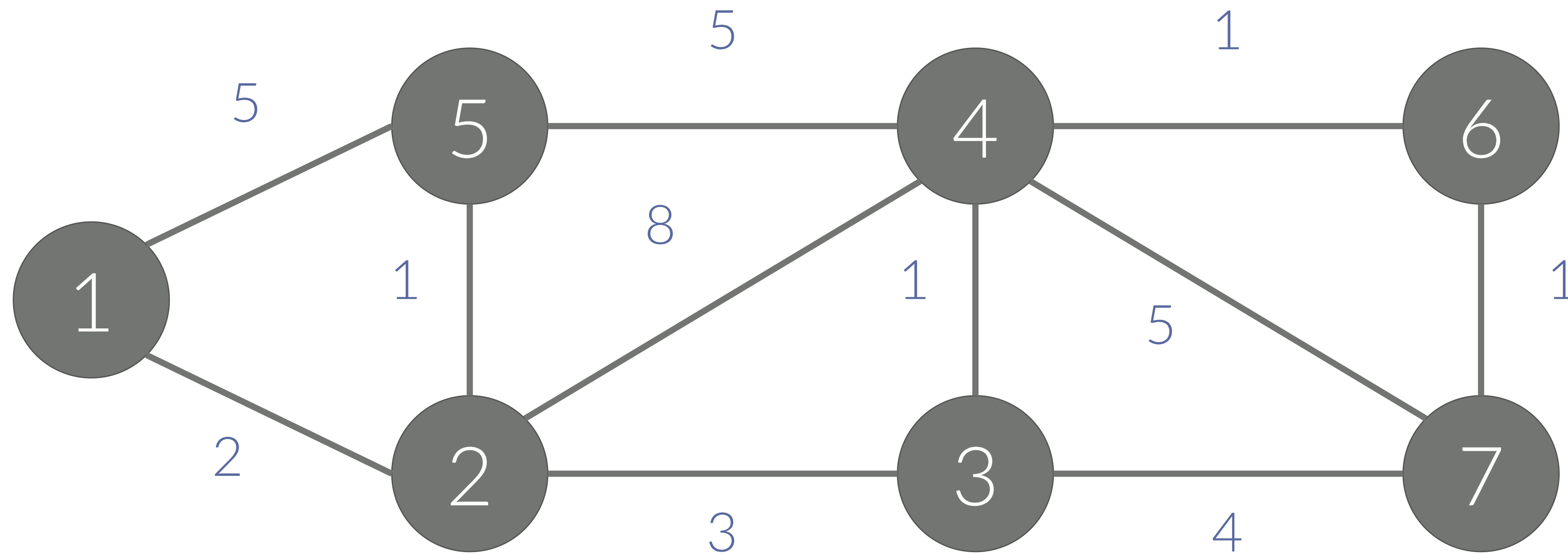
Dijkstra

플로이드

Floyd-Warshall Algorithm

143

- 모든 쌍의 최단 경로를 구하는 알고리즘



플로이드

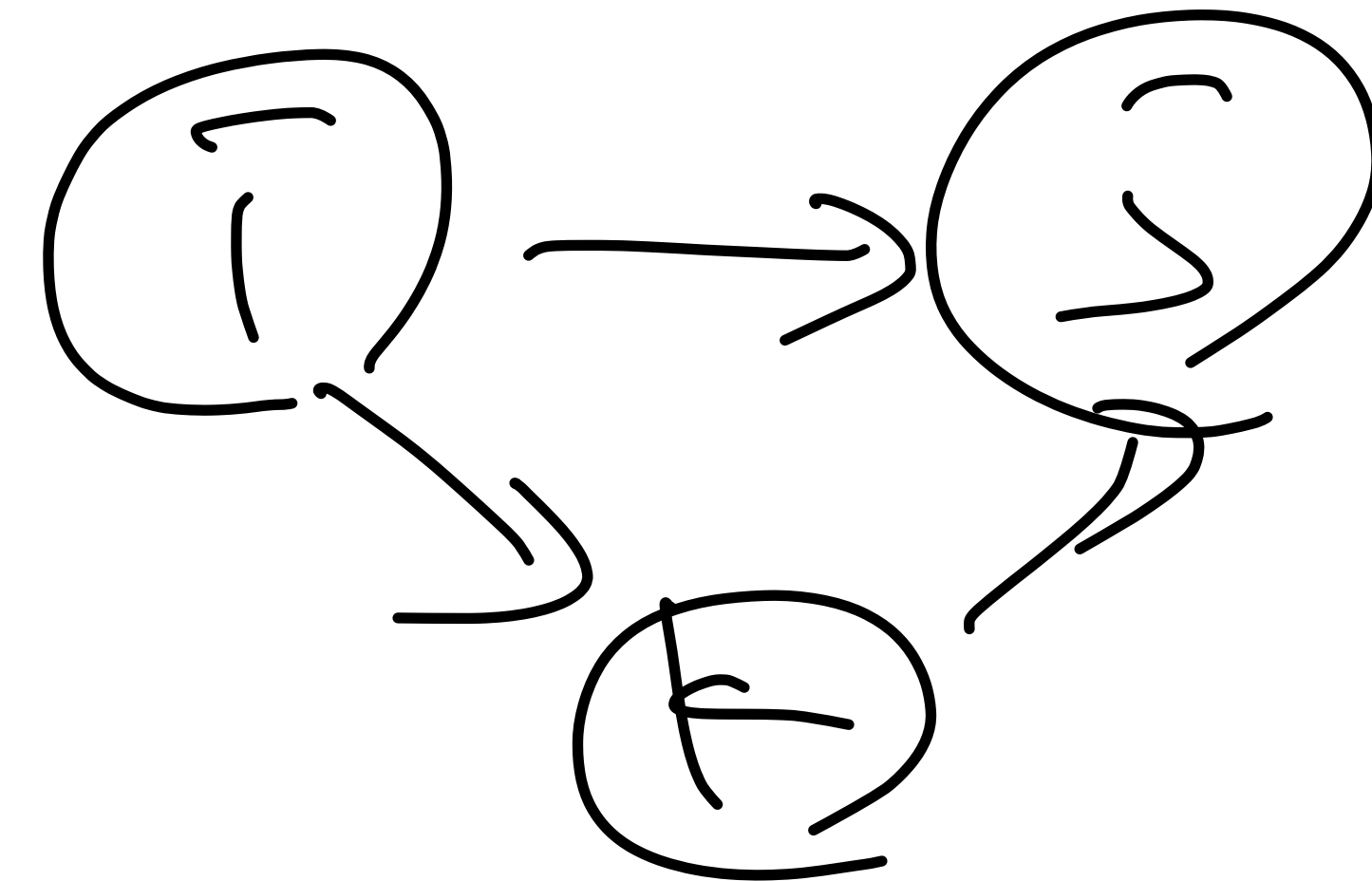
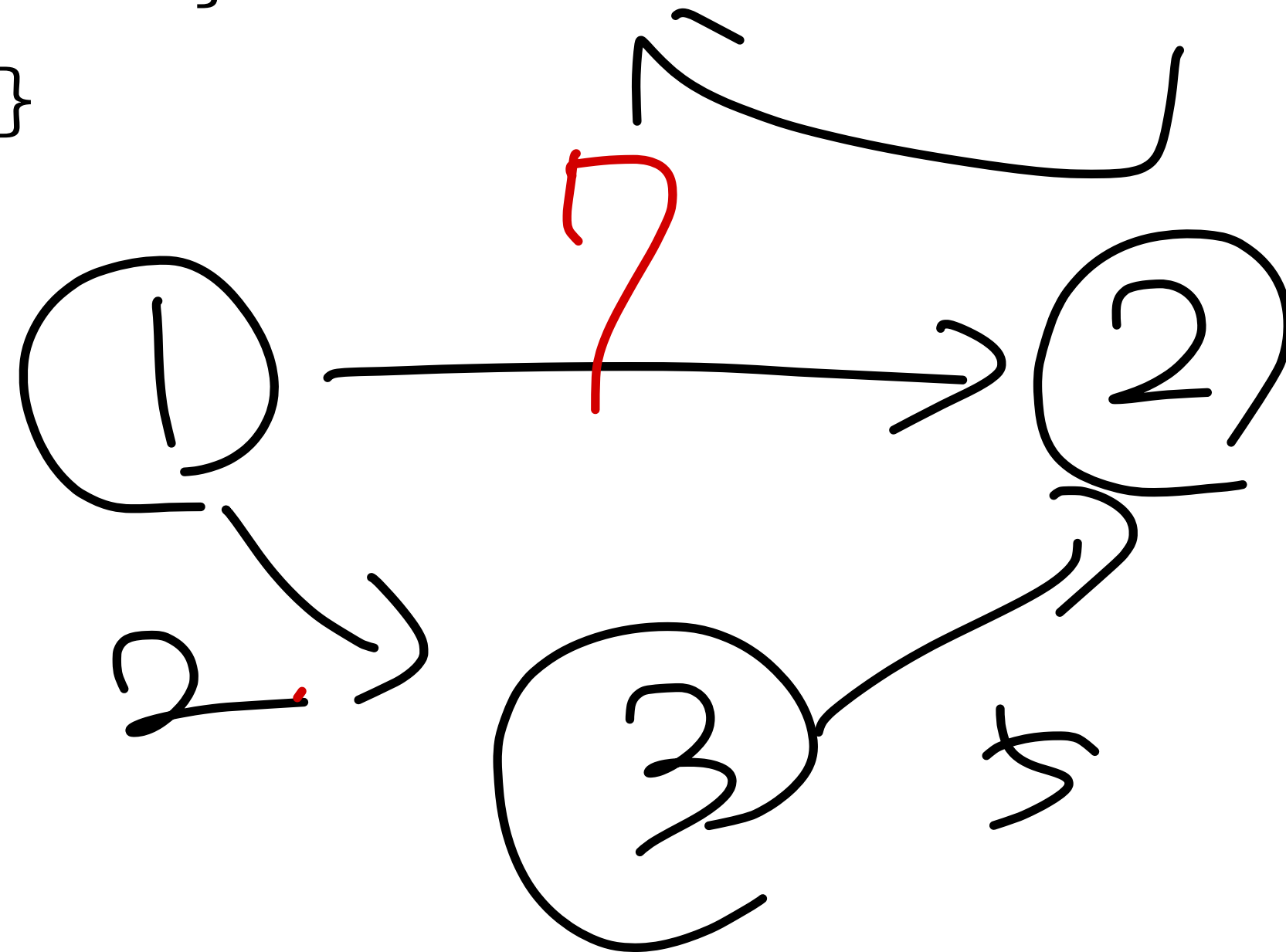
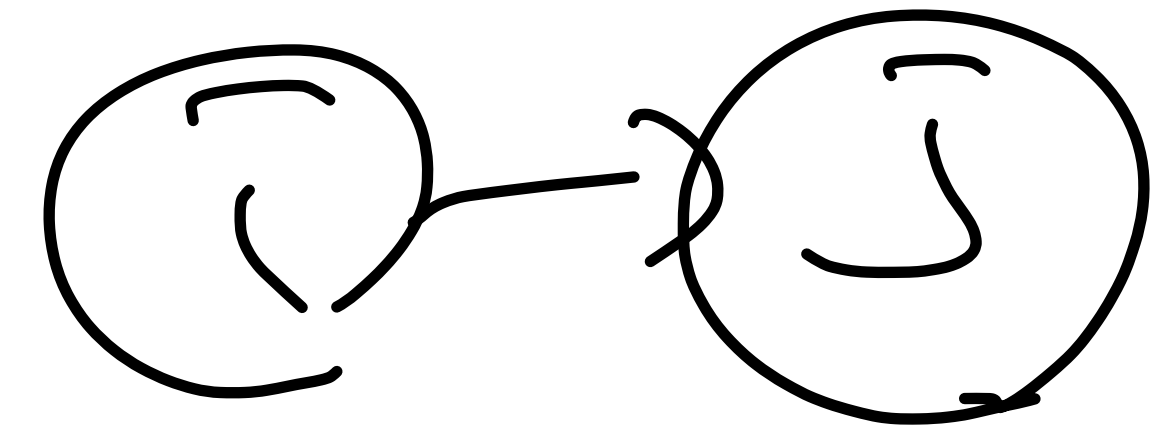
Floyd-Warshall Algorithm

$$O(V^3)$$

DP

144

```
for (int k=1; k<=n; k++) {  
    for (int i=1; i<=n; i++) {  
        for (int j=1; j<=n; j++) {  
            if (d[i][j] > d[i][k] + d[k][j]) {  
                d[i][j] = d[i][k] + d[k][j];  
            }  
        }  
    }  
}
```



플로이드

Floyd-Warshall Algorithm

- 1~N까지 정점이 있을 때
- $d[k][i][j]$ 를 다음과 같이 정의
 - $i \rightarrow j$ 로 이동하는 최단 경로
 - 이 때, 중간에 방문할 수 있는 정점은 $\{1, 2, \dots, k\}$

- 그럼 $d[k][i][j]$ 를 구해보자

- k가 경로에 없는 경우
- k가 경로에 있는 경우

$d[k][i][j]$

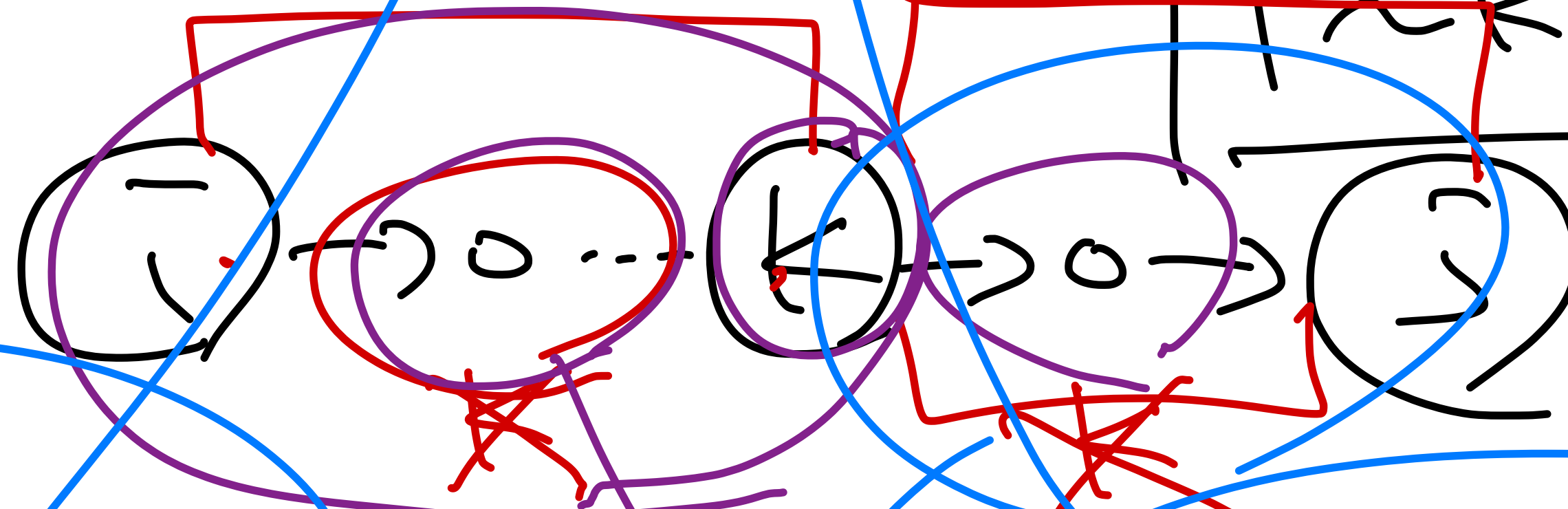
최단 경로

$1 \sim k$

i

경로

j



$d[k-1][i][j]$

$d[k-1][i][k] + d[k-1][k][j]$

플로이드

Floyd-Warshall Algorithm

- 1~N 까지 정점이 있을 때
- $d[k][i][j]$ 를 다음과 같이 정의
 - $i \rightarrow j$ 로 이동하는 최단 경로
 - 이 때, 중간에 방문할 수 있는 정점은 $\{1, 2, \dots, k\}$
- 그럼 $d[k][i][j]$ 를 구해보자
 - k가 경로에 없는 경우
 - $d[k-1][i][j]$
 - k가 경로에 있는 경우
 - $d[k-1][i][k] + d[k-1][k][j]$

플로이드

Floyd-Warshall Algorithm

147

- $d[k][i][j] = a[i][j]$ ($k == 0$)
- $\min(d[k-1][i][j], d[k-1][i][k] + d[k-1][k][j])$ ($k \geq 1$)

처음 문제

128542

문제 문제
문제 문제

~~$D[i][j]$~~ = [이 문제, 문제] 일 때
가장 최

$\rightarrow \max(\text{ ~~$D[i][j]$~~ , $D[i-1][j] - w[i]$ + $v[i]$)$

플로이드

Floyd-Warshall Algorithm

- 구현할 때는 2차원 배열로 구현하면 된다
- `for (int k=1; k<=n; k++) { d[k-1]을 이용해 d[k]를 구한다`

플로이드

Floyd-Warshall Algorithm

149

$O(V^3)$

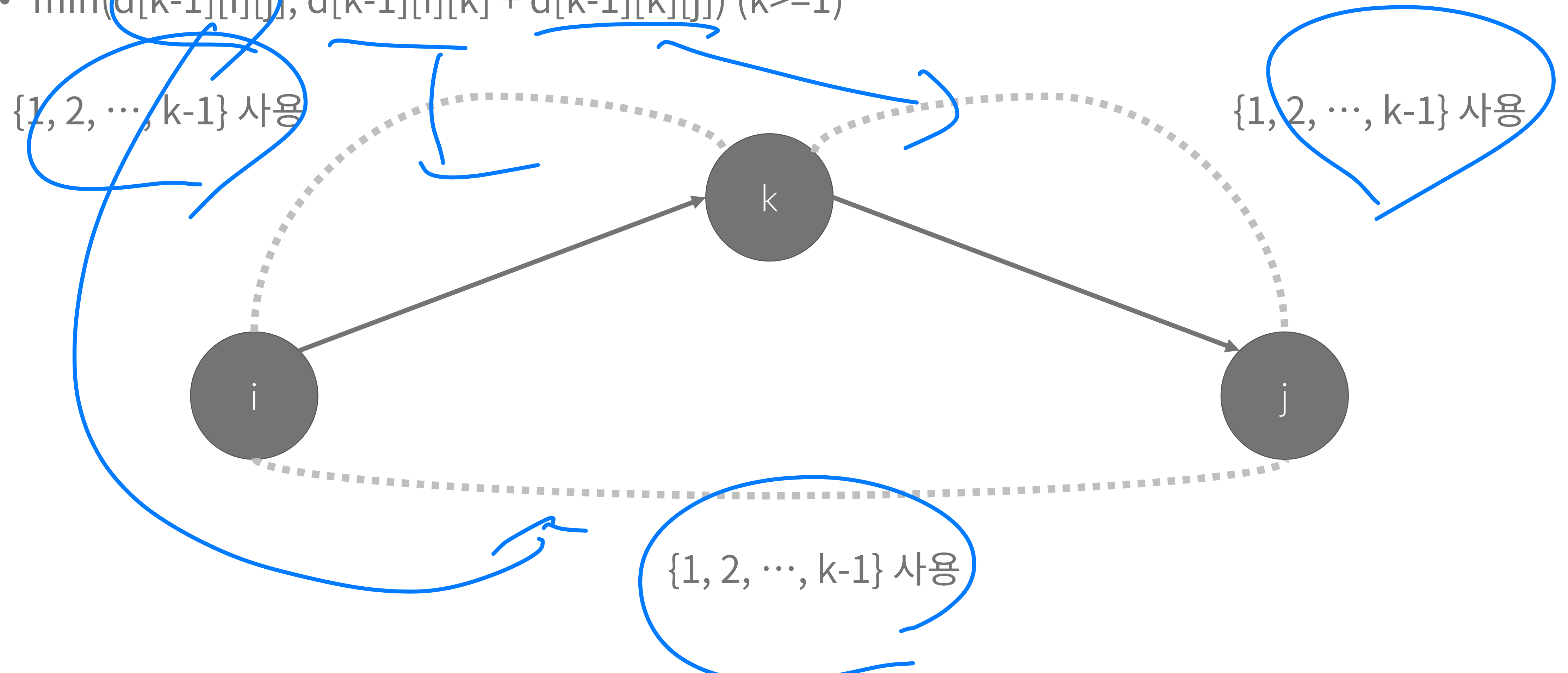
```
for (int k=1; k<=n; k++) {  
    for (int i=1; i<=n; i++) {  
        for (int j=1; j<=n; j++) {  
            if (d[i][j] > d[i][k] + d[k][j]) {  
                d[i][j] = d[i][k] + d[k][j];  
            }  
        }  
    }  
}
```

플로이드

150

Floyd-Warshall Algorithm

- $d[k][i][j] = a[i][j]$ ($k == 0$)
- $\min(d[k-1][i][j], d[k-1][i][k] + d[k-1][k][j])$ ($k \geq 1$)



경로 찾기

151

<https://www.acmicpc.net/problem/11403>

- 가중치 없는 방향 그래프 G 가 주어졌을 때, 모든 정점 (i, j) 에 대해서, i 에서 j 로 가는 경로가 있는지 없는지 구하는 프로그램을 작성하시오.

경로 찾기

152

<https://www.acmicpc.net/problem/11403>

- 소스: <http://codeplus.codes/cf911cf77c414280827c6cdcdeb71f90>

플로이드

153

<https://www.acmicpc.net/problem/11404>

- $n(1 \leq n \leq 100)$ 개의 도시가 있다
- 한 도시에서 출발하여 다른 도시에 도착하는 $m(1 \leq m \leq 100,000)$ 개의 버스가 있다
- 각 버스는 한 번 사용할 때 필요한 비용이 있다
- 모든 도시의 쌍 (A, B) 에 대해서 도시 A에서 B로 가는데 필요한 비용의 최소값을 구하는 프로그램을 작성하시오

플로이드

154

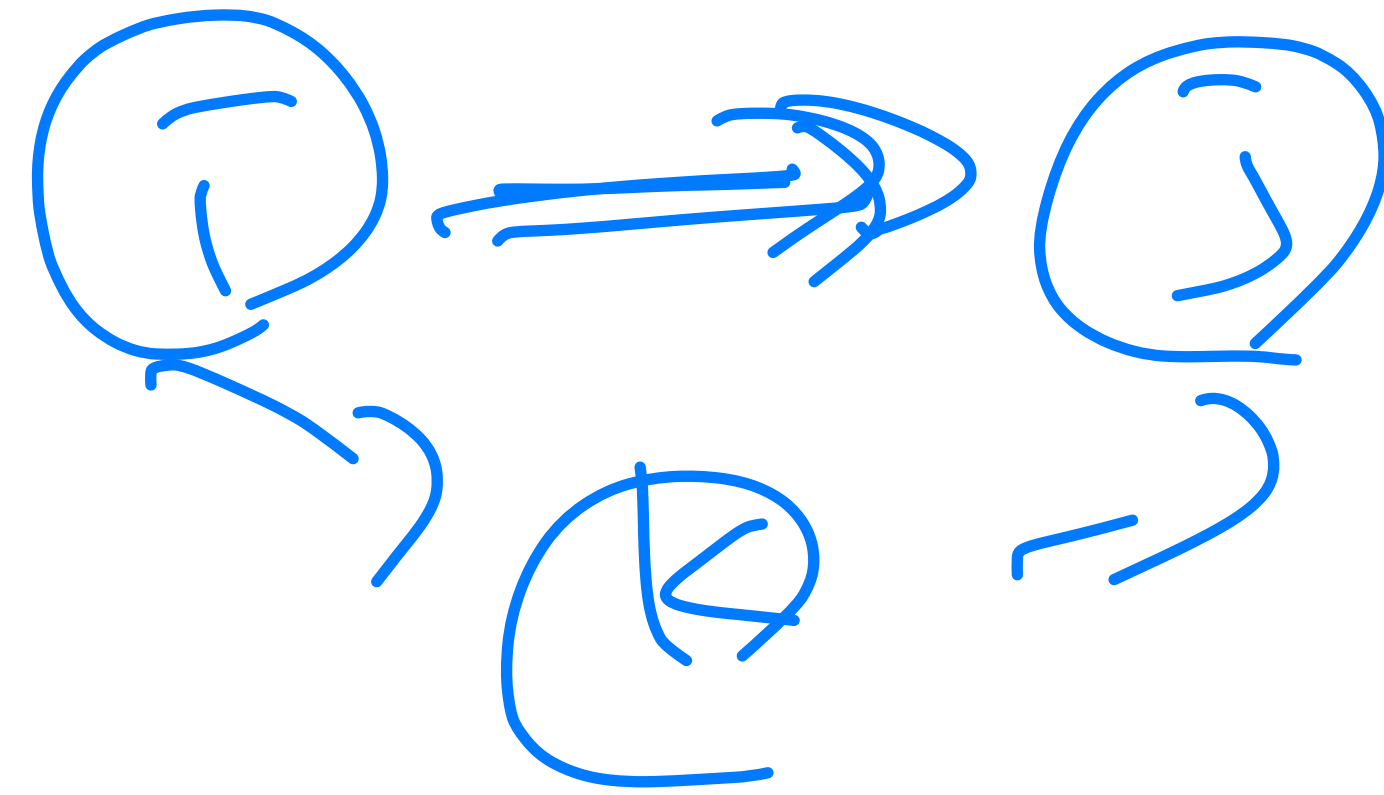
<https://www.acmicpc.net/problem/11404>

- 소스: <http://codeplus.codes/ac2ab8f030b145b69655fffc5f54bd87>

플로이드 2

<https://www.acmicpc.net/problem/11780>

- 플로이드 구현을 보면
- $i \rightarrow k, k \rightarrow j$ 로 가는 간선을 $i \rightarrow j$ 로 바꿔줬기 때문에
- $i \rightarrow j$ 이 원래 어떤 정점을 가르키고 있었는지를 알아야 한다.
- 구현을 조금 변경해야 한다.

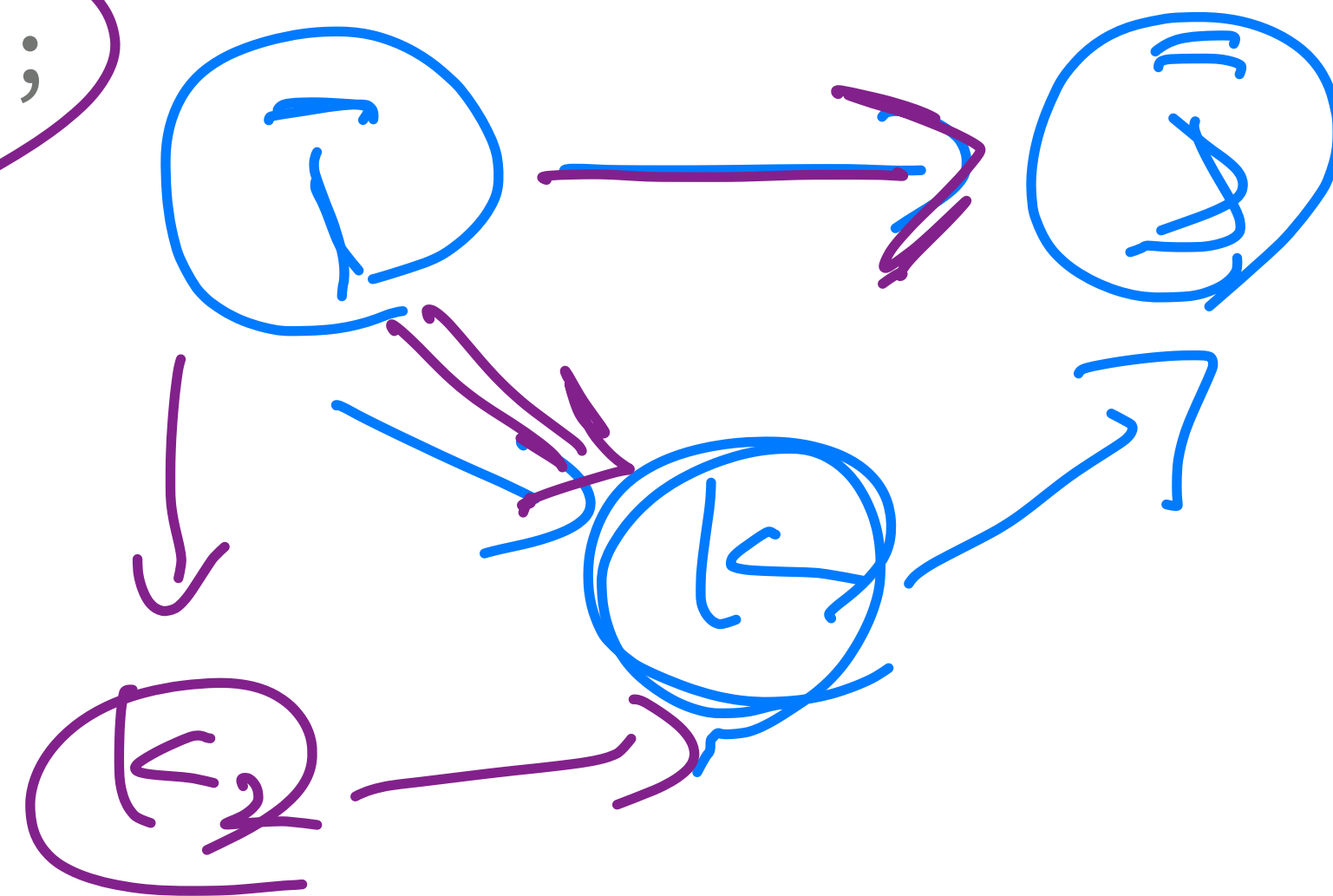
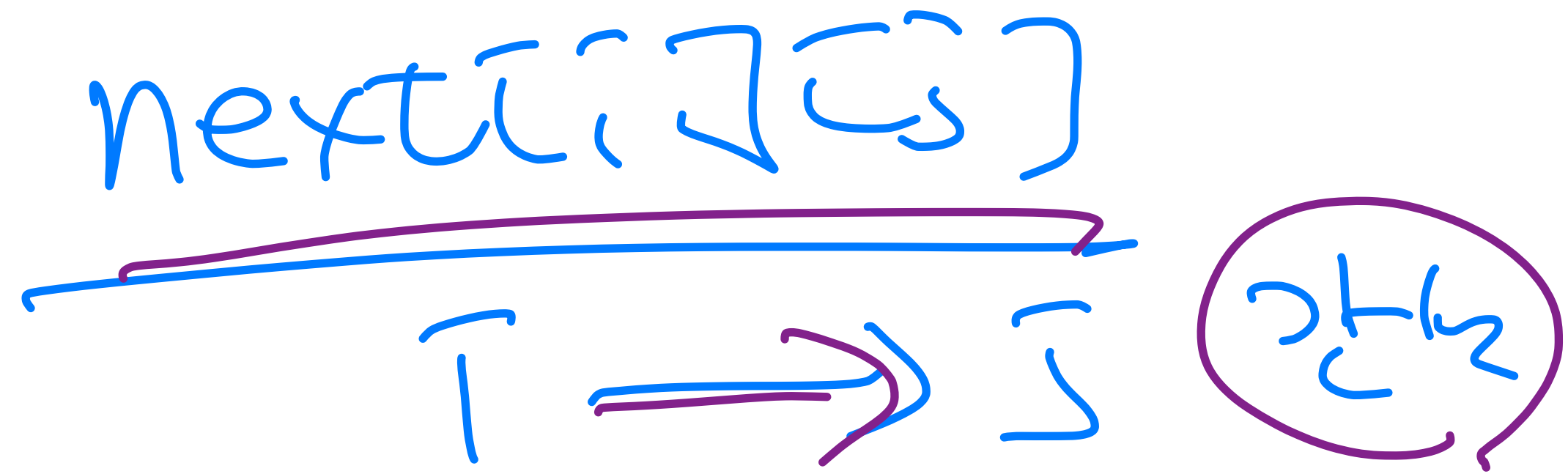


플로이드 2

<https://www.acmicpc.net/problem/11780>

156

```
for (int k=1; k<=n; k++) {  
    for (int i=1; i<=n; i++) {  
        for (int j=1; j<=n; j++) {  
            if (a[i][j] > a[i][k] + a[k][j]) {  
                a[i][j] = a[i][k] + a[k][j];  
                next[i][j] = next[i][k];  
            }  
        }  
    }  
}
```



플로이드 2

157

<https://www.acmicpc.net/problem/11780>

- 소스: <http://codeplus.codes/4afd8d9e27eb44ebb8efa1000eab2beb>

케빈 베이컨의 6단계 법칙

158

<https://www.acmicpc.net/problem/1389>

- 플로이드 알고리즘을 이용해 모든 쌍의 최단 경로를 구한 다음에 구할 수 있다

케빈 베이컨의 6단계 법칙

159

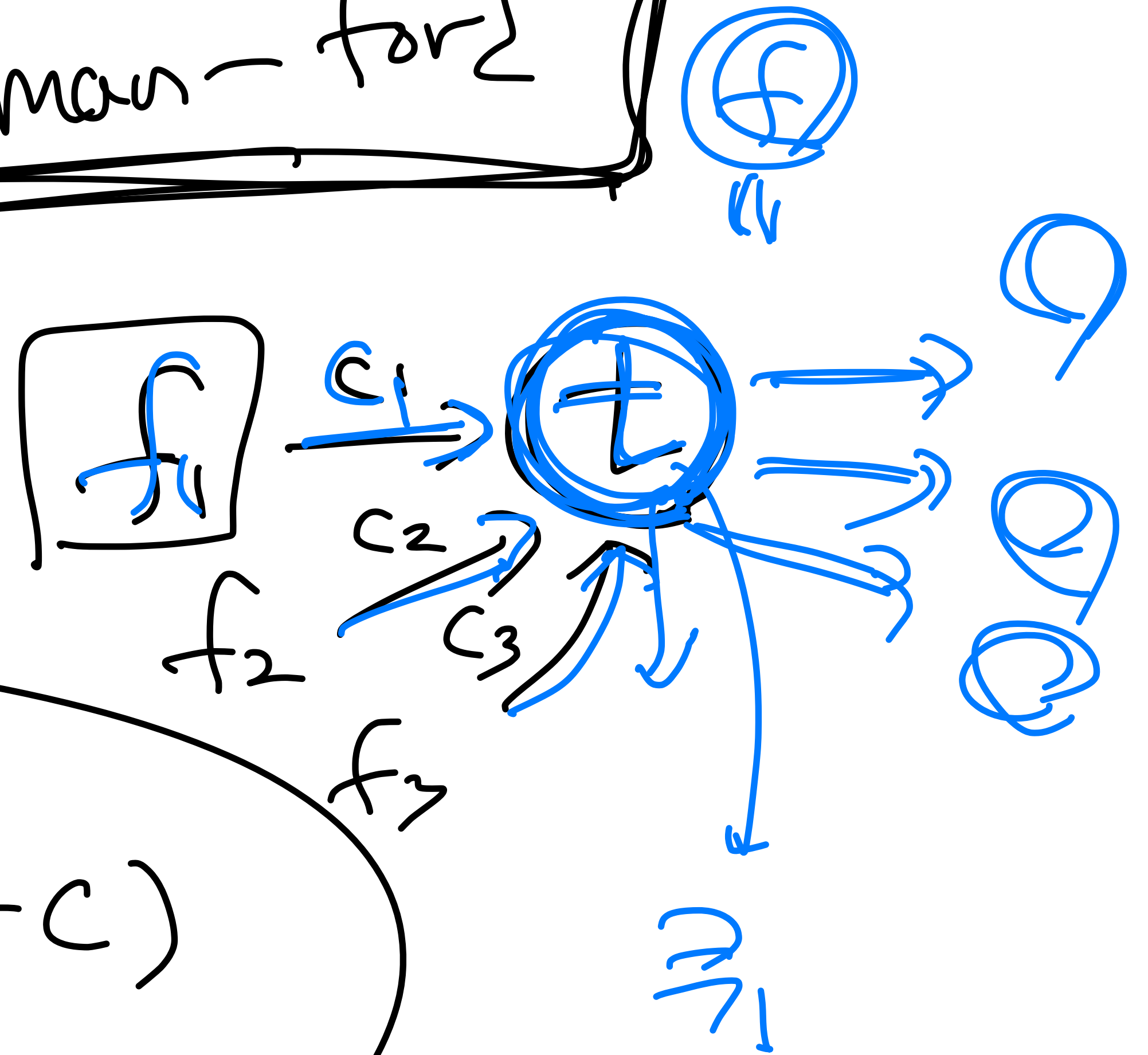
<https://www.acmicpc.net/problem/1389>

- 소스: <http://codeplus.codes/6b311efaffc646b38d9021a215d10640>


dist \Rightarrow 2

Bellman-Ford

SPFA



$$\begin{aligned} f(\sqrt{t}) &= \sqrt{f} + c \\ \sqrt{t} &= \sqrt{f} + c \end{aligned}$$



SPFA

Shorteest Path Faster Algorithm

- 벨만포드의 성능을 향상시킨 알고리즘
- 최악의 경우에는 벨만 포드의 시간복잡도와 같지만 평균적으로 $O(E)$ 이다
- 벨만포드의 아이디어와 같은 아이디어이다

```
for (int j = 0; j < n; j++) {  
    for (int k = 0; k < m; k++) {  
        int from = edges[k].from;  
        int to = edges[k].to;  
        int cost = edges[k].cost;  
        if (distance[to] > distance[from] + cost) {  
            distance[to] = distance[from] + cost;  
        }  
    }  
}
```

SPFA

Shorteest Path Faster Algorithm

- 벨만포드는 모든 간선에 대해서 업데이트를 진행하고
- SPFA는 아래 if문에 의해서 바뀐 정점과 연결된 간선에 대해서만 업데이트를 진행한다.
- 따라서, 인접 리스트의 구현이 필요하다.

```
for (int j = 0; j < n; j++) {  
    for (int k = 0; k < m; k++) {  
        int from = edges[k].from;  
        int to = edges[k].to;  
        int cost = edges[k].cost;  
        if (distance[to] > distance[from] + cost) {  
            distance[to] = distance[from] + cost;  
        }  
    }  
}
```

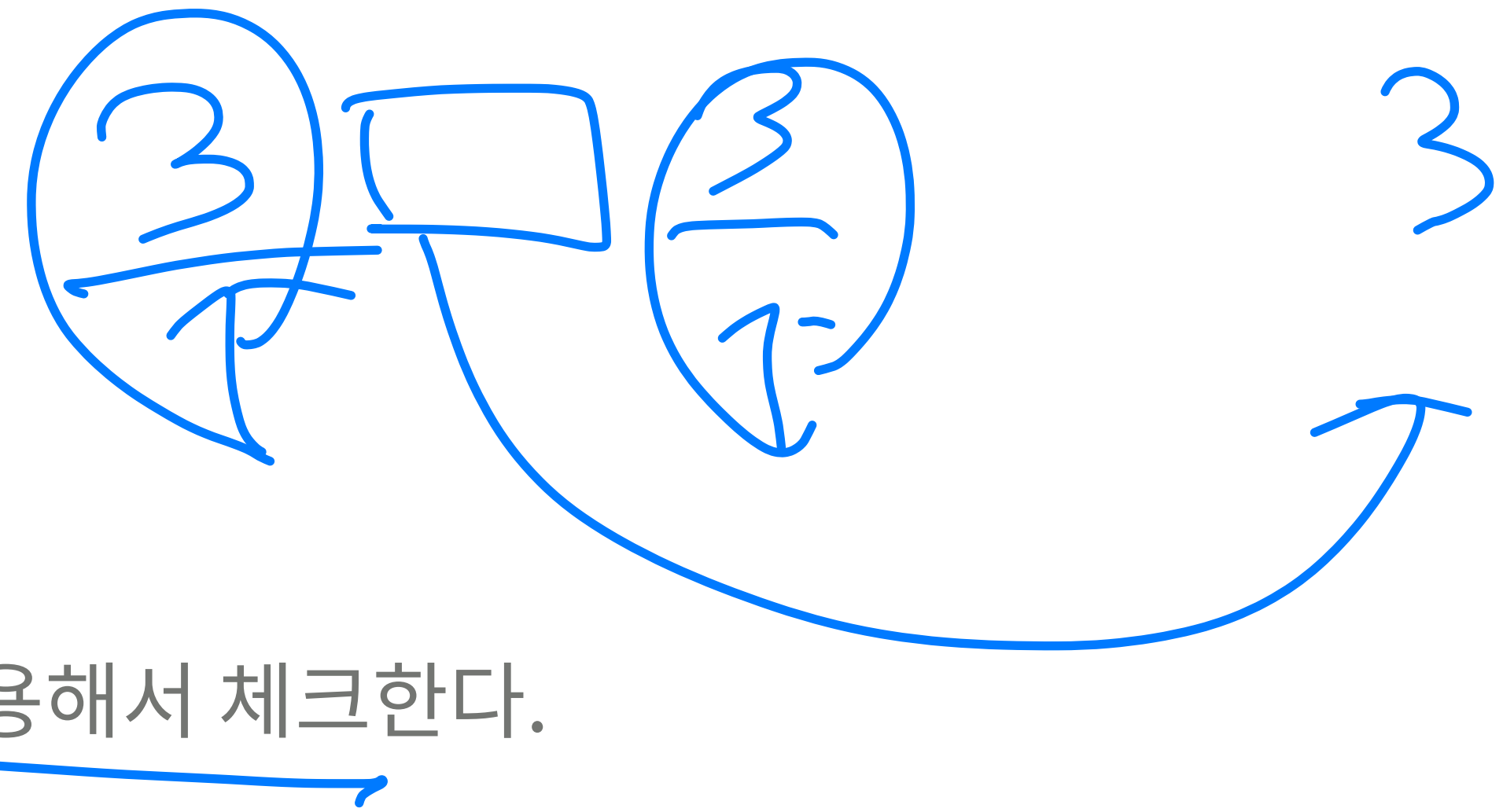
SPFA

Shorteest Path Faster Algorithm

- 바뀐 정점은 큐를 이용해서 관리하고
- 큐에 들어가있는지, 안 들어가있는지를 배열을 이용해서 체크한다.
- 초기화를 하고, 큐에 시작점을 넣어주고

```
for (int i=1; i<=n; i++) {  
    d[i] = inf;  
}
```

```
d[1] = 0;  
queue<int> q;  
q.push(1);  
c[1] = true;
```



SPFA

Shortest Path Faster Algorithm

Dijkstra

0/5/7

164

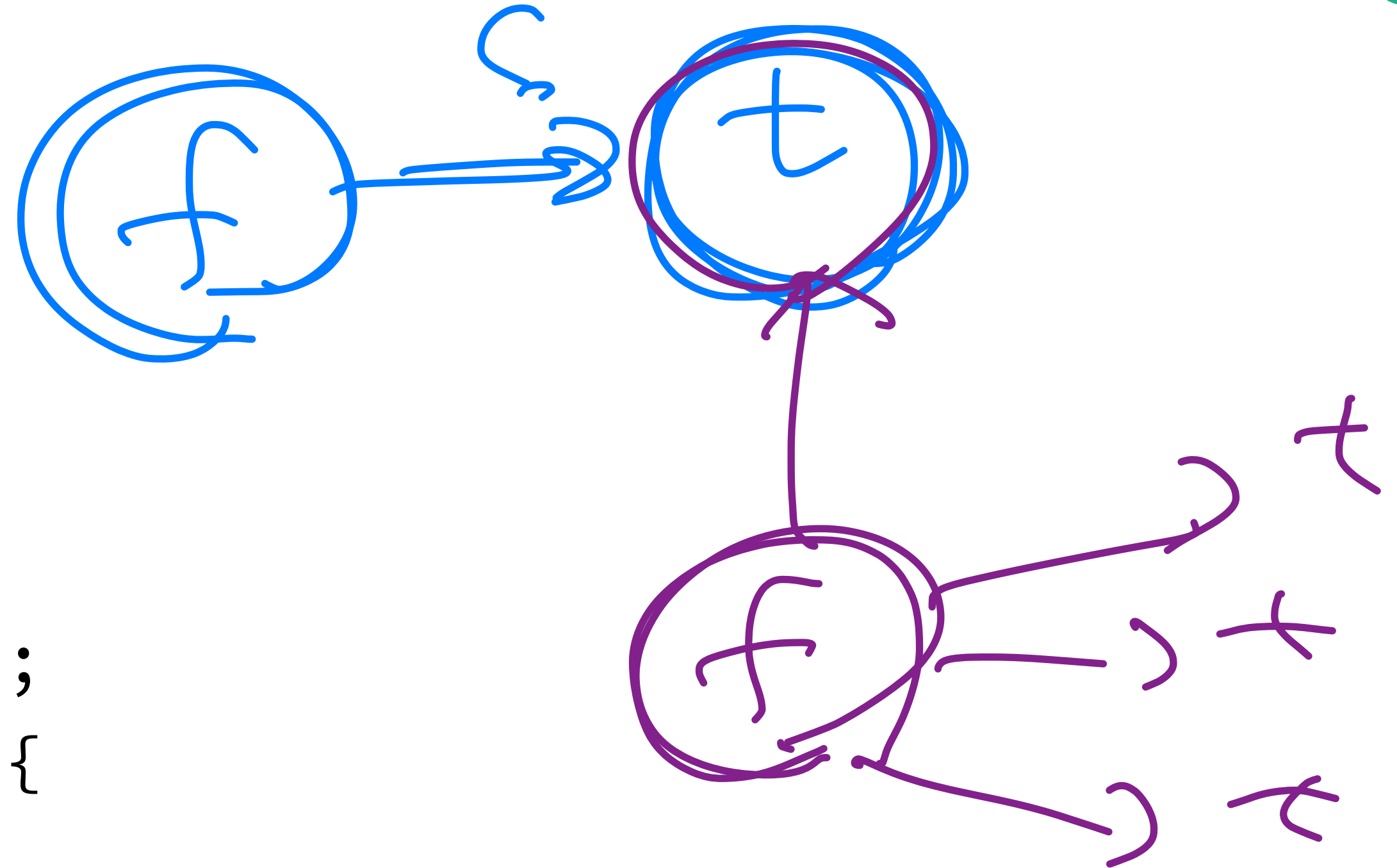
```
while (!q.empty()) {  
    int from = q.front();  
    c[from] = false; q.pop();  
    for (Edge &e : a[from]) {  
        int to = e.to, cost = e.cost;  
        if (d[to] > d[from] + cost) {  
            d[to] = d[from] + cost;  
            if (c[to] == false) {  
                q.push(to);  
                c[to] = true;  
            }  
        }  
    }  
}
```

0/5/7

MC4F

① $\xrightarrow{-5}$ ②

Worst



~~0/5/7~~
0/5/7

$O(E)$

0/5/7

타임머신

165

<https://www.acmicpc.net/problem/11657>

- 소스: <http://codeplus.codes/075358f07aab4922a64e5e3598aac0f8>