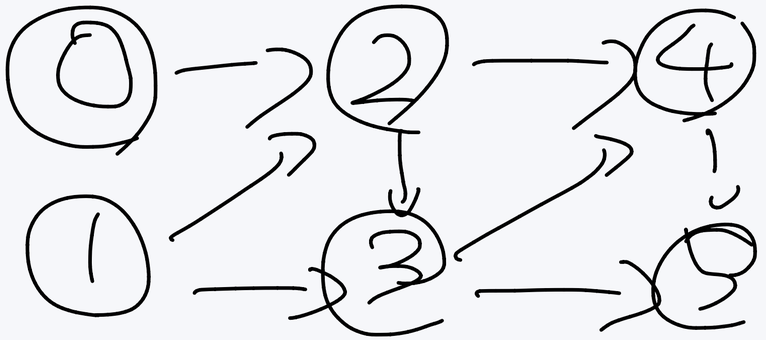
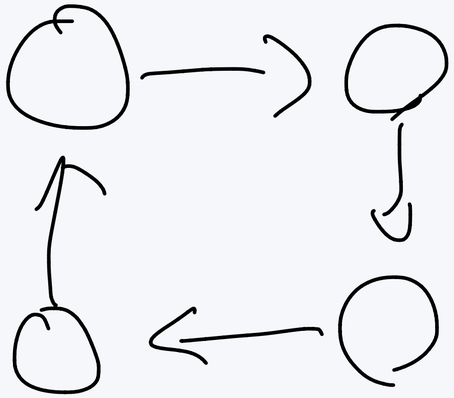
그래프 2

최백준 choi@startlink.io

DAG

$$D[i] = D[i-1] + D[i-2]$$

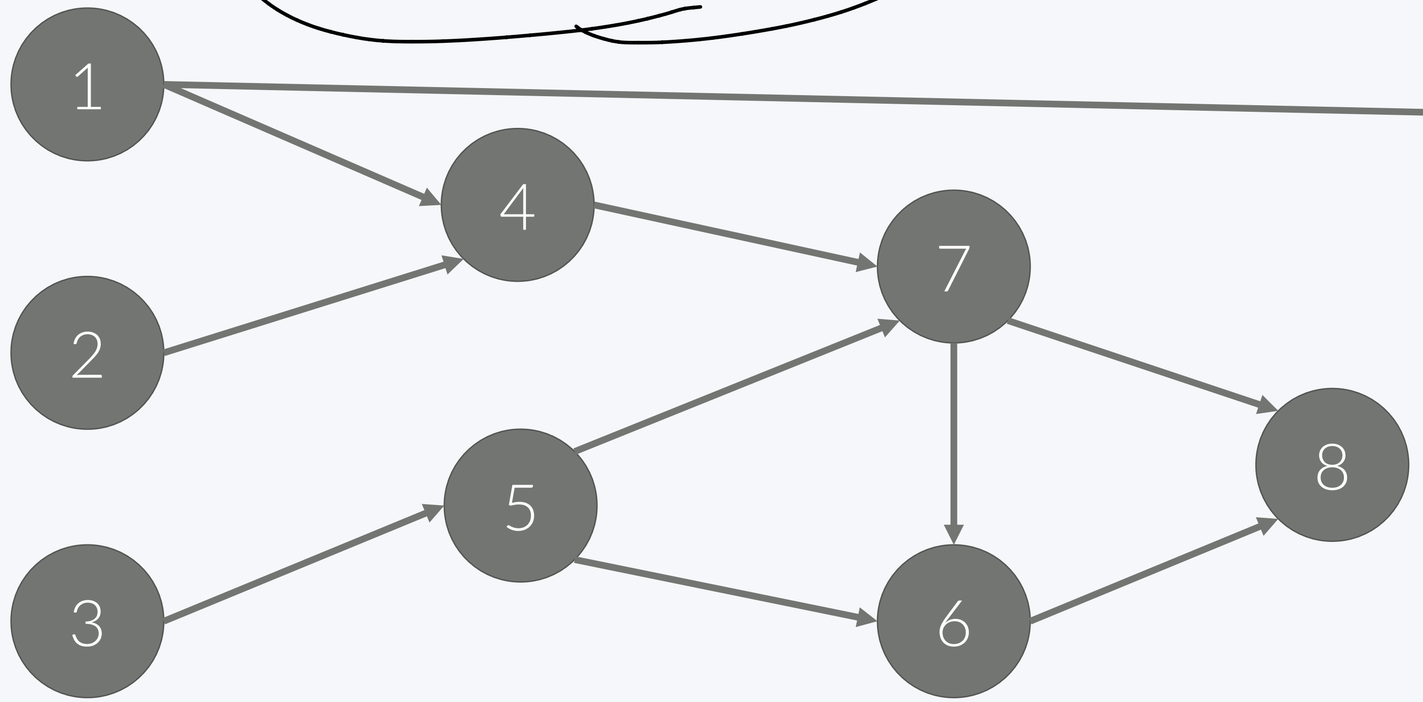


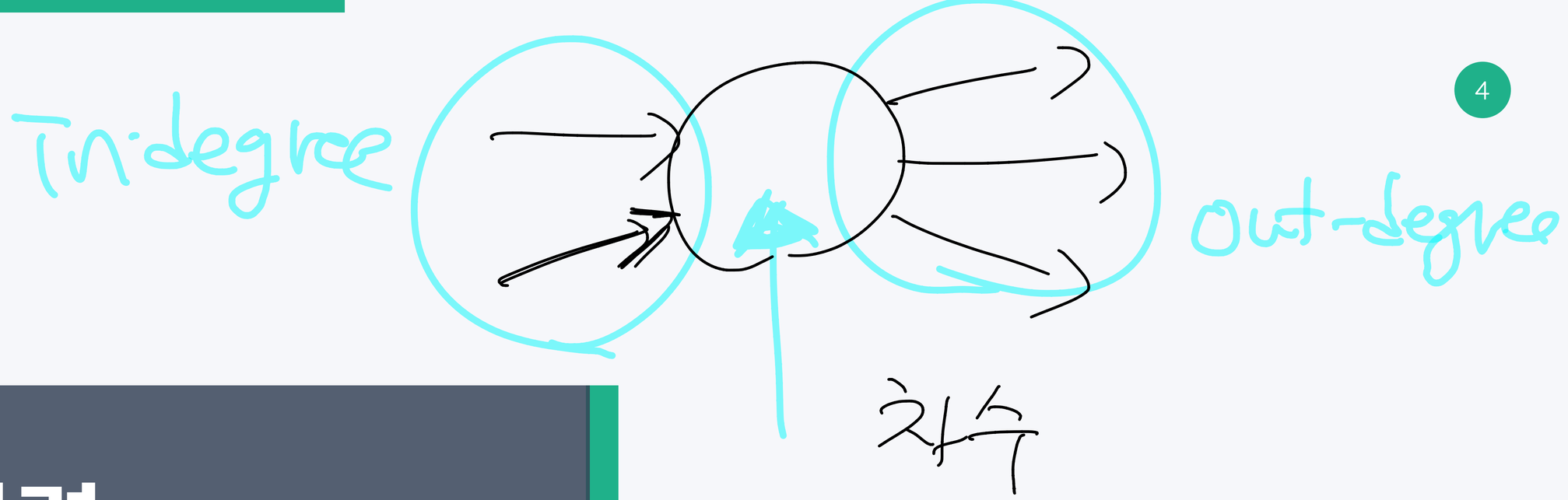


DAG

Directed Acyclic Graph

- 사이클이 없는 방향 있는 그래프를 DAG라고 한다.





위상 정렬

위상 정렬

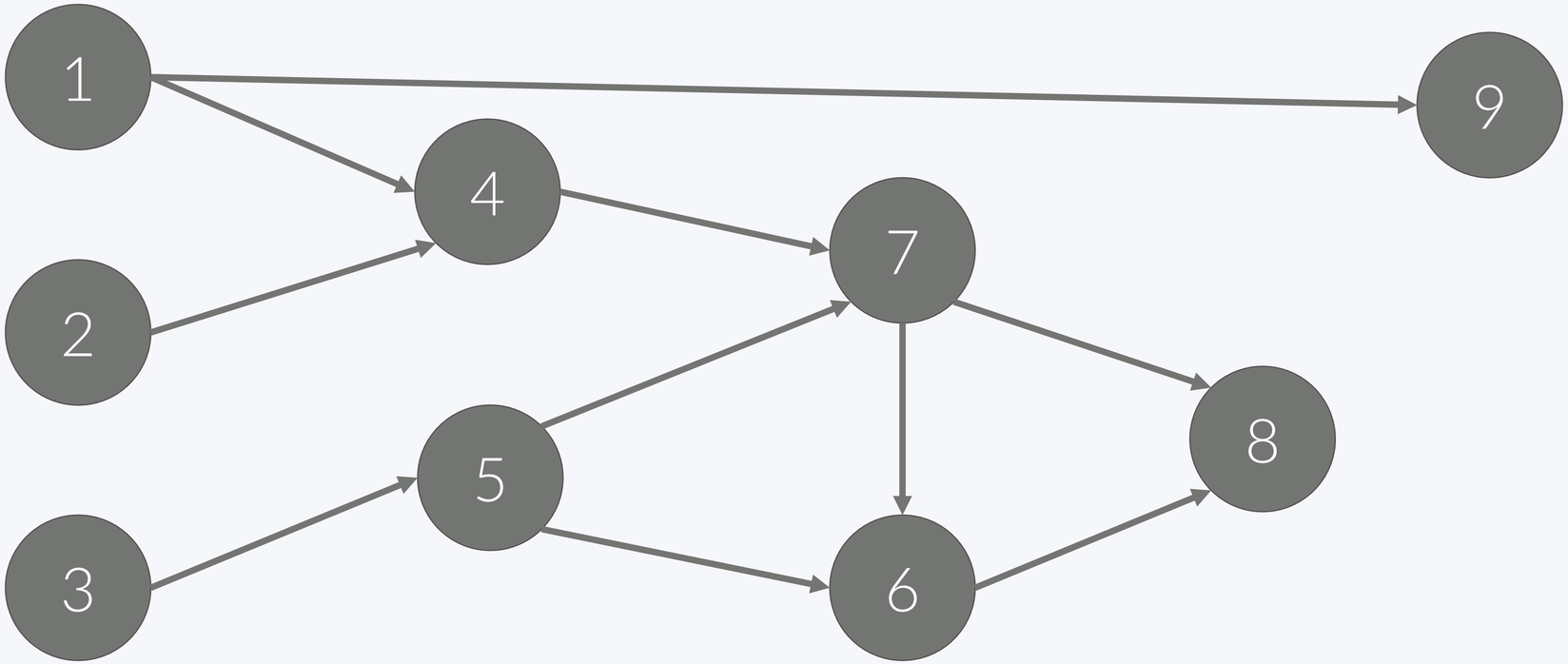
Topological Sort

- 어떤 일을 하는 순서를 찾는 알고리즘이다
- 1 -> 2
- 2를 하기 전에 1을 먼저 해야 한다.

위상 정렬

Topological Sort

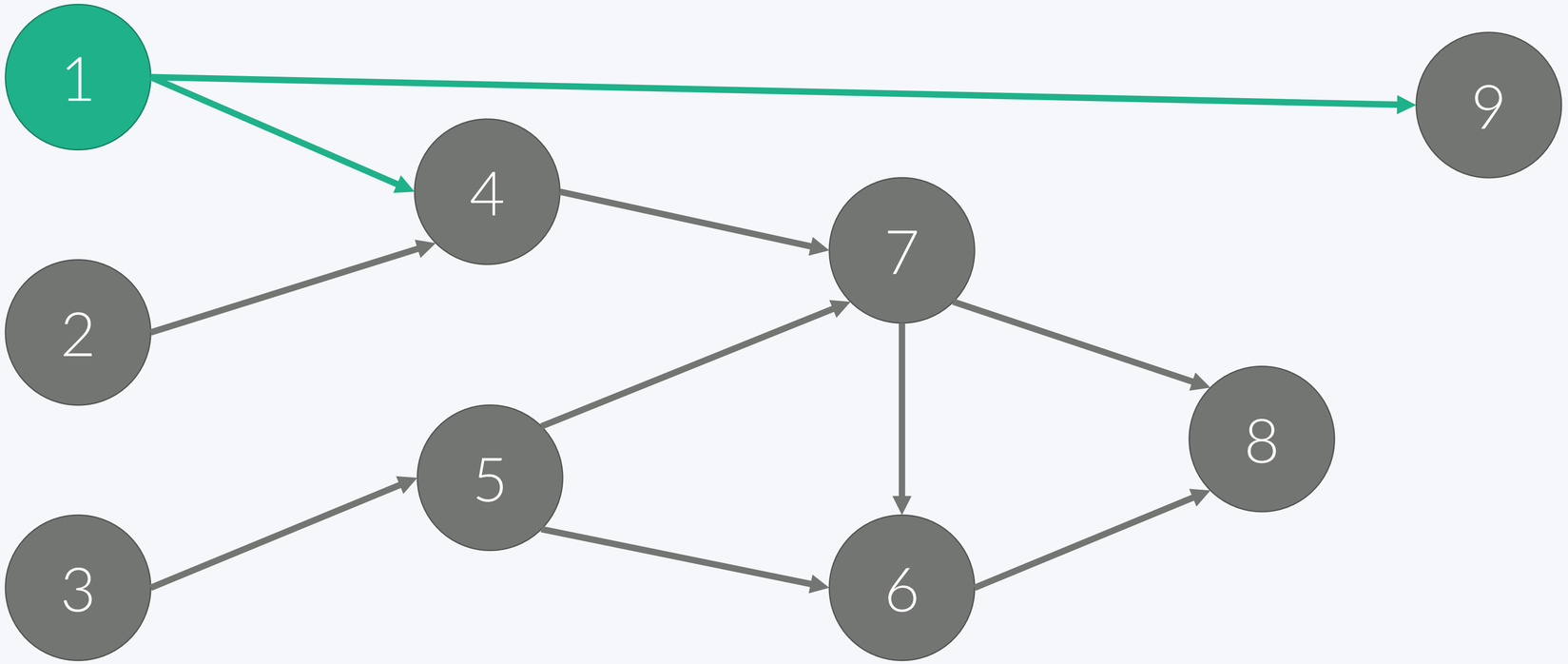
- 큐에 가장 들어있는 것은
- 들어오는 간선의 개수가 0인 것이다.
- 순서:
- 큐: 1 2 3



위상 정렬

Topological Sort

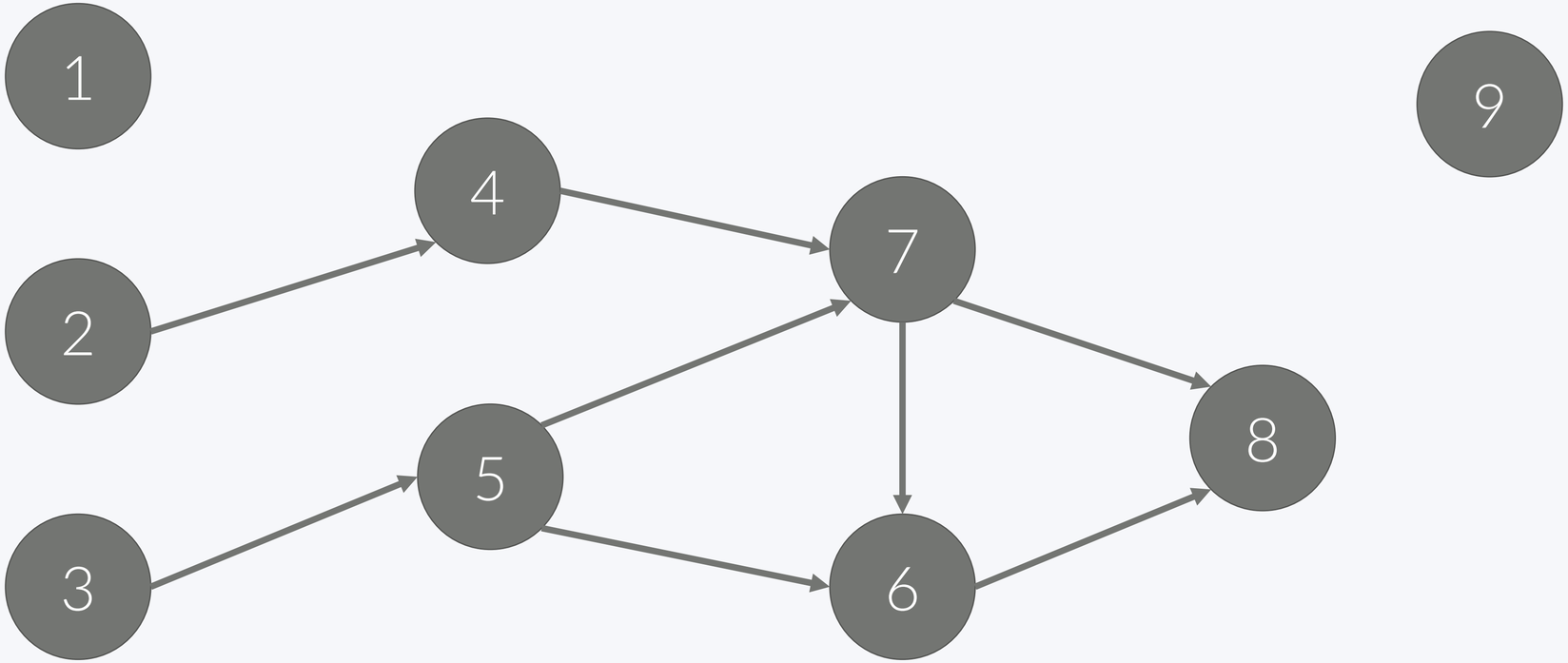
- 순서: 1
- 큐: 2 3



위상 정렬

Topological Sort

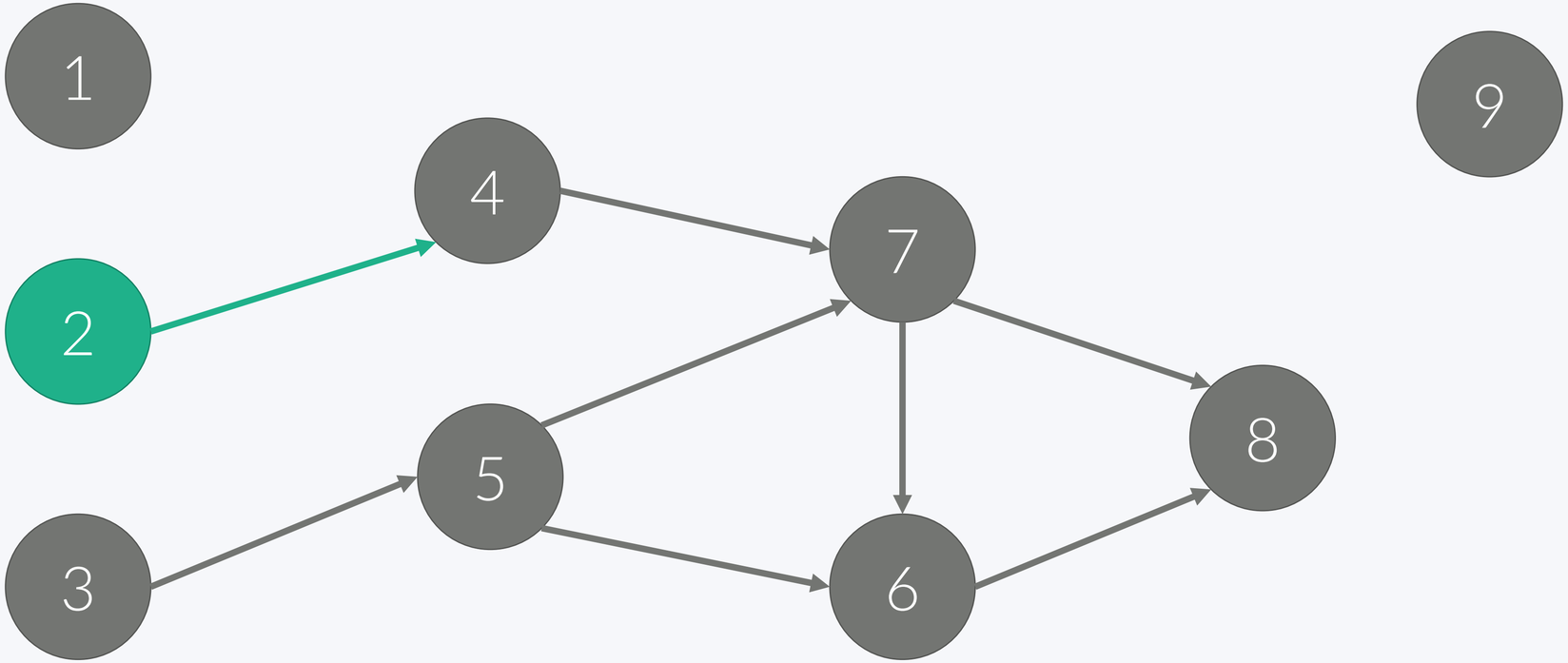
- 순서: 1
- 큐: 2 3 9



위상 정렬

Topological Sort

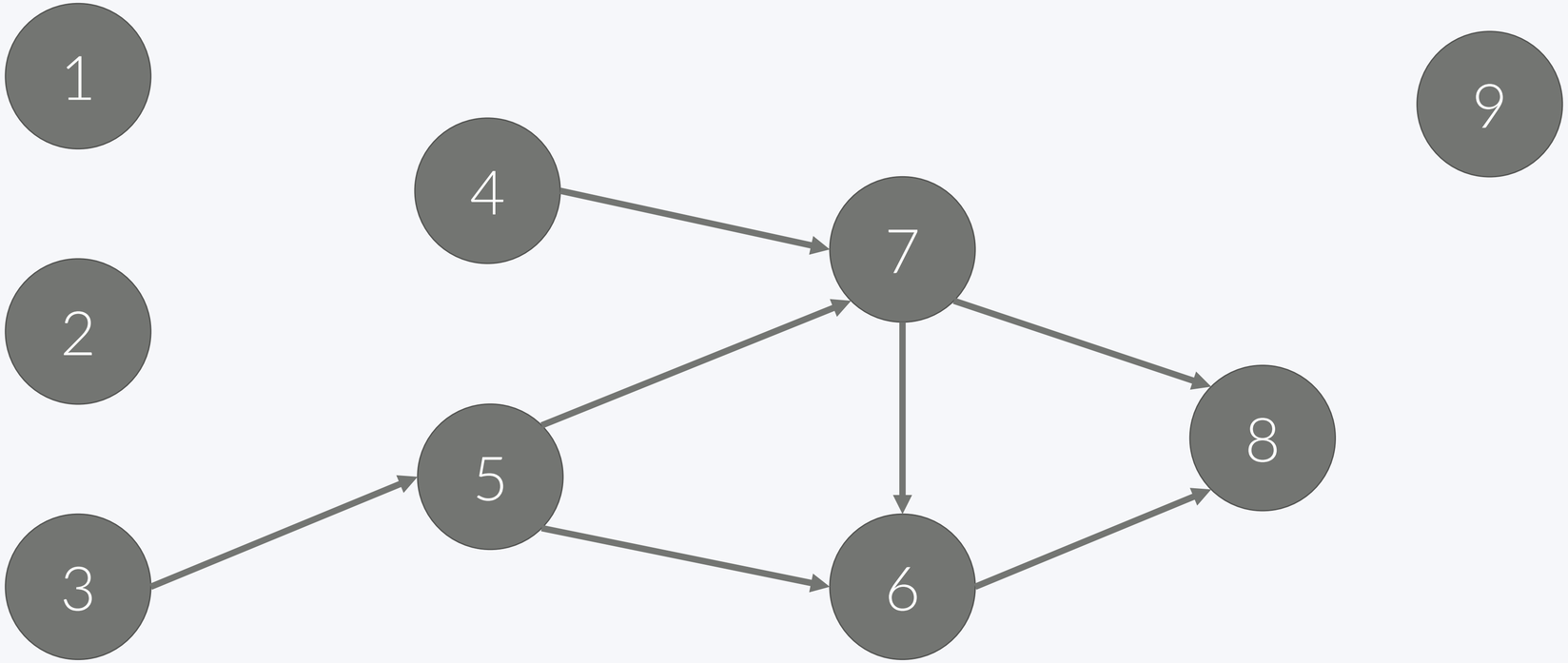
- 순서: 1 2
- 큐: 3 9



위상 정렬

Topological Sort

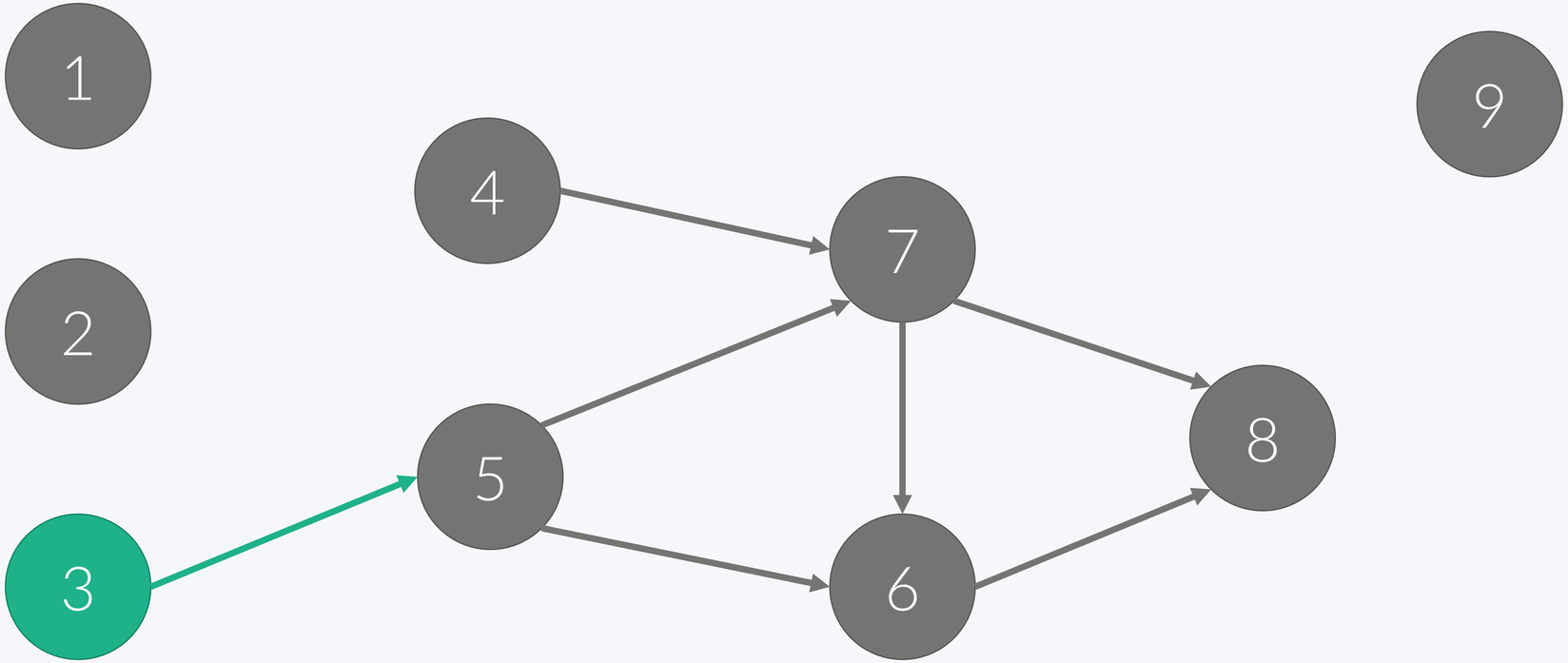
- 순서: 1 2
- 큐: 3 9 4



위상 정렬

Topological Sort

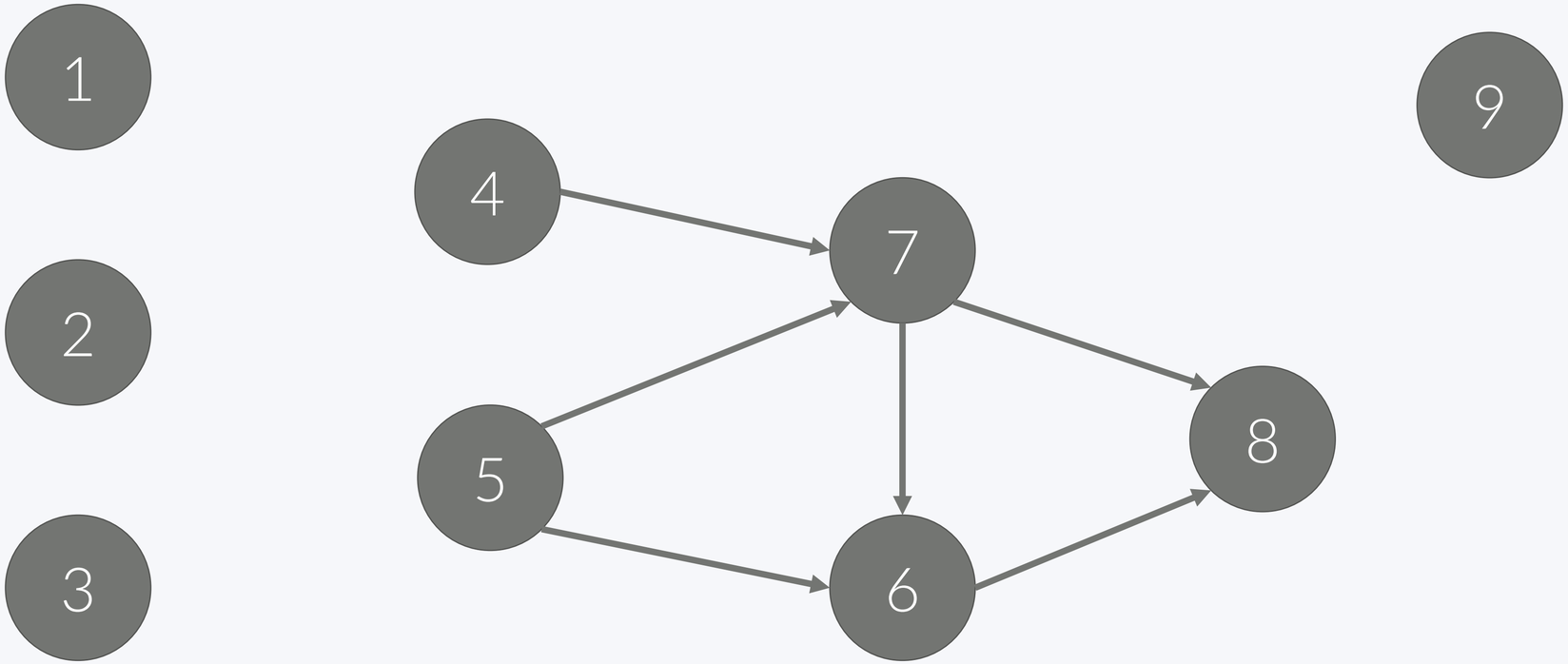
- 순서: 1 2 3
- 큐: 9 4



위상 정렬

Topological Sort

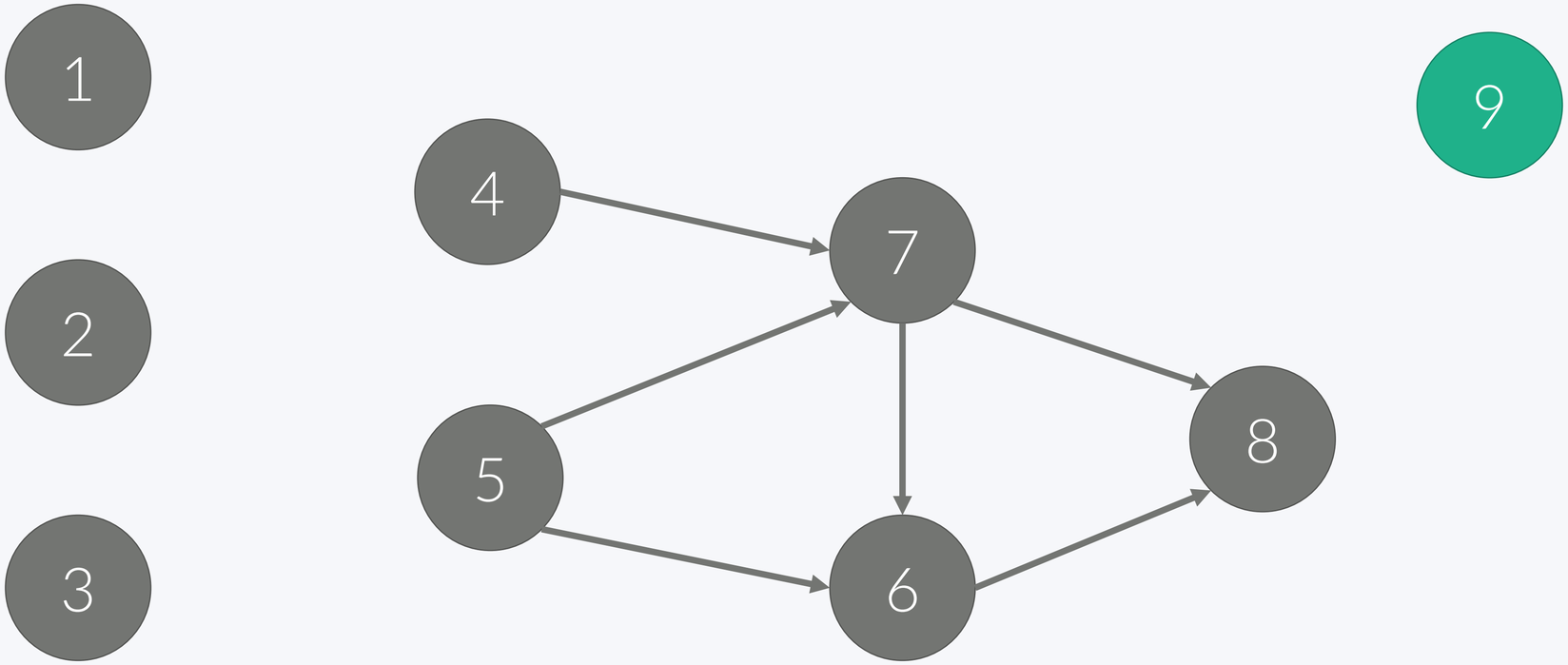
- 순서: 1 2 3 4
- 큐: 9 4 5



위상 정렬

Topological Sort

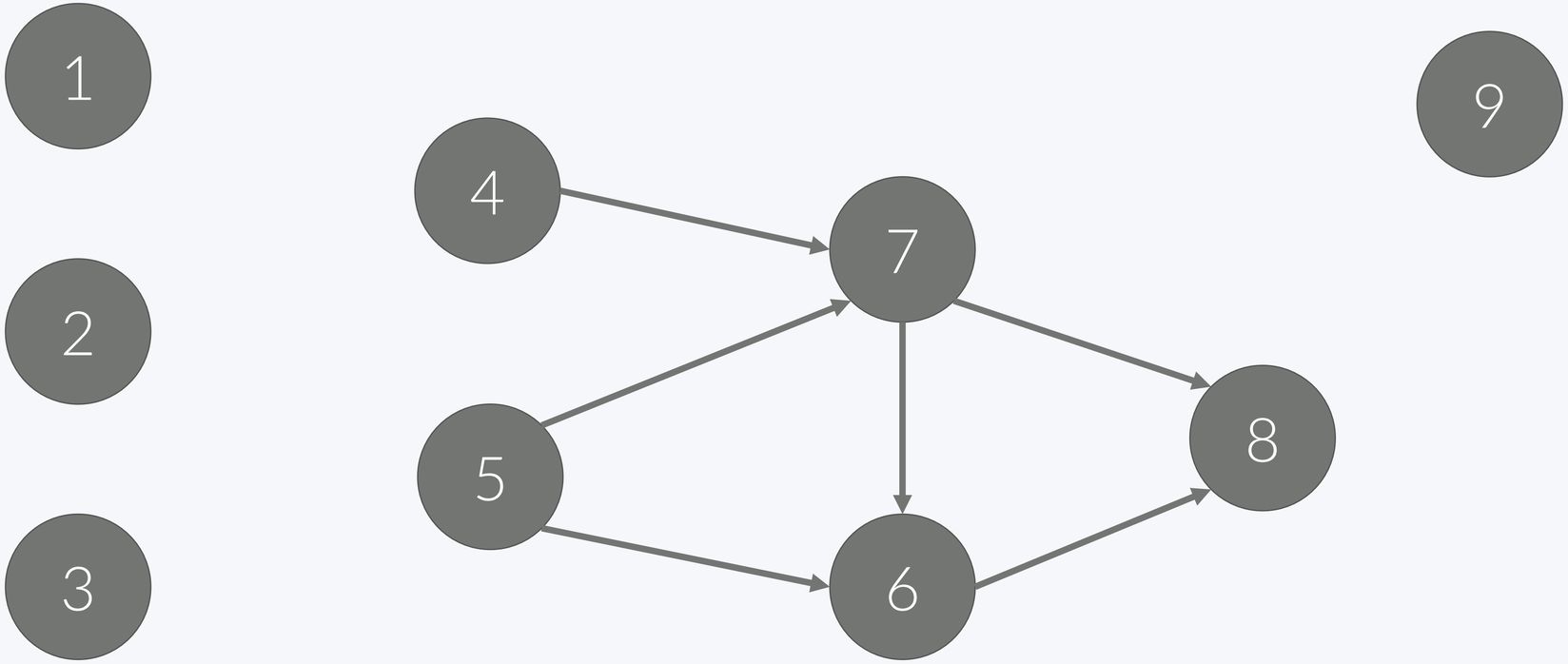
- 순서: 1 2 3 9
- 큐: 4 5



위상 정렬

Topological Sort

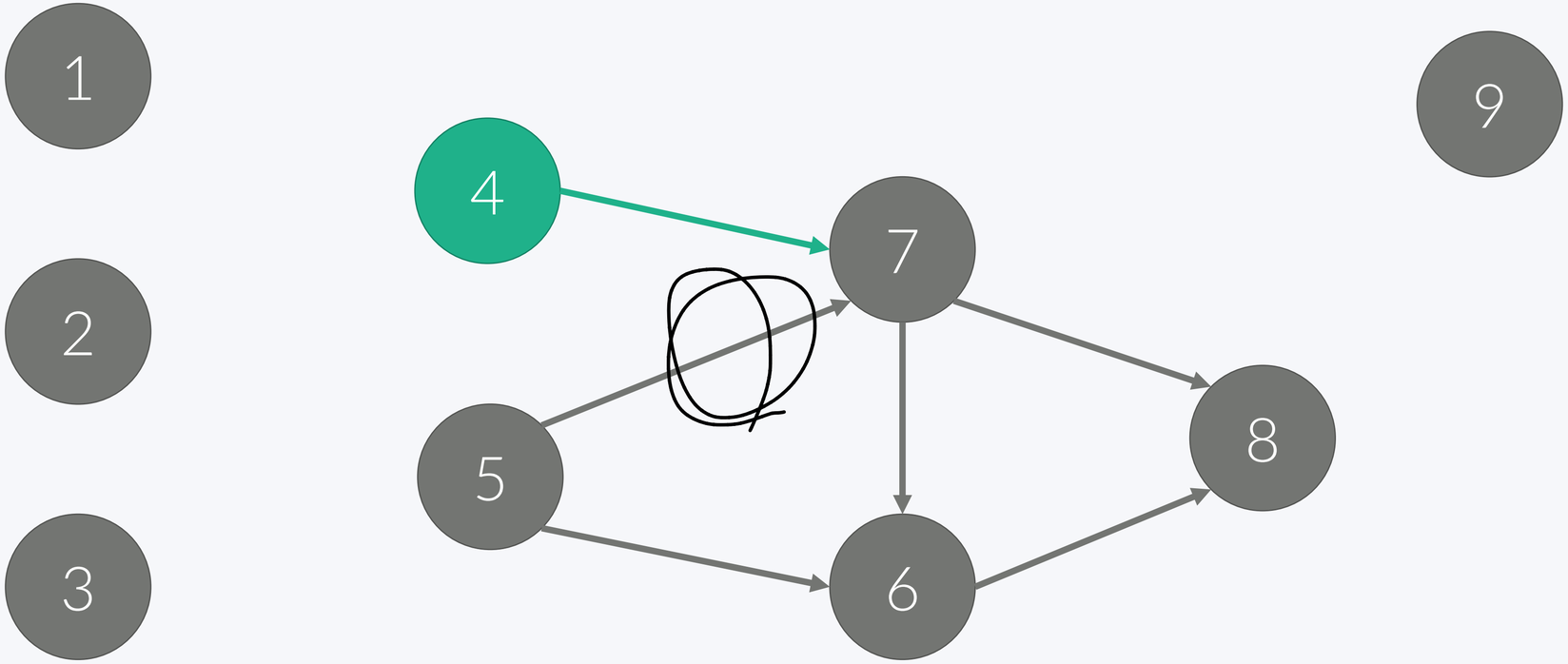
- 순서: 1 2 3 9
- 큐: 4 5



위상 정렬

Topological Sort

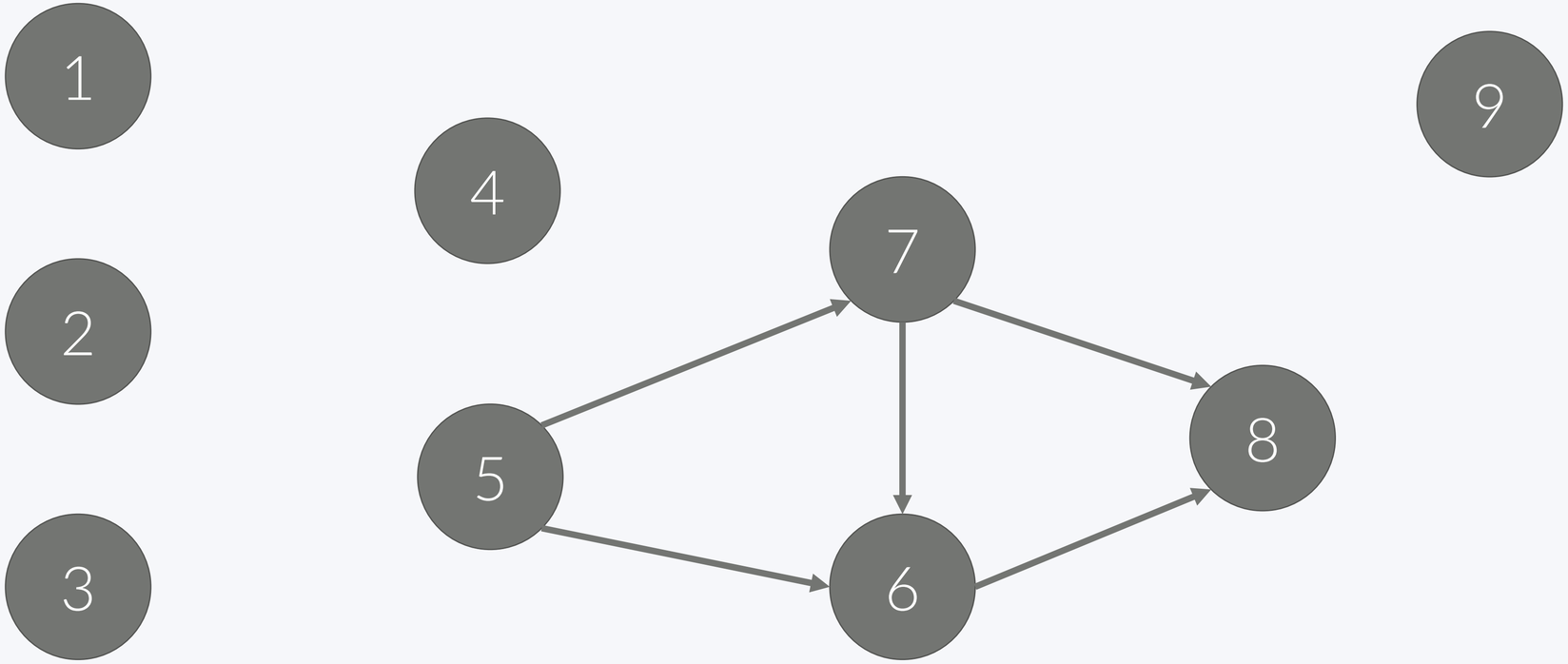
- 순서: 1 2 3 9 4
- 큐: 5



위상 정렬

Topological Sort

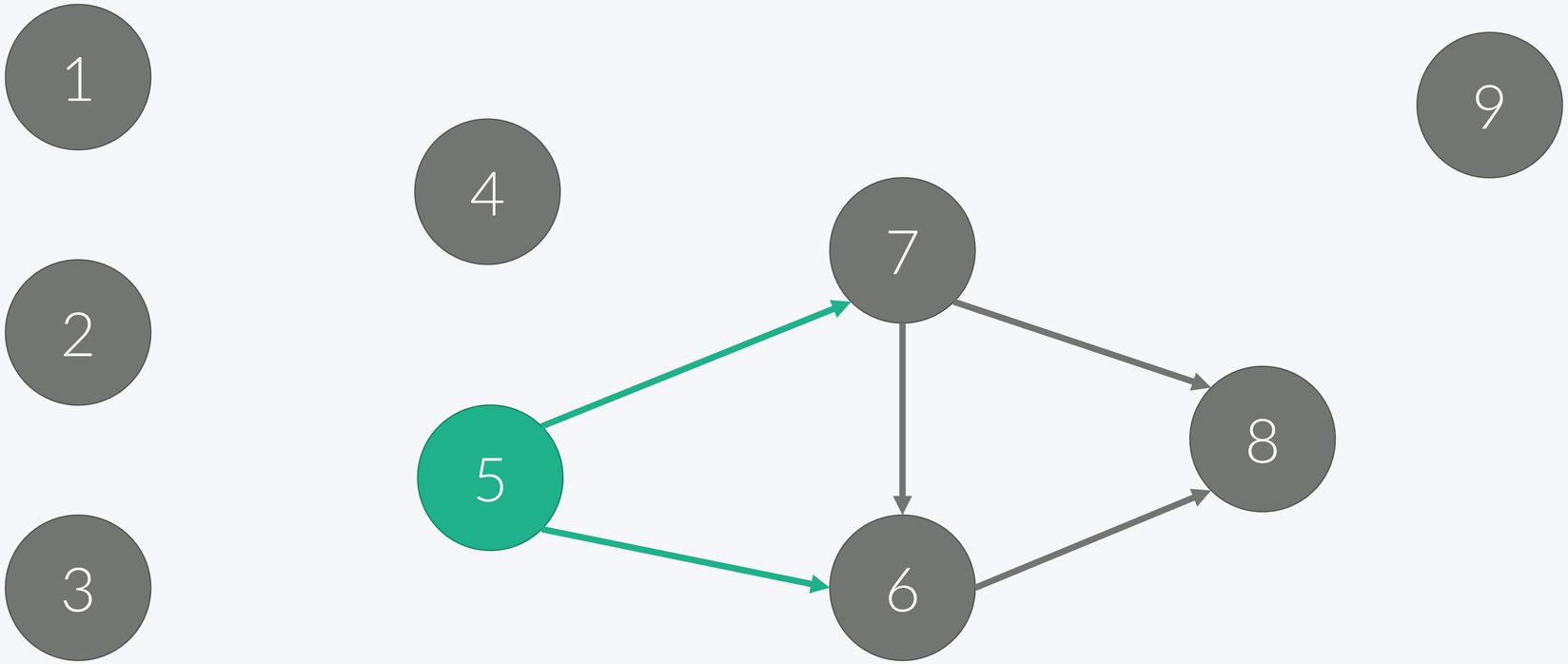
- 순서: 1 2 3 9 4
- 큐: 5



위상 정렬

Topological Sort

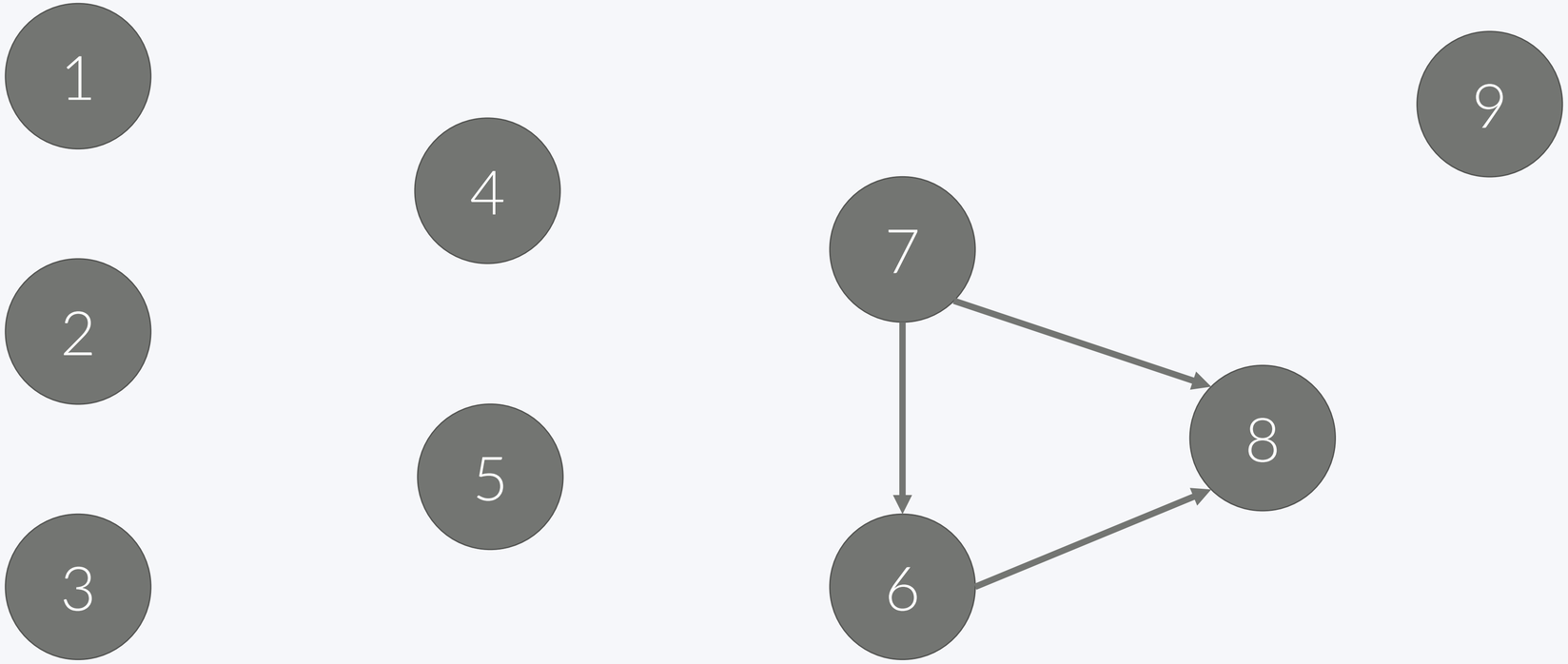
- 순서: 1 2 3 9 4 5
- 큐:



위상 정렬

Topological Sort

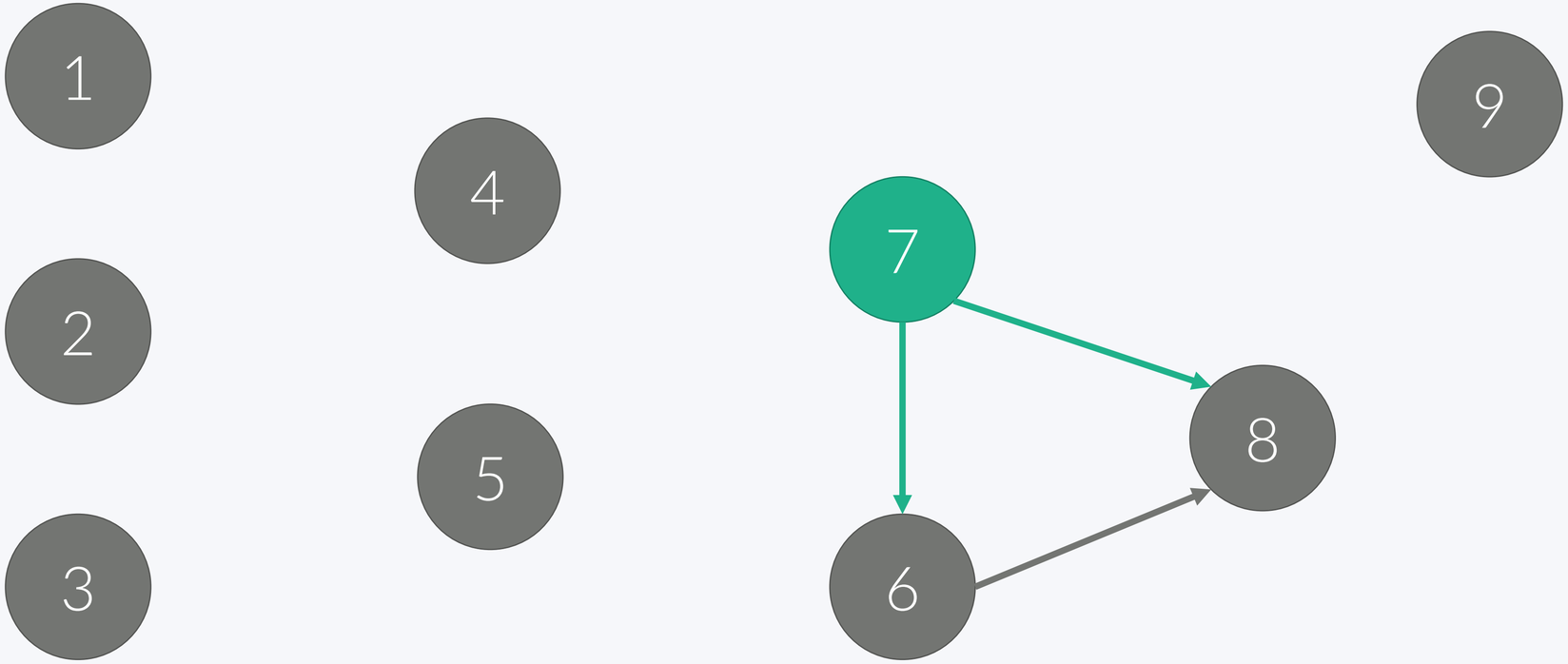
- 순서: 1 2 3 9 4 5
- 큐: 7



위상 정렬

Topological Sort

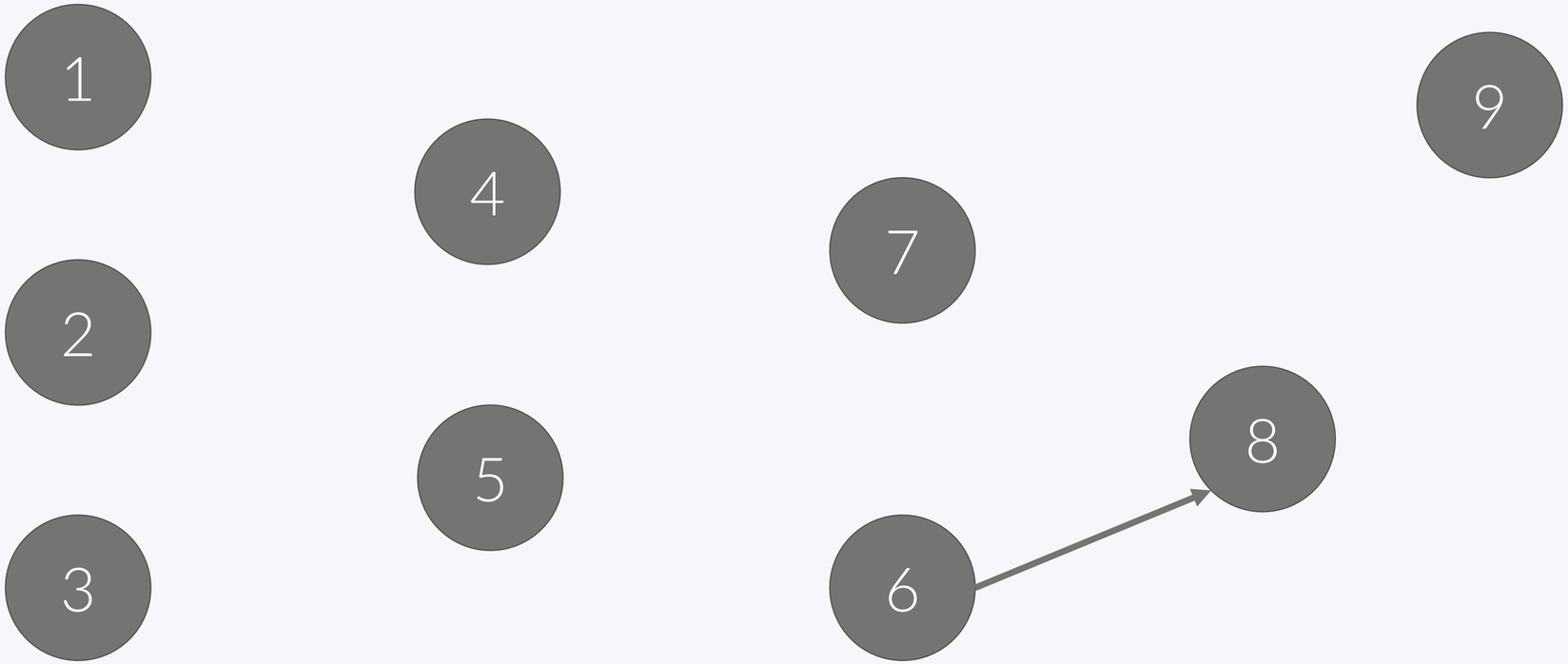
- 순서: 1 2 3 9 4 5 7
- 큐:



위상 정렬

Topological Sort

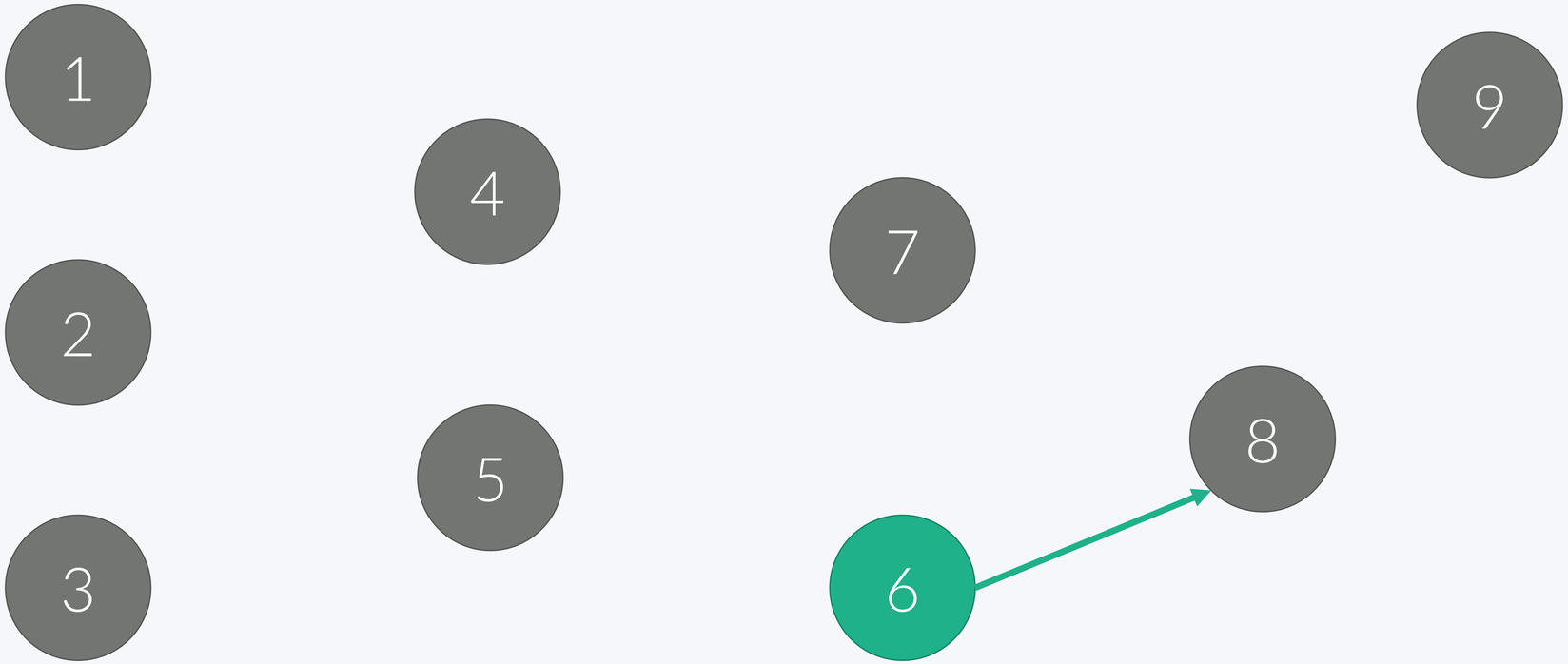
- 순서: 1 2 3 9 4 5 7
- 큐: 6



위상 정렬

Topological Sort

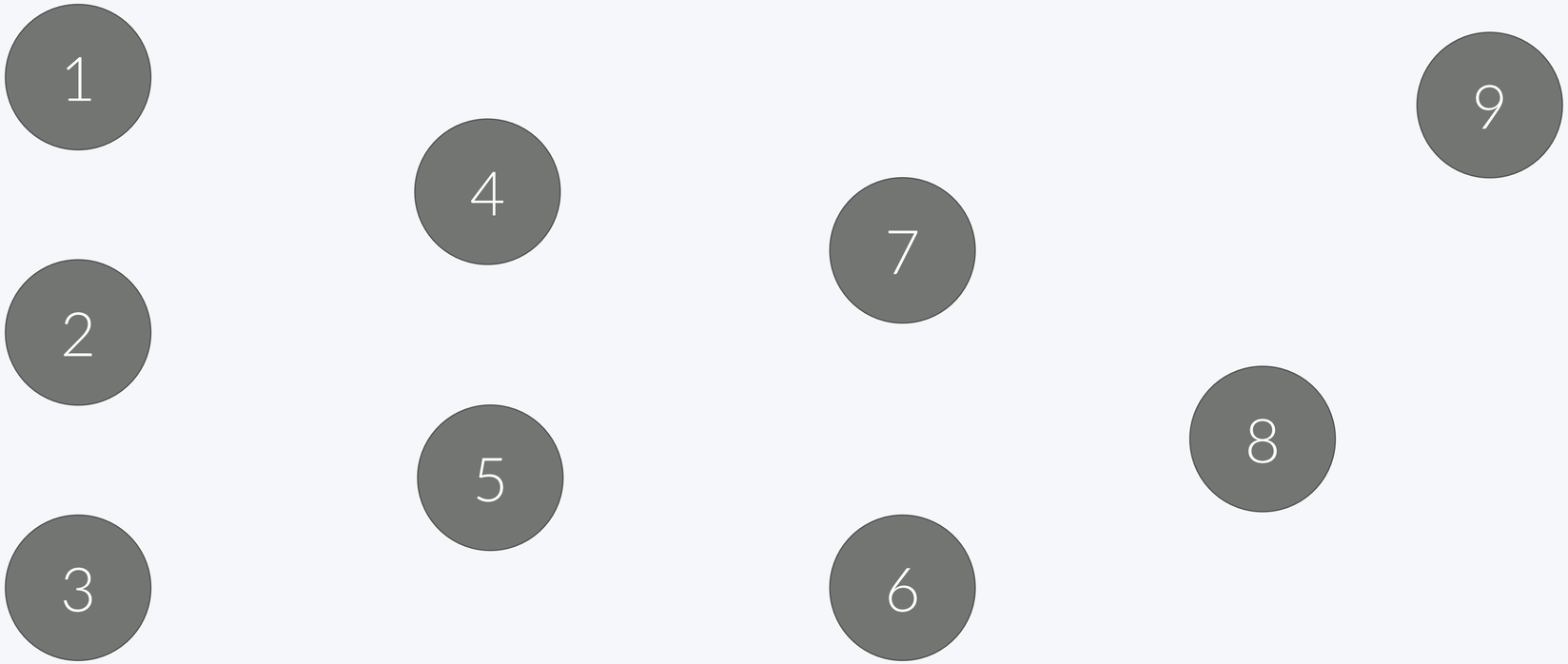
- 순서: 1 2 3 9 4 5 7 6
- 큐:



위상 정렬

Topological Sort

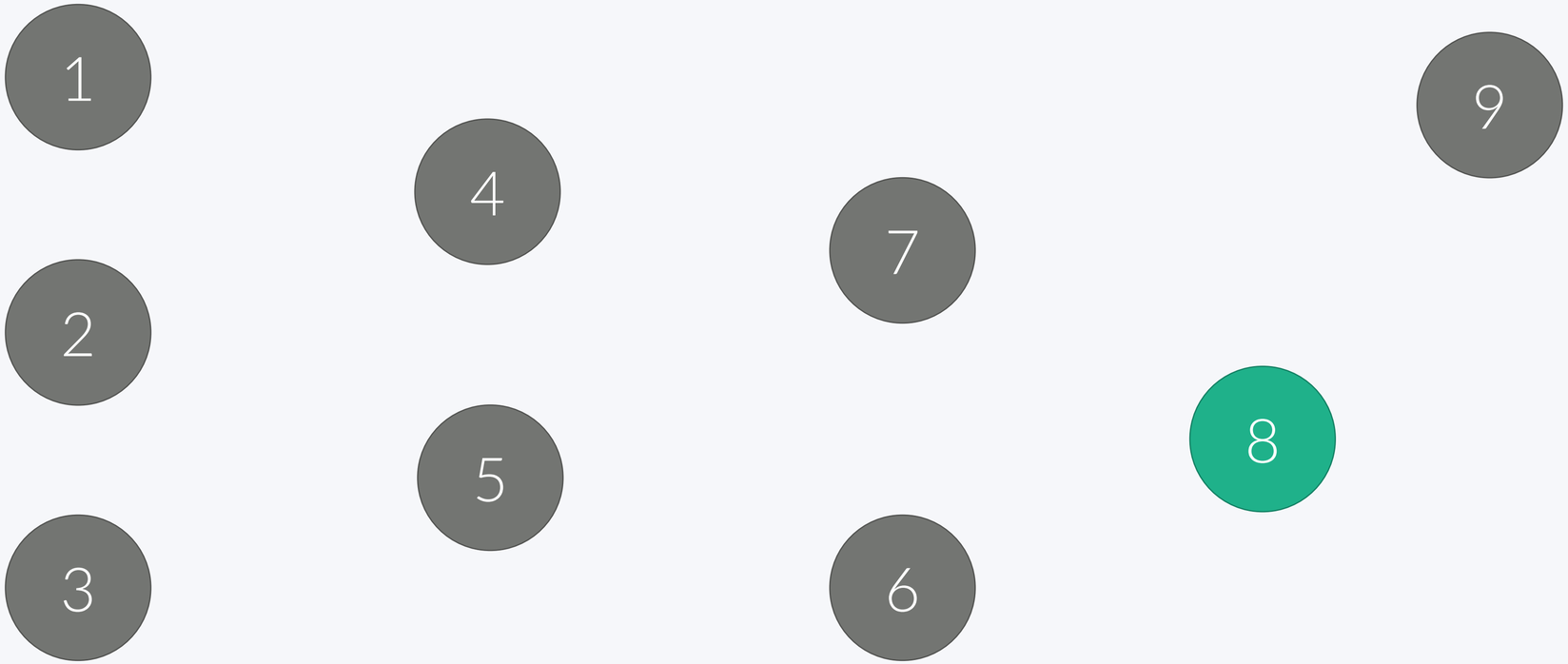
- 순서: 1 2 3 9 4 5 7 6
- 큐: 8



위상 정렬

Topological Sort

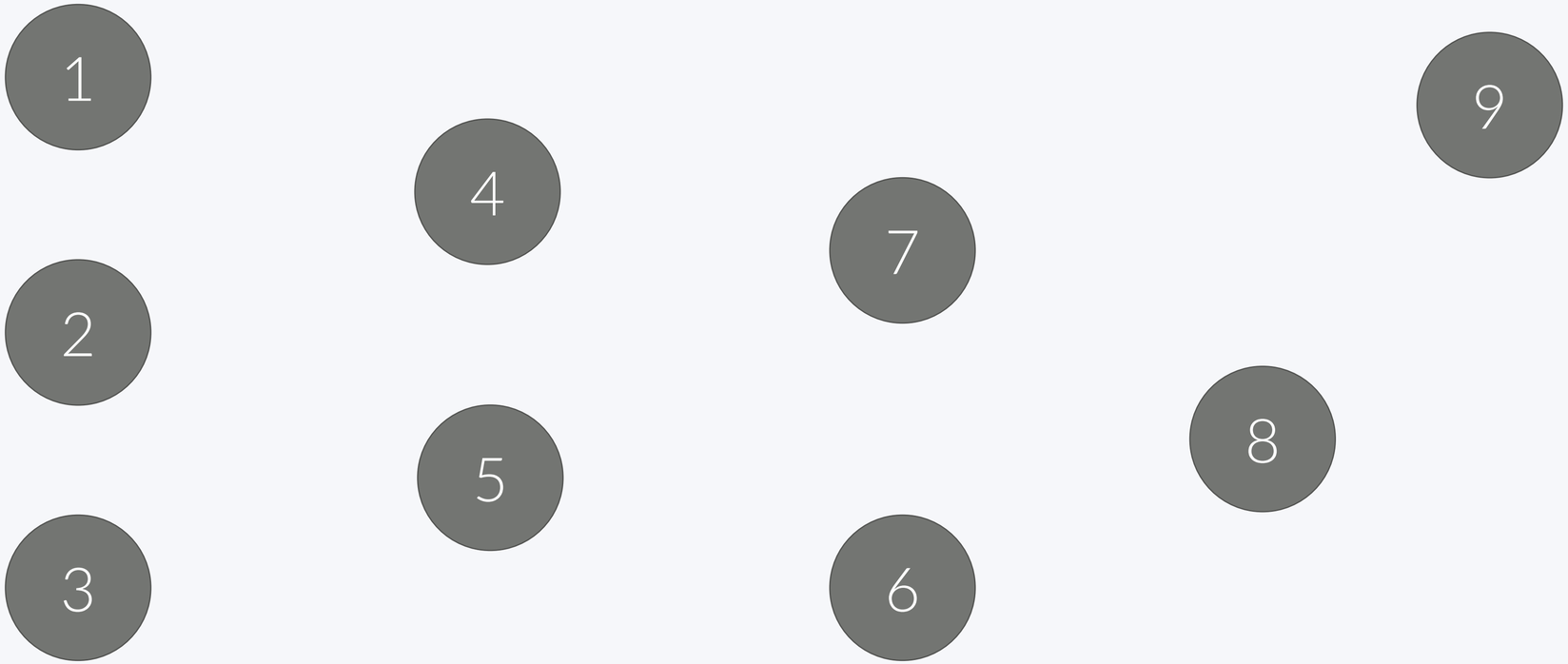
- 순서: 1 2 3 9 4 5 7 6 8
- 큐:



위상 정렬

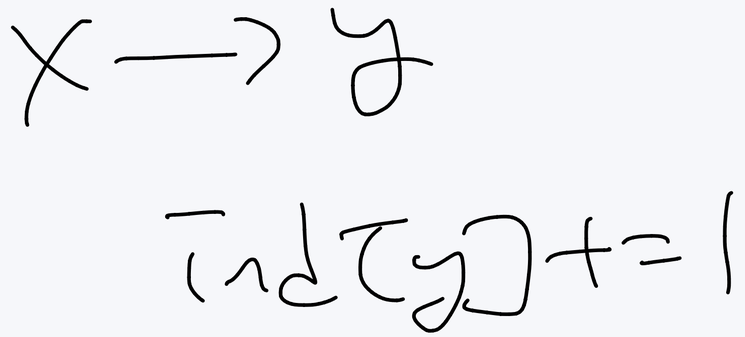
Topological Sort

- 순서: 1 2 3 9 4 5 7 6 8
- 큐:



위상 정렬

Topological Sort



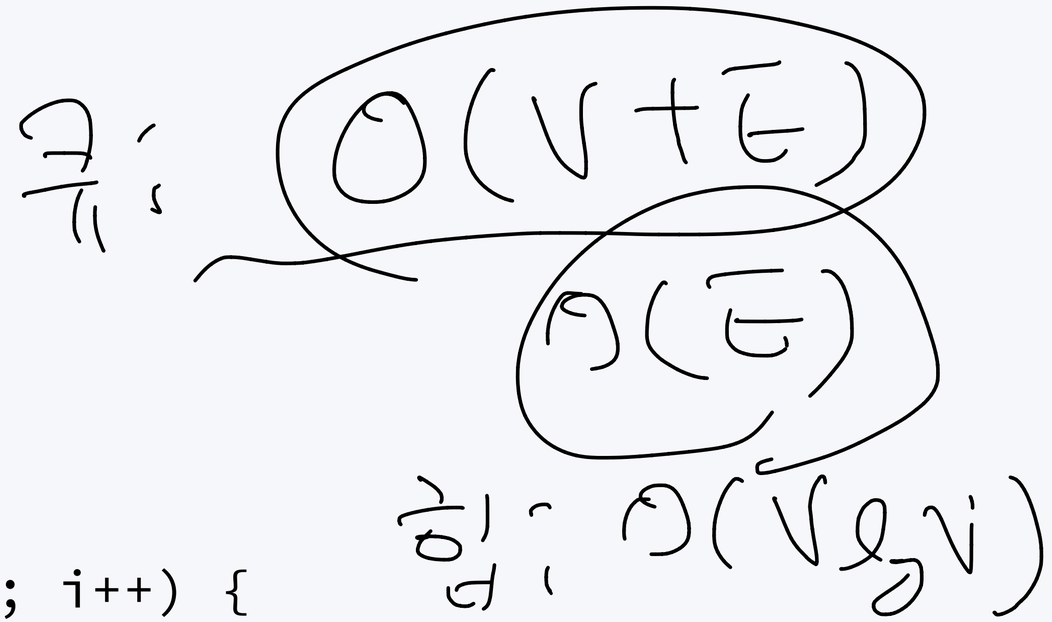
- 가장 처음

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

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);
        }
    }
}
```




```
if( check[y] == 0 ) {
    check[y] = 1;
    q.push(y);
}
```

줄 세우기

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

- 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

- 소스: http://boj.kr/12418e7f4fff41aba6a83918afafc009

문제집

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

- 위상 정렬에서 큐 대신에 우선 순위 큐를 사용해야 한다

문제집

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

- 소스: http://boj.kr/027ff63c629e479b9088adc0f8a6ec33

작업

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

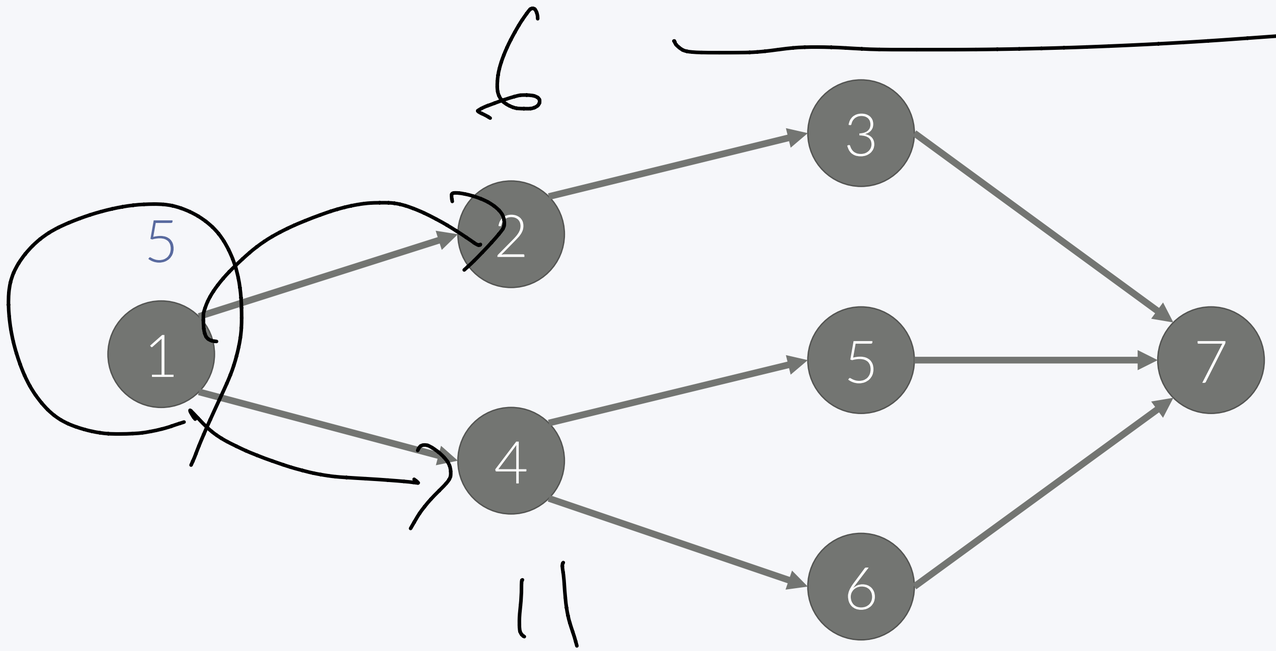
- 작업의 선행관계가 주어졌을 때, 모두 마치는 가장 빠른 시간
- 위상 정렬을 응용해서 풀 수 있다

작업

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

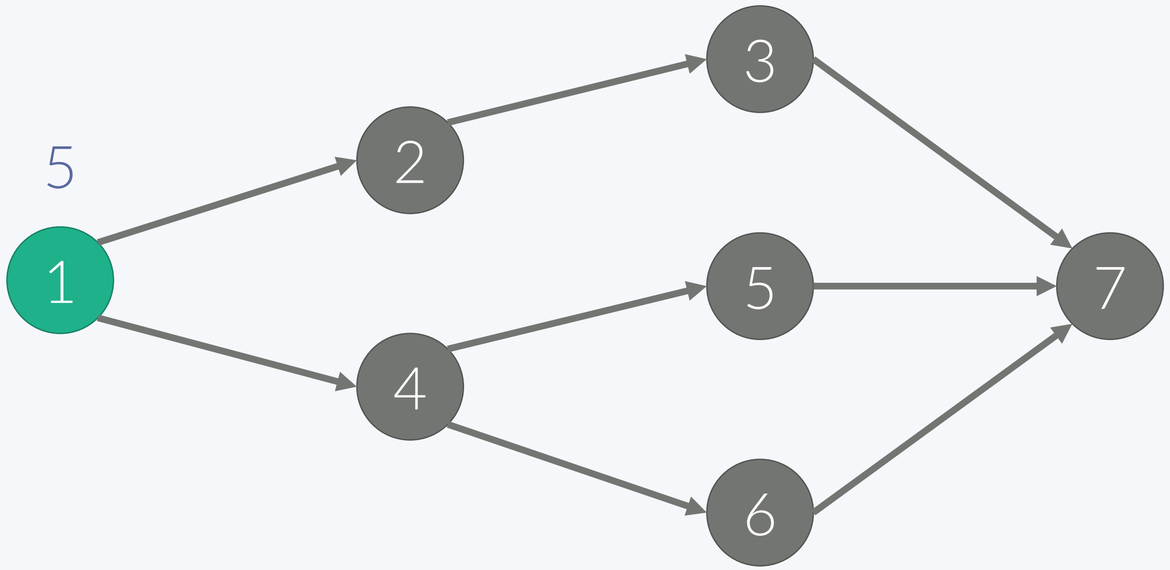


작업

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

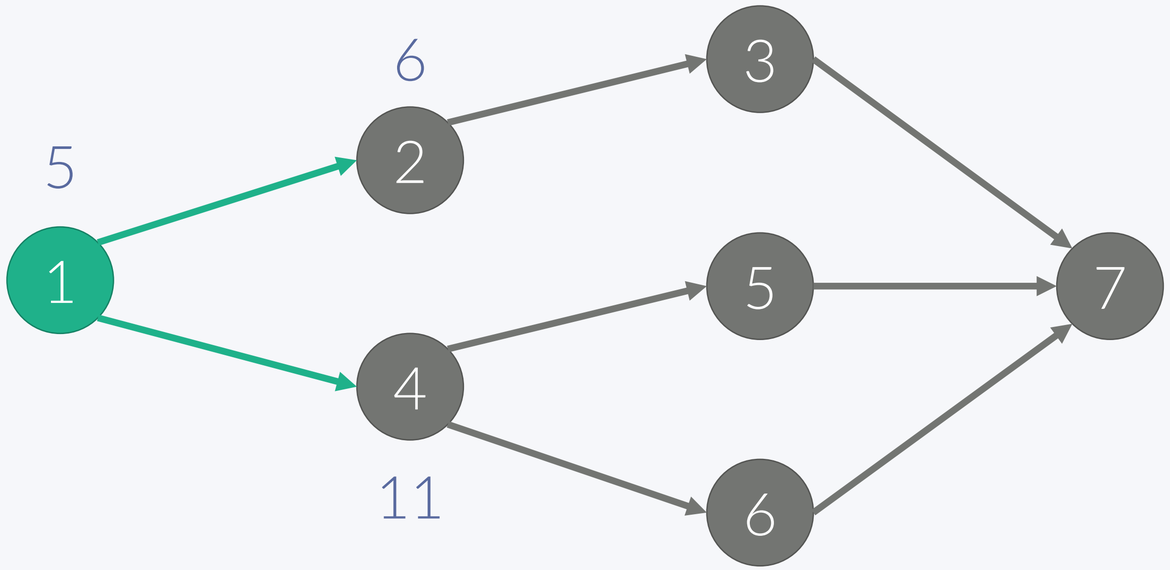


작업

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

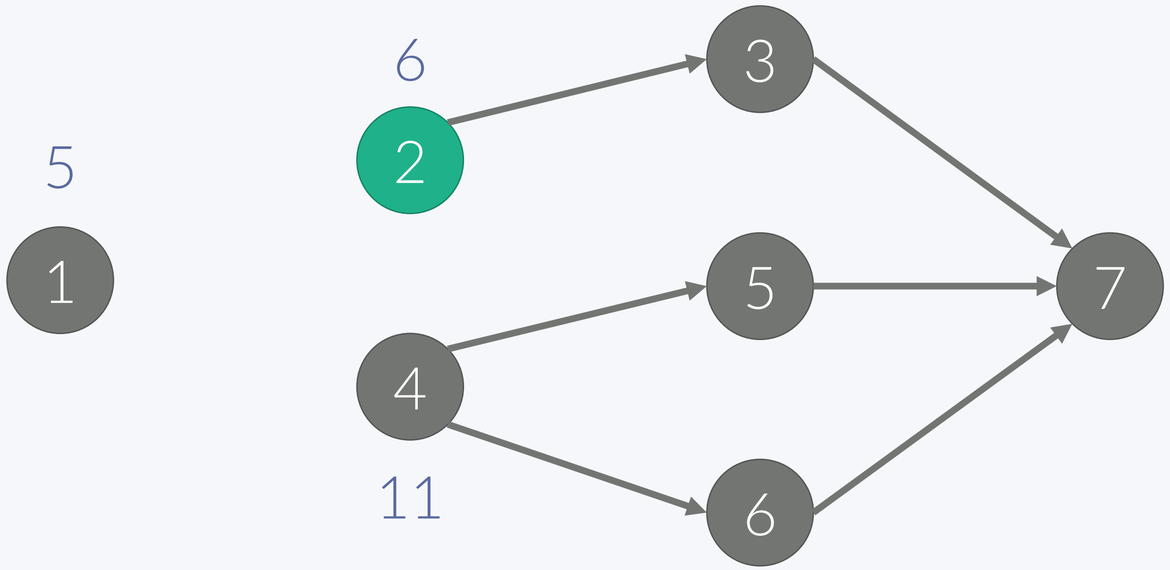


작업

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

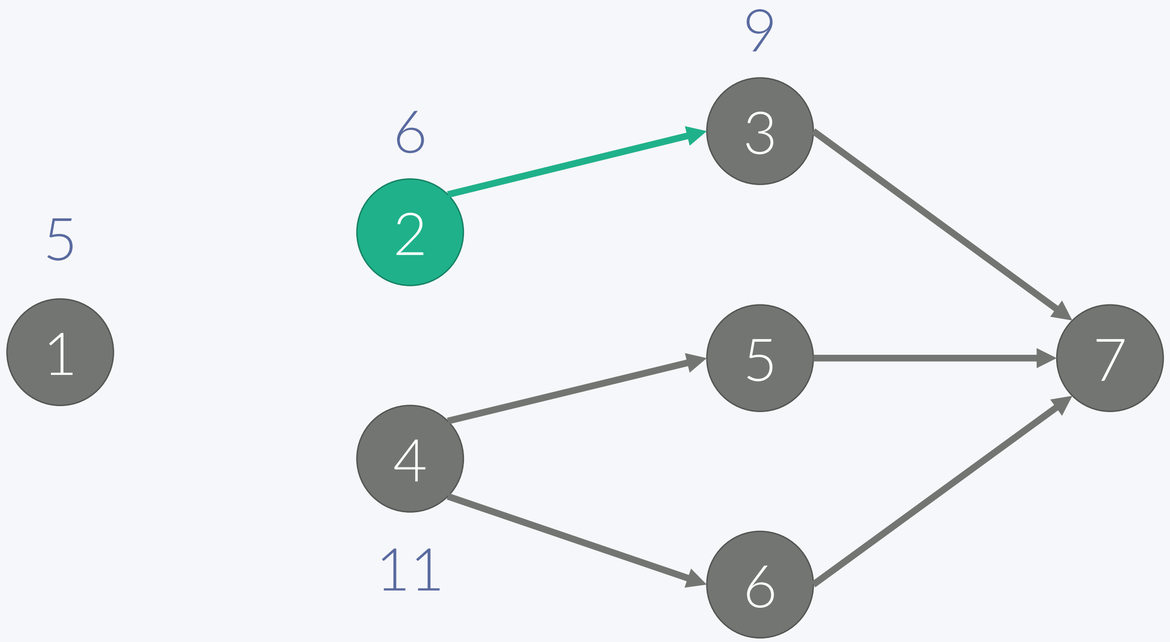


작업

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

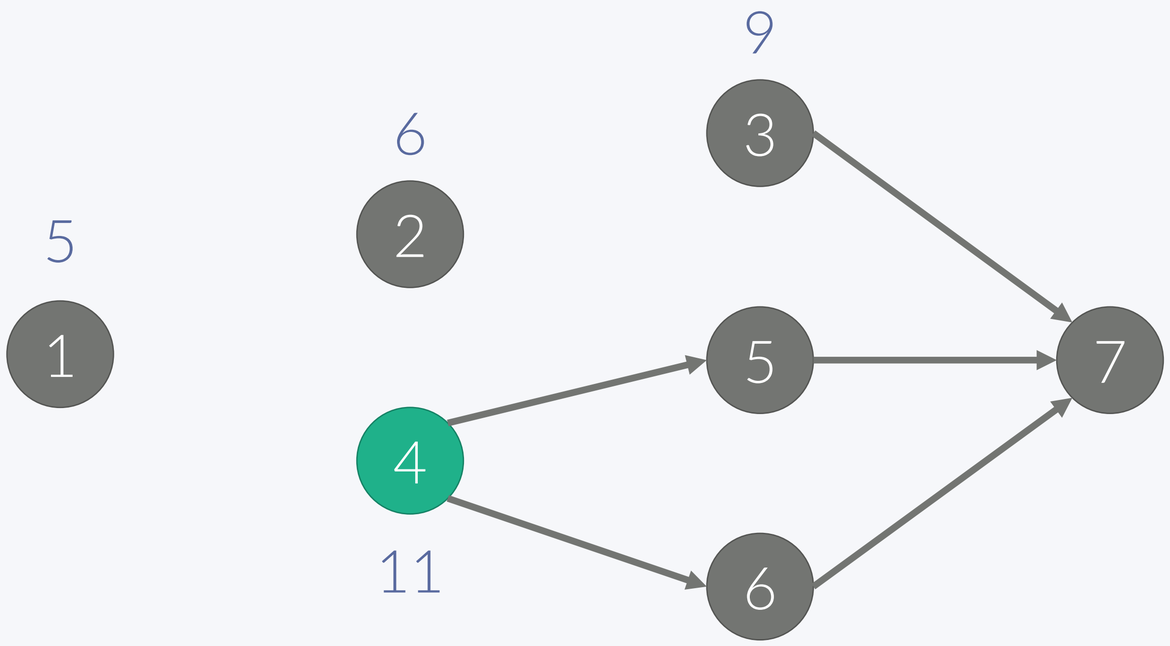


작업

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

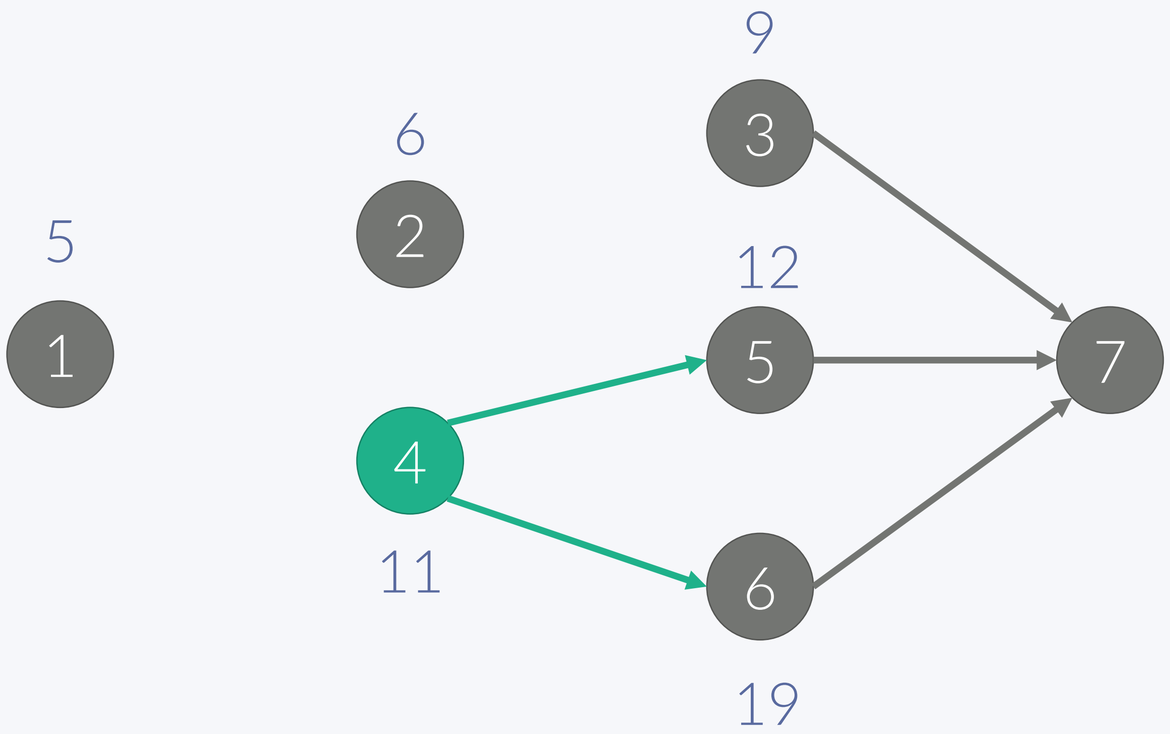


작업

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

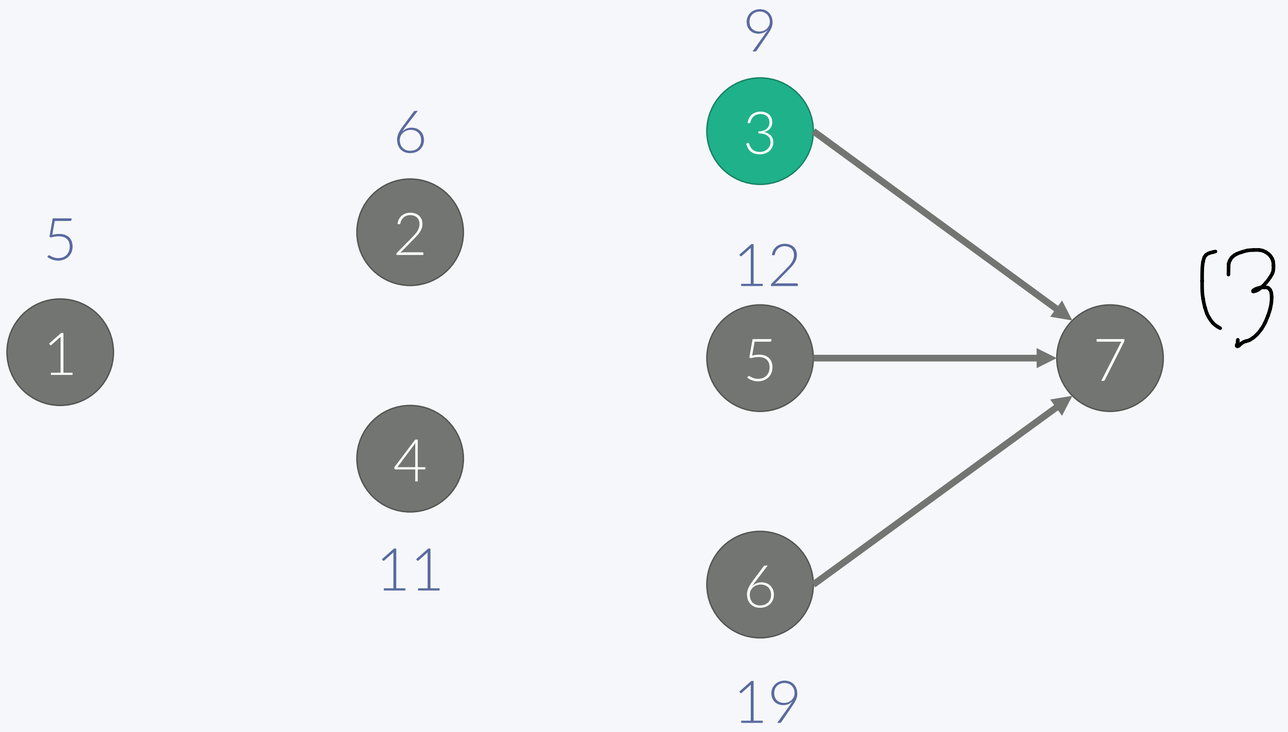


작업

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

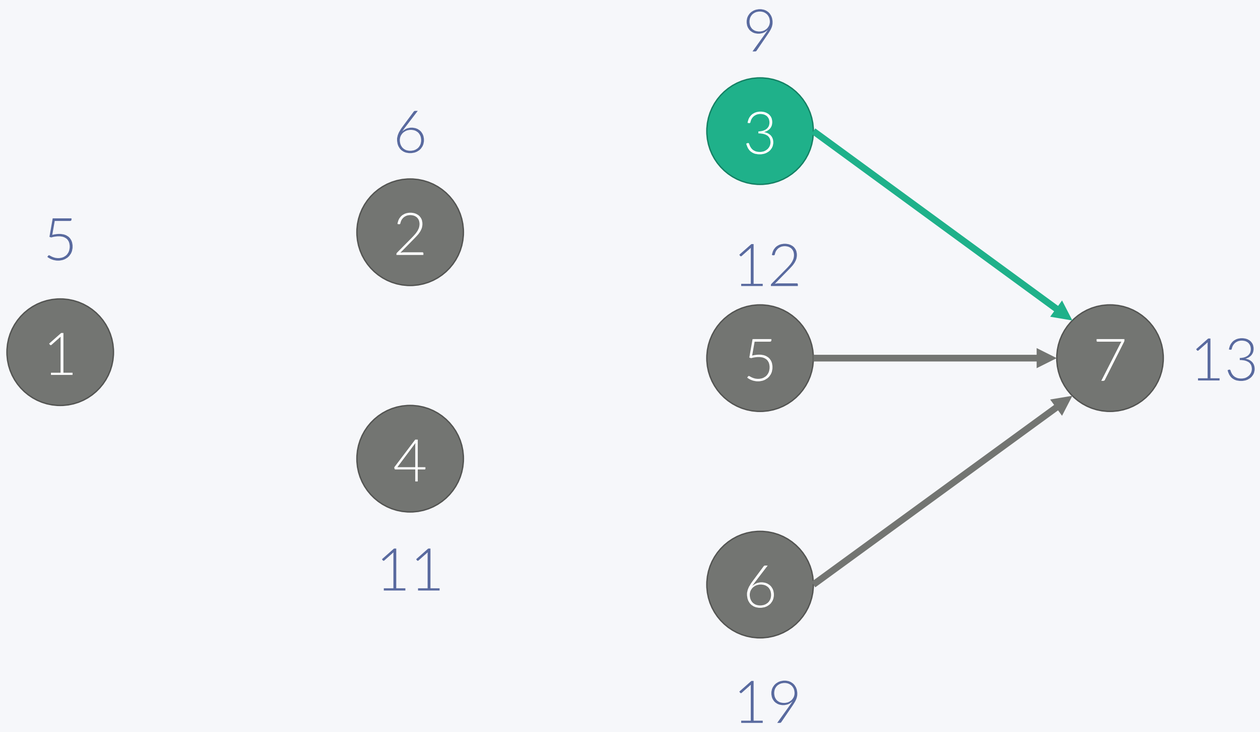


작업

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

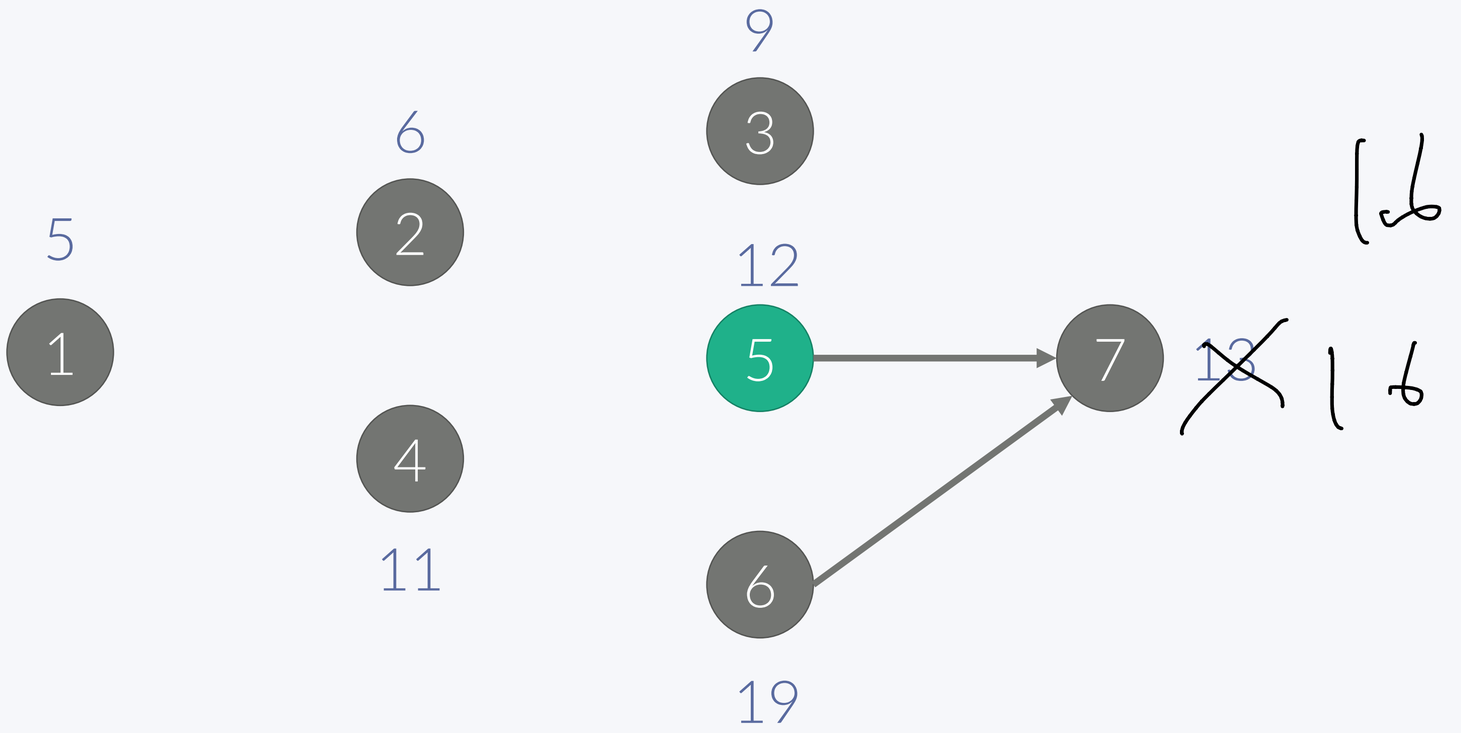


작업

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

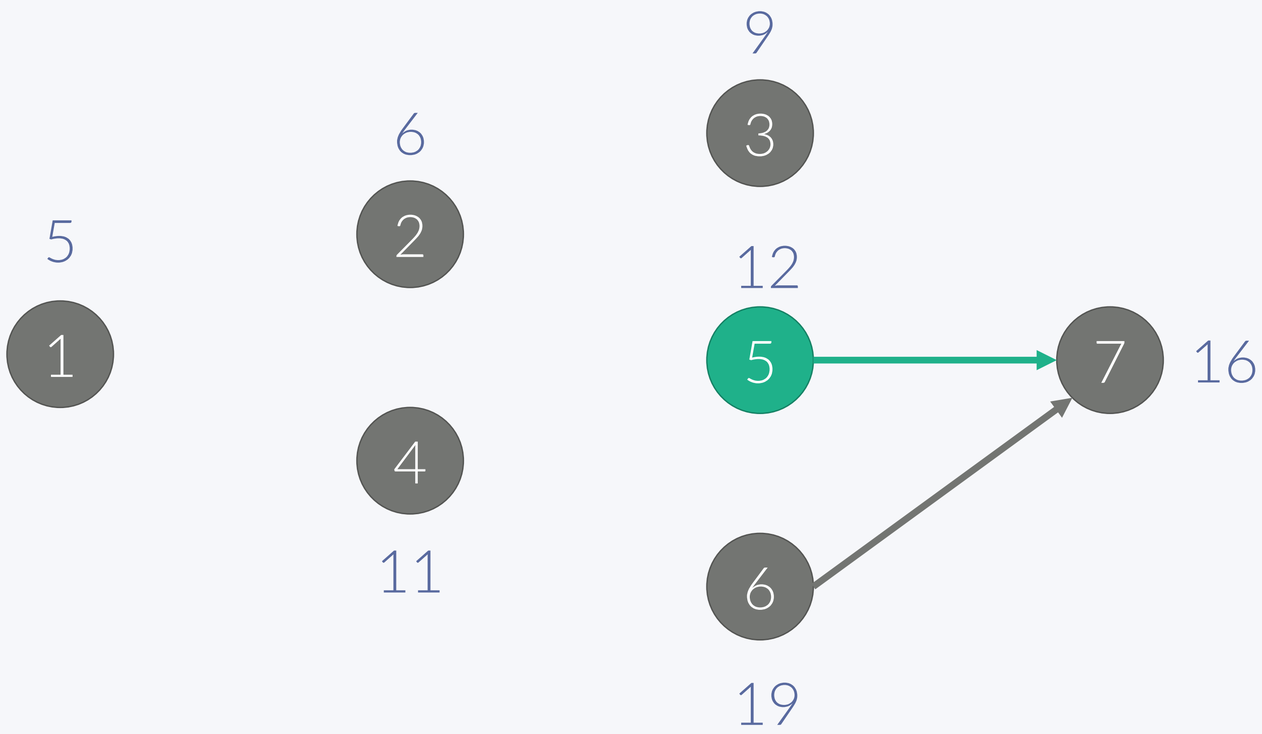


작업

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

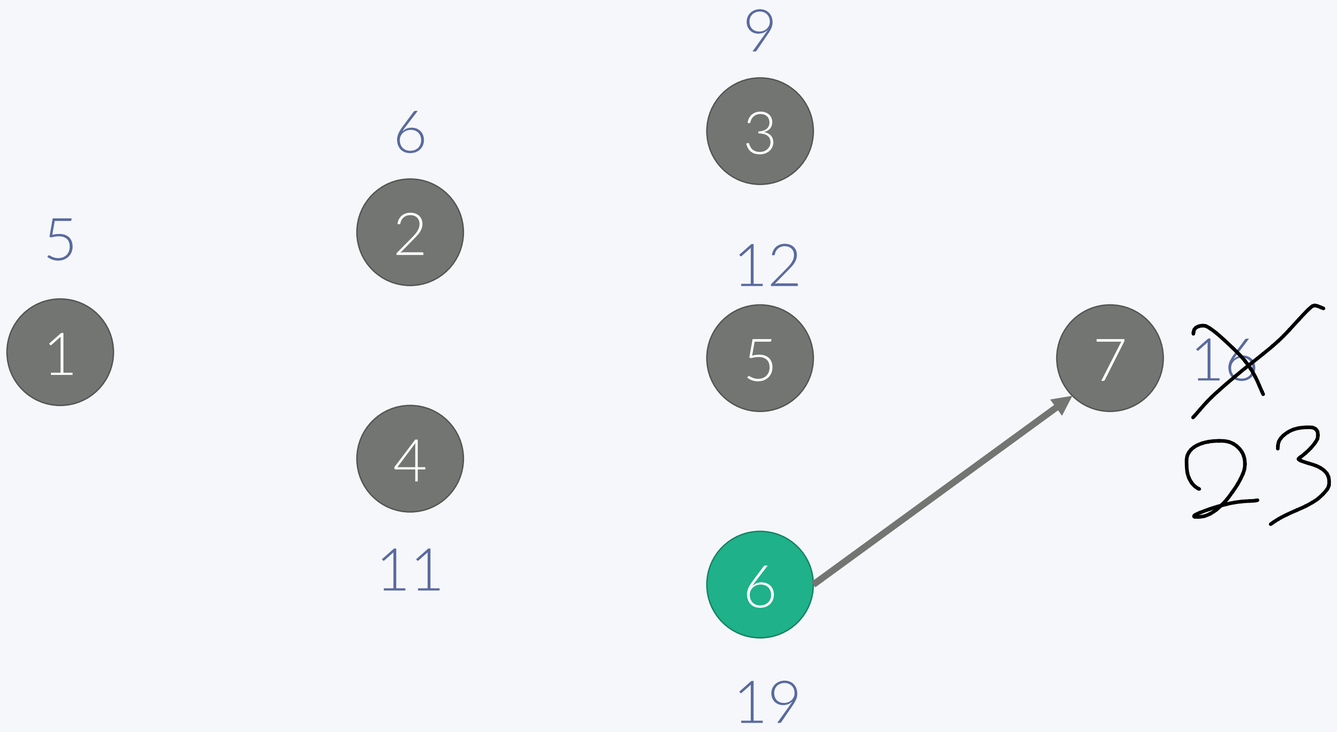


작업

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

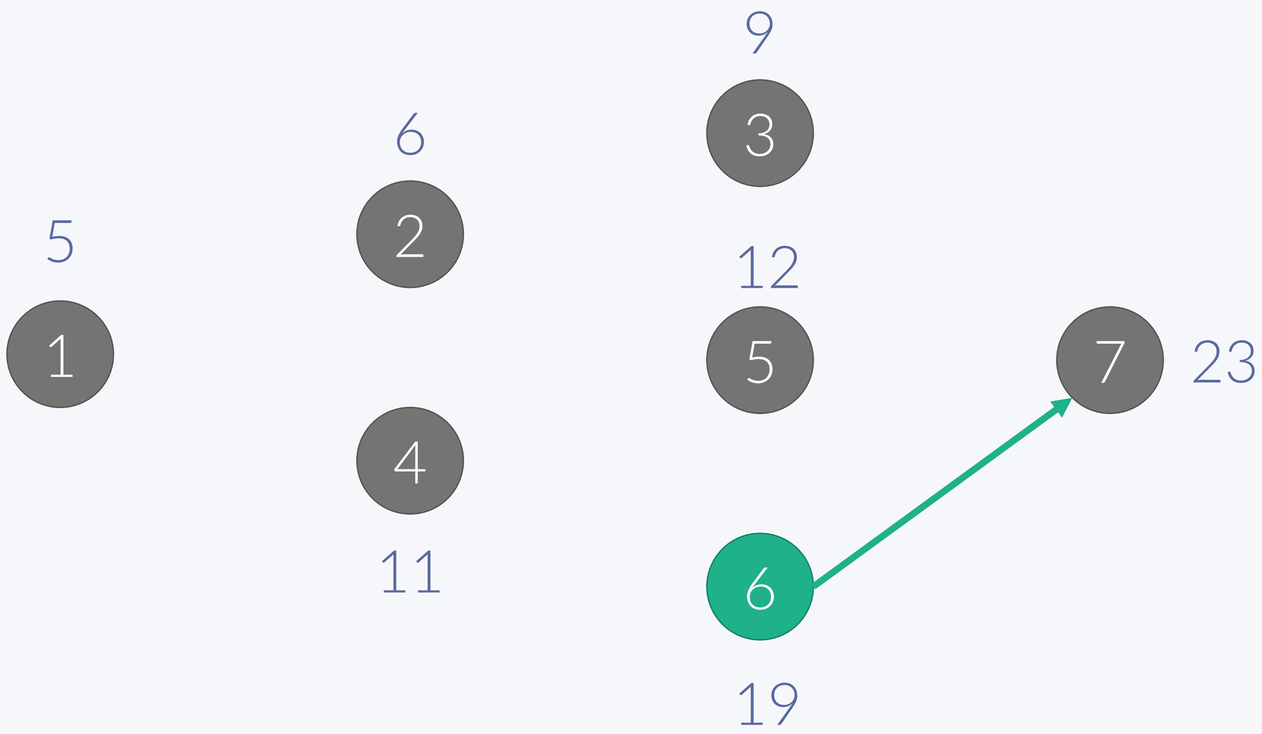


작업

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

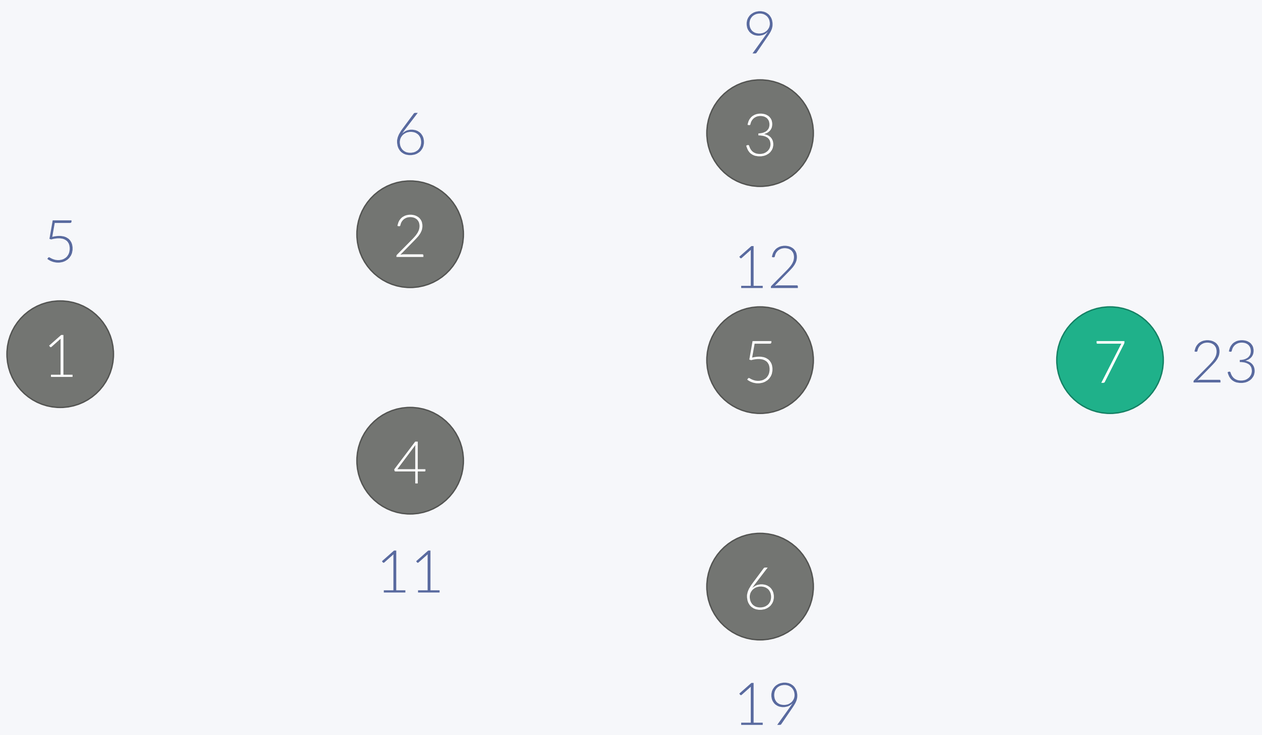


작업

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

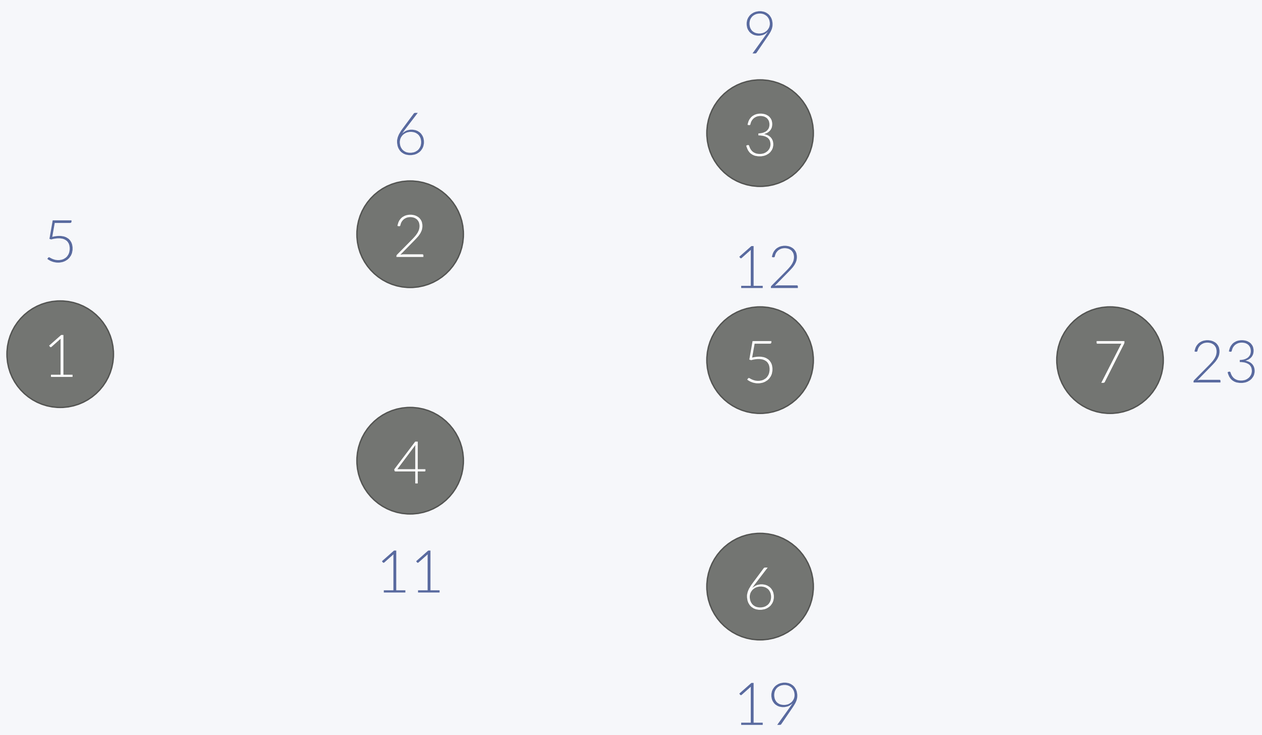


작업

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://boj.kr/c5c71501065a4ed590ea009f77cd5a6d

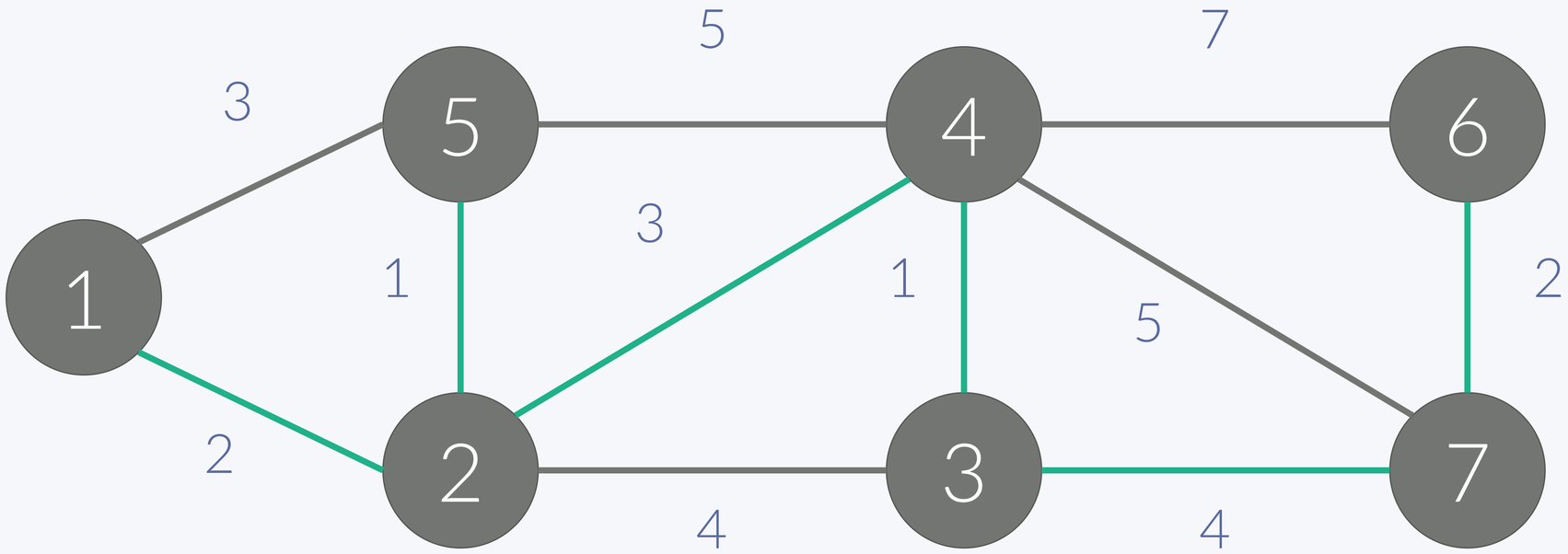
MST

최소 스패닝 트리

Minimum Spanning Tree

정점 V 간선 V-1

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



프림

프림

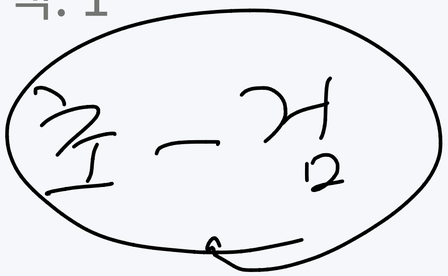
Prim

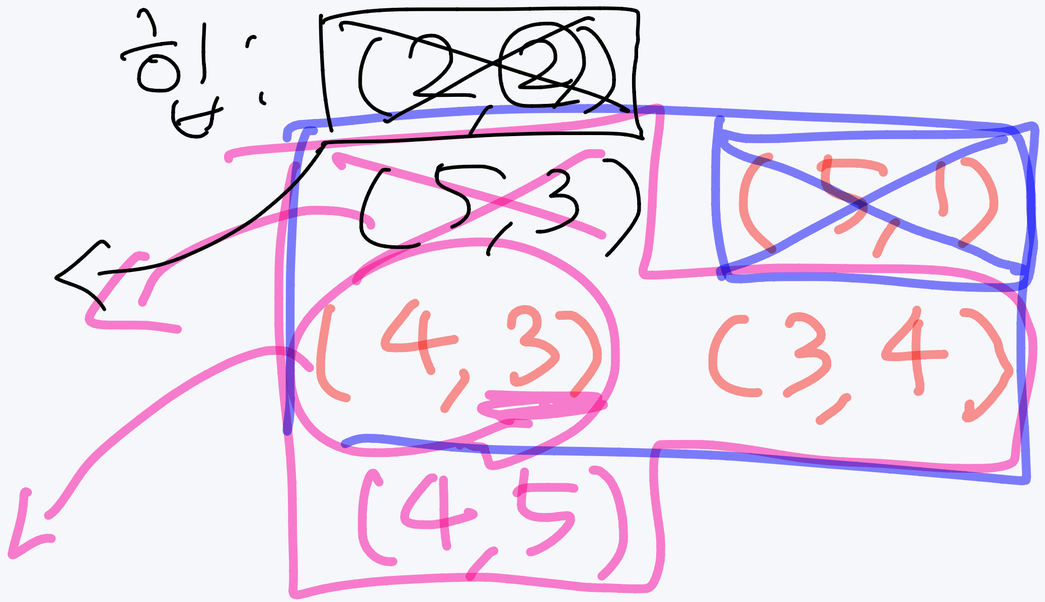
1. 그래프에서 아무 정점이나 선택한다.
2. 선택한 정점과 선택하지 않은 정점을 연결하는 간선중에 최소값을 고른다. 이 간선을 (u, v)라고 한다. (u = 선택, v = 선택하지 않음)
3. 선택한 간선을 MST에 추가하고, v를 선택한다.
4. 모든 정점선택하지 않았다면, 2번 단계로 돌아간다.

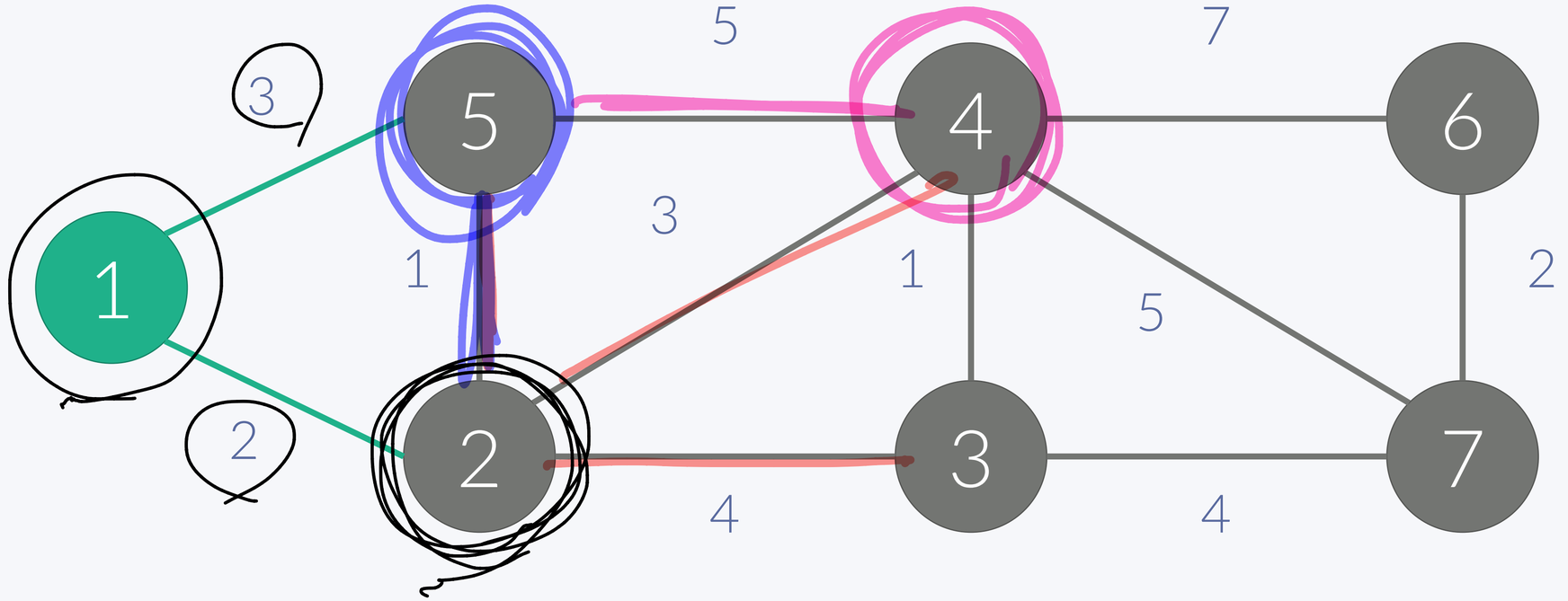
프림

Prim

- 선택: 1



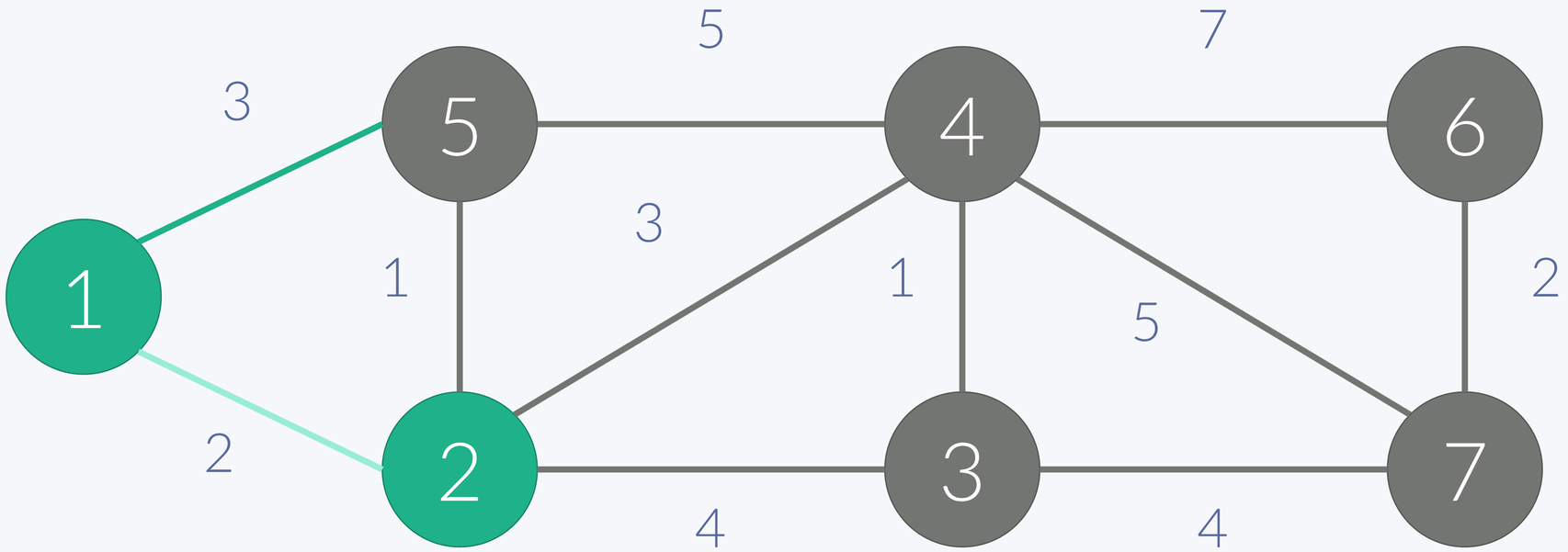




프림

Prim

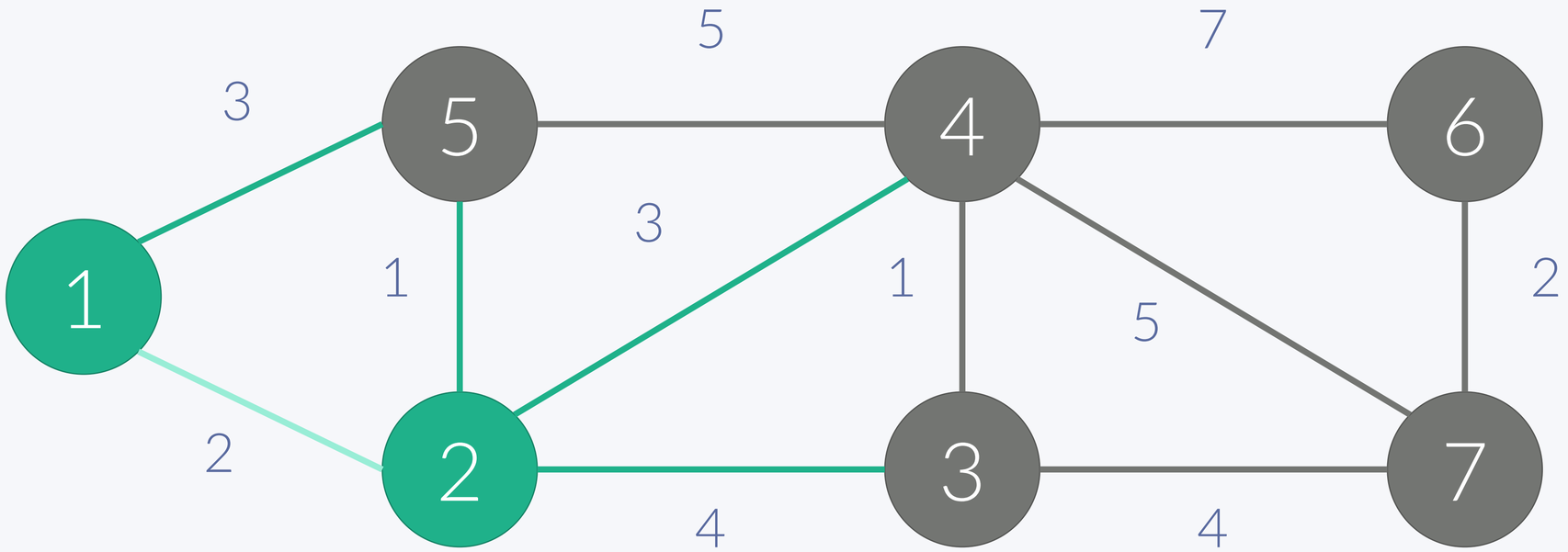
- 선택: 1 2



프림

Prim

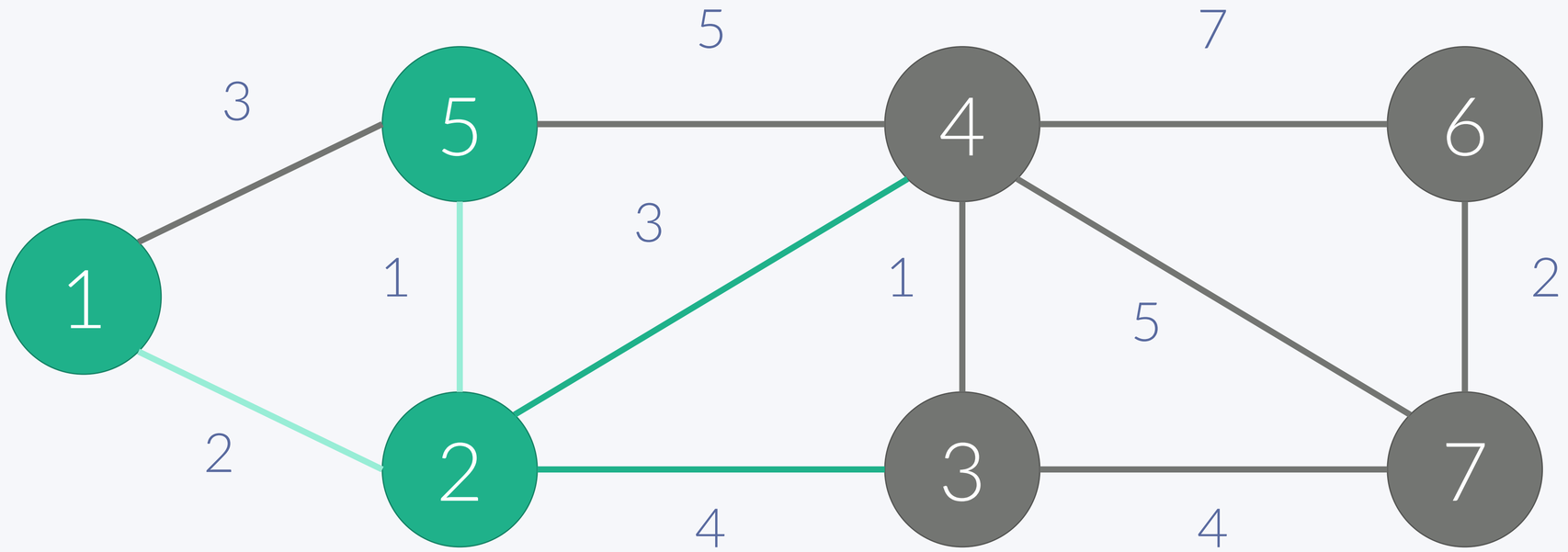
- 선택: 1 2



프림

Prim

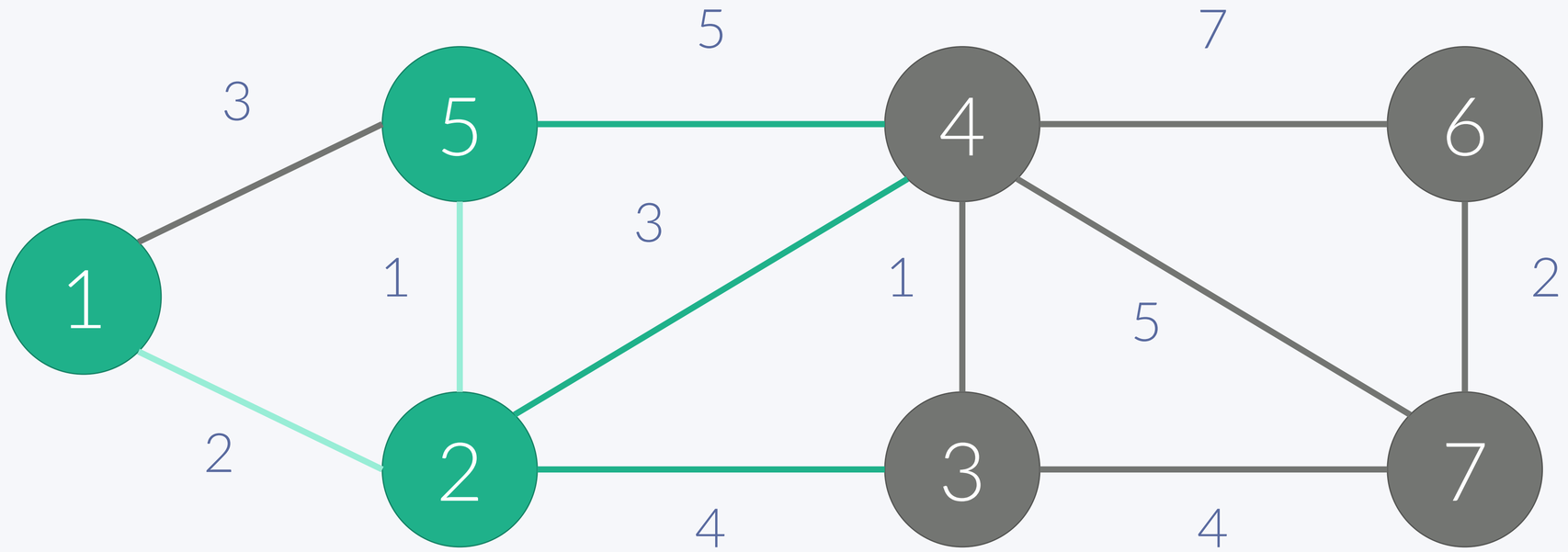
- 선택: 1 2 5



프림

Prim

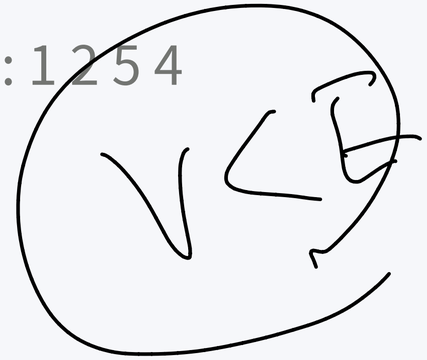
- 선택: 1 2 5



프림

Prim

- 선택: 1 2 5 4

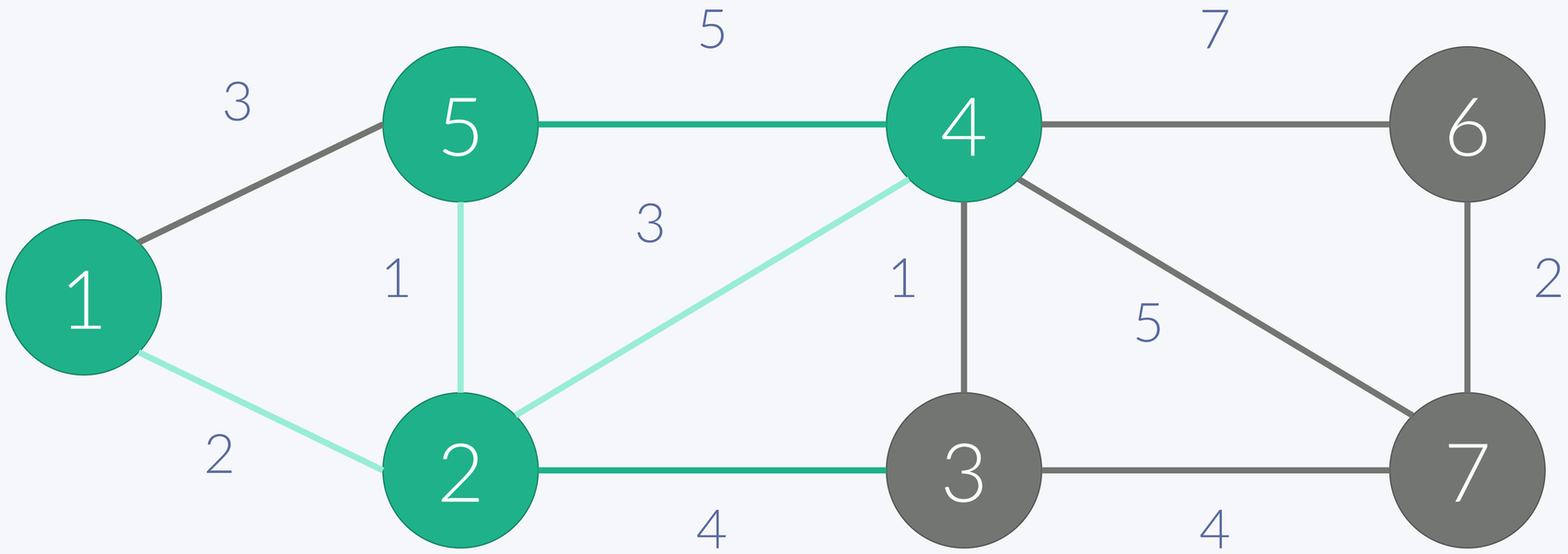


O(VE)

Heap

BST

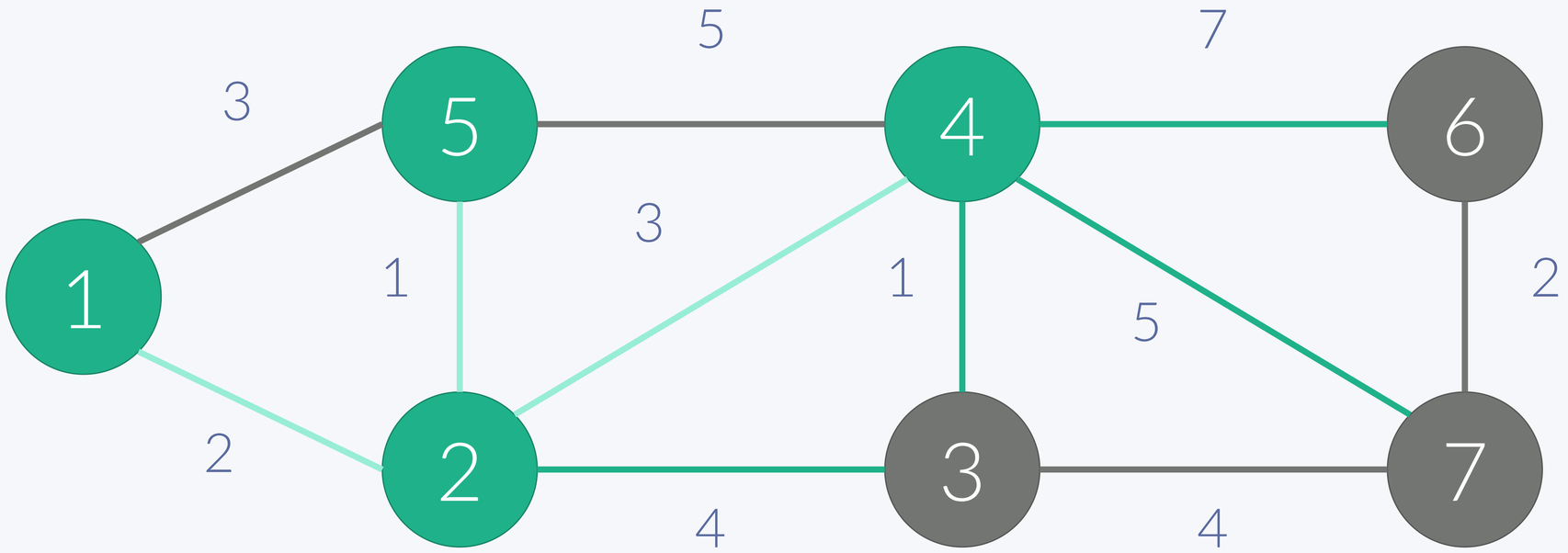
O(ElogE) / O(ElgV)



프림

Prim

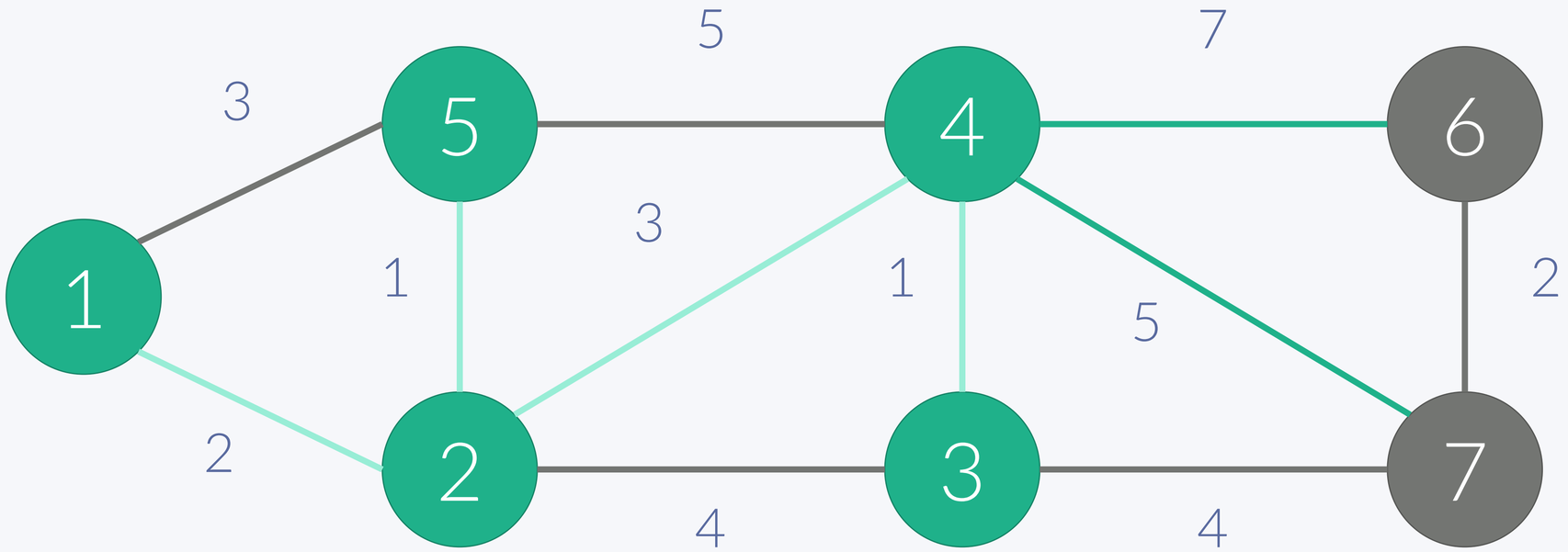
- 선택: 1 2 5 4



프림

Prim

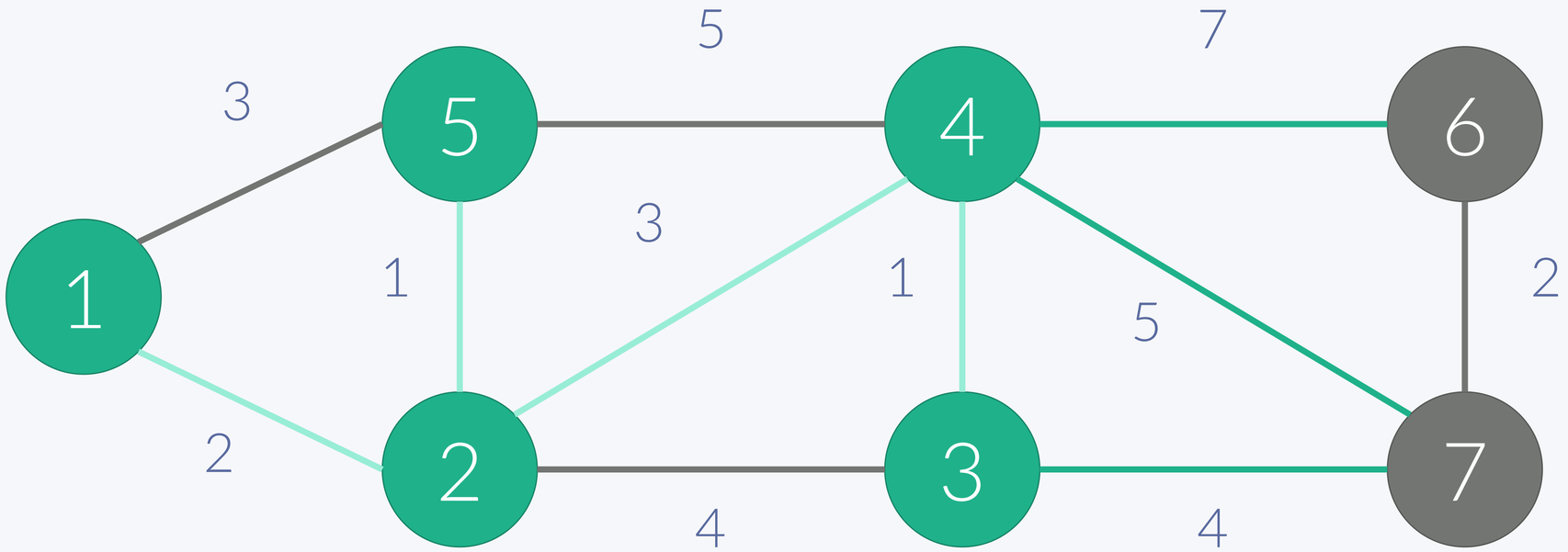
- 선택: 1 2 5 4 3



프림

Prim

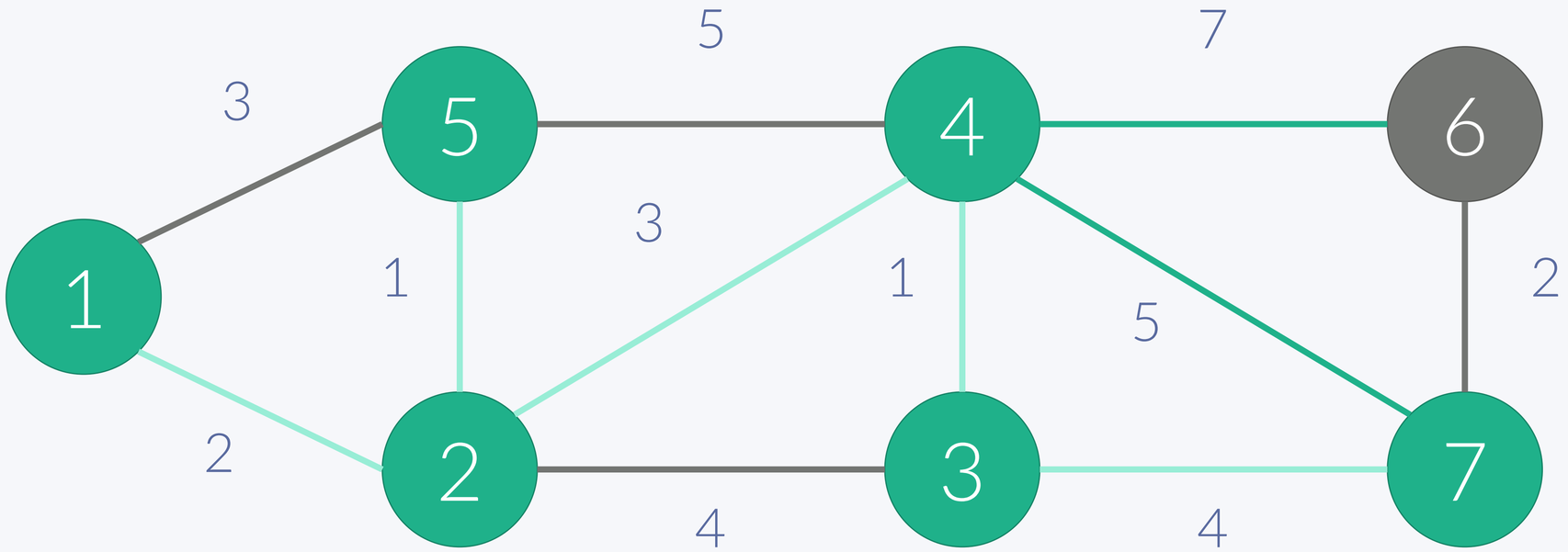
- 선택: 1 2 5 4 3



프림

Prim

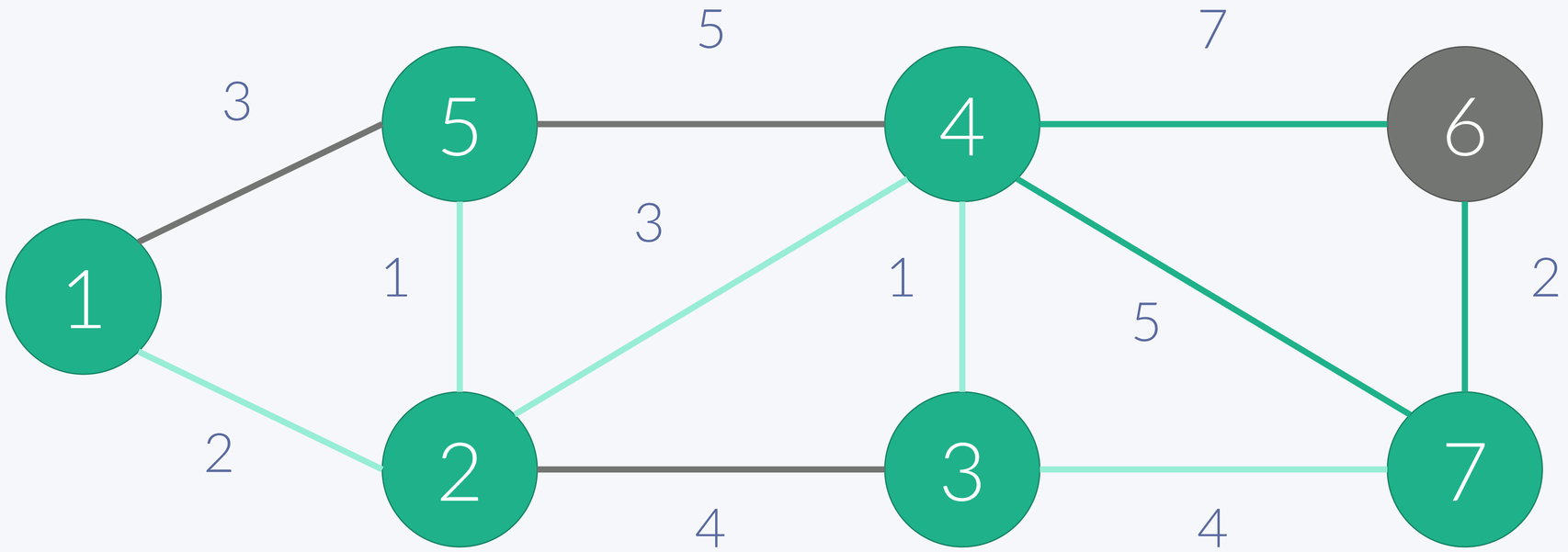
- 선택: 1 2 5 4 3 7



프림

Prim

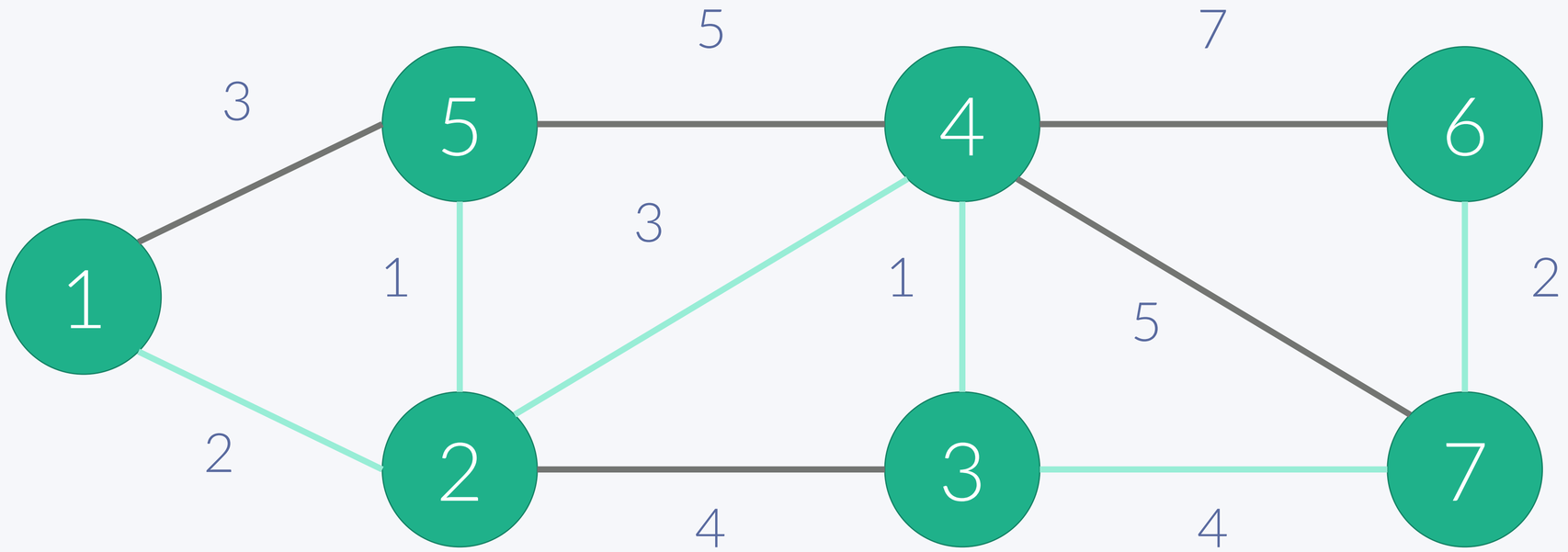
- 선택: 1 2 5 4 3 7



프림

Prim

- 선택: 1 2 5 4 3 7 6



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, v를 선택한다.
4. 모든 정점선택하지 않았다면, 2번 단계로 돌아간다.

- 각각의 정점을 선택하고 모든 간선을 살펴봐야 한다.
- 시간 복잡도: O(V*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, v를 선택한다.
4. 모든 정점선택하지 않았다면, 2번 단계로 돌아간다.

- 최소값을 우선 순위 큐를 이용하면 최소값을 lgE만에 찾을 수 있다
- 시간 복잡도: O(ElgE)

네트워크 연결

Prim

- 그래프가 주어졌을 때, 그 그래프의 최소 스패닝 트리를 구하기

네트워크 연결

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

- 소스: http://boj.kr/67533cb9eb684081856eb6a87b923bce

MST

트리: 사이클 X

Union Find

크루스칼

크루스칼

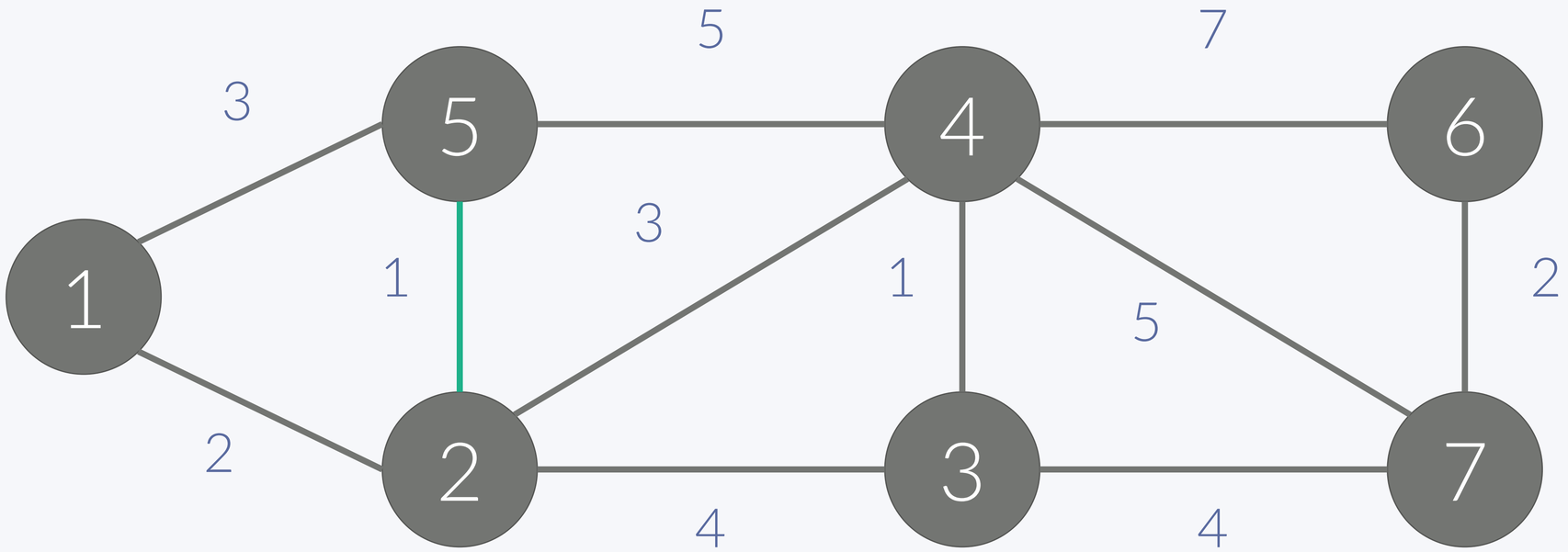
Kruskal

- 가중치가 작은 Edge부터 순서대로 살펴본다.

- Edge e가 (u, v, c) 일 때
- u와 v가 다른 집합이면 e를 MST에 추가한다

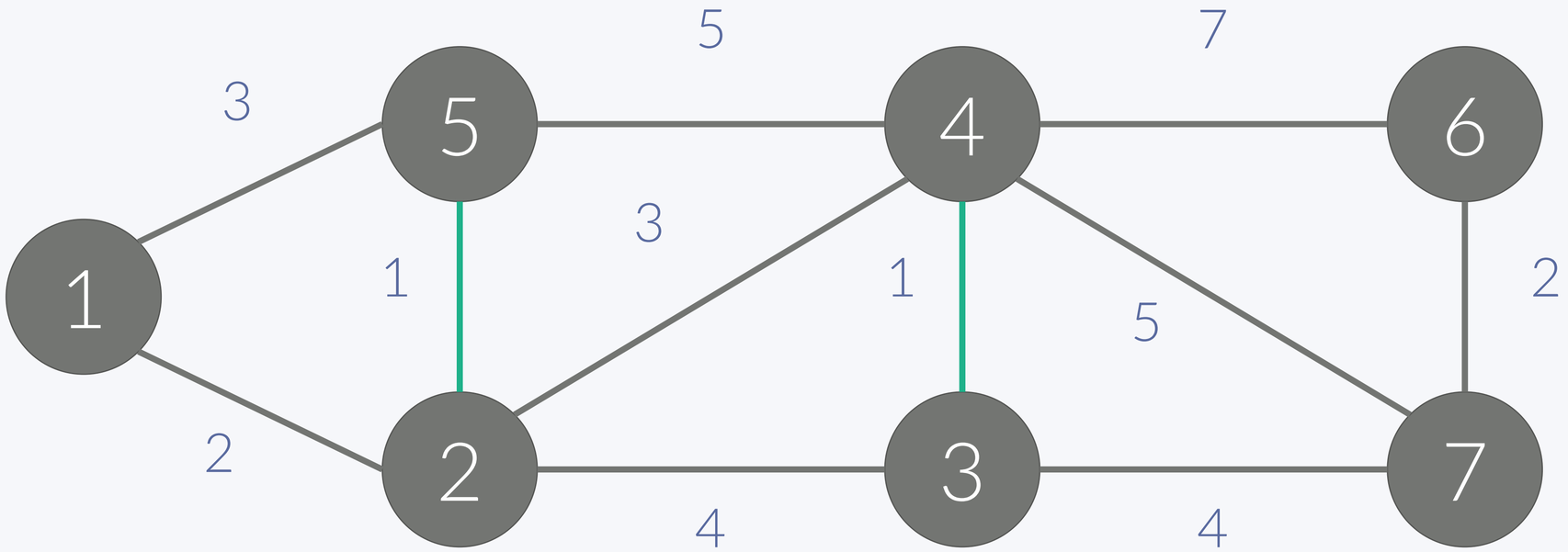
크루스칼

Kruskal



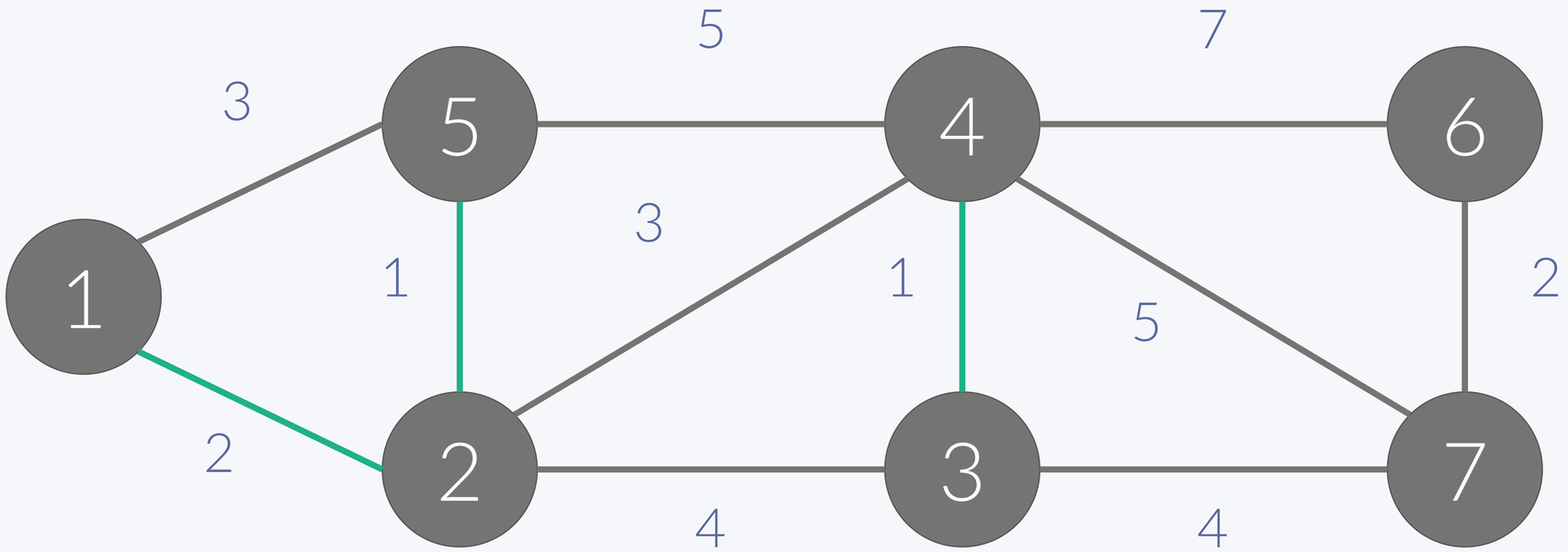
크루스칼

Kruskal



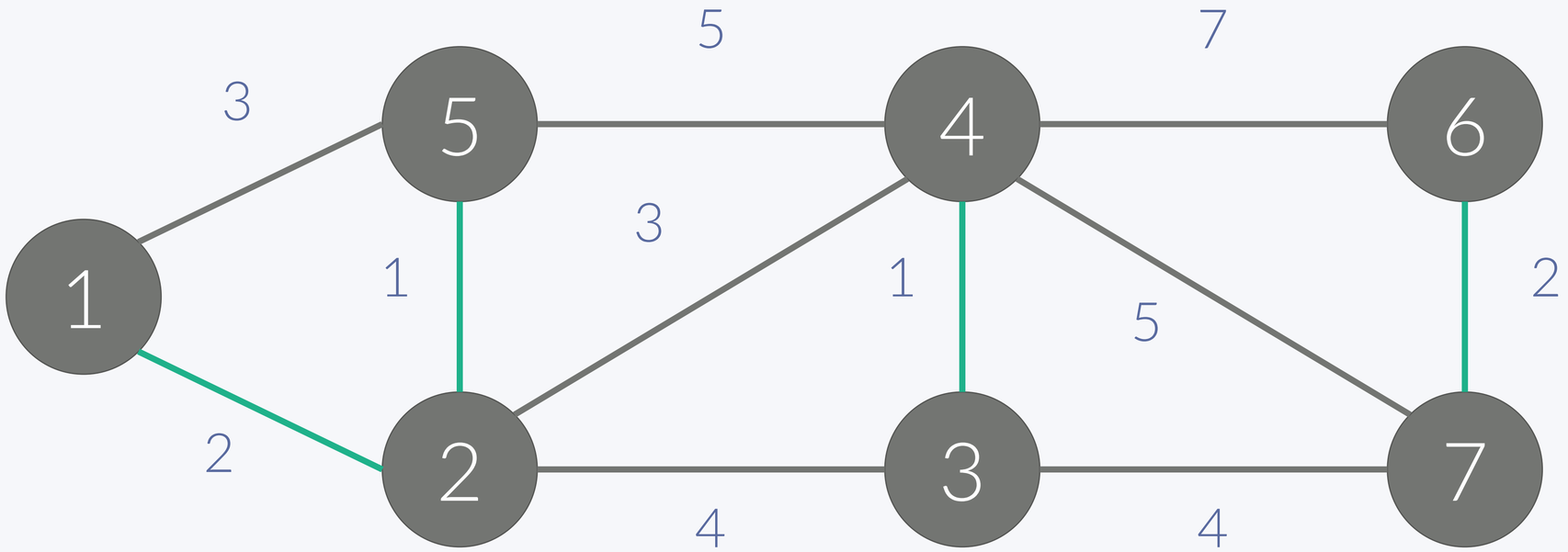
크루스칼

Kruskal



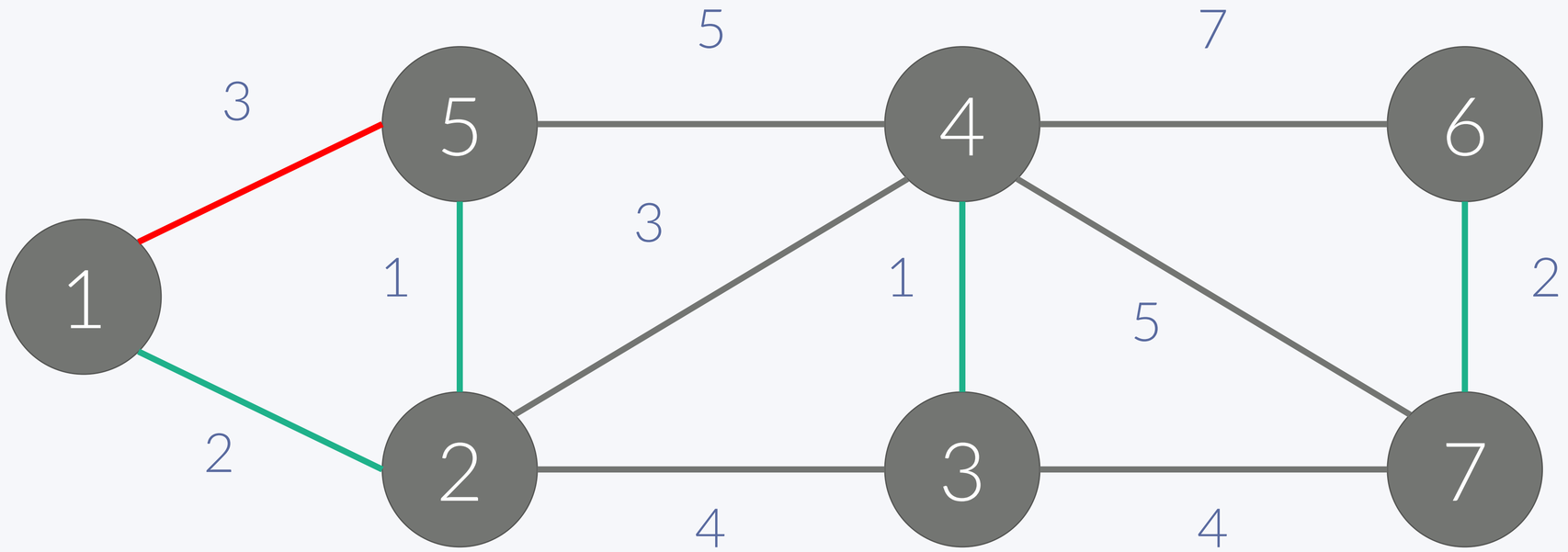
크루스칼

Kruskal



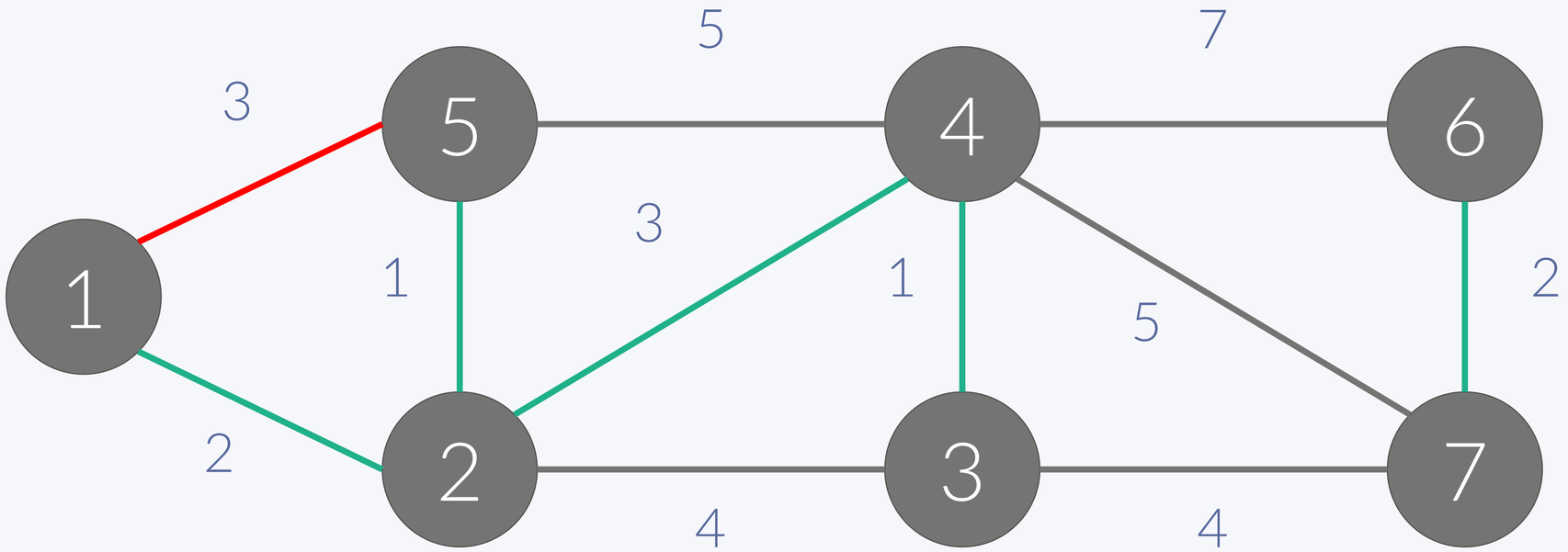
크루스칼

Kruskal



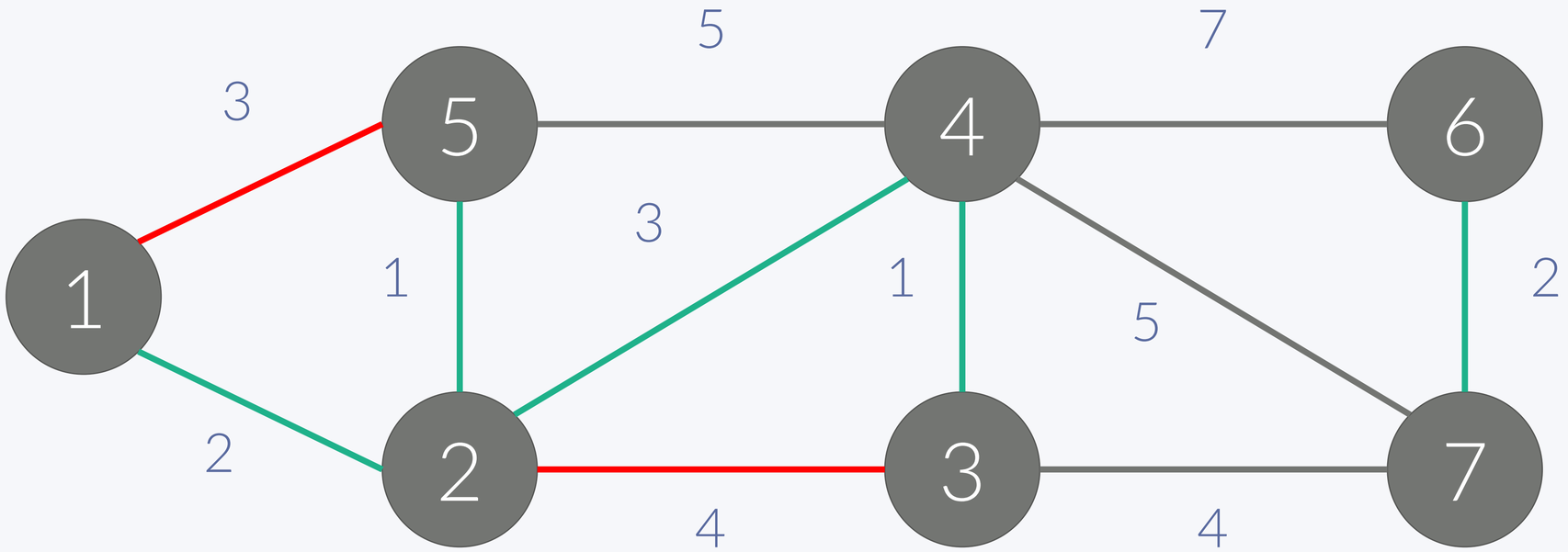
크루스칼

Kruskal



크루스칼

Kruskal

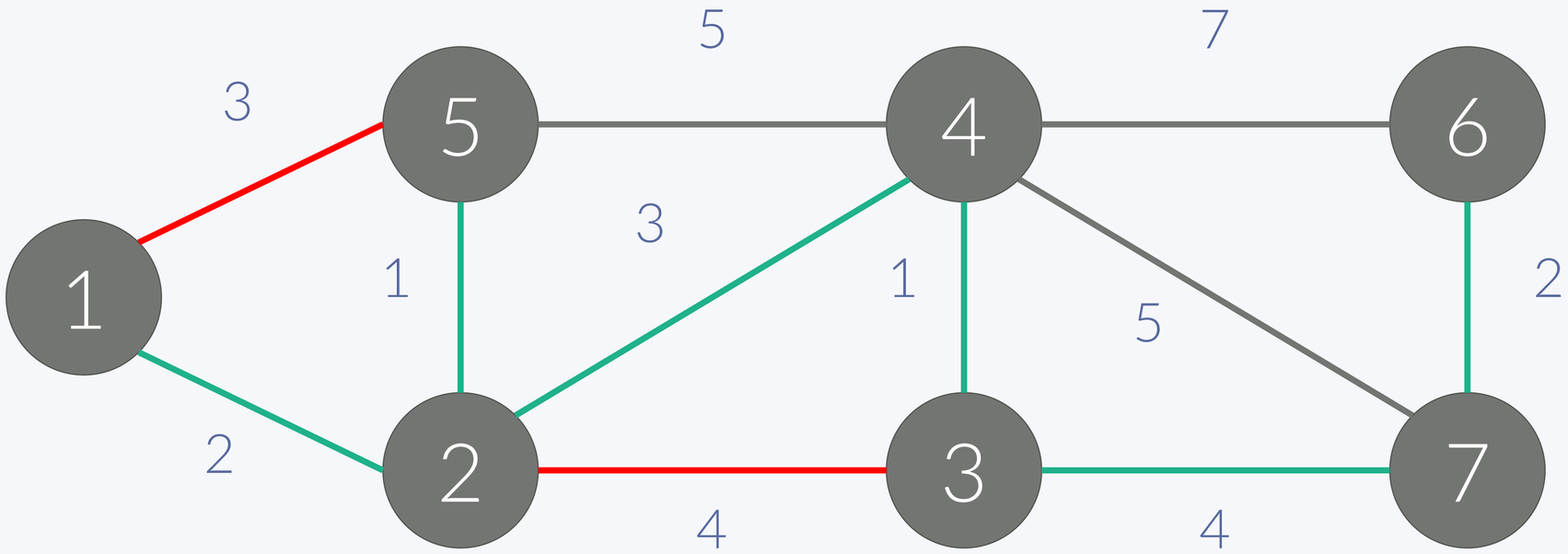


크루스칼

Kruskal

시간복잡도: O(E)

O(ElgE)



최소 스패닝 트리

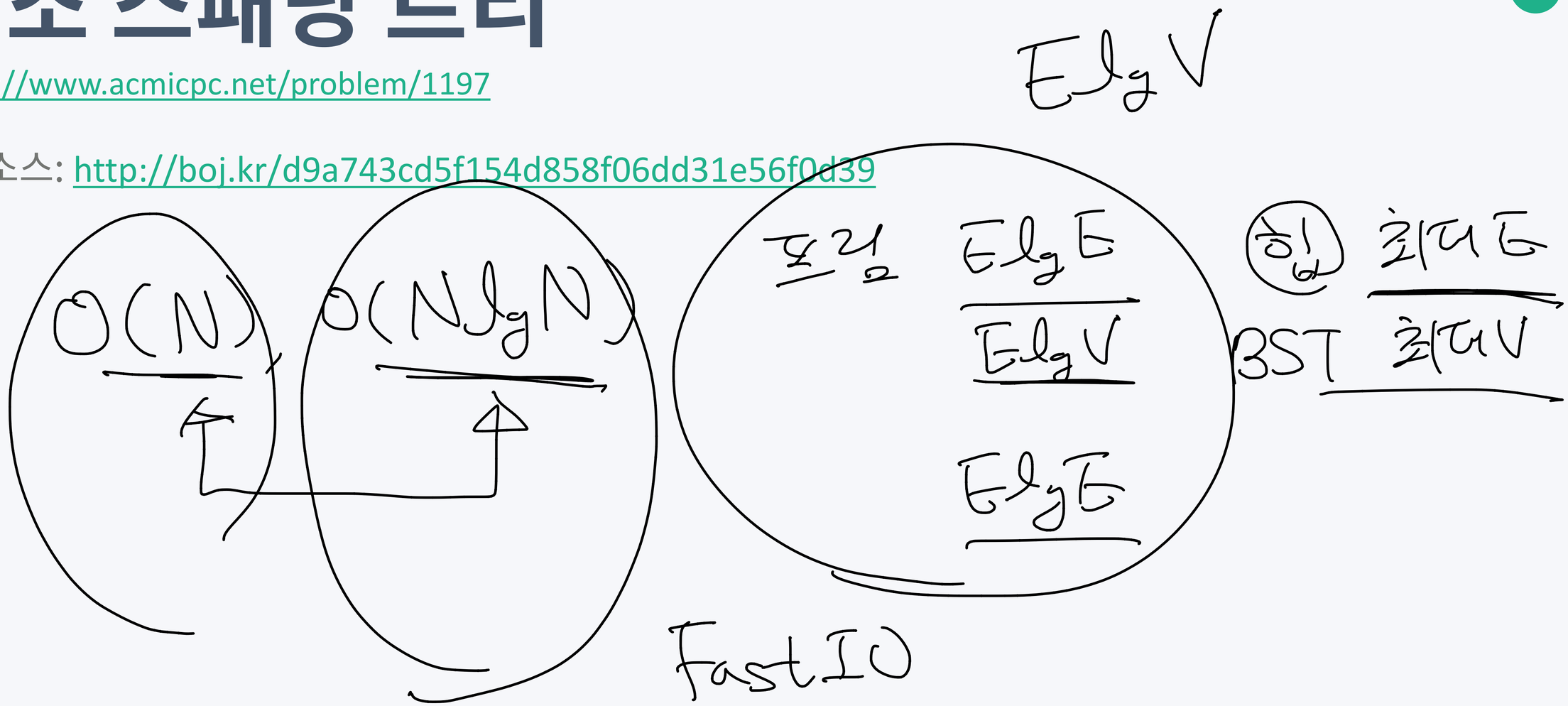
https://www.acmicpc.net/problem/1197

- 그래프가 주어졌을 때, 그 그래프의 최소 스패닝 트리를 구하기

최소 스패닝 트리

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

- 소스: http://boj.kr/d9a743cd5f154d858f06dd31e56f0d39



① 시작점 1개

② 시작점 V개 = ①xV번

플로이드 V^3

V, E

최단 경로

가중치 1 : BFS

트리 : DFS, BFS

음수가중치O: 벨만포드 VE

음수가중치X: 다익스트라

시간복잡도 -

V^2

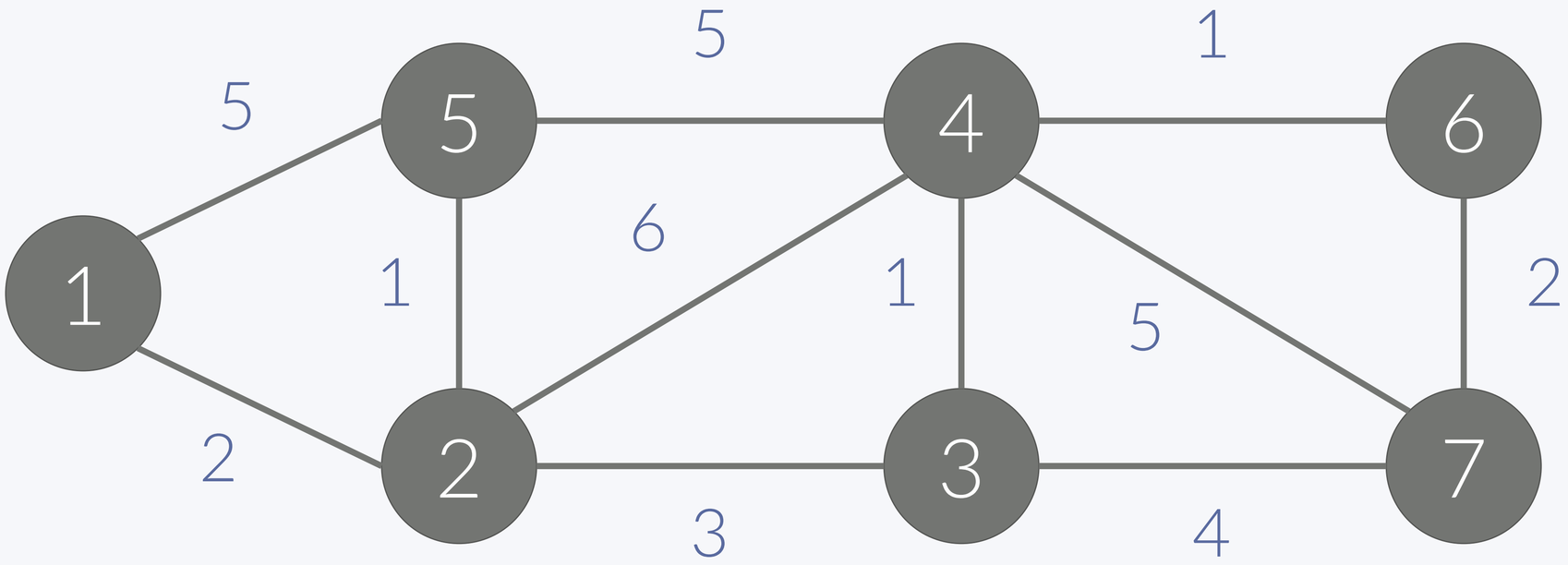
E log E

-> E log V

최단 경로

Single Source Shortest Path

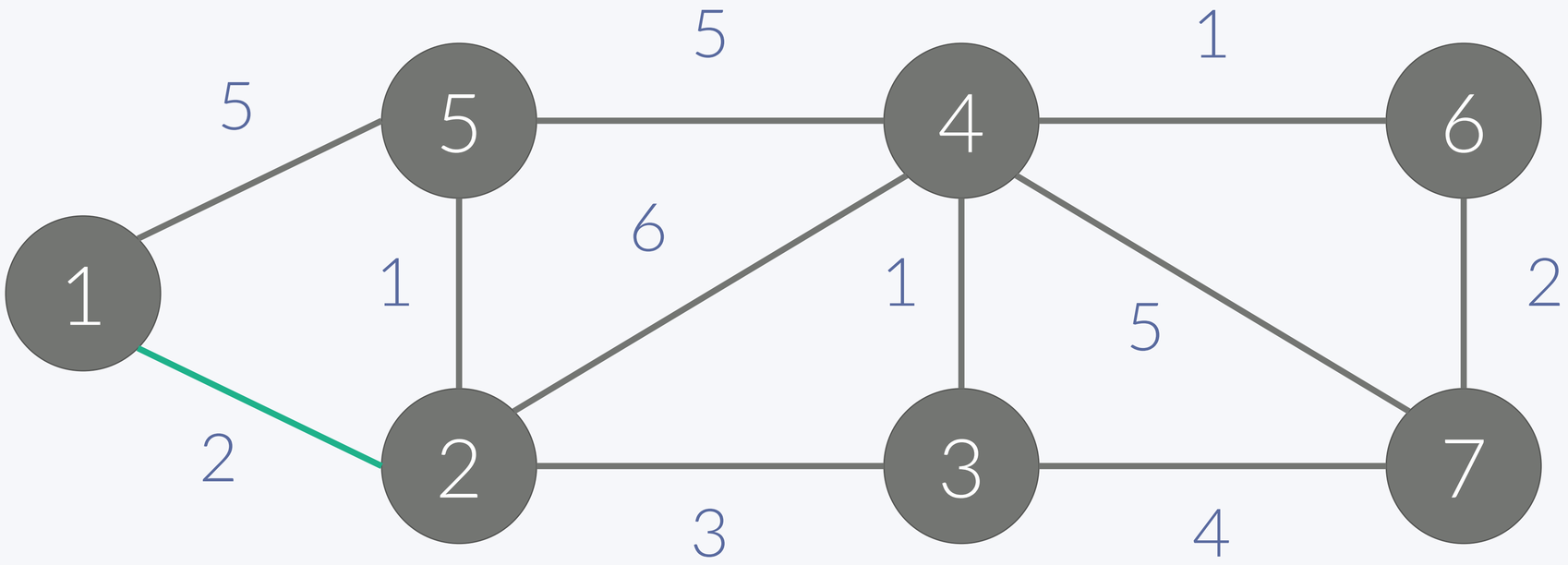
- 시작점이 1개일 때, 다른 모든 곳으로 가는 최단 경로 구하기



최단 경로

Single Source Shortest Path

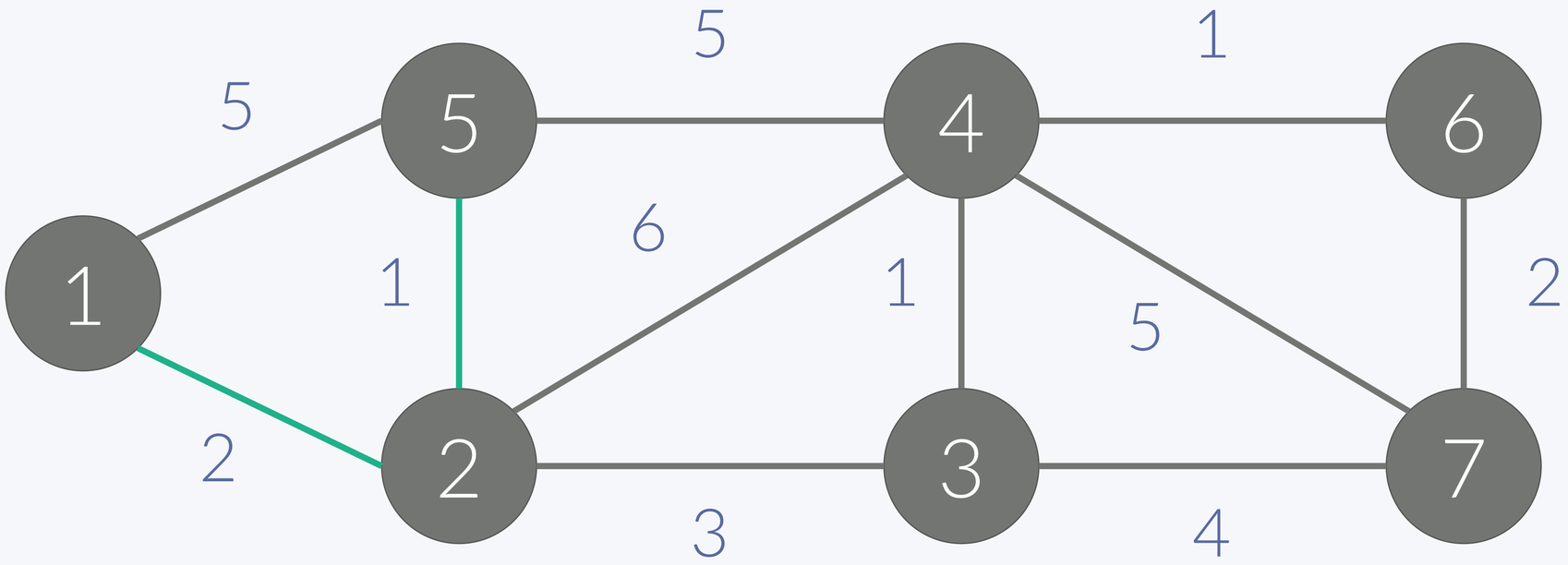
- 1에서 2까지 최단 경로



최단 경로

Single Source Shortest Path

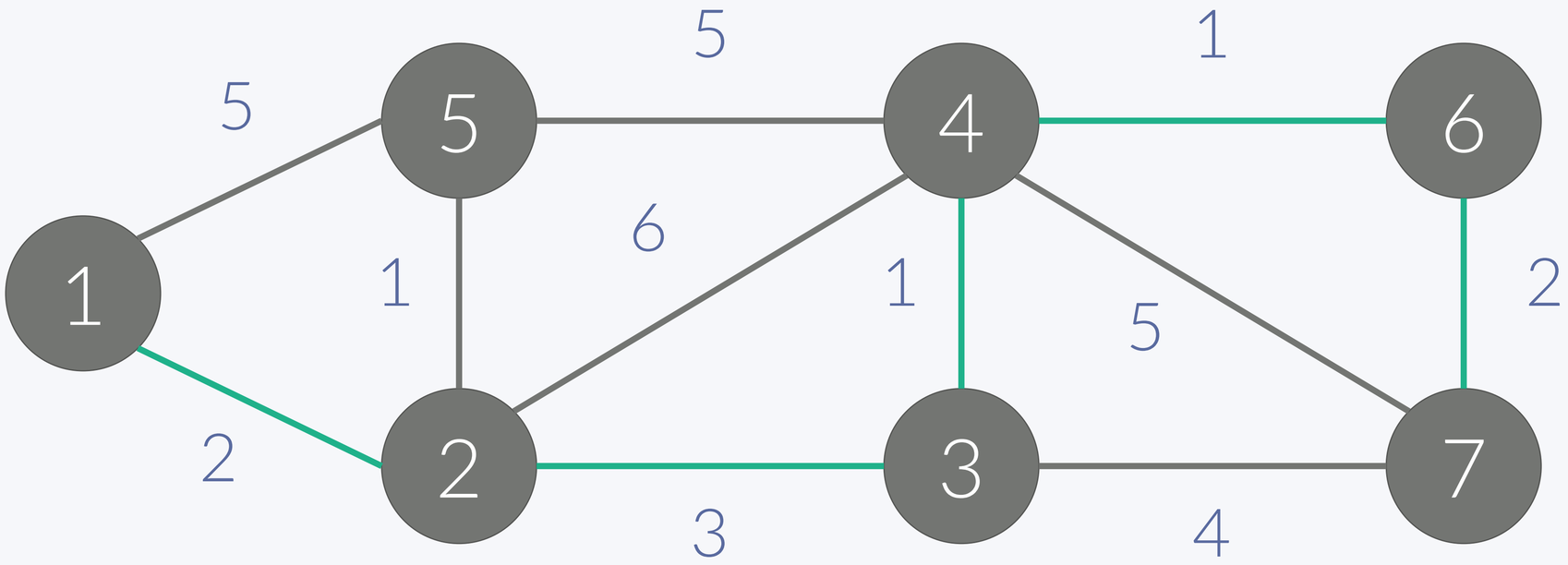
- 1에서 5까지 최단 경로



최단 경로

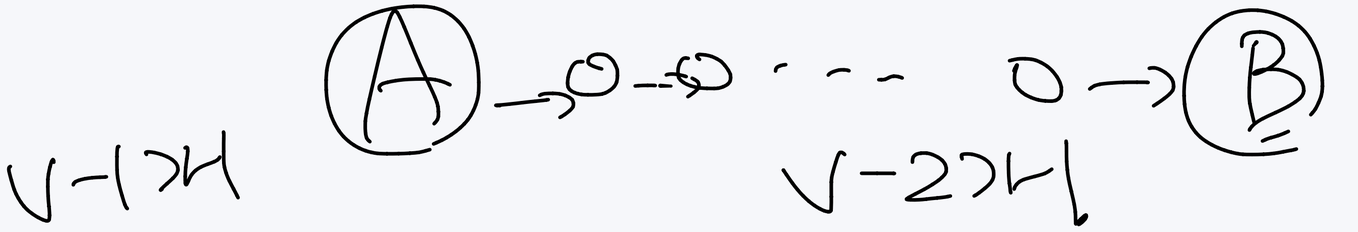
Single Source Shortest Path

- 1에서 7까지 최단 경로

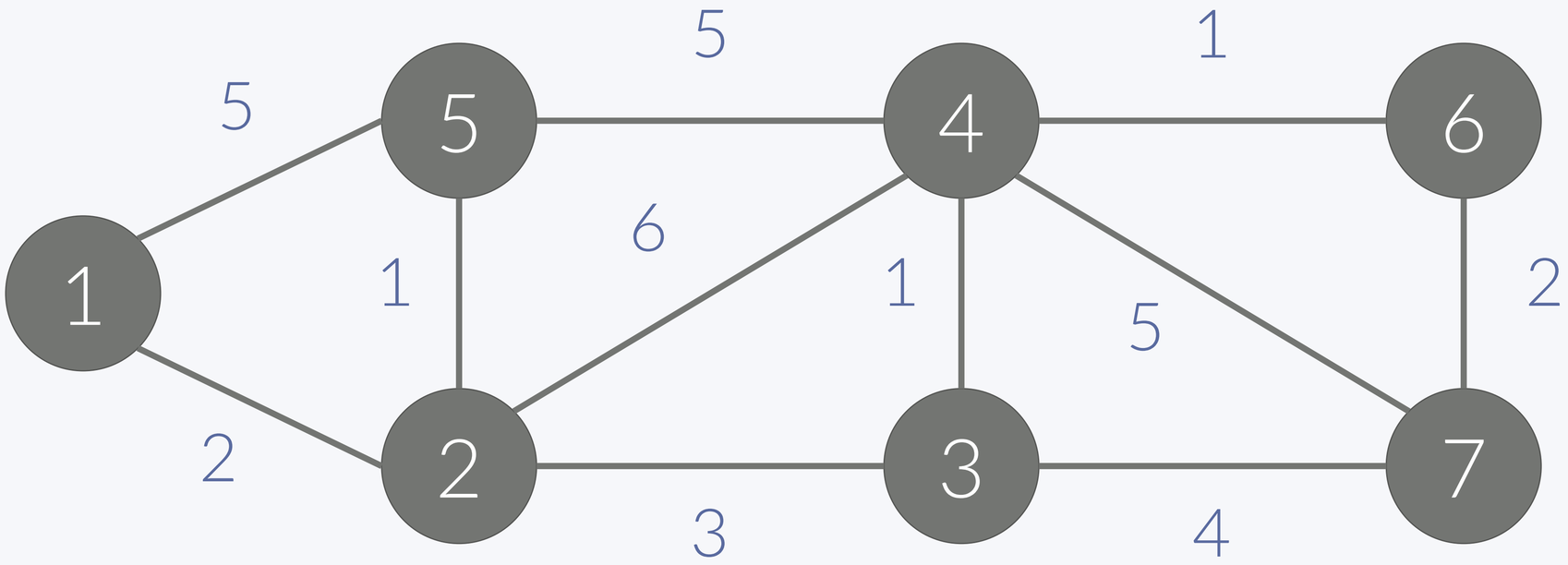


최단 경로

Single Source Shortest Path



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



벨만포드

X(V-1) 번

벨만포드

Bellman-Ford Algorithm

- dist[i] = 시작점에서 i로 가는 최단경로

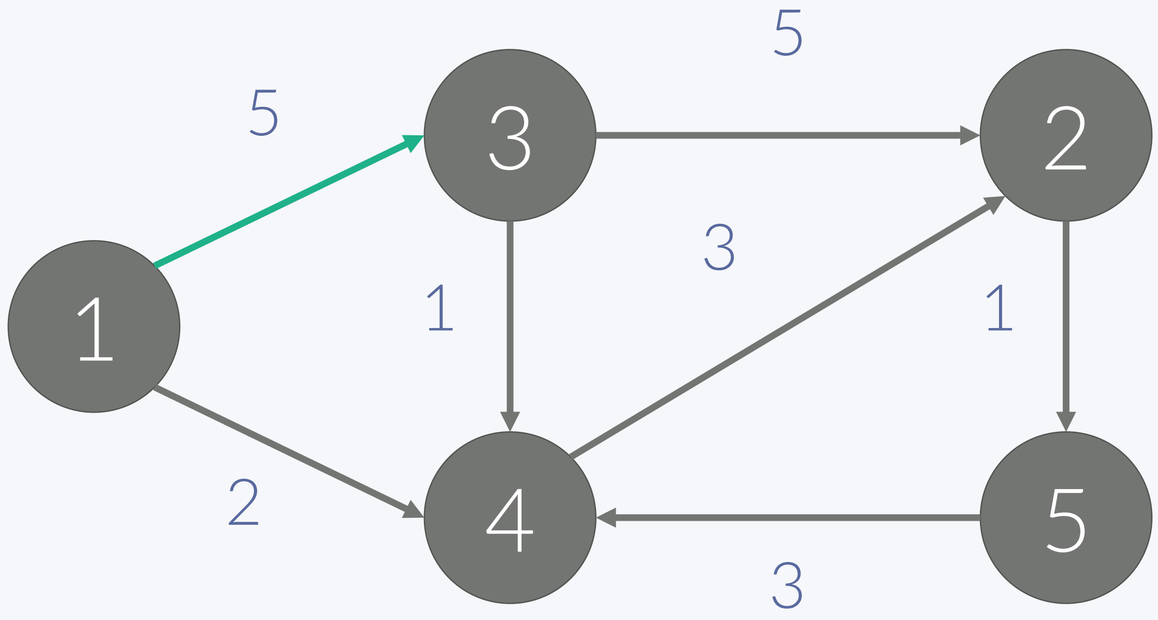
1. 모든 간선 e (u,v,c)에 대해서 다음을 검사한다.
    - dist[v] = Min(dist[v], dist[u] + c)

- 1번 과정을 총 N-1번 반복한다.

벨만포드

Bellman-Ford Algorithm

- 1단계

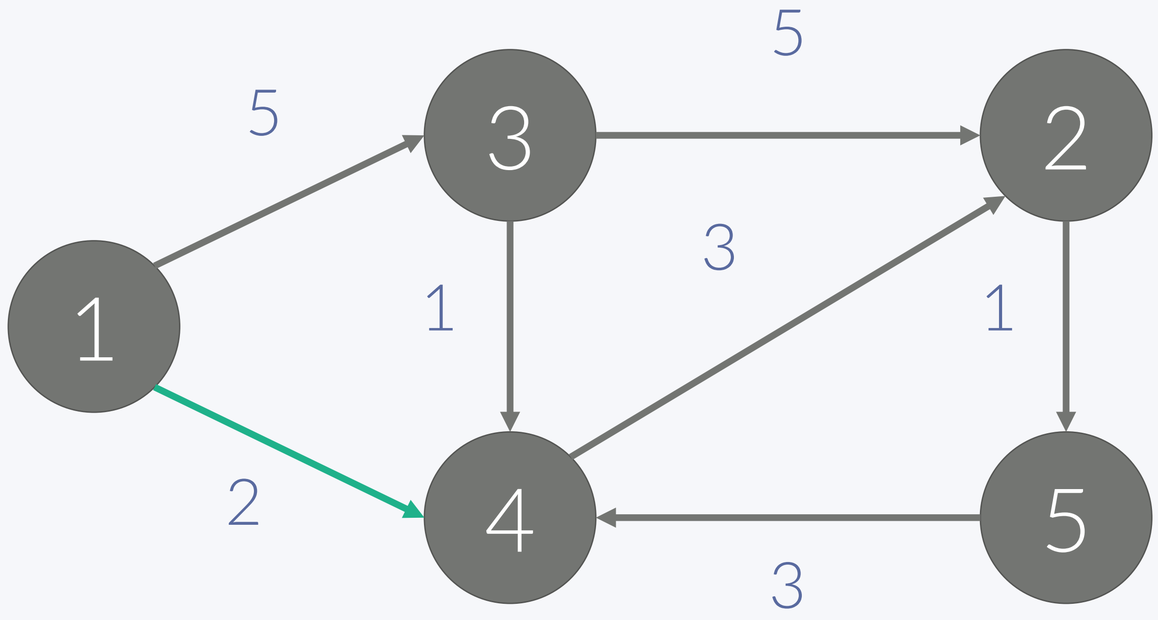


i	1	2	3	4	5
dist[i]	0	∞	5	∞	∞

벨만포드

Bellman-Ford Algorithm

- 1단계

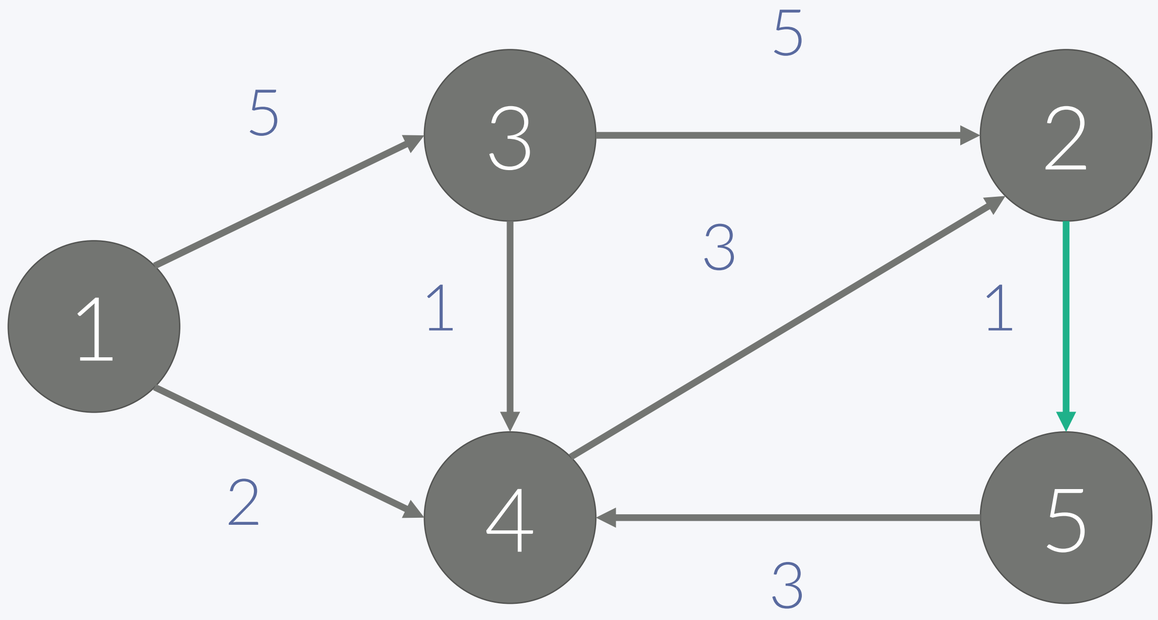


i	1	2	3	4	5
dist[i]	0	∞	5	2	∞

벨만포드

Bellman-Ford Algorithm

- 1단계

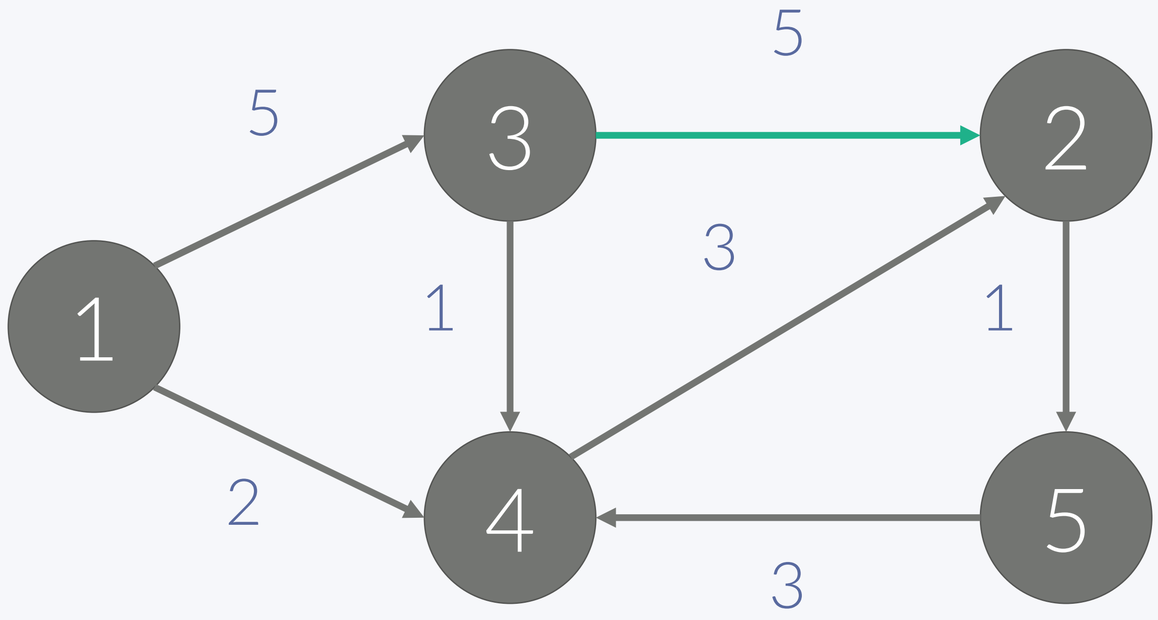


i	1	2	3	4	5
dist[i]	0	∞	5	2	∞

벨만포드

Bellman-Ford Algorithm

- 1단계

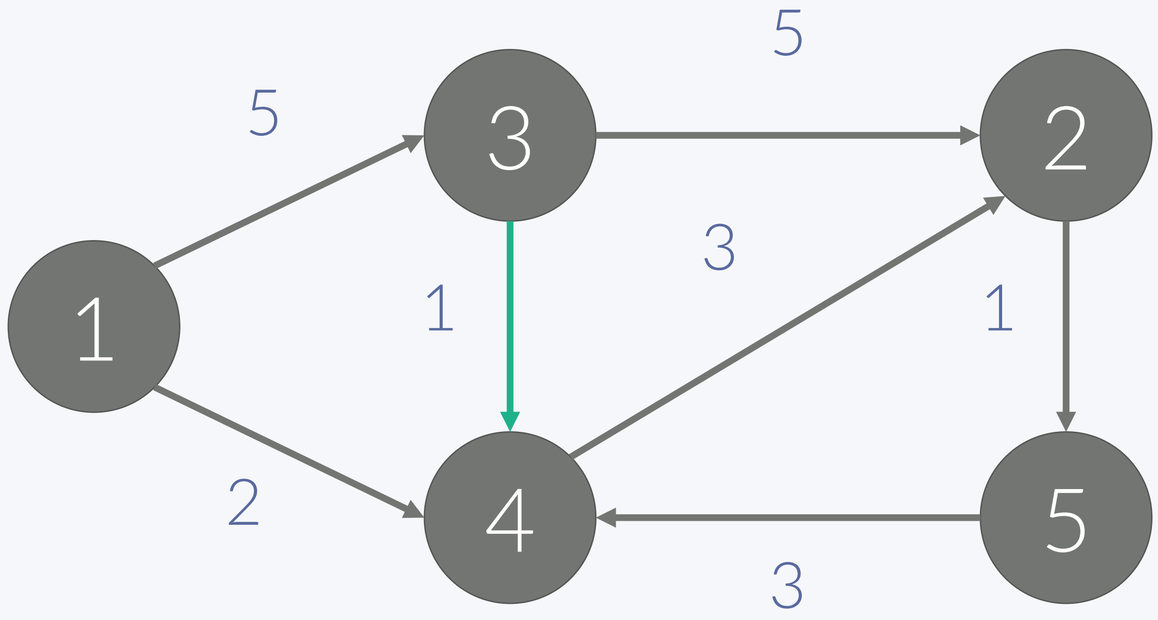


i	1	2	3	4	5
dist[i]	0	10	5	2	∞

벨만포드

Bellman-Ford Algorithm

- 1단계

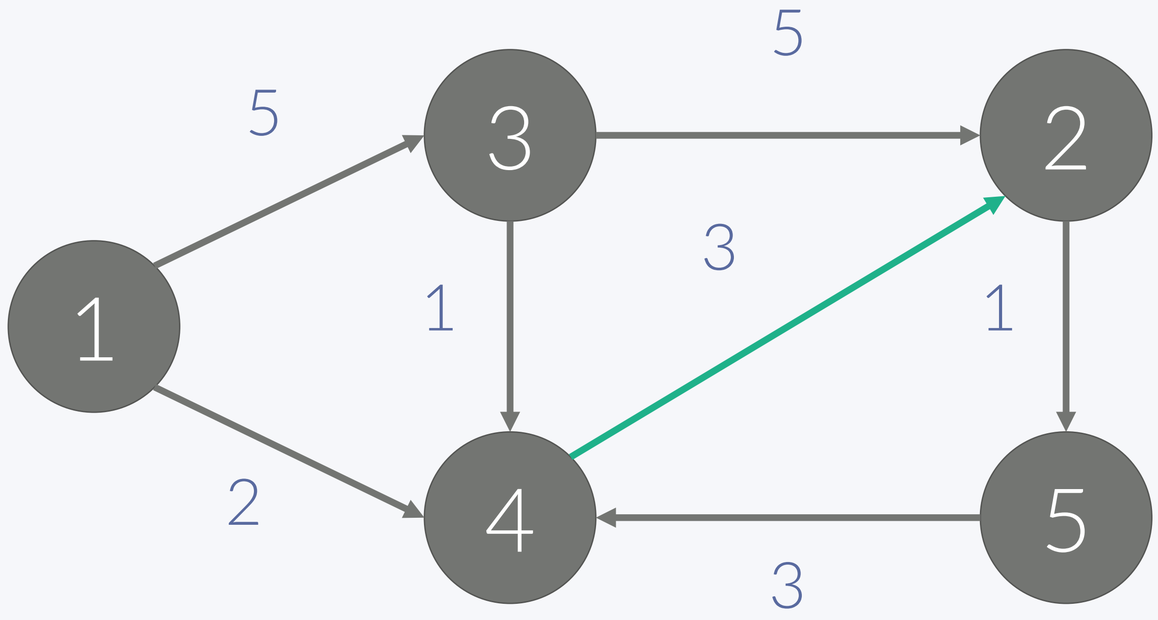


i	1	2	3	4	5
dist[i]	0	10	5	2	∞

벨만포드

Bellman-Ford Algorithm

- 1단계

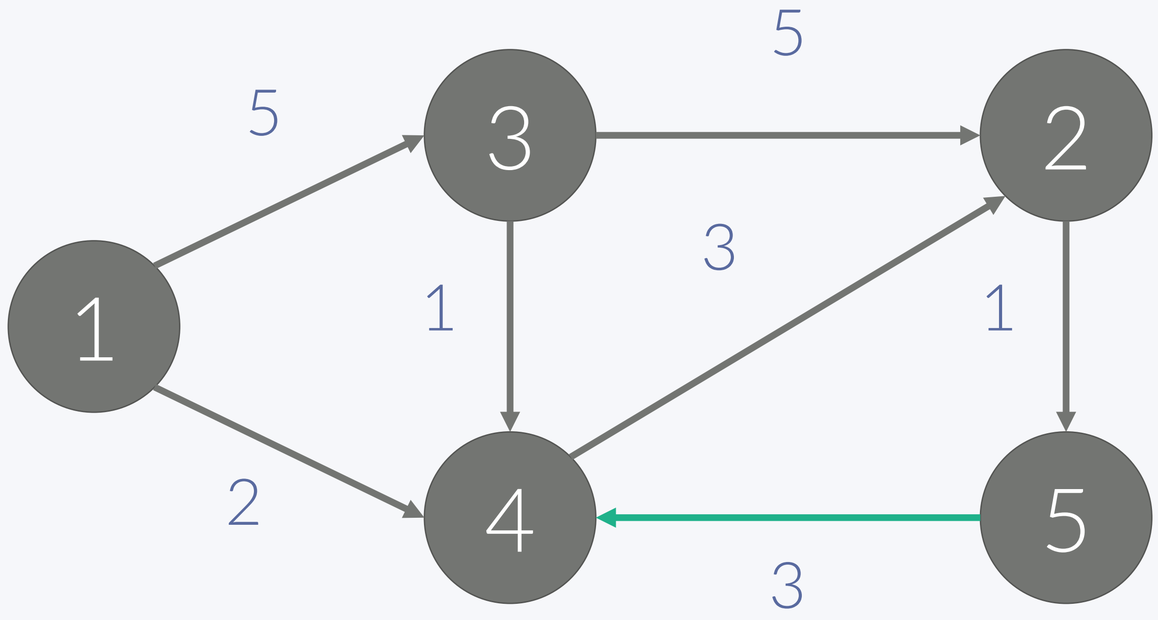


i	1	2	3	4	5
dist[i]	0	5	5	2	∞

벨만포드

Bellman-Ford Algorithm

- 1단계

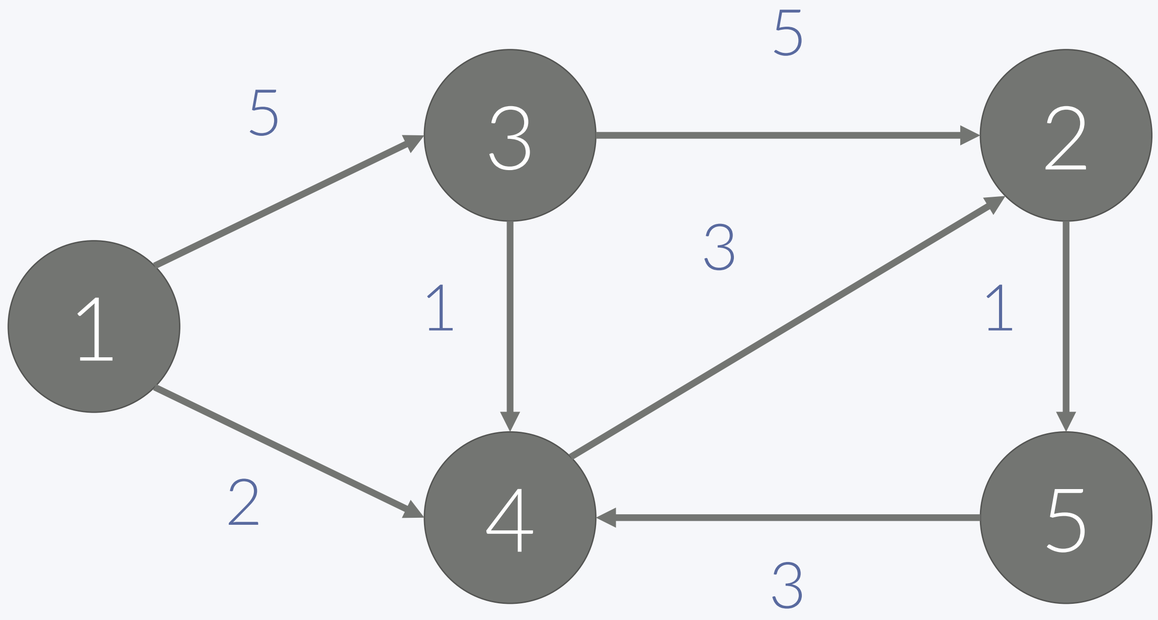


i	1	2	3	4	5
dist[i]	0	5	5	2	∞

벨만포드

Bellman-Ford Algorithm

- 이런 단계를 N-1번 반복한다

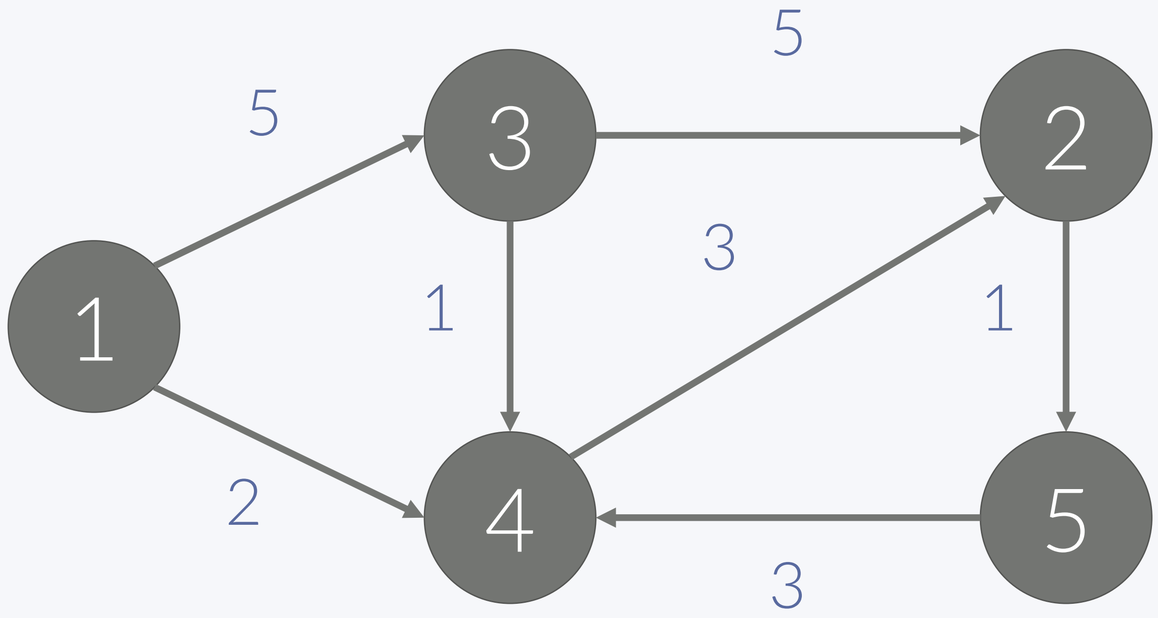


i	1	2	3	4	5
dist[i]	0	5	5	2	∞

벨만포드

Bellman-Ford Algorithm

- 이런 단계를 N-1번 반복한다

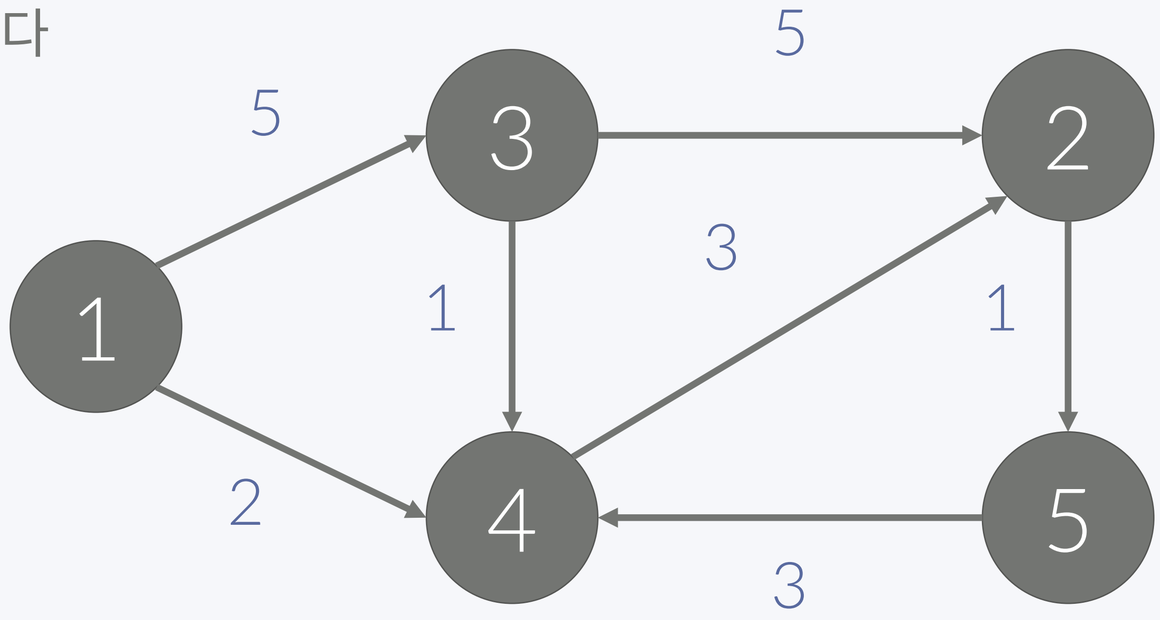


i	1	2	3	4	5
dist[i]	0	5	5	2	6

벨만포드

Bellman-Ford Algorithm

- 시간 복잡도: O(VE)
- E ≤ 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

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 = 0인 경우는 순간 이동을 하는 경우, C < 0인 경우는 타임머신으로 시간을 되돌아가는 경우이다.
- 1번 도시에서 출발해서 나머지 도시로 가는 가장 빠른 시간을 구하는 프로그램을 작성하시오.

타임머신

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

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

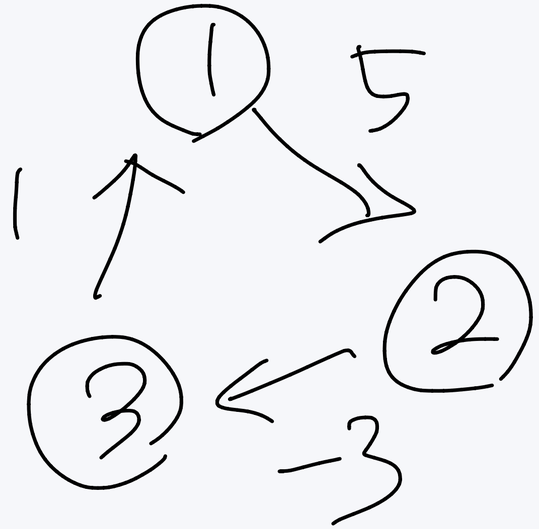
타임머신

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

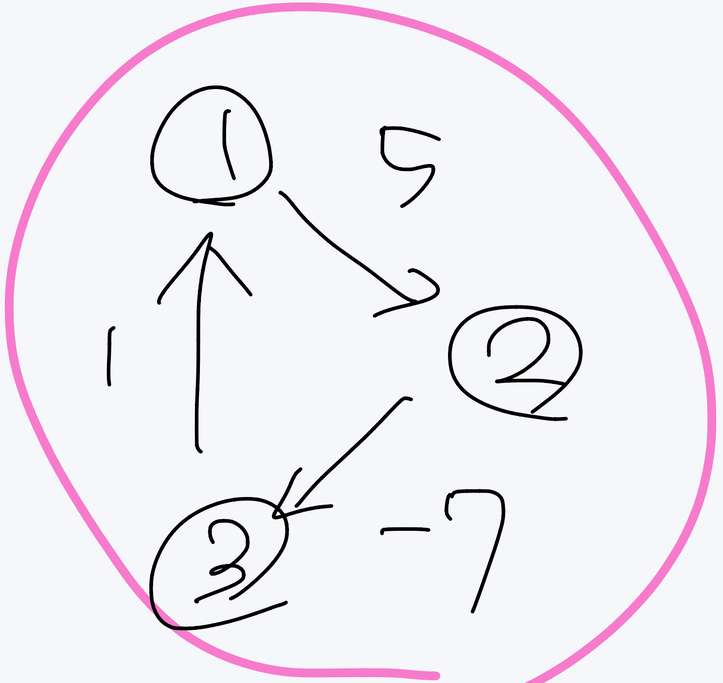
음수 사이클

1 ~> 3

- 소스: http://boj.kr/3791ec5dab67436cad788d914dfee0ee



거리: 2



거리: -∞

웜홀

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

- N개의 지점 사이에는 M개의 도로와 W개의 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

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

V 노드수
E 노드수

- 음수로 되어있는 사이클을 찾는 문제

1 : 모든 간선 조사

2 :

V-1번

결과 : 최단거리

+1번 :

결과 : 같음 (음수 사이클 X), 다름 (있음)

웜홀

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

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

웜홀

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

- 소스: http://boj.kr/a7186d9fd5204f06af8e6db22daf75d1

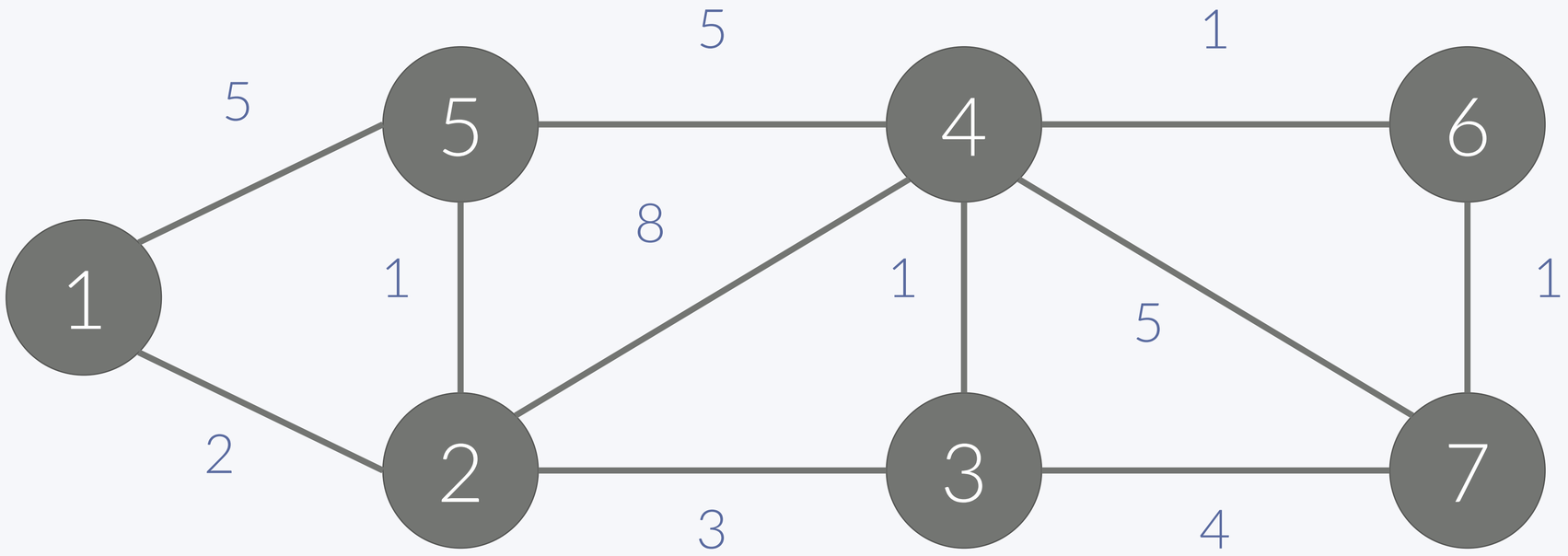
$O(V^2) \rightarrow O(E \log E)$

다익스트라

다익스트라

Dijkstra Algorithm

- dist[i] = 시작 -> i로 가는 최단 거리
- check[i] = i가 체크되어 있으면 true, 아니면 false



다익스트라

Dijkstra Algorithm

1. 체크되어 있지 않은 정점 중에서 dist의 값이 가장 작은 정점 x를 선택한다.
2. x를 체크한다.
3. x와 연결된 모든 정점을 검사한다.
    - 간선을 (x, y, z)라고 했을 때
    - dist[y] > dist[x] + z 이면 갱신해준다.

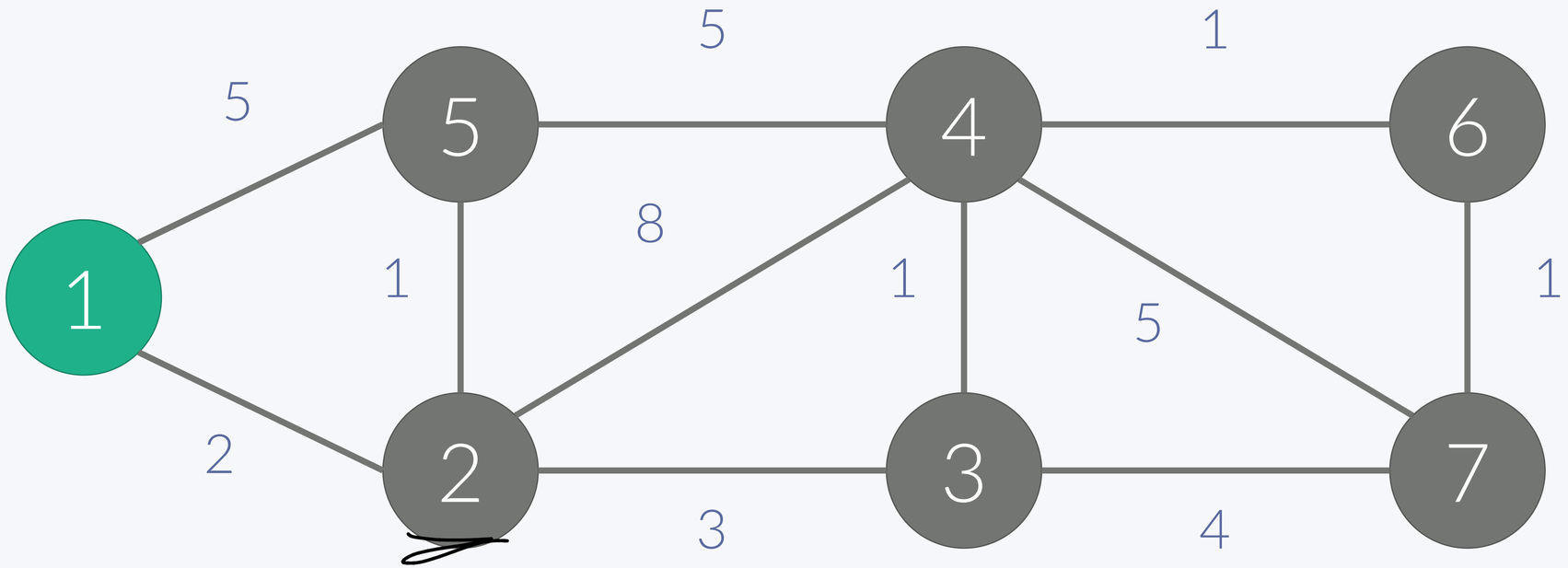
- 1, 2, 3단계를 모든 정점을 체크할 때까지 계속한다.

다익스트라

Dijkstra Algorithm

- 선택: 1

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

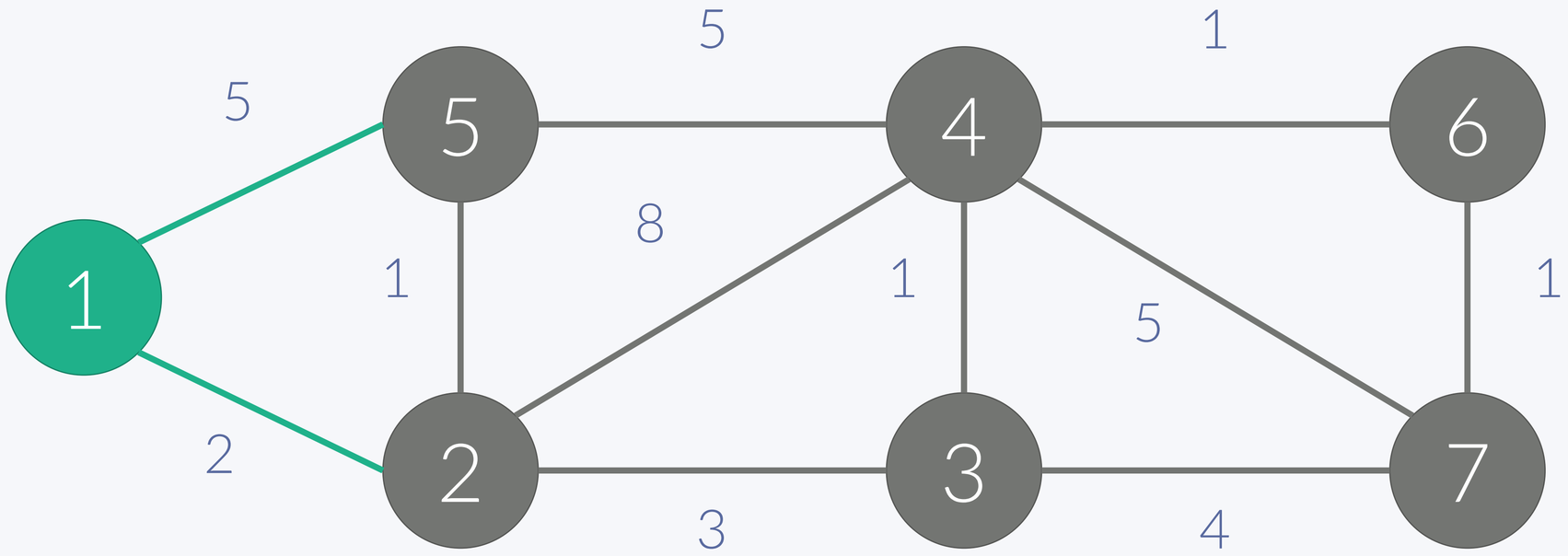


다익스트라

Dijkstra Algorithm

- 선택: 1

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

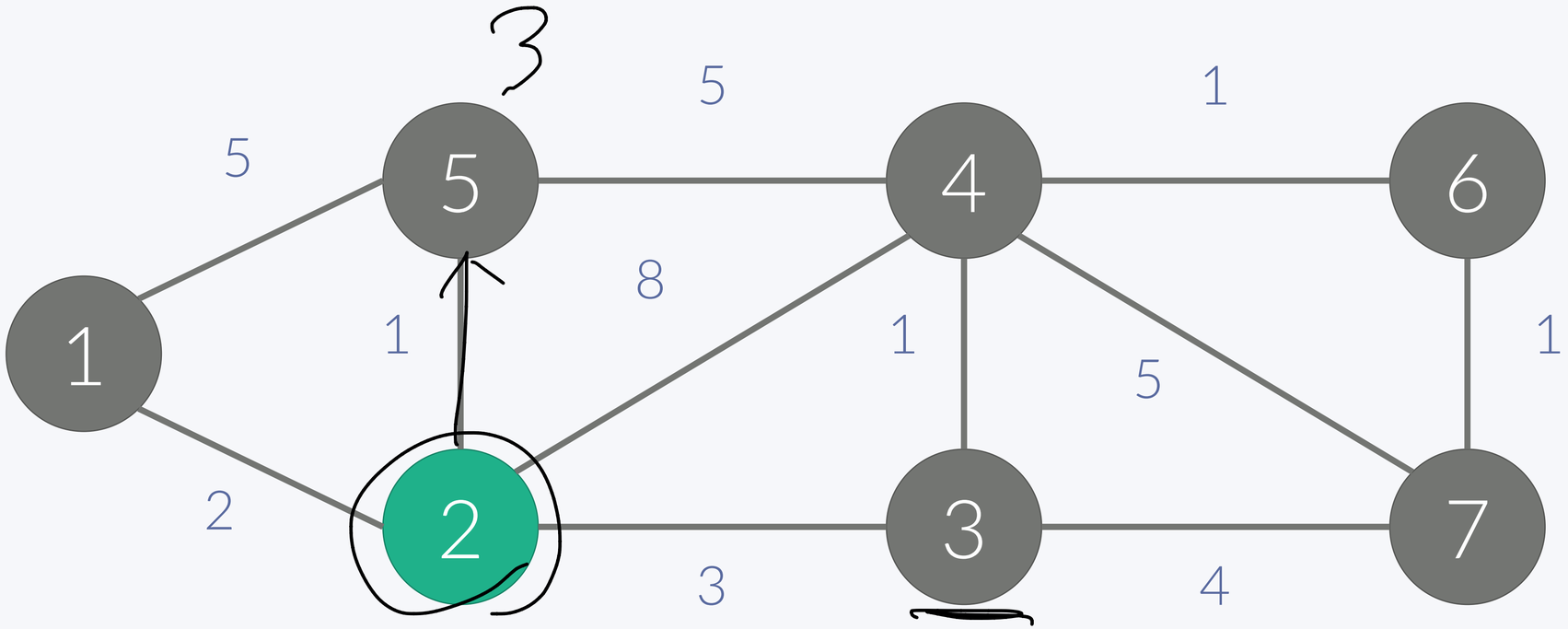


다익스트라

Dijkstra Algorithm

- 선택: 2

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

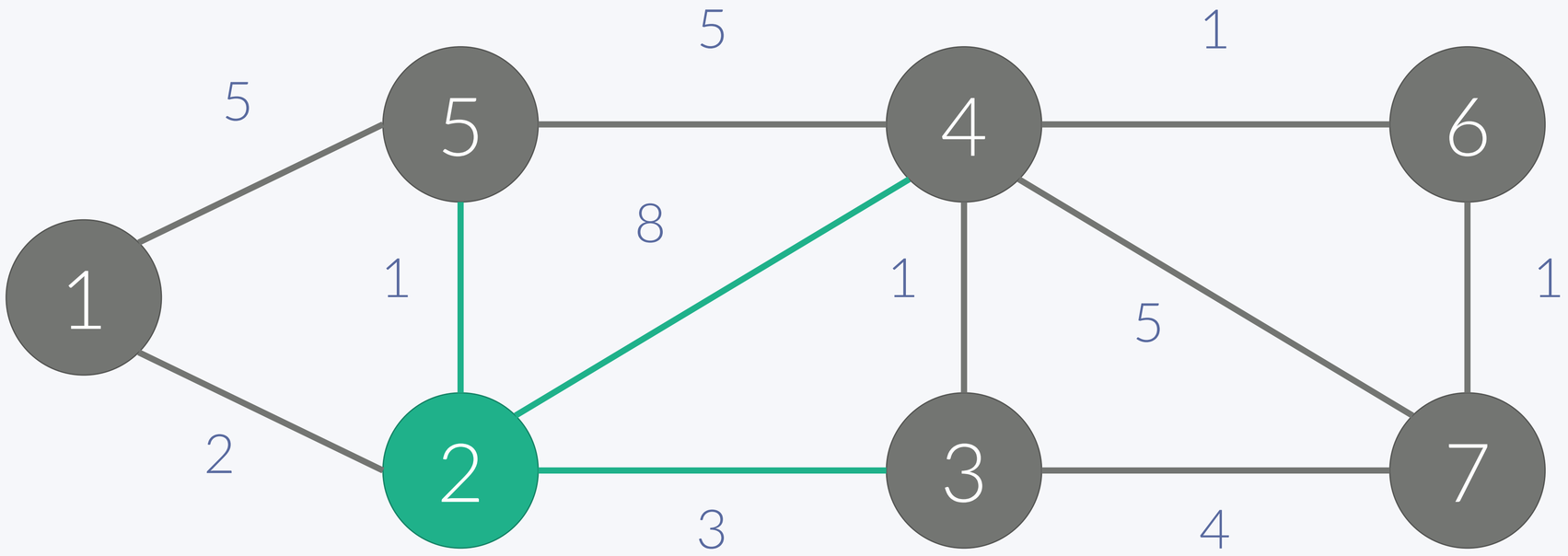


다익스트라

Dijkstra Algorithm

- 선택: 2

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

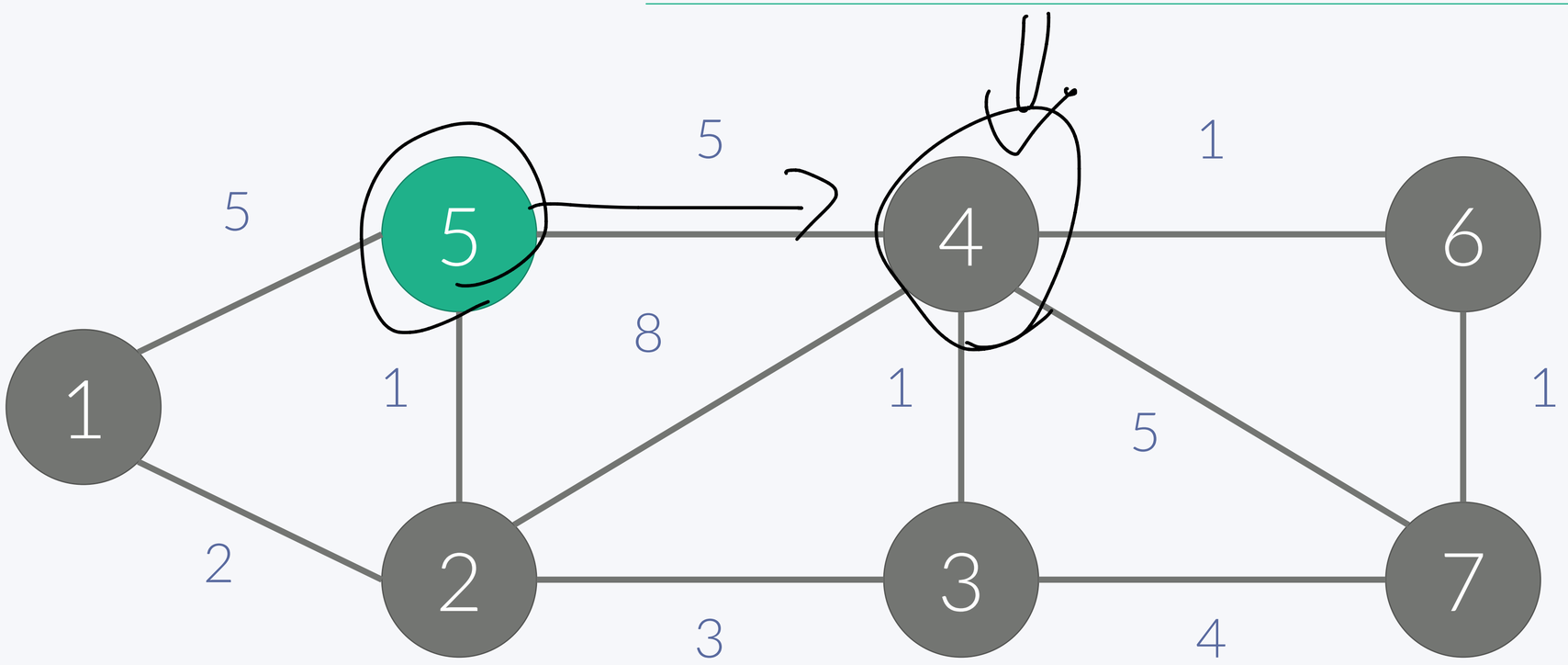


다익스트라

Dijkstra Algorithm

- 선택: 5

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

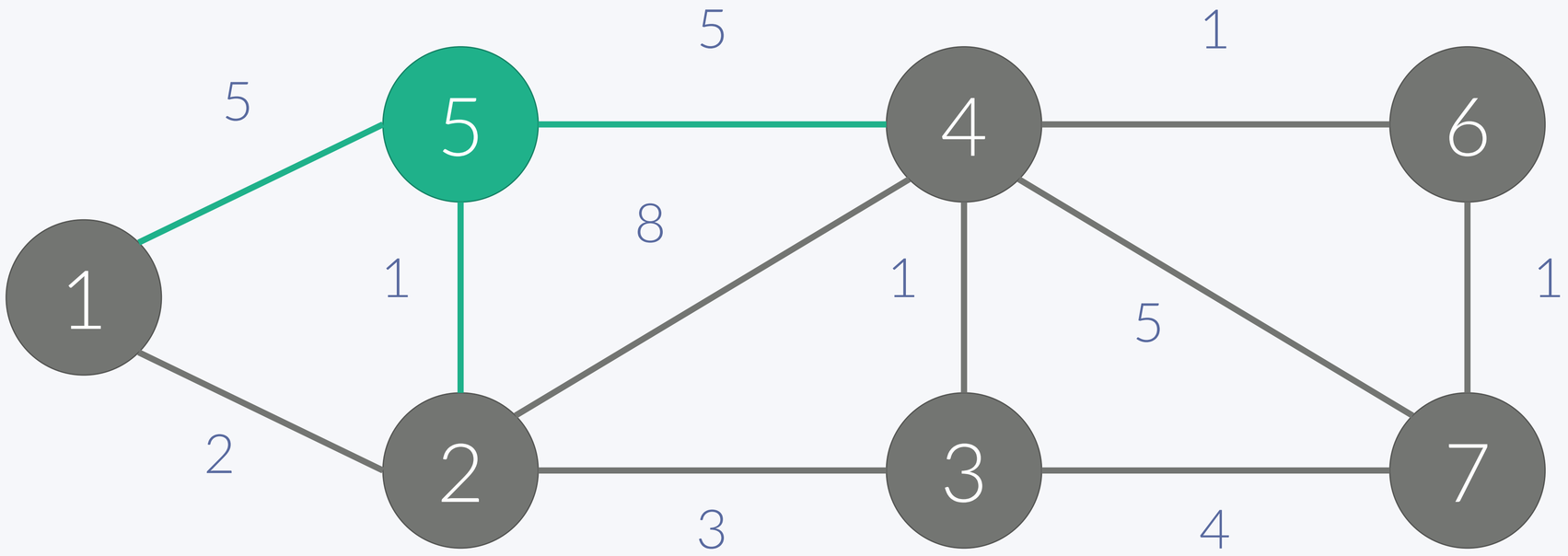


다익스트라

Dijkstra Algorithm

- 선택: 5

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

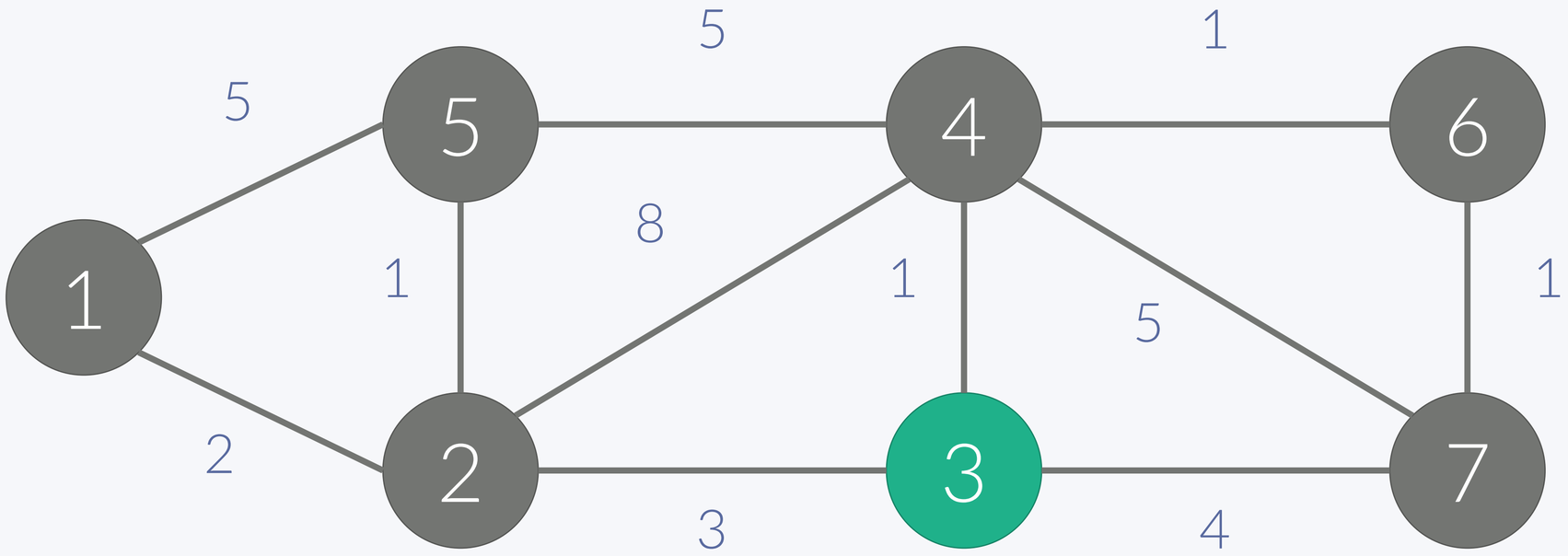


다익스트라

Dijkstra Algorithm

- 선택: 3

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

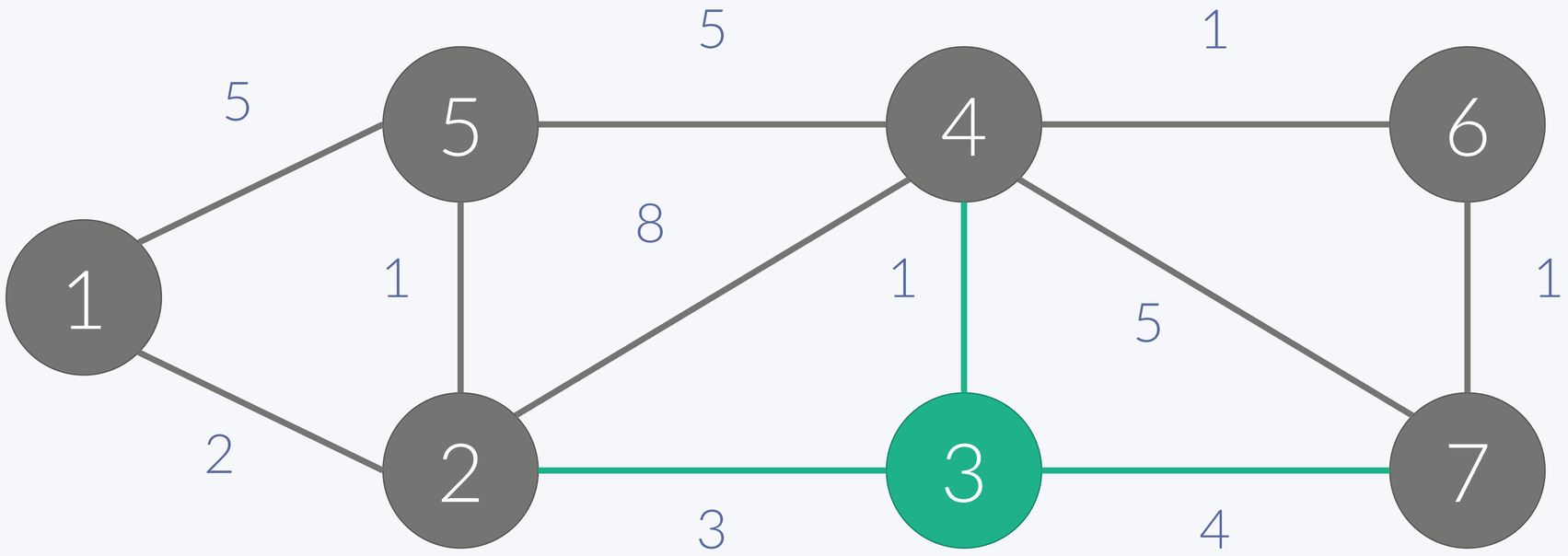


다익스트라

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		

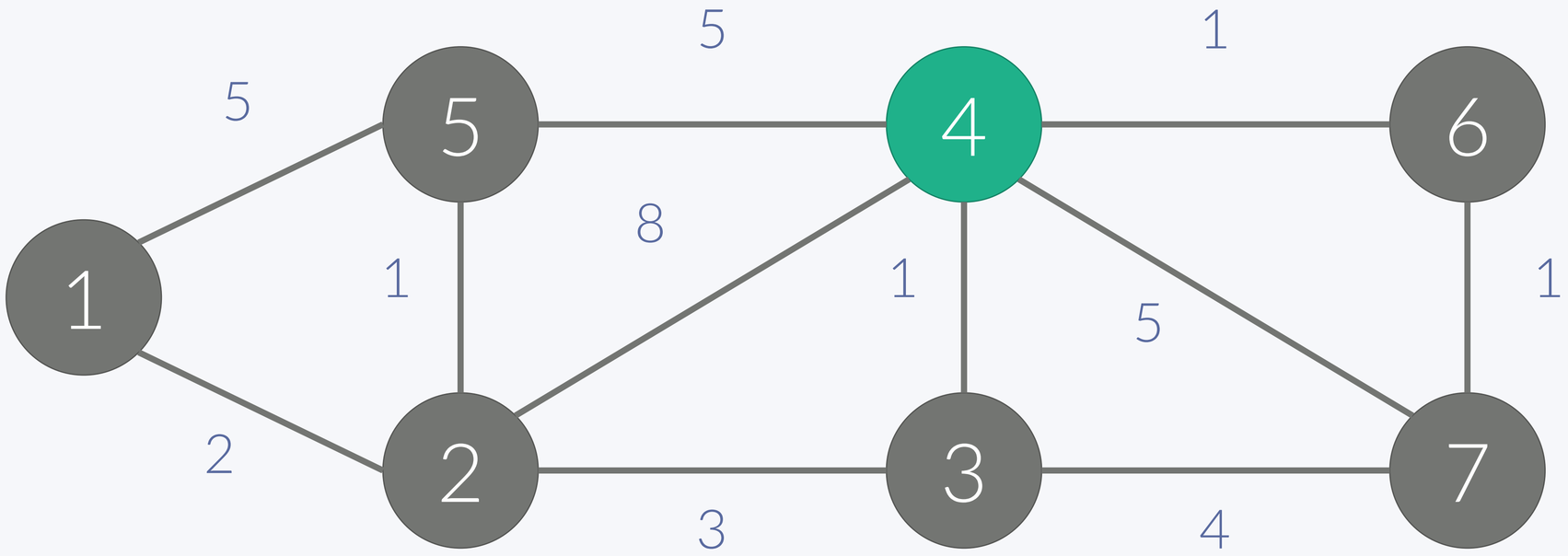


다익스트라

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		

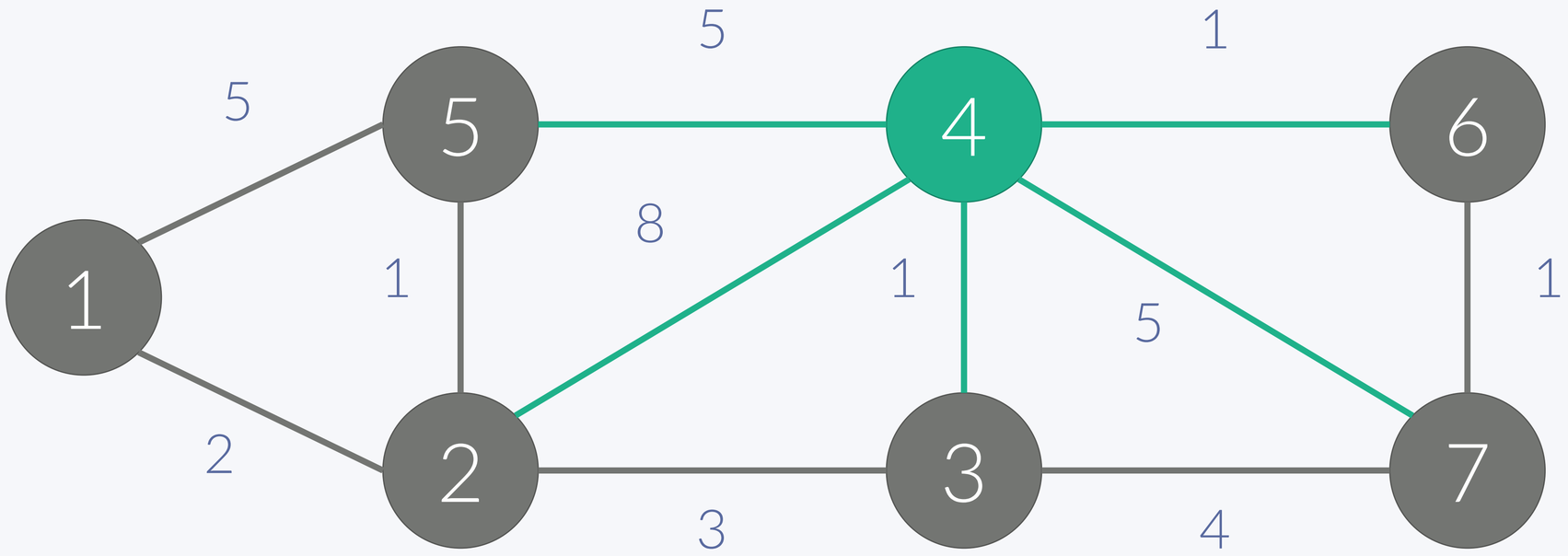


다익스트라

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		

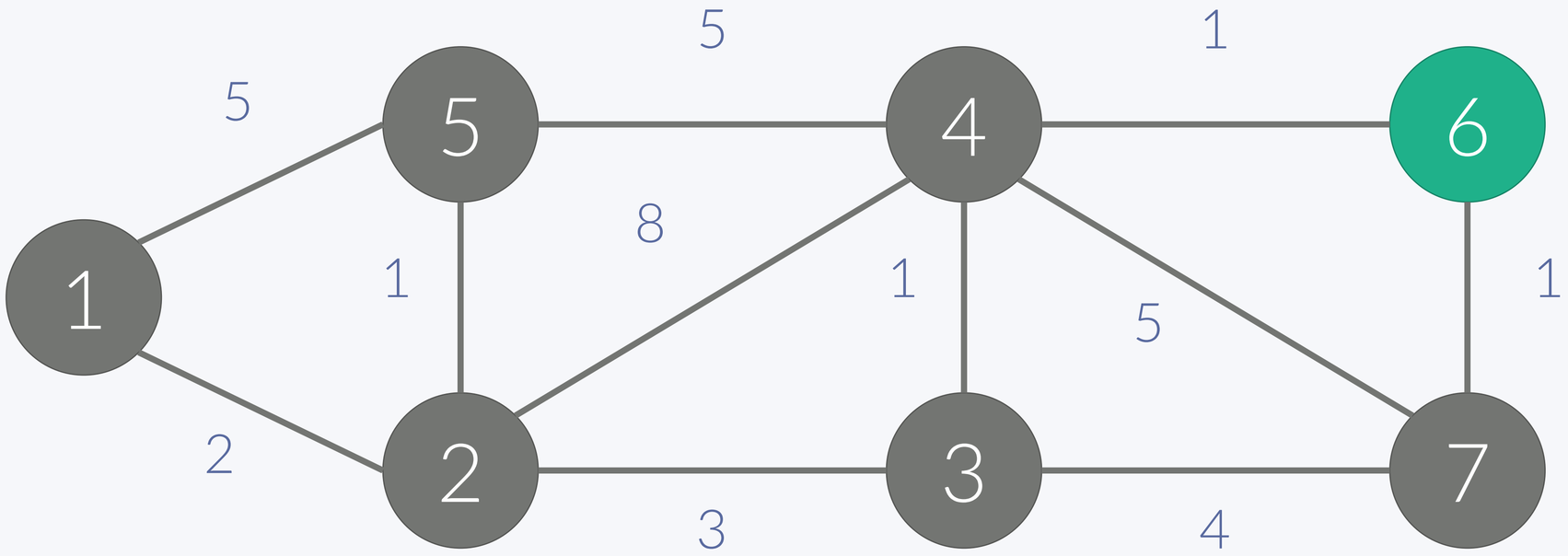


다익스트라

Dijkstra Algorithm

- 선택: 6

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

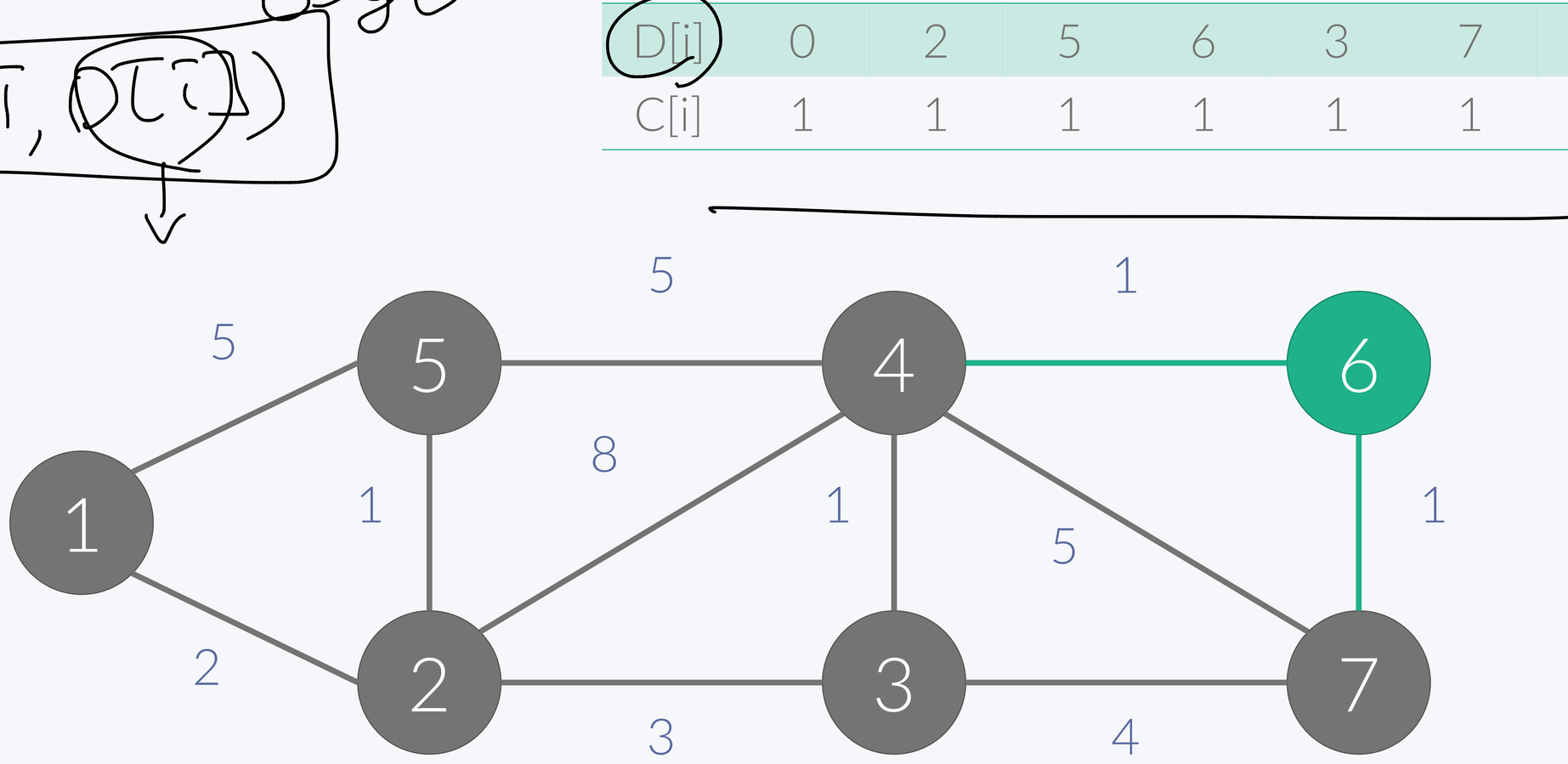


다익스트라

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	

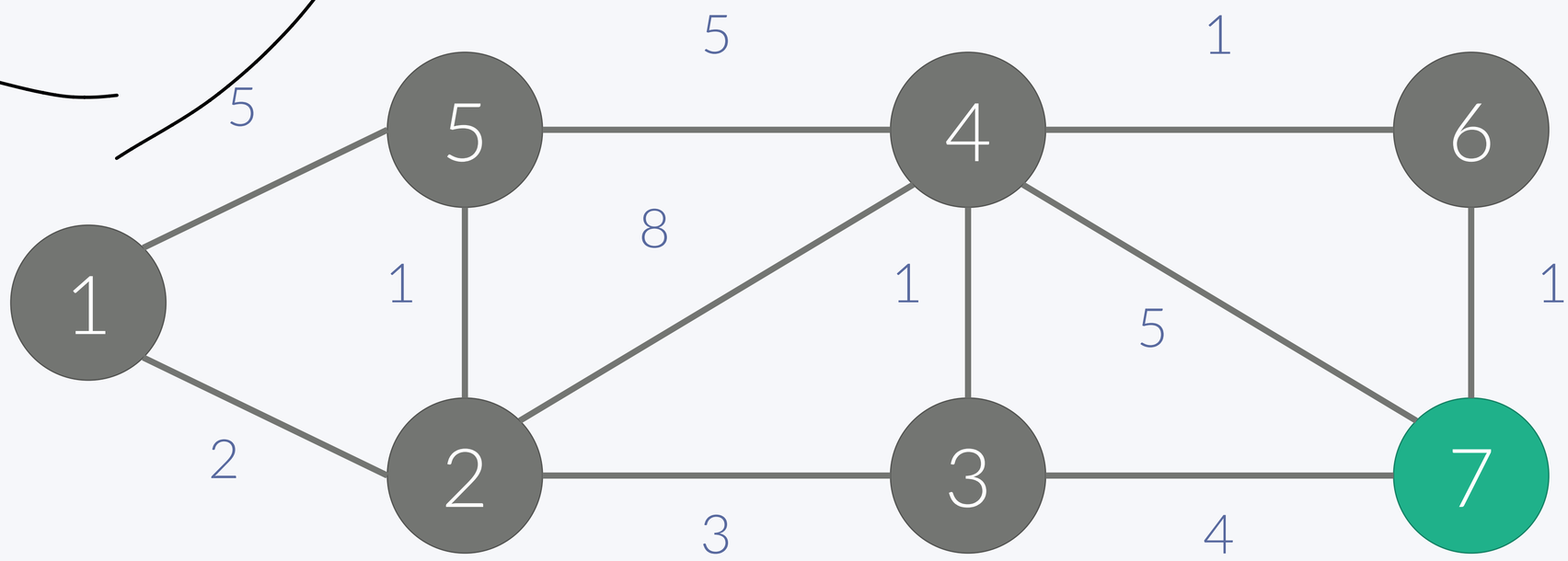


다익스트라

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

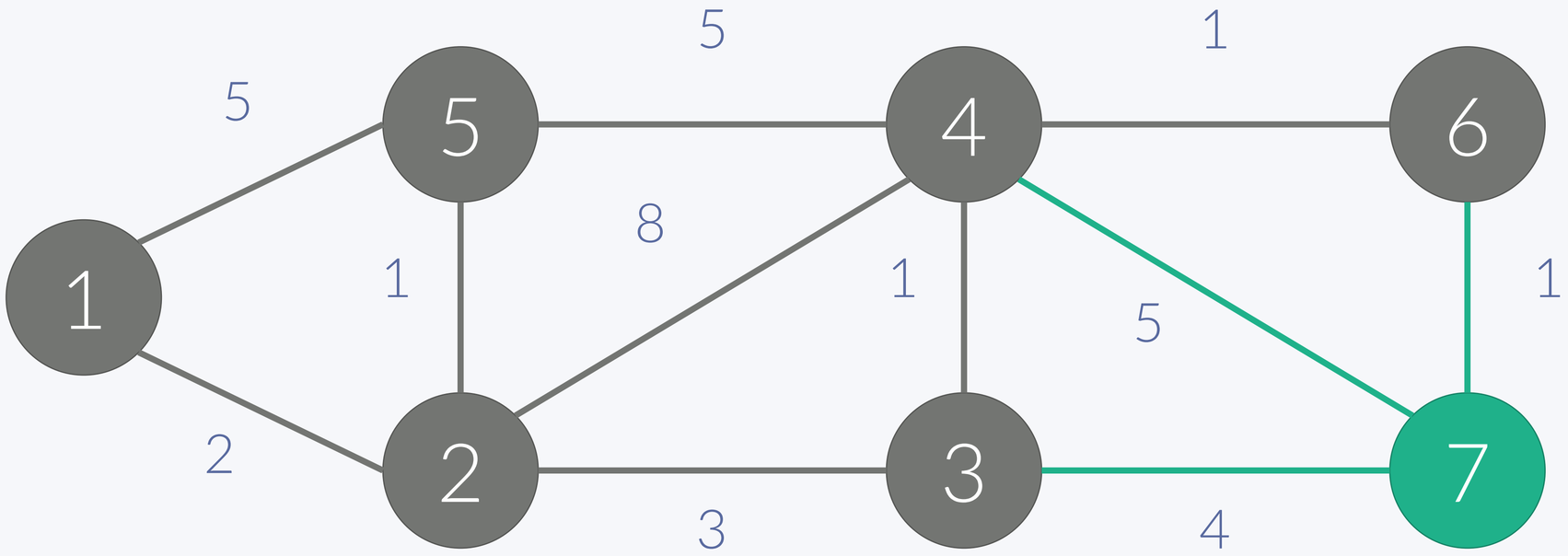


다익스트라

Dijkstra Algorithm

- 선택: 7

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

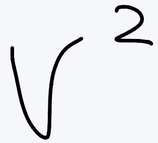
- 시간 복잡도
- 인접 행렬: O(V^2)
- 인접 리스트: O(V^2)

최소비용 구하기

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

- A에서 B로 가는 최단 경로

최소비용 구하기



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

- 인접 행렬 소스: http://boj.kr/44c16c96c6c34c6dbd9d62177eb4f380
- 인접 리스트 소스: http://boj.kr/234862a31bc9401ebcef53065caa9822

최소비용 구하기 2

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

- A에서 B로 가는 최단 경로
- 최단 경로도 구해야 함

- 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;
}
```

최소비용 구하기 2

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();
}
```

최소비용 구하기 2

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

- 소스: http://boj.kr/ff6179d1a97e46c48e1b4d061c60c7f5

특정한 최단 경로

https://www.acmicpc.net/problem/1504

- 1 -> N으로 가는데, V1, V2를 꼭 거쳐서 가는 최단 경로

- 가능한 경우 2가지
- 1 -> V1 -> V2 -> N
- 1 -> V2 -> V1 -> N

- 다익스트라의 시작점을 1, V1, V2로 총 3번 알고리즘을 수행한 다음에 답을 구할 수 있다.

특정한 최단 경로

https://www.acmicpc.net/problem/1504

- 소스: http://boj.kr/e1d61ef1a00740bd890d1226e2894e9a

최단 경로

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

- 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?

최단 경로

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

1. 체크되어 있지 않은 정점 중에서 D의 값이 가장 작은 정점 x를 선택한다.
2. x를 체크한다.
3. x와 연결된 모든 정점을 검사한다.
    - 간선을 (x, y, z)라고 했을 때
    - d[y] > d[x] + z 이면 갱신해준다.

- 1, 2, 3단계를 모든 정점을 체크할 때까지 계속한다.

최단 경로

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

1. **체크되어 있지 않은 정점 중에서 D의 값이 가장 작은 정점 x를 선택한다.**
2. x를 체크한다.
3. x와 연결된 모든 정점을 검사한다.
    - 간선을 (x, y, z)라고 했을 때
    - d[y] > d[x] + z 이면 갱신해준다.

- 1, 2, 3단계를 모든 정점을 체크할 때까지 계속한다.

최단 경로

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

- 힙을 사용해서 최소값을 O(lgE)만에 찾을 수 있다

최단 경로

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

- 소스: http://boj.kr/6382d43260624ea5b088fc0032fdca21

```
if(dist[x] < p.first)
    continue;
```

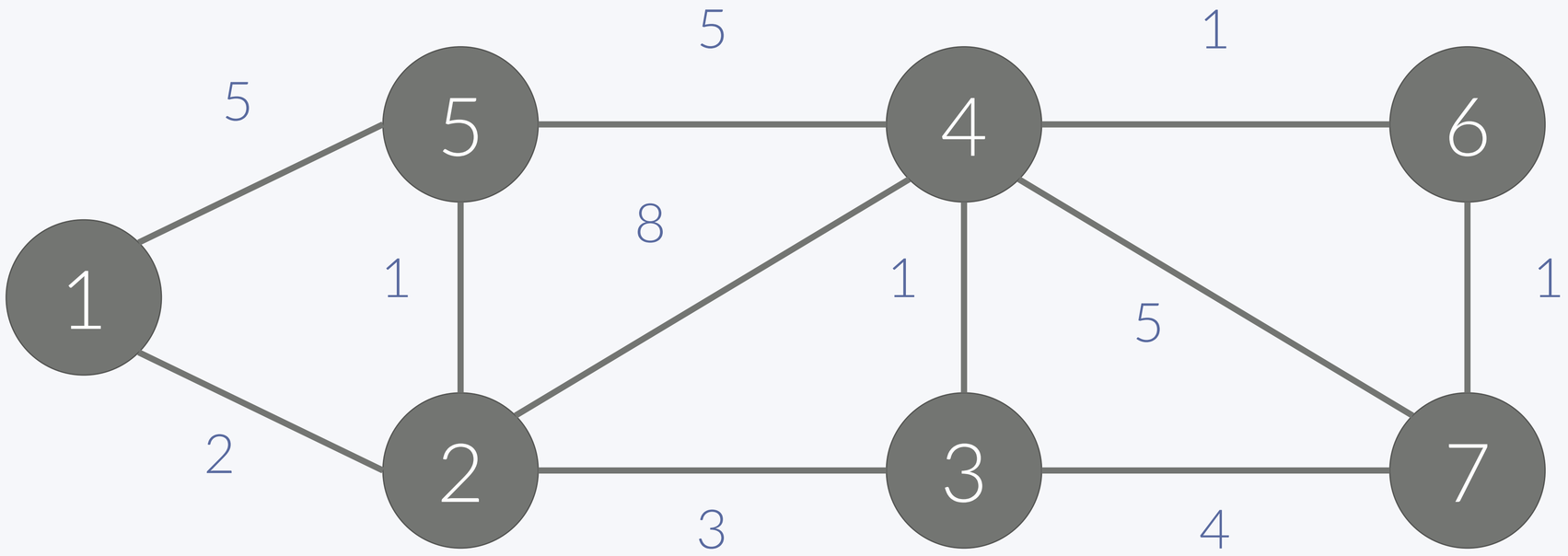
$V E \log E < V^3$

플로이드

플로이드

Floyd-Warshall Algorithm

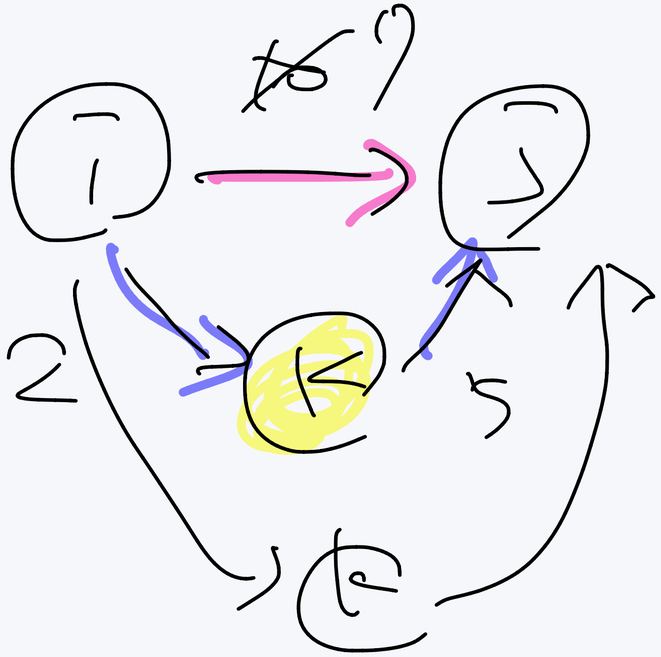
- 모든 쌍의 최단 경로를 구하는 알고리즘



플로이드

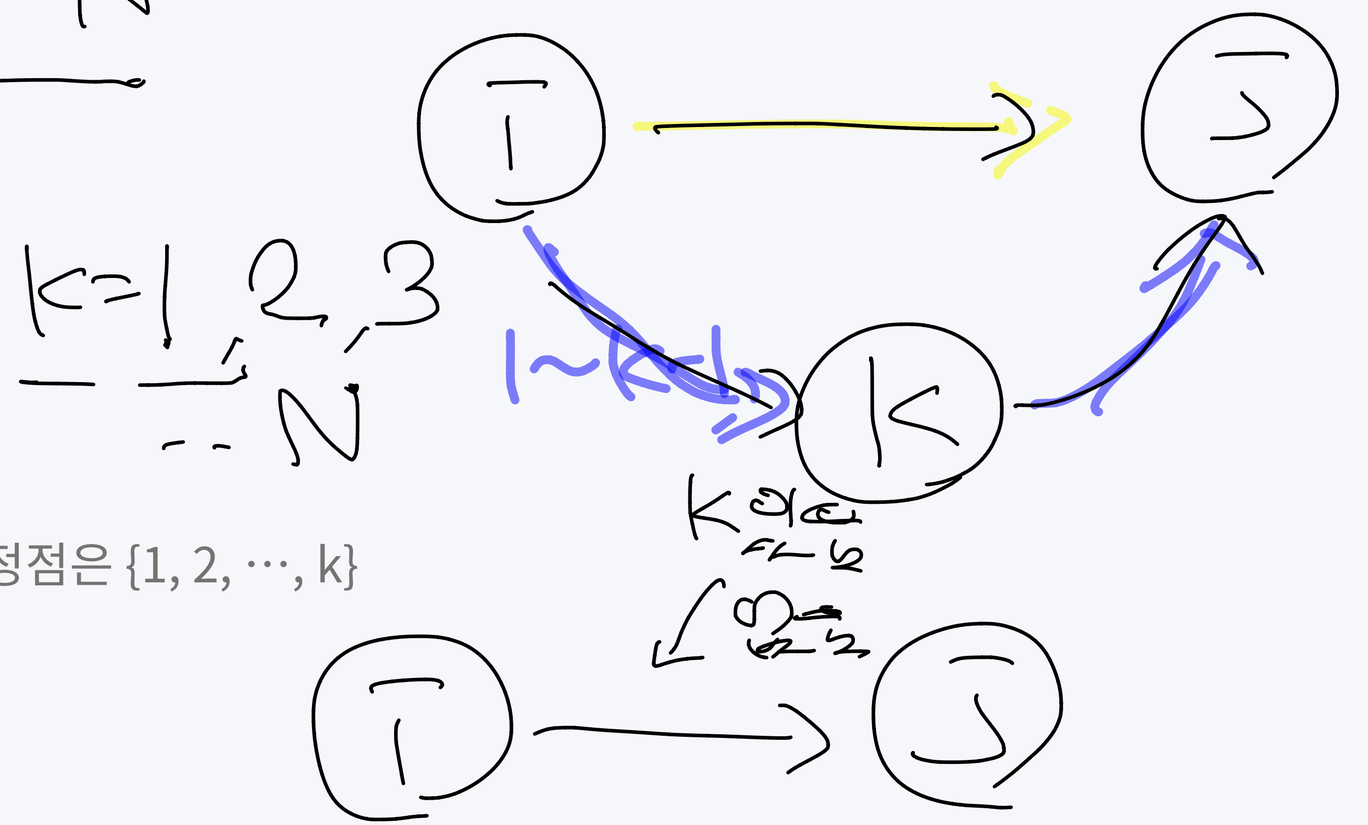
Floyd-Warshall Algorithm

```
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~N 까지 정점이 있을 때
- d[k][i][j]를 다음과 같이 정의
  - i -> j로 이동하는 최단 경로
  - 이 때, 중간에 방문할 수 있는 정점은 {1, 2, ⋯, k}
- 그럼 d[k][i][j]를 구해보자
  - k가 경로에 없는 경우
  - k가 경로에 있는 경우

플로이드

Floyd-Warshall Algorithm

- 1~N 까지 정점이 있을 때
- d[k][i][j]를 다음과 같이 정의
  - i -> j로 이동하는 최단 경로
  - 이 때, 중간에 방문할 수 있는 정점은 {1, 2, ⋯, 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

- 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>=1)

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

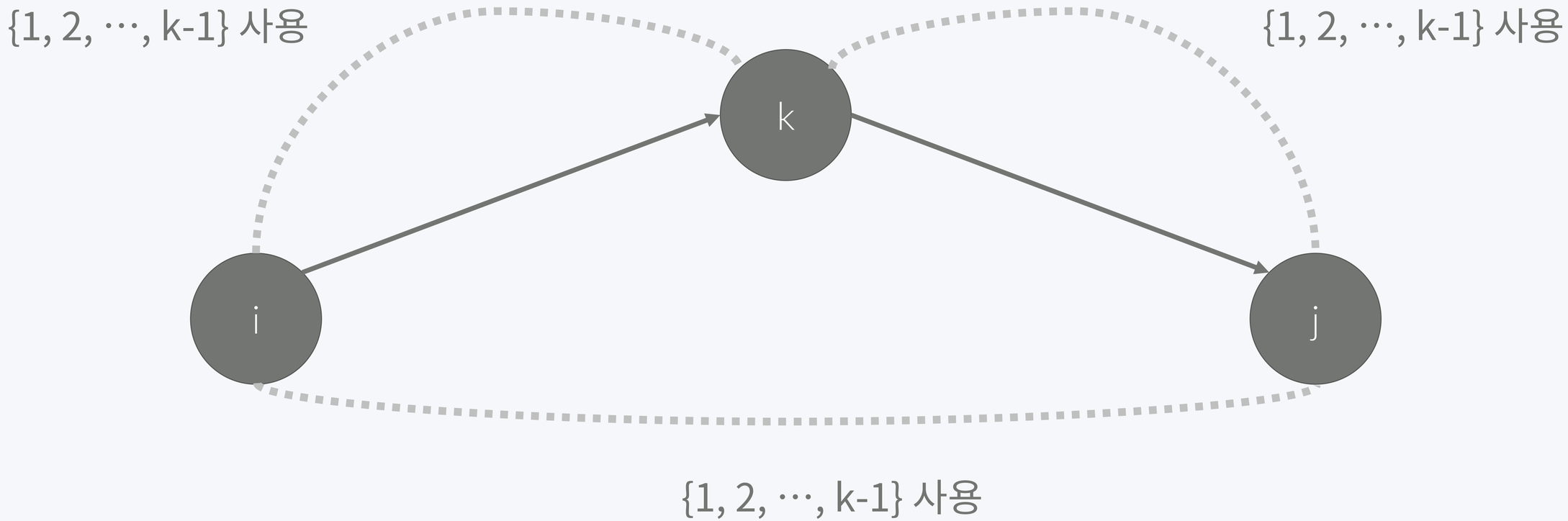

```
for (int k=1; k<=n; k++) {
    for (int i=1; i<=n; i++) {
        for (int j=1; j<=n; j++) {
            if (d[i][j] > d[i][k] + d[k][j]) {
                d[i][j] = d[i][k] + d[k][j];
            }
        }
    }
}
```

from[i][j] = k

플로이드

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>=1)



경로 찾기

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

- 가중치 없는 방향 그래프 G가 주어졌을 때, 모든 정점 (i, j)에 대해서, i에서 j로 가는 경로가 있는지 없는지 구하는 프로그램을 작성하시오.

경로 찾기

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

- 소스: http://boj.kr/b34dcdb5786348cb89b376d29e6e450e

플로이드

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

- n(1≤n≤100)개의 도시가 있다
- 한 도시에서 출발하여 다른 도시에 도착하는 m(1≤m≤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

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

- 소스: http://boj.kr/56f787d7c799483b8ae49e57f3da22ef

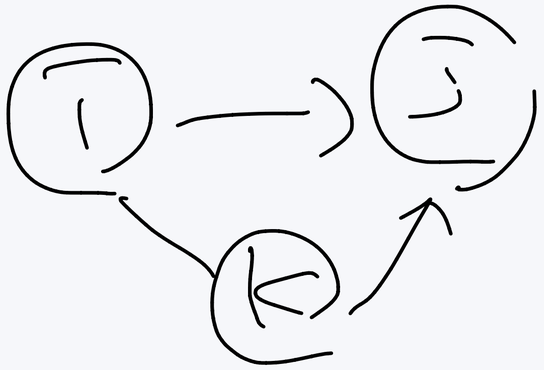
플로이드 2

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

- 플로이드 구현을 보면
- i -> k, k -> j로 가는 간선을 i -> j로 바꿔줬기 때문에
- i -> j 이 원래 어떤 정점을 가르키고 있었는지를 알아야 한다.
- 따라서 구현을 조금 변경할 수 있다.

플로이드 2

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




```
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

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

- 소스: http://boj.kr/da5884122de244de82c7c6fc97dcf593

케빈 베이컨의 6단계 법칙

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

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

케빈 베이컨의 6단계 법칙

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

- 소스: http://boj.kr/c234f2380f984bfc969e01d9ec8a22bf

궁금한 민호

https://www.acmicpc.net/problem/1507

- 강호는 모든 쌍의 도시에 대해서 최소 이동 시간을 구해놓았다. 민호는 이 표를 보고 원래 도로가 몇 개 있는지를 구해보려고 한다.
- 모든 쌍의 도시 사이의 최소 이동 시간이 주어졌을 때, 이 나라에 존재할 수 있는 도로의 개수의 최소값과 그 때, 모든 도로의 시간의 합을 구하는 프로그램을 작성하시오.

궁금한 민호

https://www.acmicpc.net/problem/1507

- A에서 B로 가는 비용이 x 일 때
- A->C->B로 가는 비용이 x이면
- A->B로 가는 도로는 필요가 없다
- 이렇게 모든 도시의 쌍을 보면서 필요없는 도로를 제거

궁금한 민호

https://www.acmicpc.net/problem/1507

- 소스: http://boj.kr/d20ebfd3bba940cbbf3b3ef8abefb27f

운동

https://www.acmicpc.net/problem/1956

- 그래프에서 사이클 길이 중 최소길이를 찾는 문제

운동

https://www.acmicpc.net/problem/1956

- 그래프에서 사이클 길이 중 최소길이를 찾는 문제
- 플로이드를 이용한 다음에 d[i][i]를 검사하면 된다.

운동

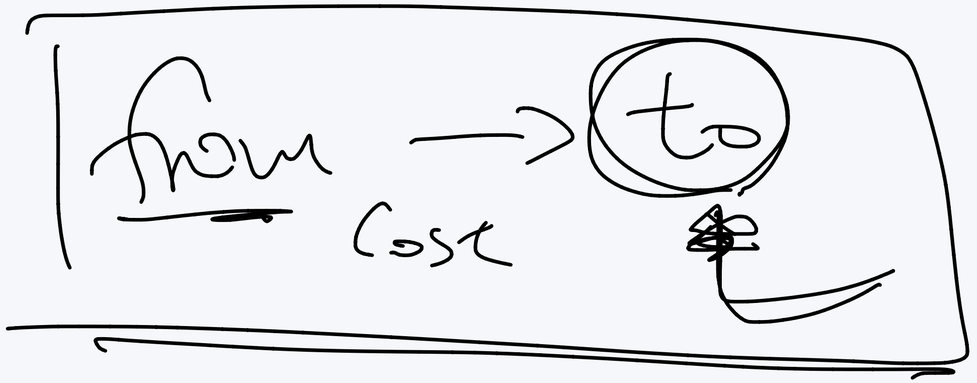
https://www.acmicpc.net/problem/1956

- 소스http://boj.kr/e240c4669742498d92d575b343017a18

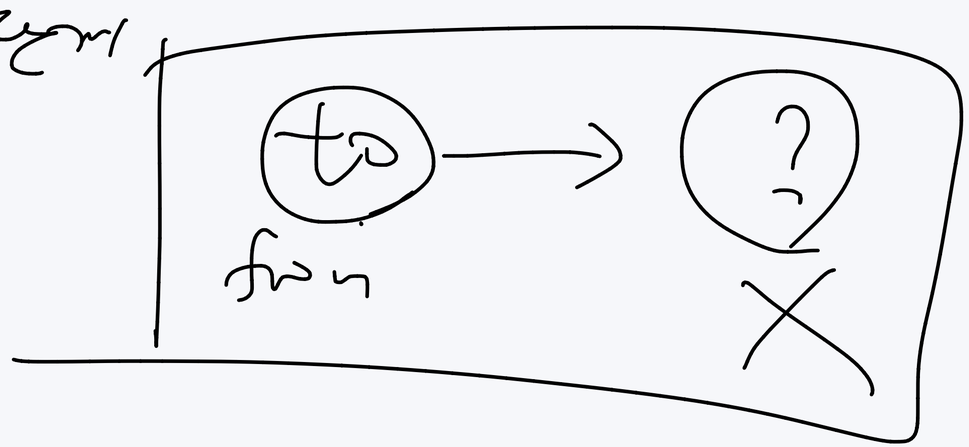
SPFA

SPFA

Shortest 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

Shorteest Path Faster Algorithm

```
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;
            }
        }
    }
}
```

O(VE)

O(E)

타임머신

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

- 소스: http://boj.kr/e49e0e0595774da4b836202e017a7180