

알튜브비뷰

최단 경로

간선에 가중치가 있는 그래프가 주어질 때, 정점 사이의 최단 경로를 구하는 알고리즘입니다.
대표적으로 다익스트라, 플로이드-워셜, 벨만-포드 알고리즘이 있습니다.

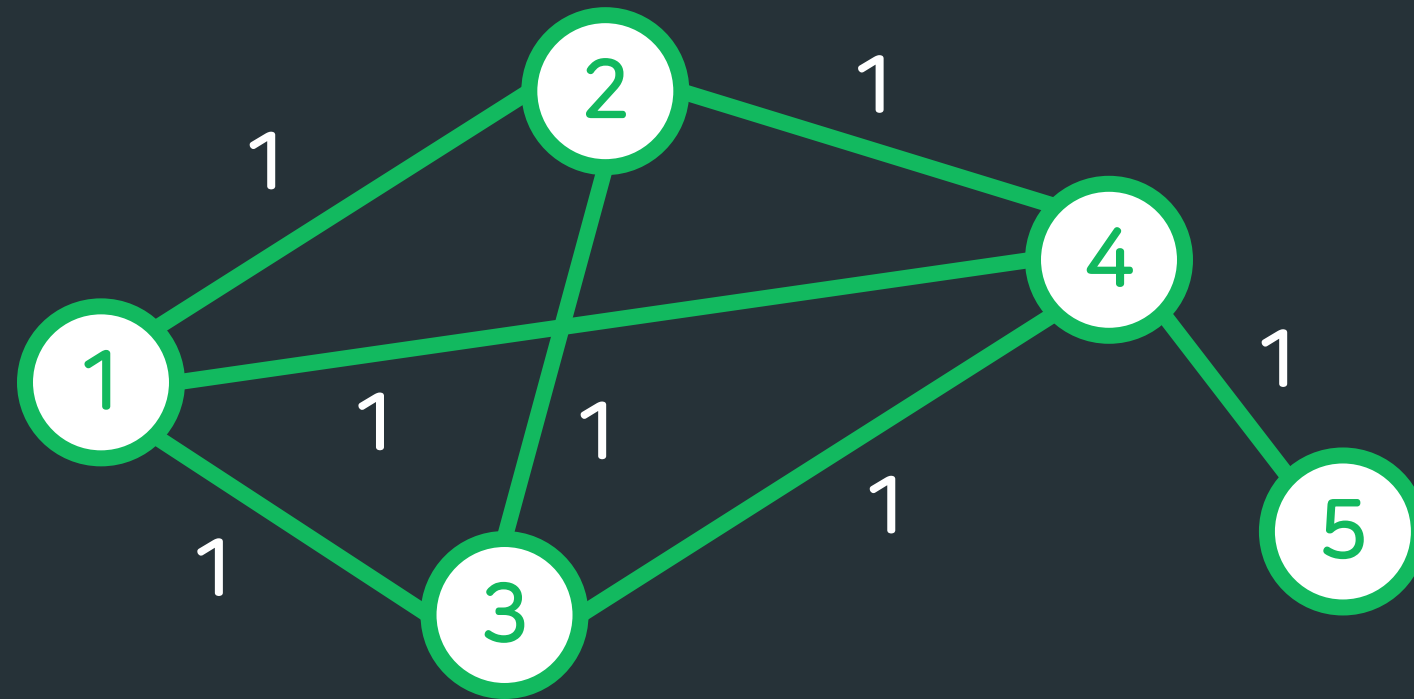
코딩테스트에 자주 나오진 않지만 한 번 나오면 난이도 있는 문제로 나오곤 해요.

■ 지난 시간에 이런 얘기를 했었어요



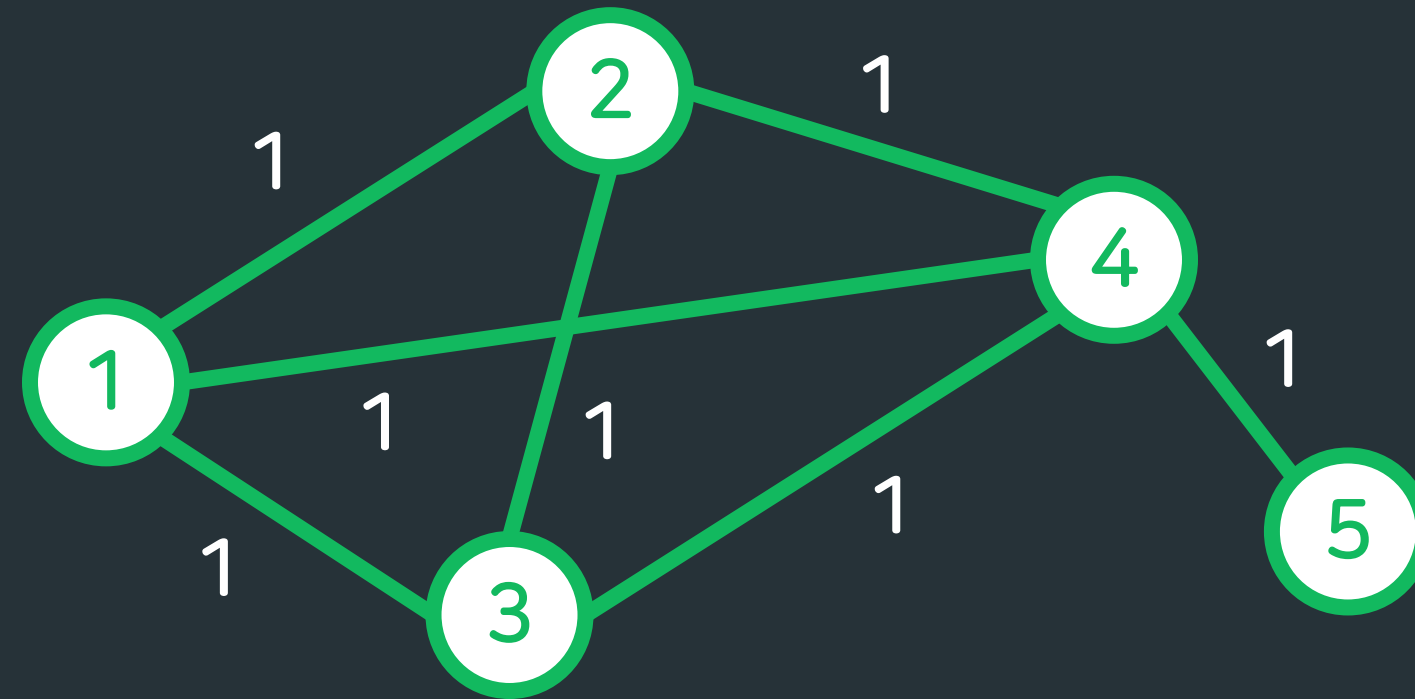
두 정점 사이의 **최단 거리**를 구할 땐 **BFS**

지난 시간에 이런 얘기를 했었어요



사실은 **가중치가 1**인 그래프의 최단 경로를 구한 것과 같음

지난 시간에 이런 얘기를 했었어요



사실은 **가중치가 1**인 그래프의 최단 경로를 구한 것과 같음

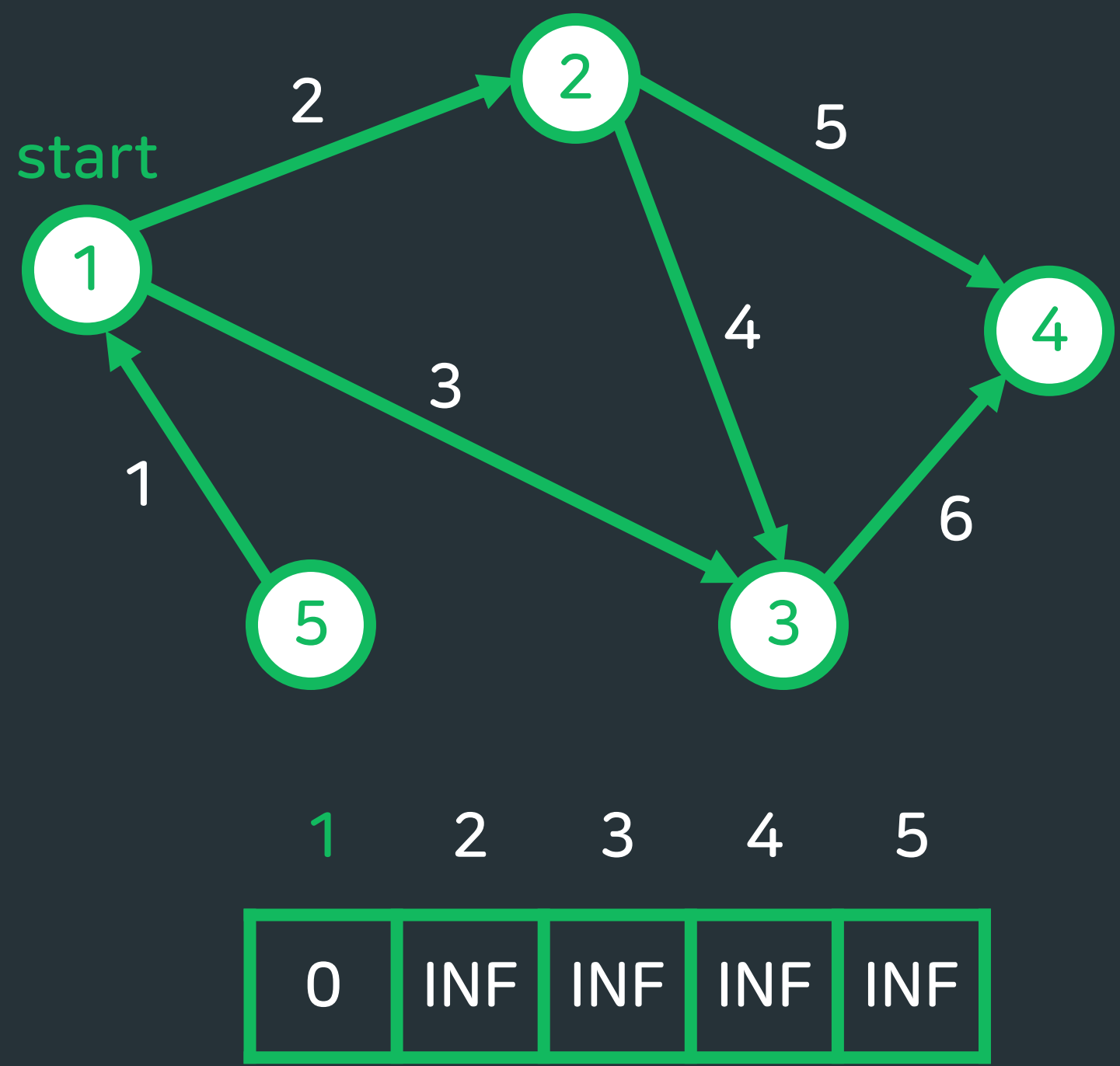
가중치가 **다양**하다면?

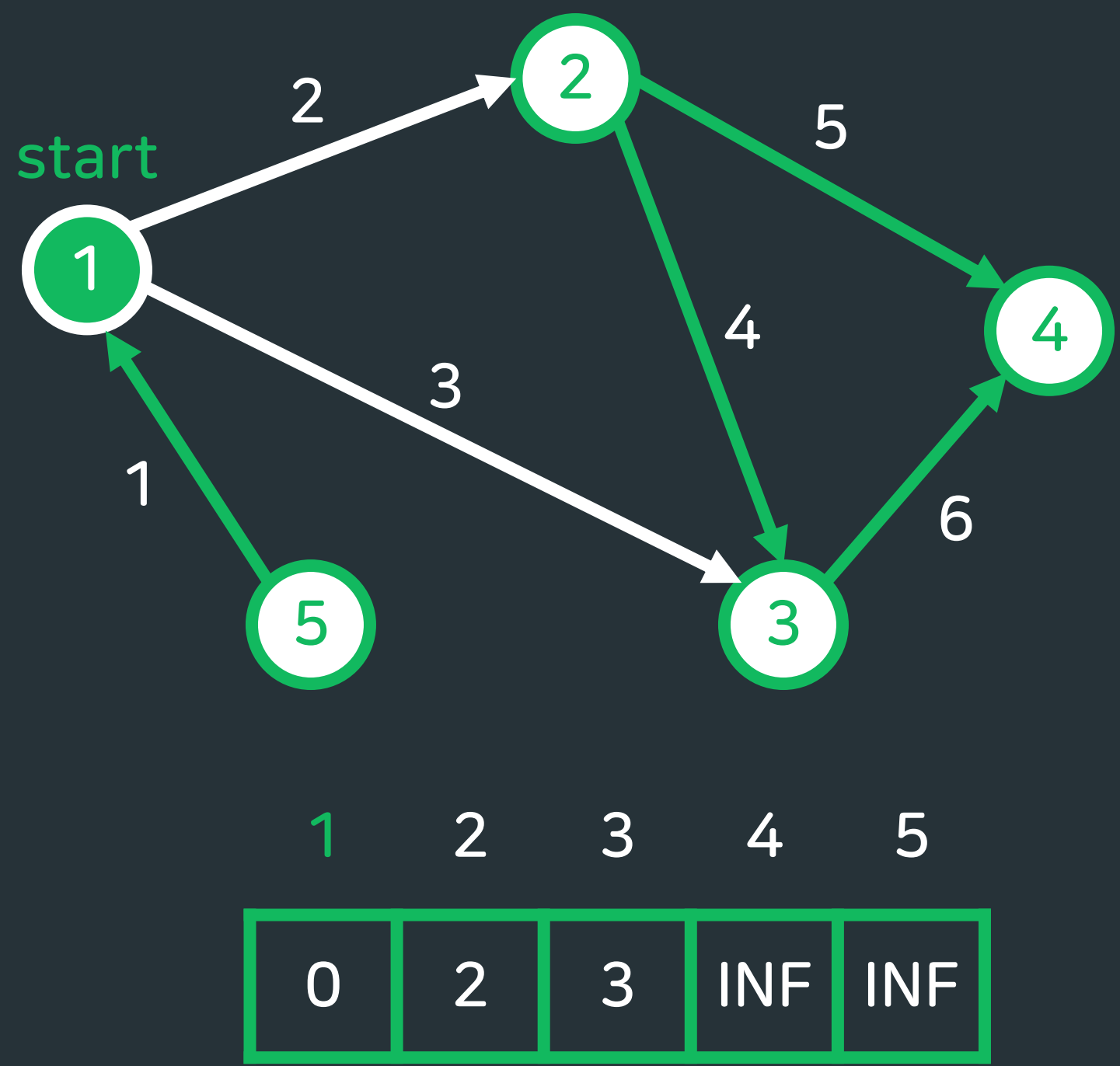
Shortest Path

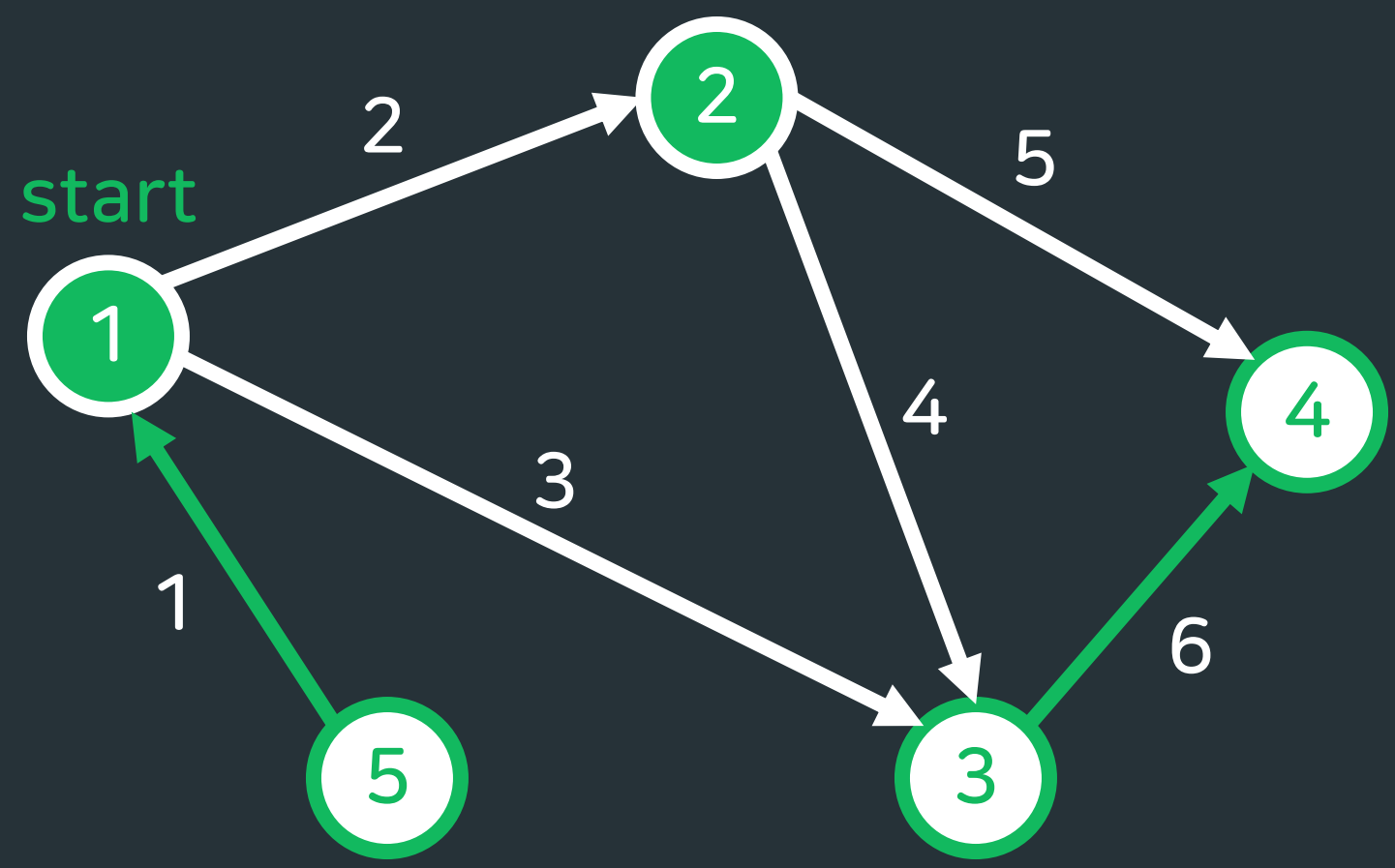
- 그래프에서 정점 사이의 최단 경로를 구하는 알고리즘
 - Single-Source (SSP) : 하나의 시작점에 대한 모든 정점까지의 최단 경로
 - Single-Destination : 모든 정점으로부터 하나의 도착점까지의 최단 경로. SSP를 뒤집어서 구현
 - Single-Pair : 특정 정점 2개 사이의 최단 경로. SSP의 sub-problem
 - All-Pairs (ASP) : 가능한 모든 정점 2개의 조합에 대한 최단 경로
-
- SSP : 다익스트라, 벨만-포드
 - ASP : 플로이드-워셜

Dijkstra

- 하나의 시작점에서 모든 정점까지의 최단 경로를 구하는 SSP 알고리즘
- 시작 정점으로부터 가장 가까운 정점부터 탐색하는 그리디적 접근
- 가중치가 음수인 간선이 있다면, 경우에 따라 무한 루프에 빠질 수 있음
- 정점의 수를 V , 간선의 수를 E 라고 할 때, 시간 복잡도는 $O(V \log V + E \log V)$

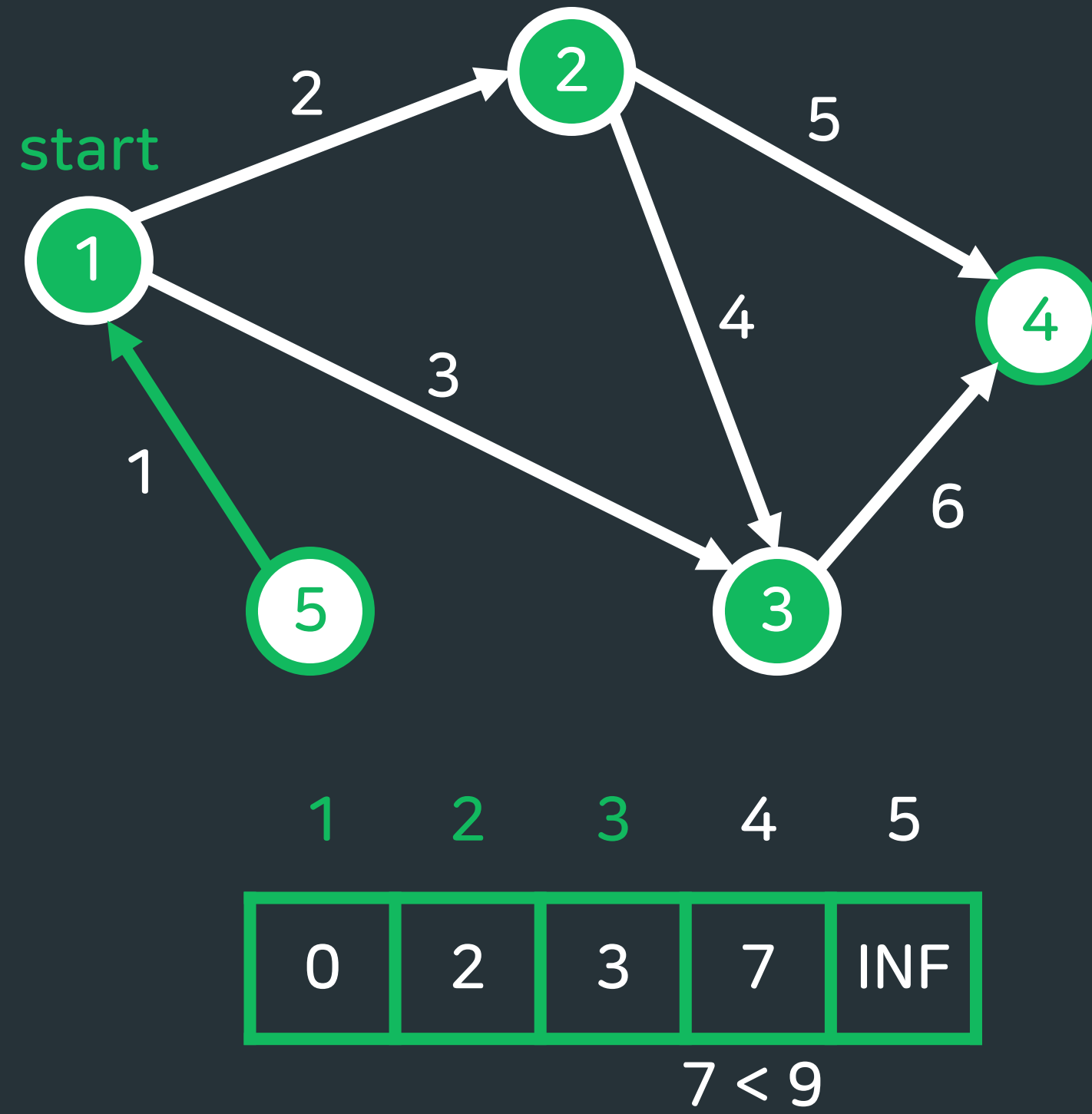


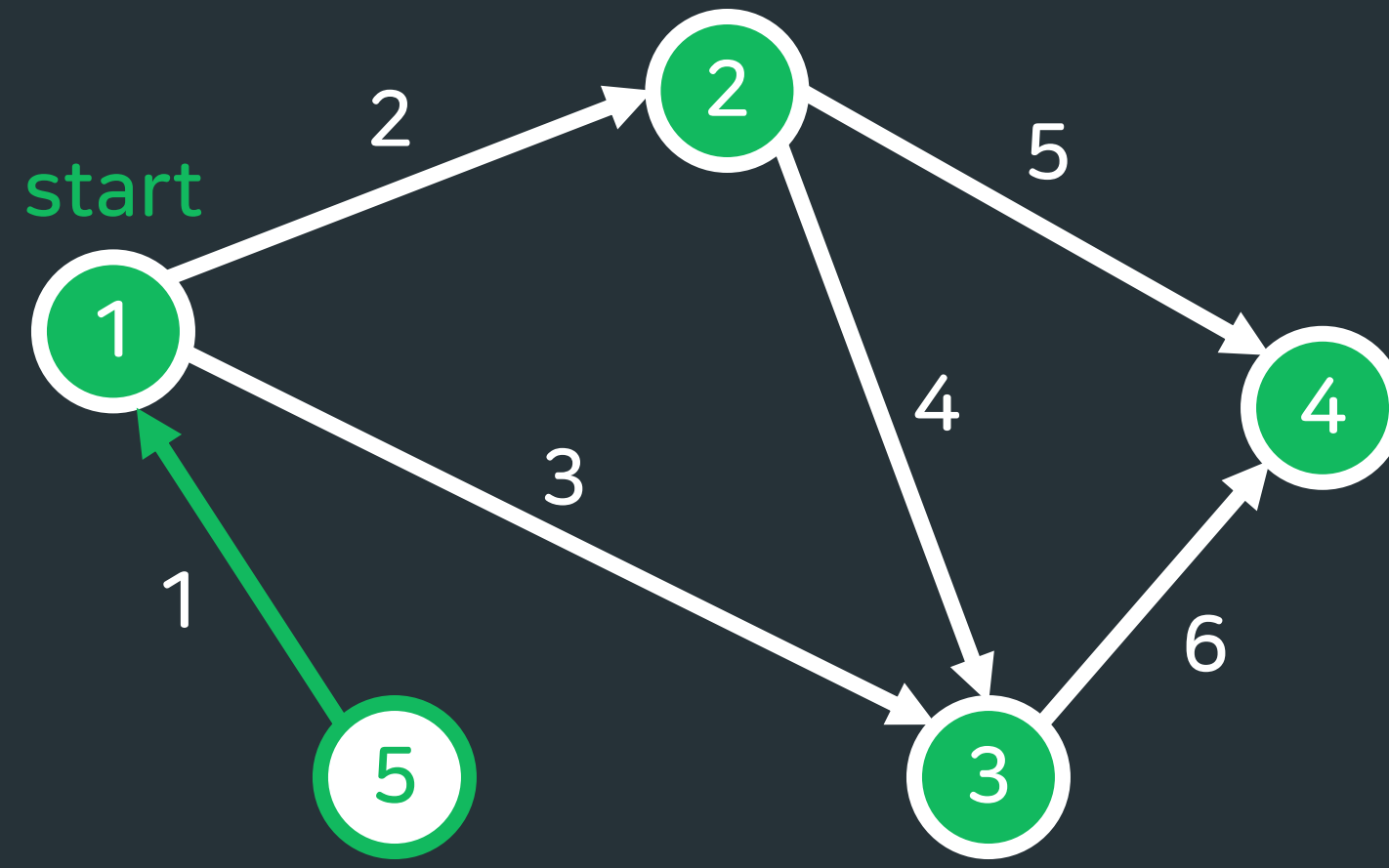




1	2	3	4	5
0	2	3	7	INF

$3 < 6$

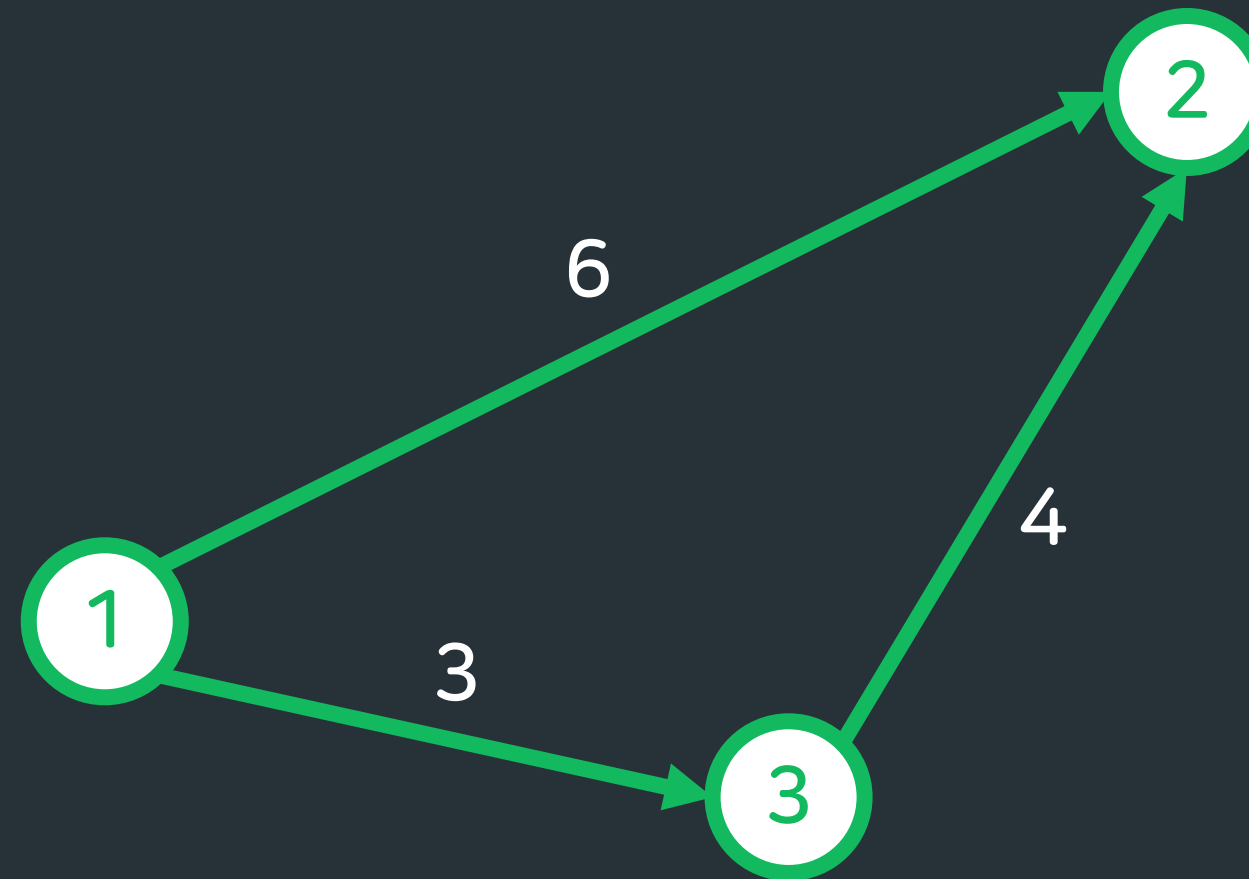




1	2	3	4	5
0	2	3	7	INF

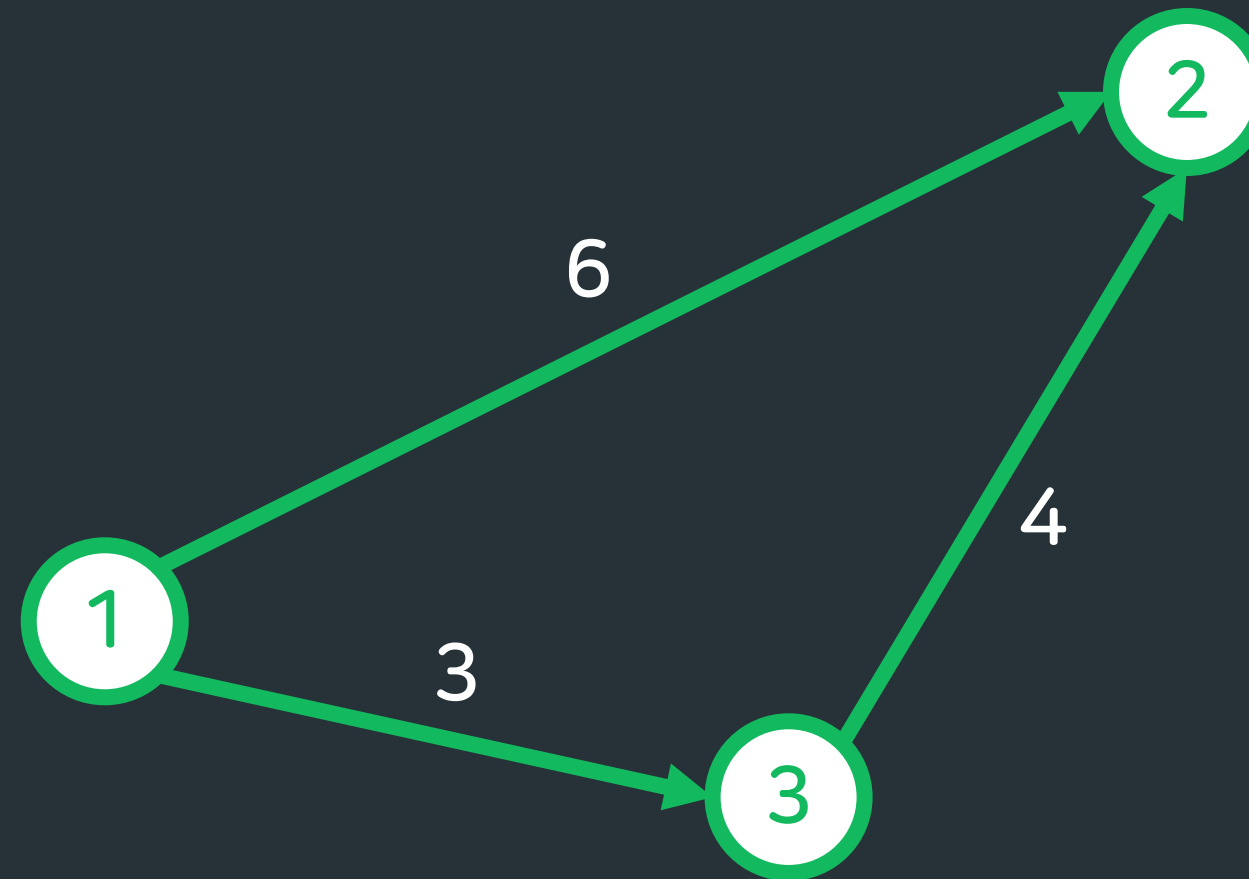
더 이상 탐색할 수 있는 정점 없음

정말 그리디가 가능할까?



1->3 간선을 먼저 선택하고 3->2 간선을 선택해 1->3->2로 갔는데
알고보니 1->2로 바로 가는게 최단 경로라면?

정말 그리디가 가능할까?



1->3 간선을 먼저 선택했더라도 3->2 간선 하나를 고려하는게 아니라 1부터의 거리를 고려하는 것!
그러므로 3->2를 통해 1->3->2를 가기 전 1->2 간선을 먼저 고려하게 될 것

cf) 시작점으로부터의 거리가 아니라 간선 자체의 가중치만 고려하는 것은 Prim 알고리즘



모든 정점까지의 거리를 담은 `dist` 배열을 `INF`으로 초기화
시작 정점까지의 거리 `0`으로 초기화

```
while (갱신할 정점이 있을 때까지) {
```

```
    int v = 탐색하지 않은 정점 중 시작점에서 가장 가까운 정점
```

← 현재 가장 가까운 정점을
정점의 수만큼(V) 찾아야 함

```
    for (v와 연결된 모든 정점에 대해){
```

```
        int u = v와 연결된 정점
```

```
        if(dist[v] + weight[v][u] < dist[u]){
```

```
            dist[u] = dist[v] + weight[v][u]
```

```
        }
```

```
    }
```

```
}
```

← 간선의 수만큼(E)
`dist`를 갱신하게 됨



모든 정점까지의 거리를 담은 `dist` 배열을 `INF`으로 초기화
시작 정점까지의 거리 `0`으로 초기화

```
while (갱신할 정점이 있을 때까지) {  
    int v = 탐색하지 않은 정점 중 시작점에서 가장 가까운 정점  
    for (v와 연결된 모든 정점에 대해){  
        int u = v와 연결된 정점  
        if(dist[v] + weight[v][u] < dist[u]){  
            dist[u] = dist[v] + weight[v][u]  
        }  
    }  
}
```

← 현재 가장 가까운 정점을 정점의 수만큼(V) 찾아야 함

← 간선의 수만큼(E) dist를 갱신하게 됨

$O(V)$ (가까운 정점 찾기) + $O(E)$ (dist값 갱신하기)

갱신 정보는 어떤 자료구조에?

배열

- 갱신된 dist는 배열에 바로 삽입할 수 있음 $O(1)$
- dist 중 가장 작은 값을 찾으려면 배열 원소를 하나하나 살펴봐야 함 $O(n)$

우선순위 큐

- 갱신된 dist는 배열에 삽입 후 우선순위에 따라 배치됨 $O(\log n)$
- dist 중 가장 작은 값을 찾으려면 우선순위 큐(min-heap)에서 원소 하나를 삭제하면 됨 $O(\log n)$

갱신 정보는 어떤 자료구조에?

배열

- 갱신된 dist는 배열에 바로 삽입할 수 있음 $O(1)$
- dist 중 가장 작은 값을 찾으려면 배열 원소를 하나하나 살펴봐야 함 $O(n)$

우선순위 큐

- 갱신된 dist는 배열에 삽입 후 우선순위에 따라 배치됨 $O(\log n)$
- dist 중 가장 작은 값을 찾으려면 우선순위 큐(min-heap)에서 원소 하나를 삭제하면 됨 $O(\log n)$

$$O(V \log V + E \log V)$$

/<> 1753번 : 최단경로 - Gold 5

문제

- 방향 그래프에서 주어진 시작점에 대한 다른 모든 정점으로의 최단 경로를 출력

제한 사항

- 정점의 개수 V 는 $1 \leq V \leq 20,000$
- 간선의 개수 E 는 $1 \leq E \leq 300,000$
- 간선의 가중치 w 는 $1 \leq w \leq 10$

*인접 행렬로 구현할 때 필요한 공간은 $20,000 * 20,000 = 4억$ -> 불가능!

* V 와 E 가 최대일 때 각 정점의 간선은 최대 15개! -> 인접 리스트로 구현

예제 입력 1

```
5 6
1
5 1 1
1 2 2
1 3 3
2 3 4
2 4 5
3 4 6
```

예제 출력 1

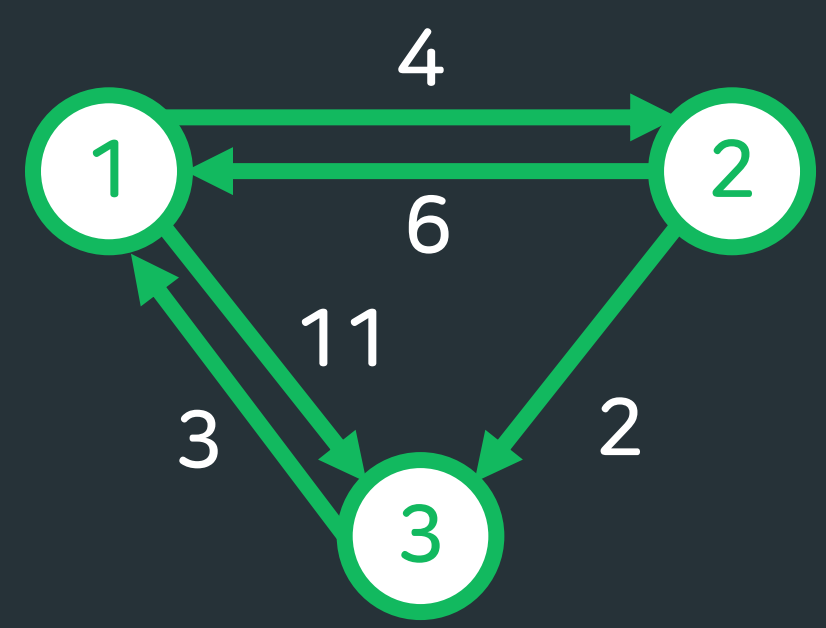
```
0
2
3
7
INF
```

Floyd-Warshall

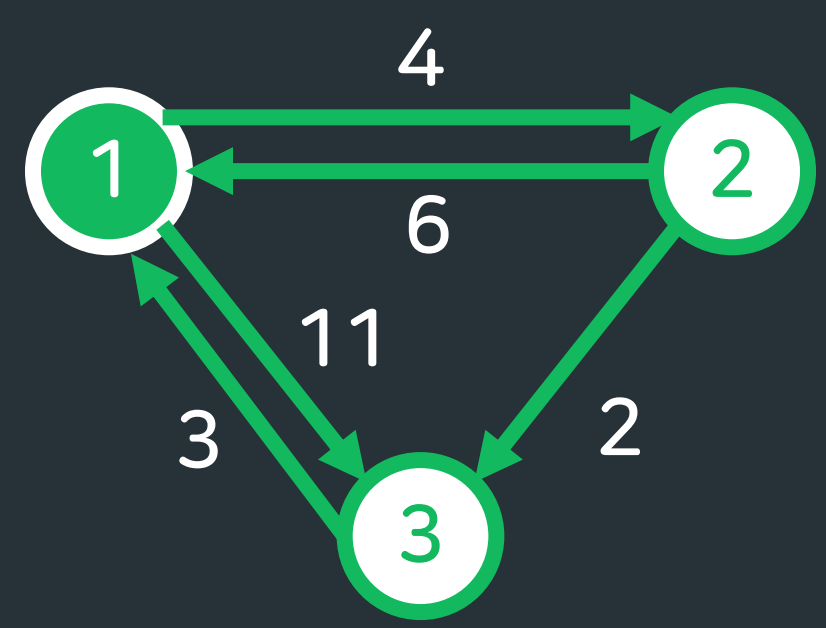
- 가능한 모든 정점 2개의 조합에 대한 최단 경로를 구하는 ASP 알고리즘
- 두 정점 사이의 최단 경로에 포함될 수 있는 모든 정점의 경우를 고려하는 dp 접근
- 정점의 수를 V , 간선의 수를 E 라고 할 때, 시간 복잡도는 $O(V^3)$

$V = 128, E = 8,000$ 일 때

- 다익스트라 V 번 수행 : $128 * (128 * \log(128) + 8,000 * \log(128)) = 7,282,688$
- 플로이드-워셜 : $128 * 128 * 128 = 2,097,152$

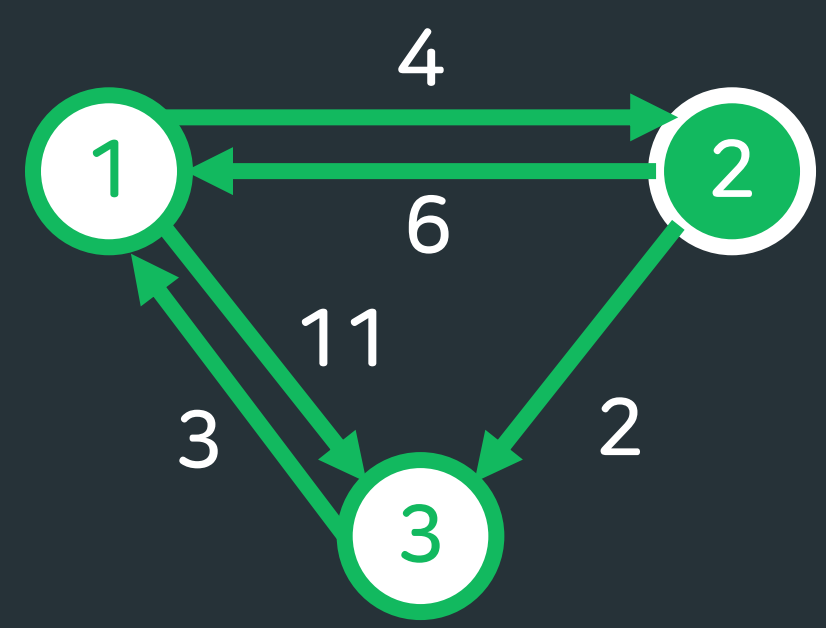


	v1	v2	v3
v1	0	4	11
v2	6	0	2
v3	3	INF	0

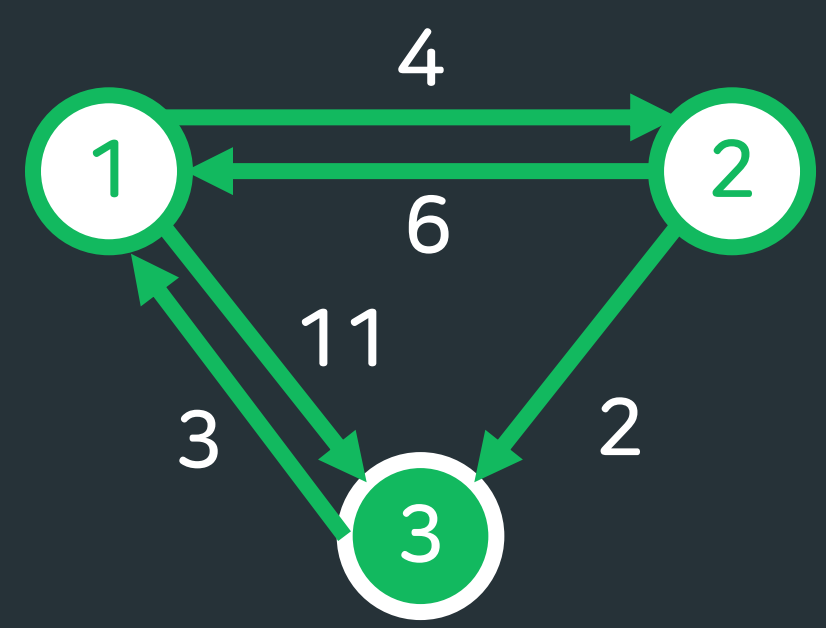


	v1	v2	v3
v1	0	4	11
v2	6	0	2
v3	3	7	0

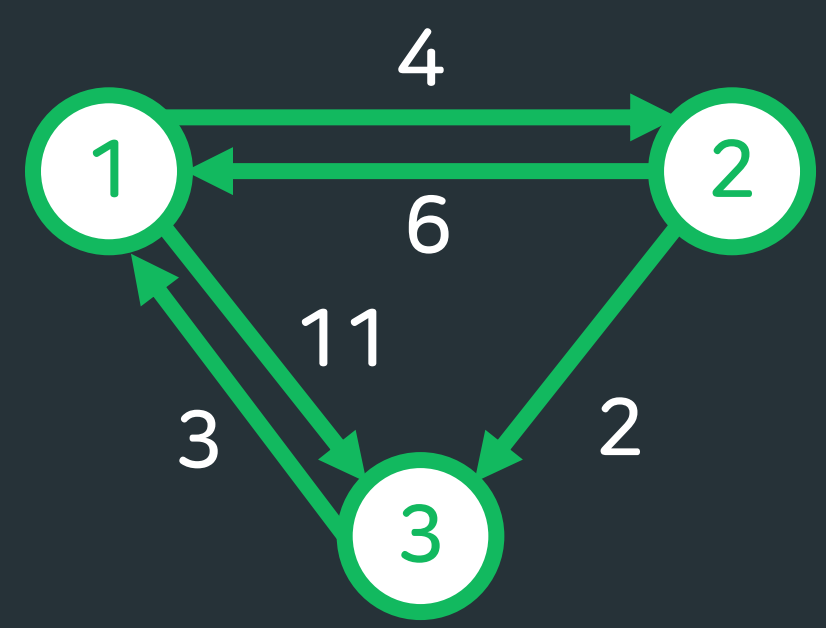
2 < 17



	v1	v2	v3
v1	0	4	6
v2	6	0	2
v3	3	7	0



	v1	v2	v3
v1	0	4	6
v2	5	0	2
v3	3	7	0



	v1	v2	v3
v1	0	4	6
v2	5	0	2
v3	3	7	0

/<> 11404번 : 플로이드 - Gold 4

문제

- 모든 도시의 쌍 (A, B)에 대해 A에서 B로 가는 비용의 최솟값은?

제한 사항

- 도시의 개수 n 은 $1 \leq n \leq 100$
- 도시 사이를 오가는 버스의 수는 $1 \leq m \leq 100,000$
- 이동 비용 c 는 $1 \leq c \leq 100,000$

*최대 100개의 도시에 어떻게 버스의 수가 100,000개?
-> (A, B)에 대해 간선이 여러 개일 수 있다!

예제 입력 1

```
5 14
1 2 2
1 3 3
1 4 1
1 5 10
2 4 2
3 4 1
3 5 1
4 5 3
3 5 10
3 1 8
1 4 2
5 1 7
3 4 2
5 2 4
```

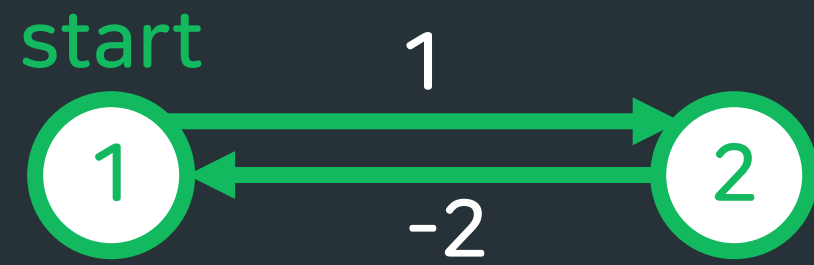
예제 출력 1

```
0 2 3 1 4
12 0 15 2 5
8 5 0 1 1
10 7 13 0 3
7 4 10 6 0
```

Bellman-Ford

- 하나의 시작점에서 모든 정점까지의 최단 경로를 구하는 SSP 알고리즘
- 가중치가 음수일 때 다익스트라 대신 사용
- 모든 정점을 $V-1$ 번 갱신한 뒤, 한 번 더 갱신을 시도하는 브루트포스적 접근
- 정점의 수를 V , 간선의 수를 E 라고 할 때, 시간 복잡도는 $O(VE)$

다익스트라는 왜?



1	2
0	INF

다익스트라는 왜?



1	2
0	1

다익스트라는 왜?



1	2
-1	1

다익스트라는 왜?



1	2
-1	0

다익스트라는 왜?



1	2
-2	0

최단 경로가 무한히 갱신됨
음의 사이클!

늘 불가능한 건 아니예요!



1	2
0	INF

늘 불가능한 건 아니예요!



1	2
0	2

늘 불가능한 건 아니예요!



1	2
0	2

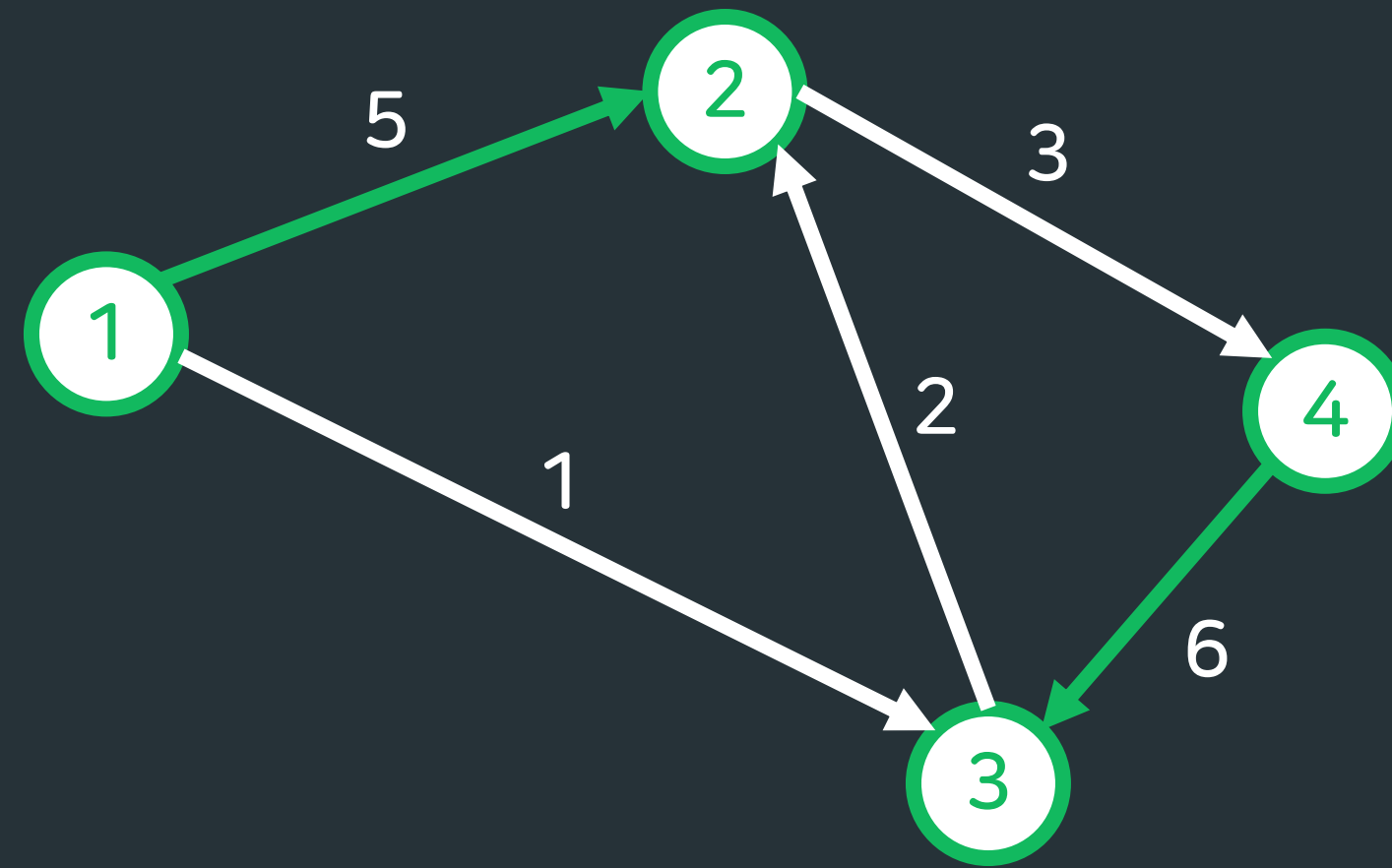
$0 < 1$

늘 불가능한 건 아니예요!

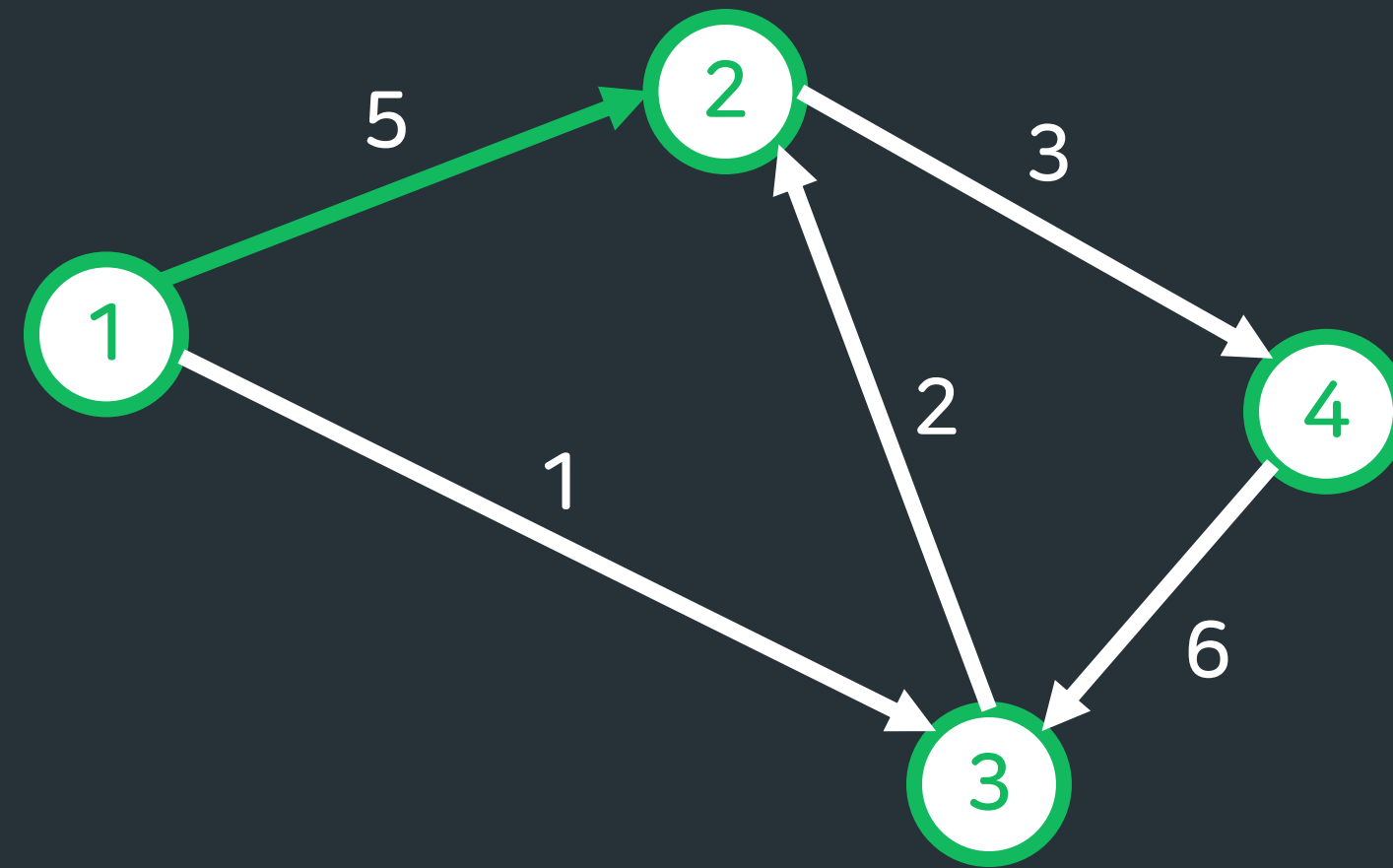


1	2
0	2

그래도 다익스트라는 음의 사이클을 잡아낼 수 없어 사용 불가



정점이 V 개일 때 정점 $A \rightarrow B$ 의 경로에는 최대 $V-1$ 개의 간선이 있을 수 있음



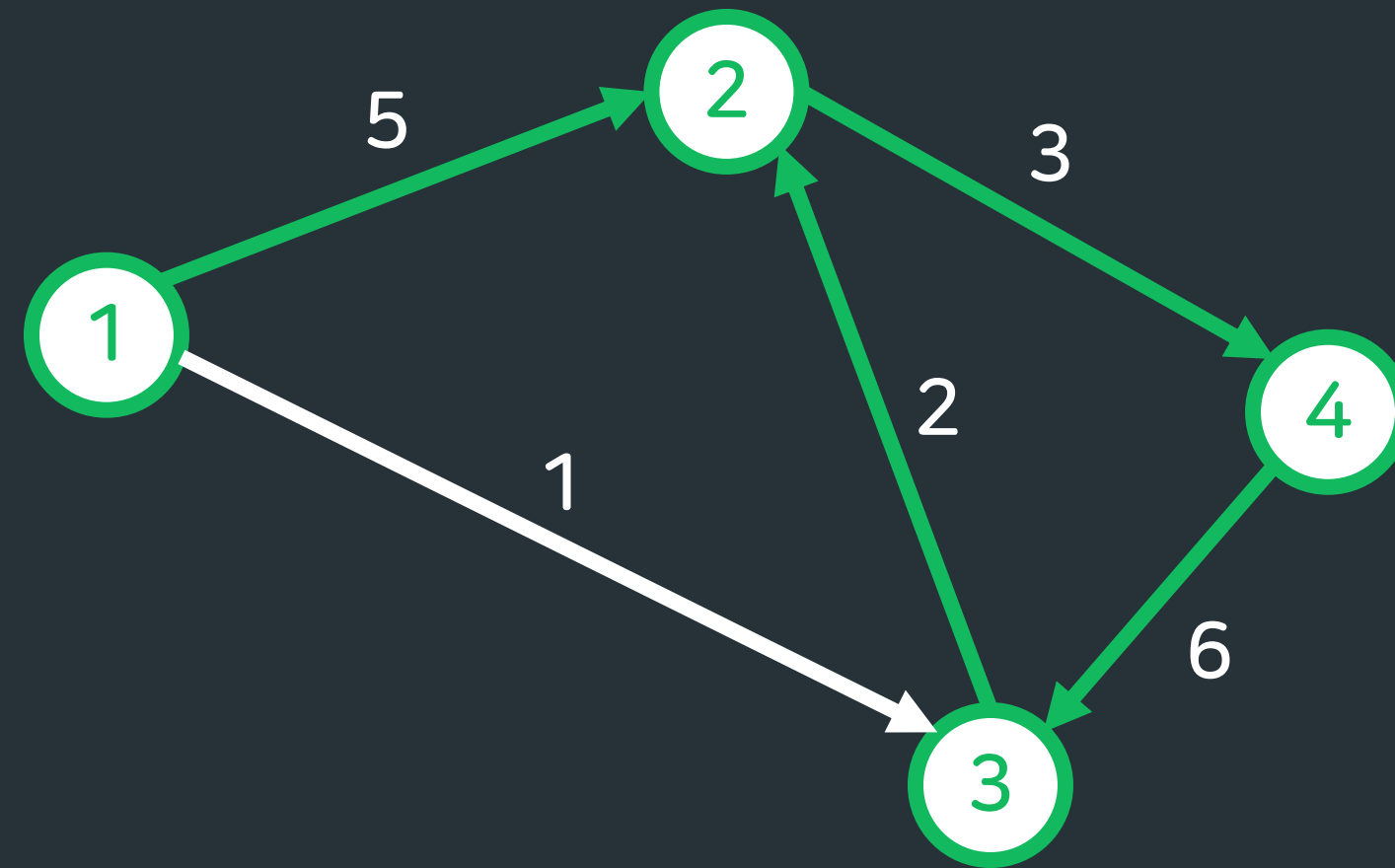
정점이 V 개일 때 정점 $A \rightarrow B$ 의 경로에는 **최대 $V-1$ 개의 간선**이 있을 수 있음

그 이상의 간선을 사용하면 **사이클** 형성!

사이클이 생겼다

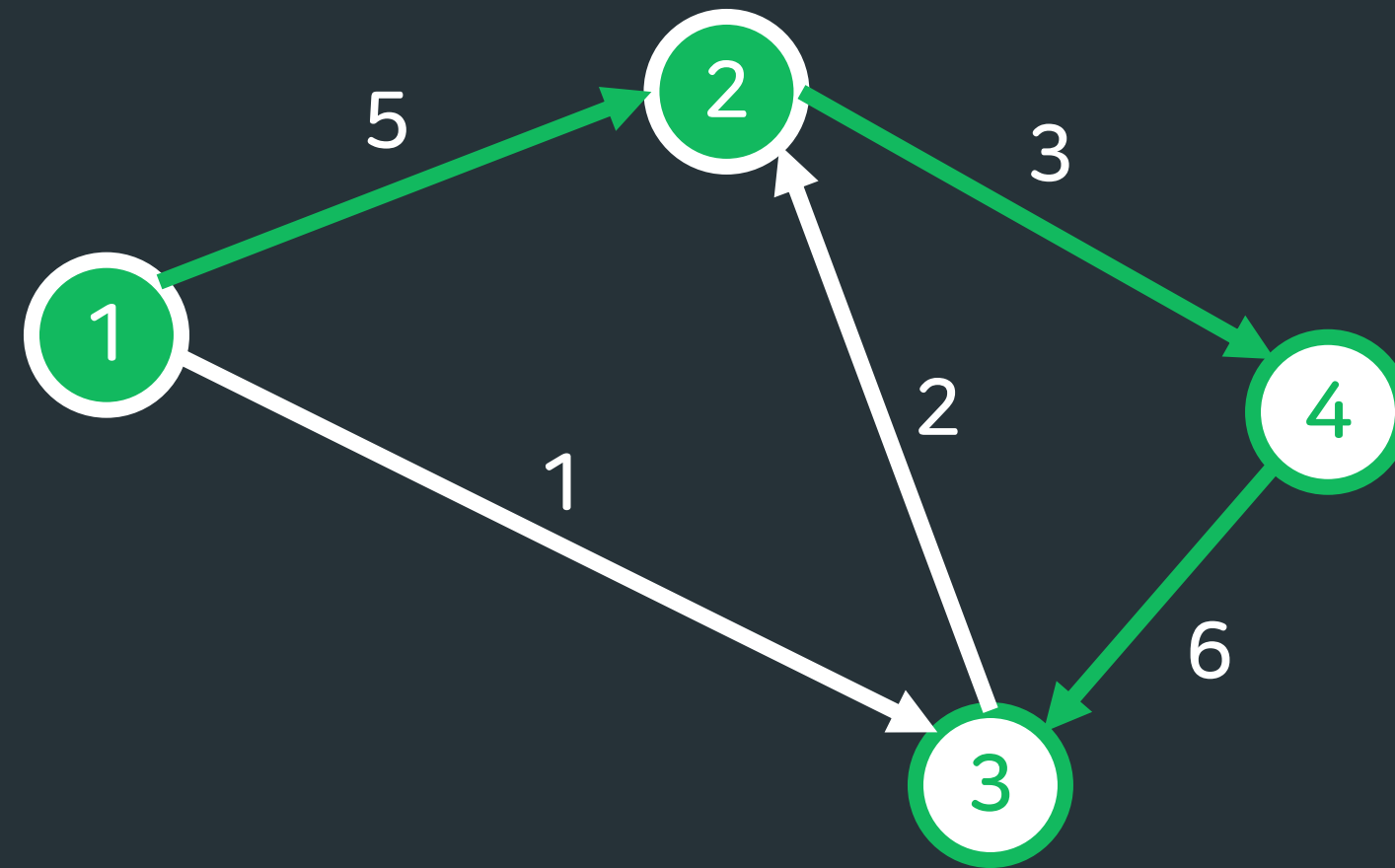
= 최단 경로를 이루는 간선이 V 개 이상인 정점 A, B가 있다
= V 번 이상 갱신되는 간선이 있다

좀 더 직관적으로 생각해볼까요?



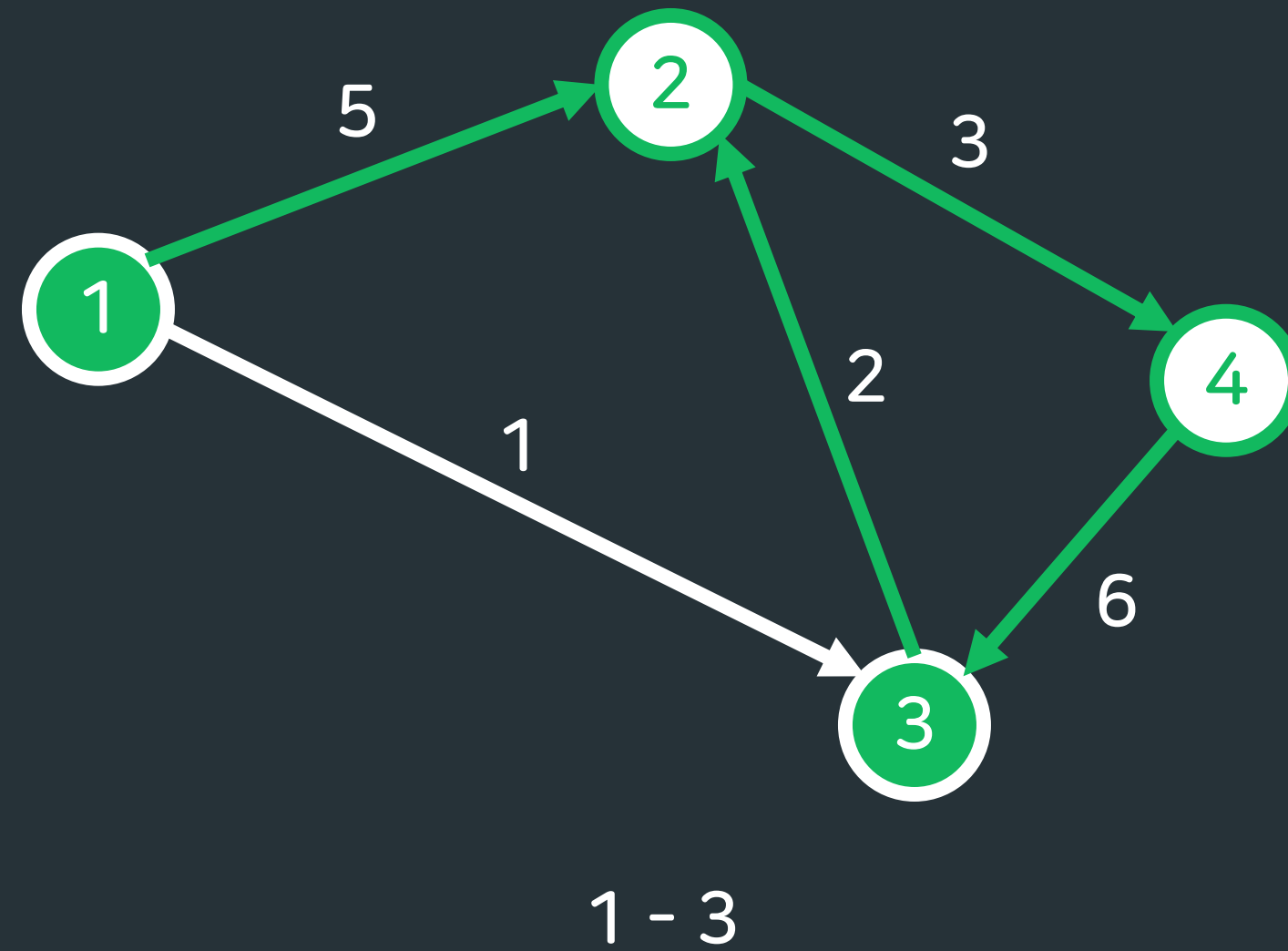
1번 정점에서 모든 정점으로의 최단 경로를 구할 때
하얀색으로 칠한 간선은 최대 **몇 번** 사용될까요?

좀 더 직관적으로 생각해볼까요?

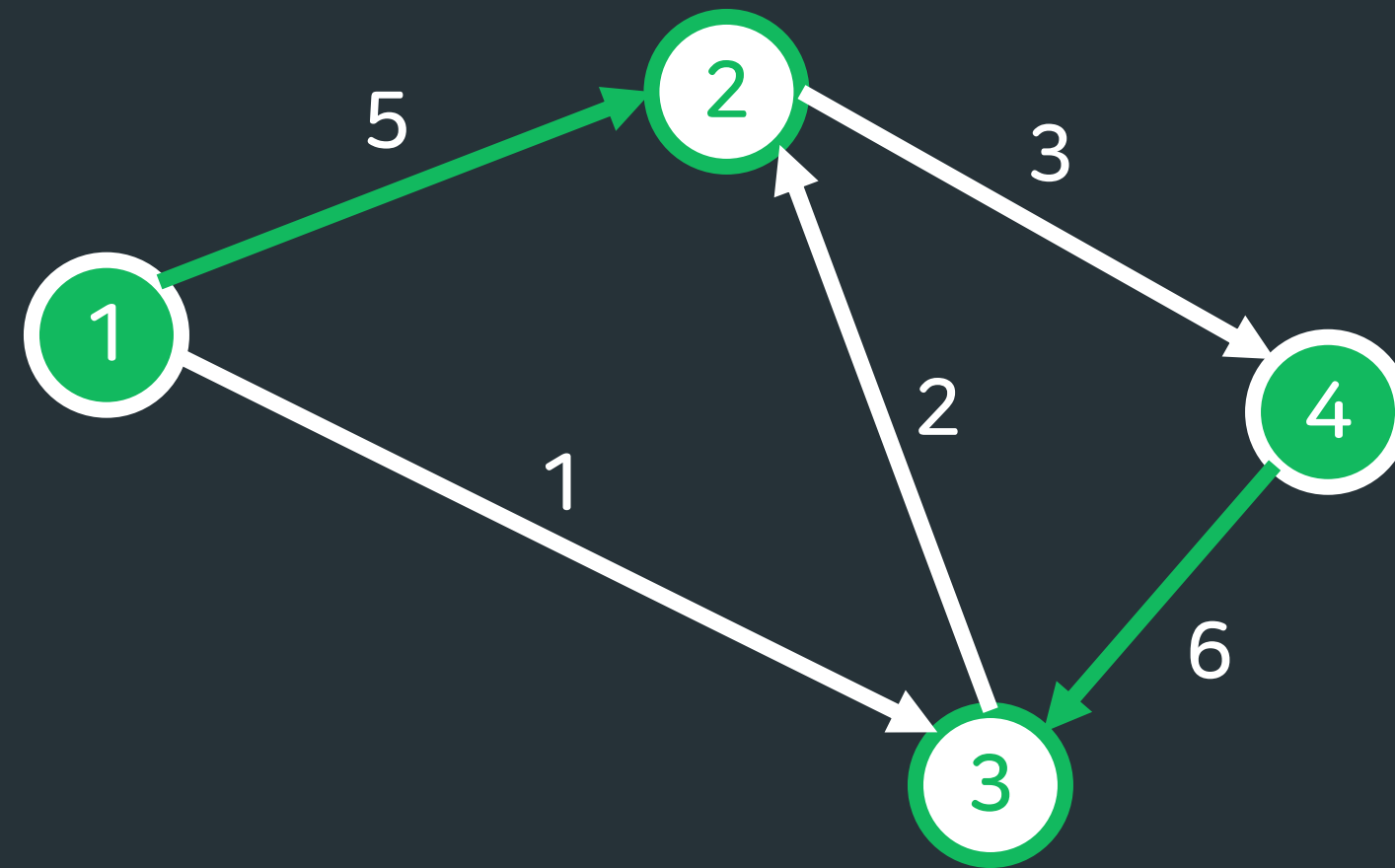


1 - 2

좀 더 직관적으로 생각해볼까요?

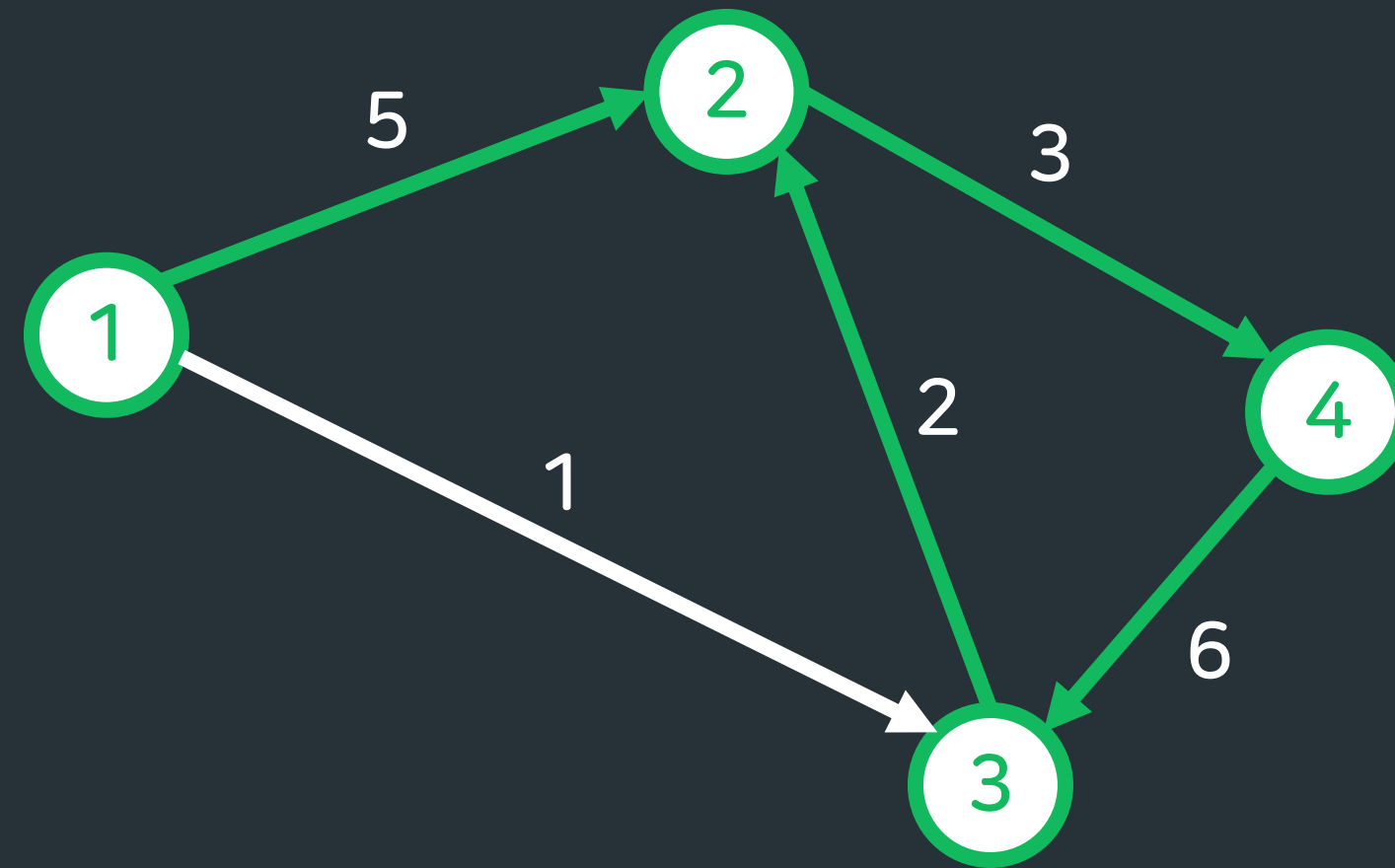


좀 더 직관적으로 생각해볼까요?

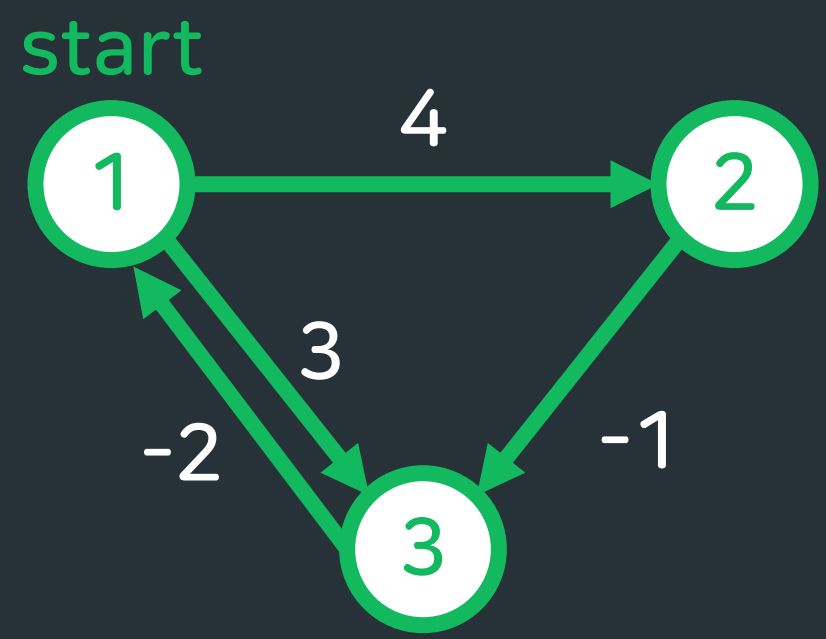


1 - 4

좀 더 직관적으로 생각해볼까요?

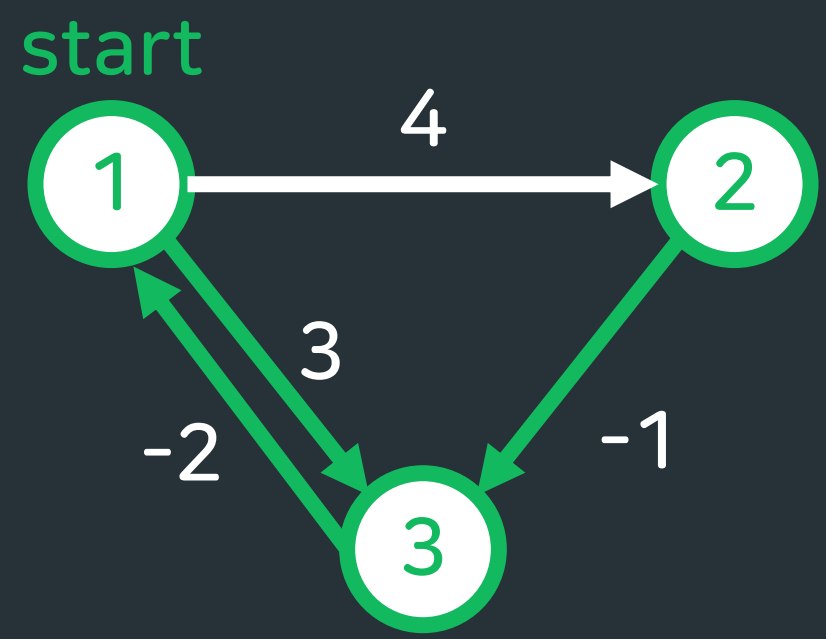


최단 경로를 구할 정점의 수가 $V-1$ 개이므로
사이클이 없다면 특정 간선은 최대 $V-1$ 번만 사용됨!



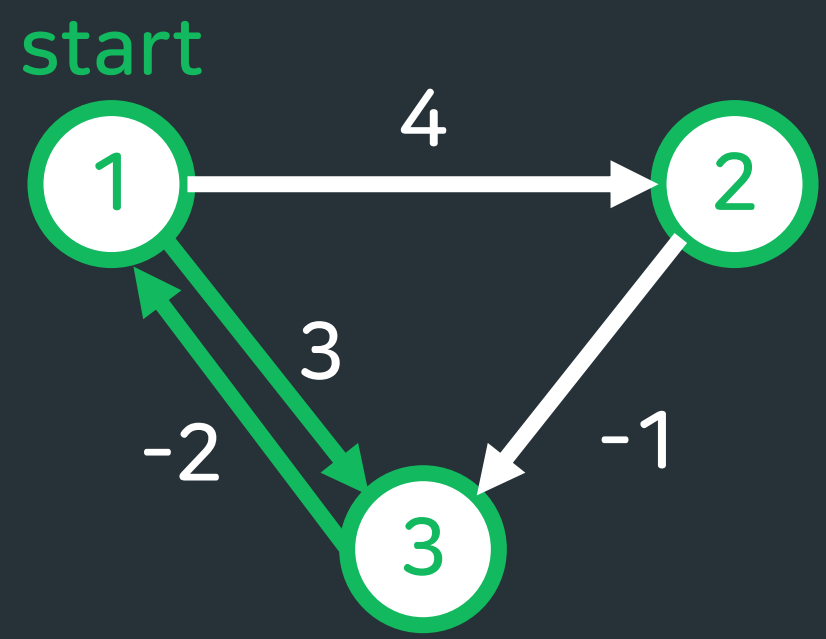
1	2	3
0	INF	INF

첫번째 반복



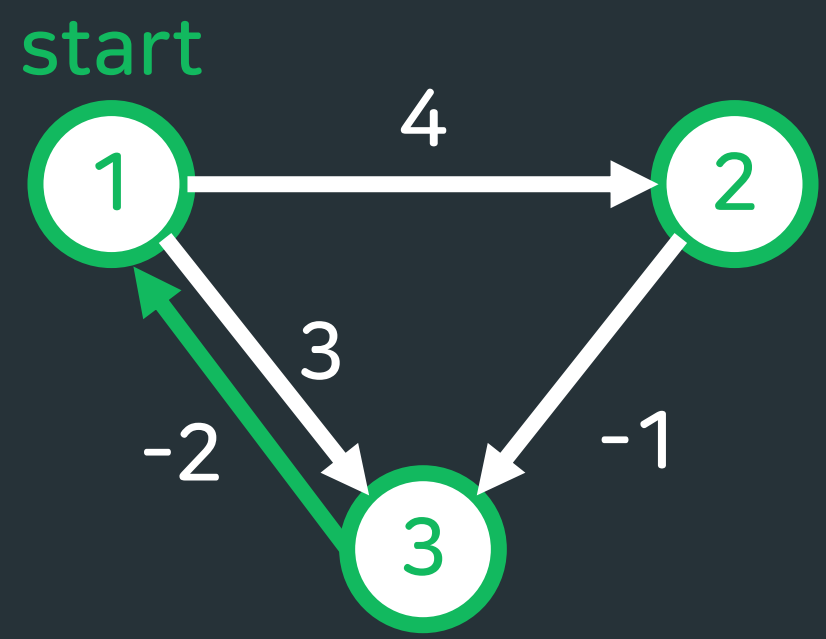
1	2	3
0	4	INF

첫번째 반복



1	2	3
0	4	3

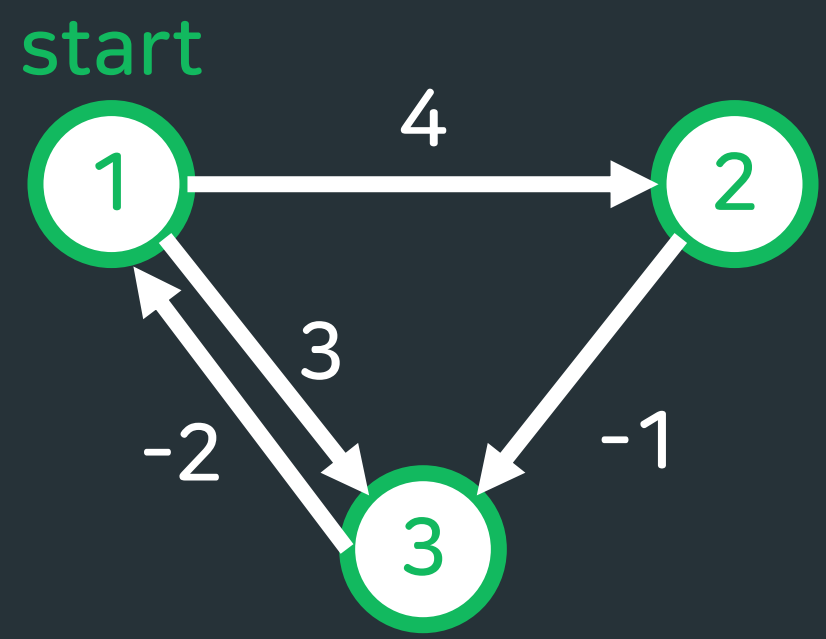
첫번째 반복



1	2	3
0	4	3

3 = 3

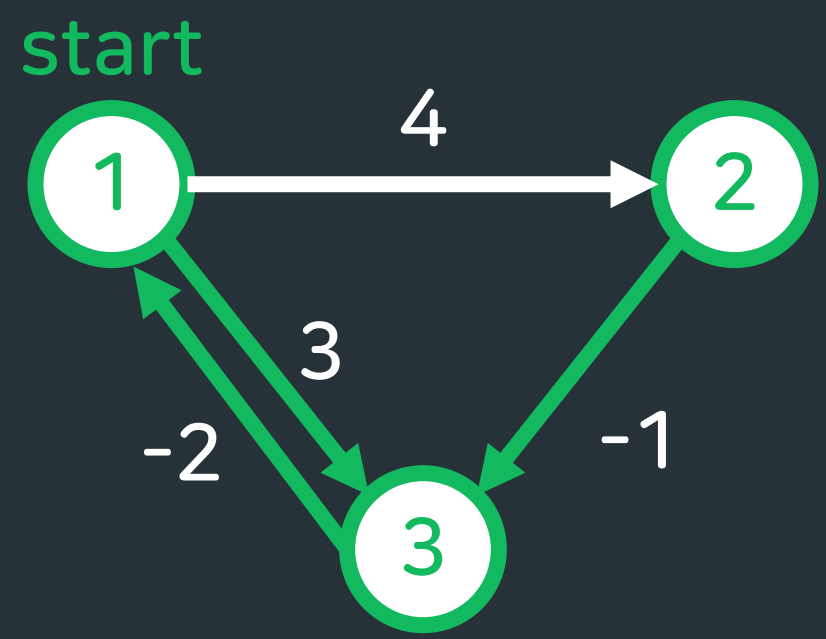
첫번째 반복



1	2	3
0	4	3

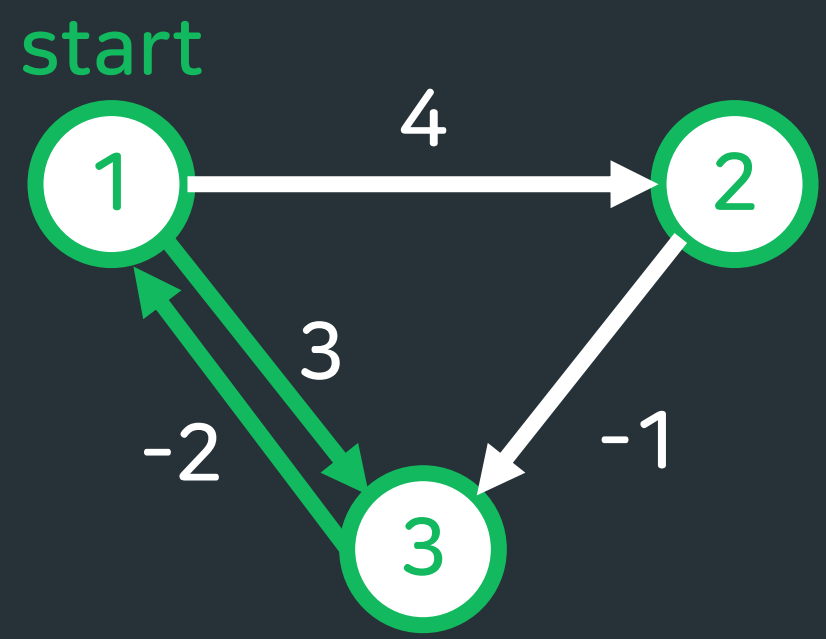
$0 < 1$

첫번째 반복



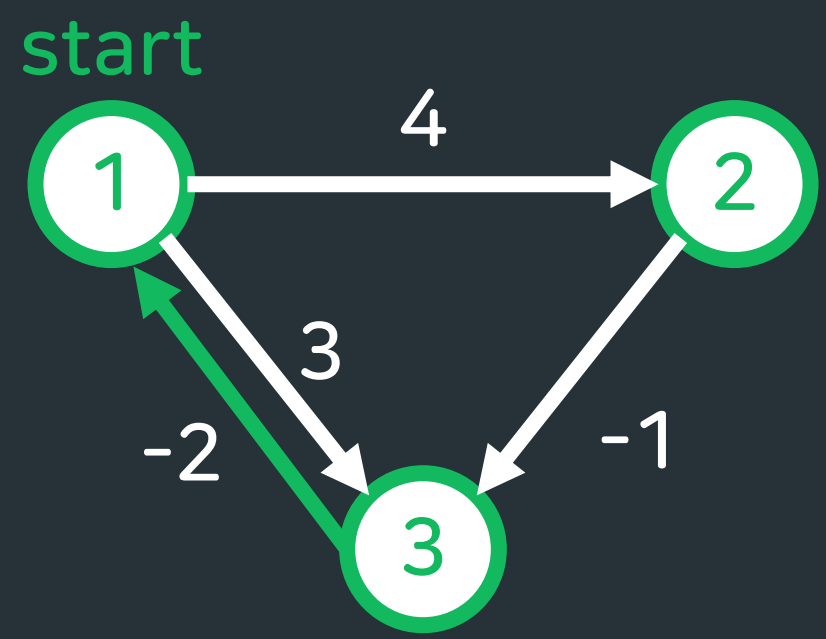
1	2	3
0	4	3

두번째 반복



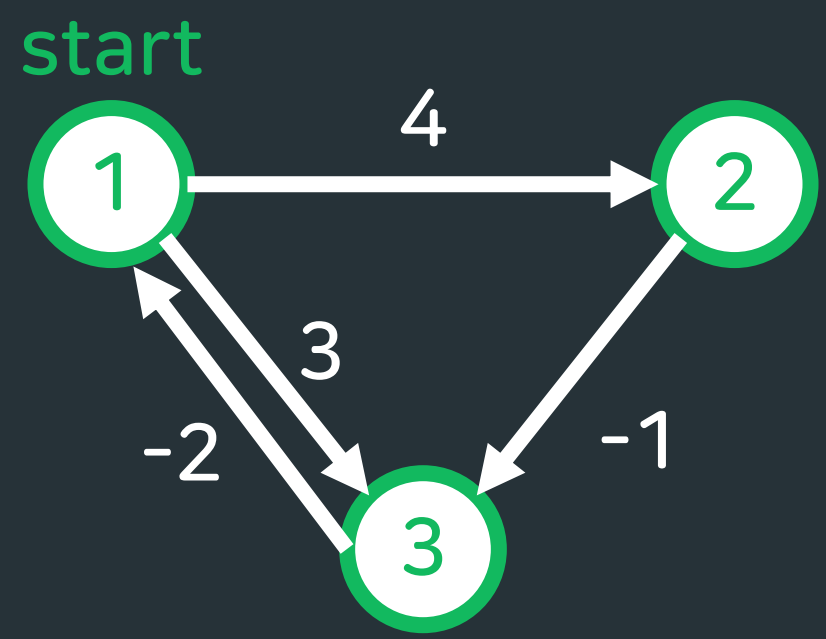
1	2	3
0	4	3

두번째 반복



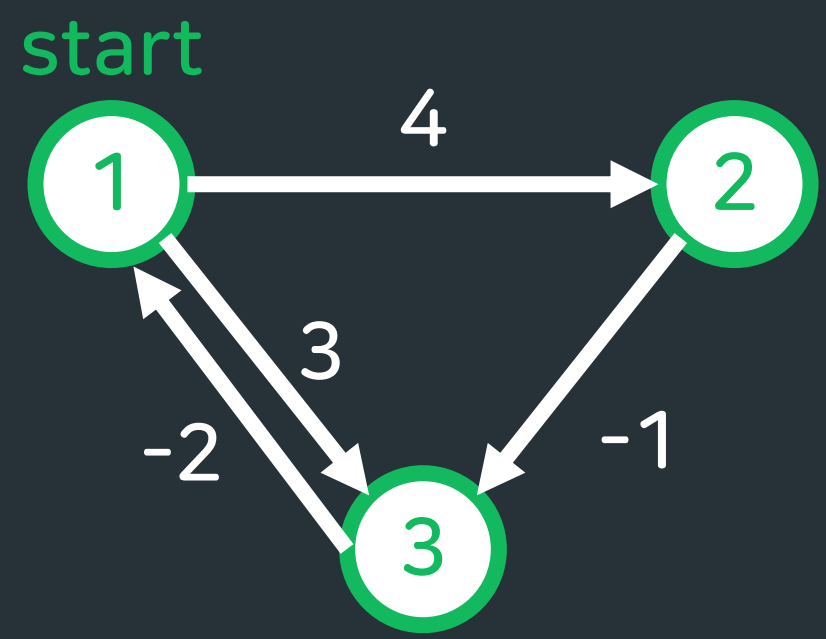
1	2	3
0	4	3

두번째 반복



1	2	3
0	4	3

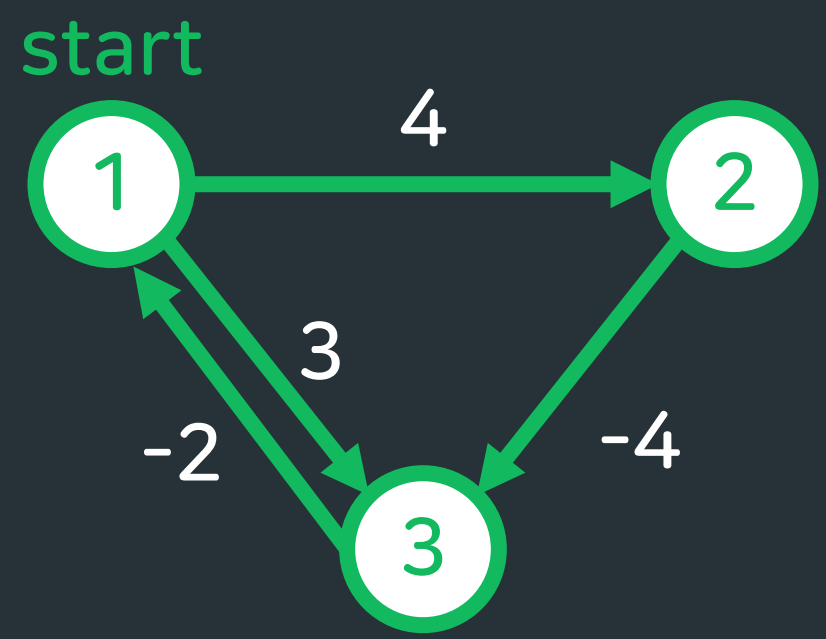
두번째 반복



1	2	3
0	4	3

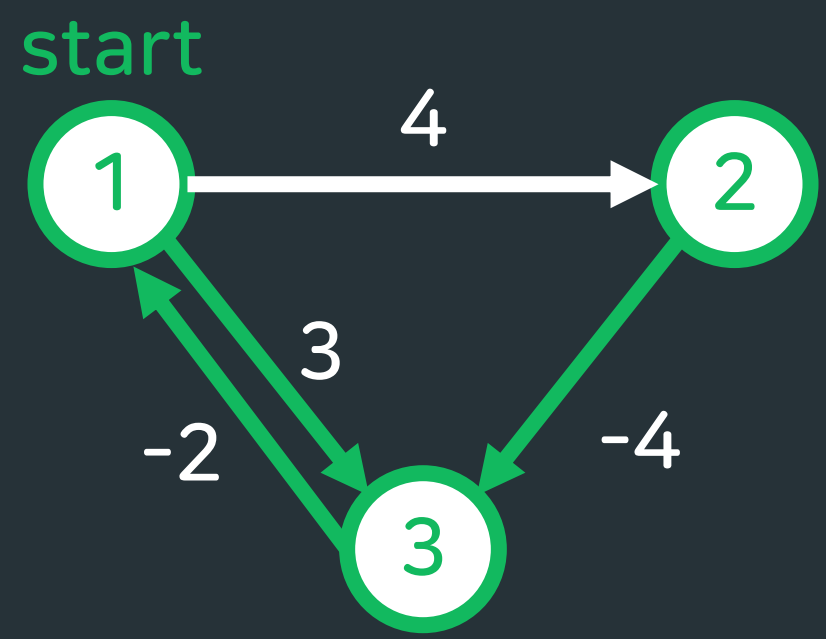
더 이상 값이 갱신되지 않음
= 음의 사이클 없음

두번째 반복



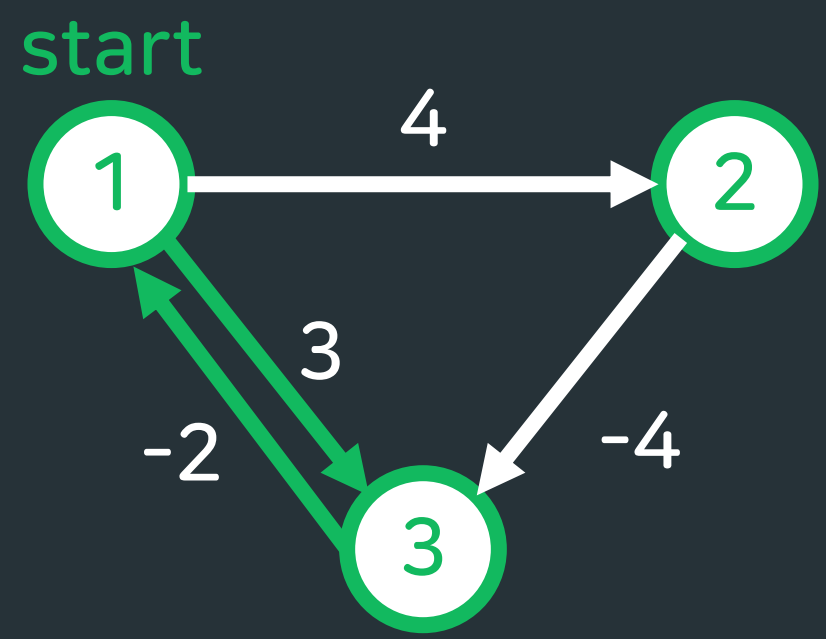
1	2	3
0	INF	INF

첫번째 반복



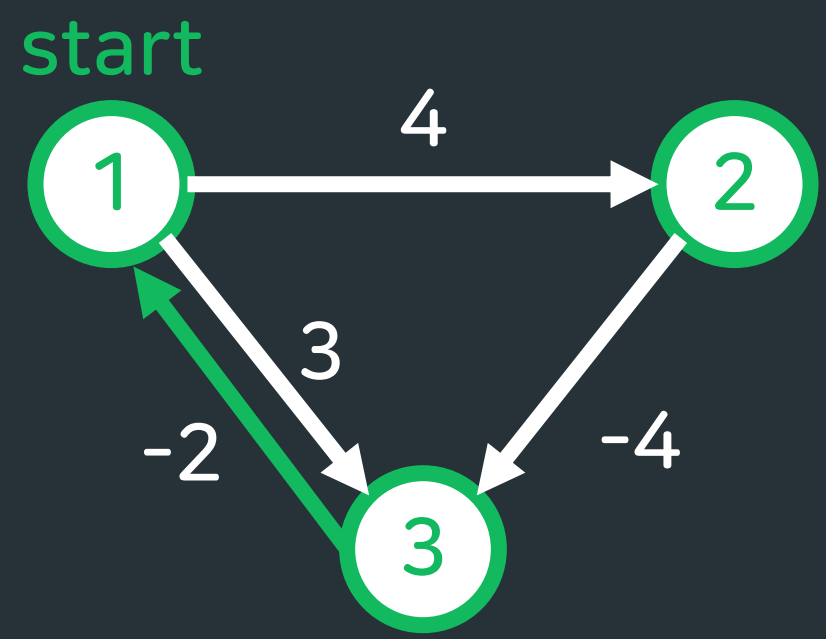
1	2	3
0	4	INF

첫번째 반복



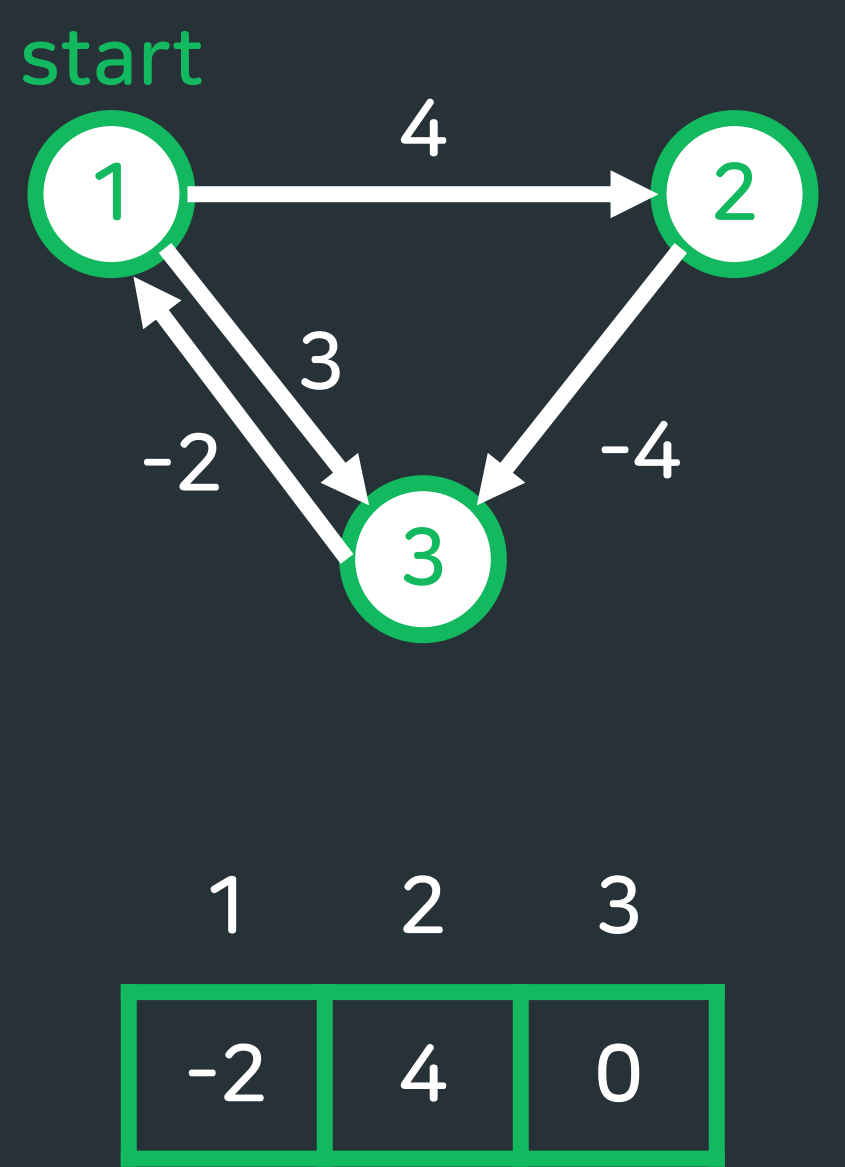
1	2	3
0	4	0

첫번째 반복

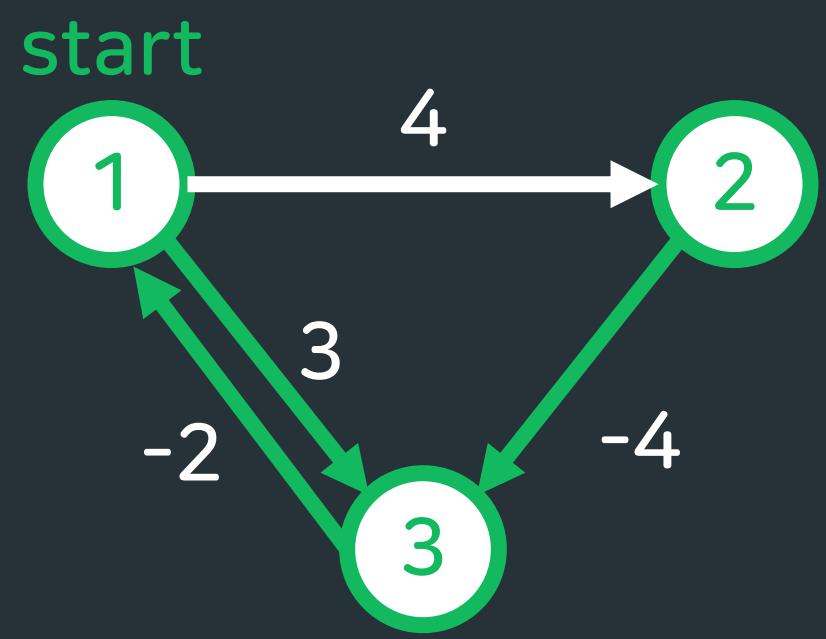


1	2	3
0	4	0

첫번째 반복

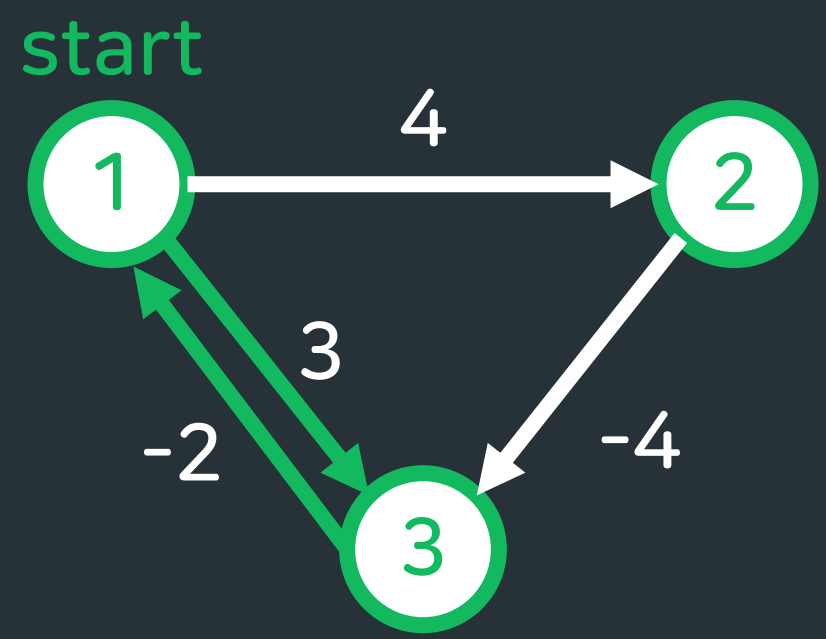


첫번째 반복



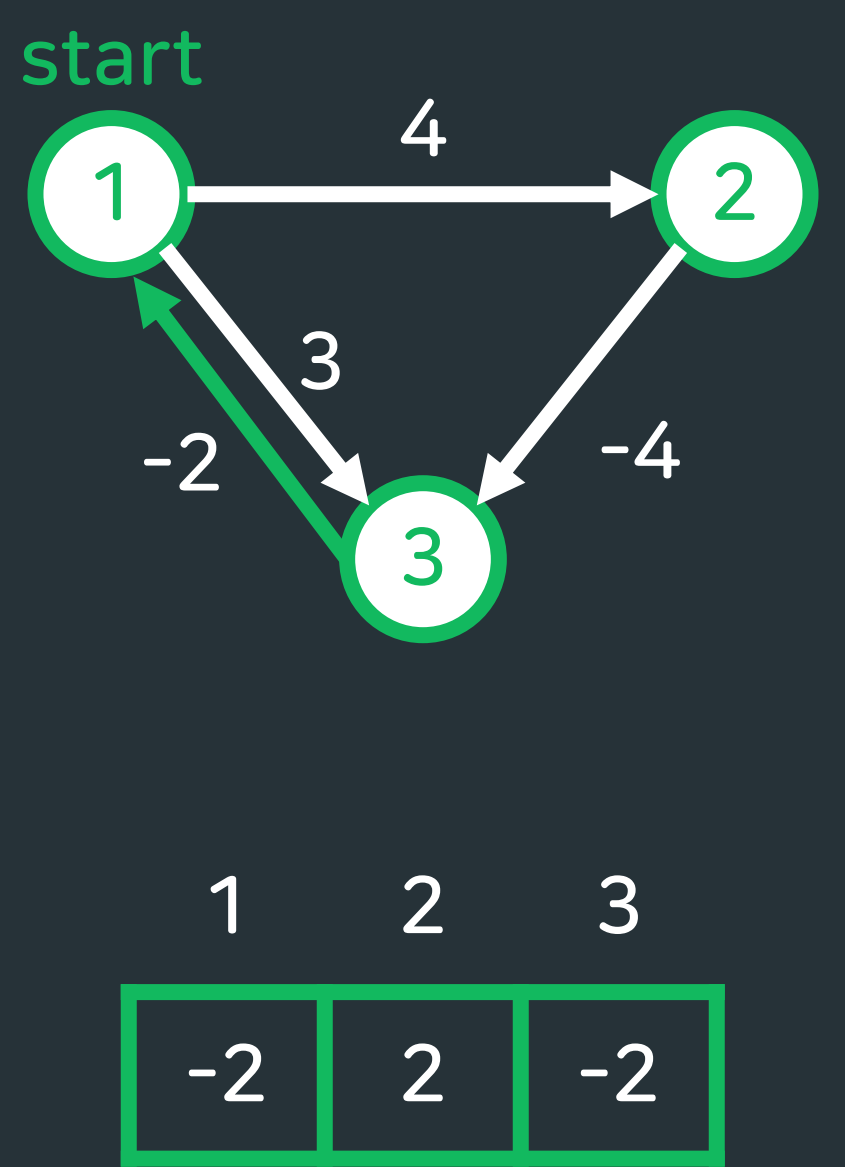
1	2	3
-2	2	0

두번째 반복

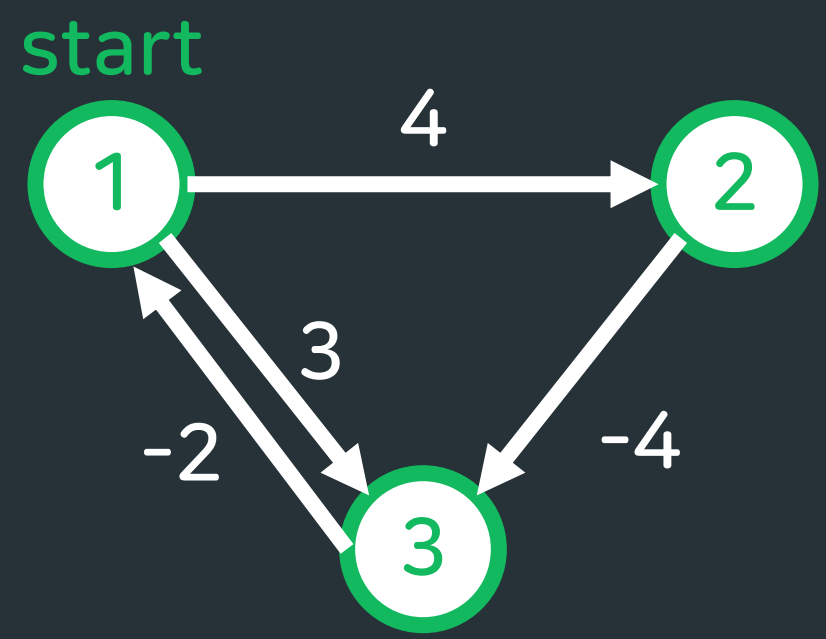


1	2	3
-2	2	-2

두번째 반복

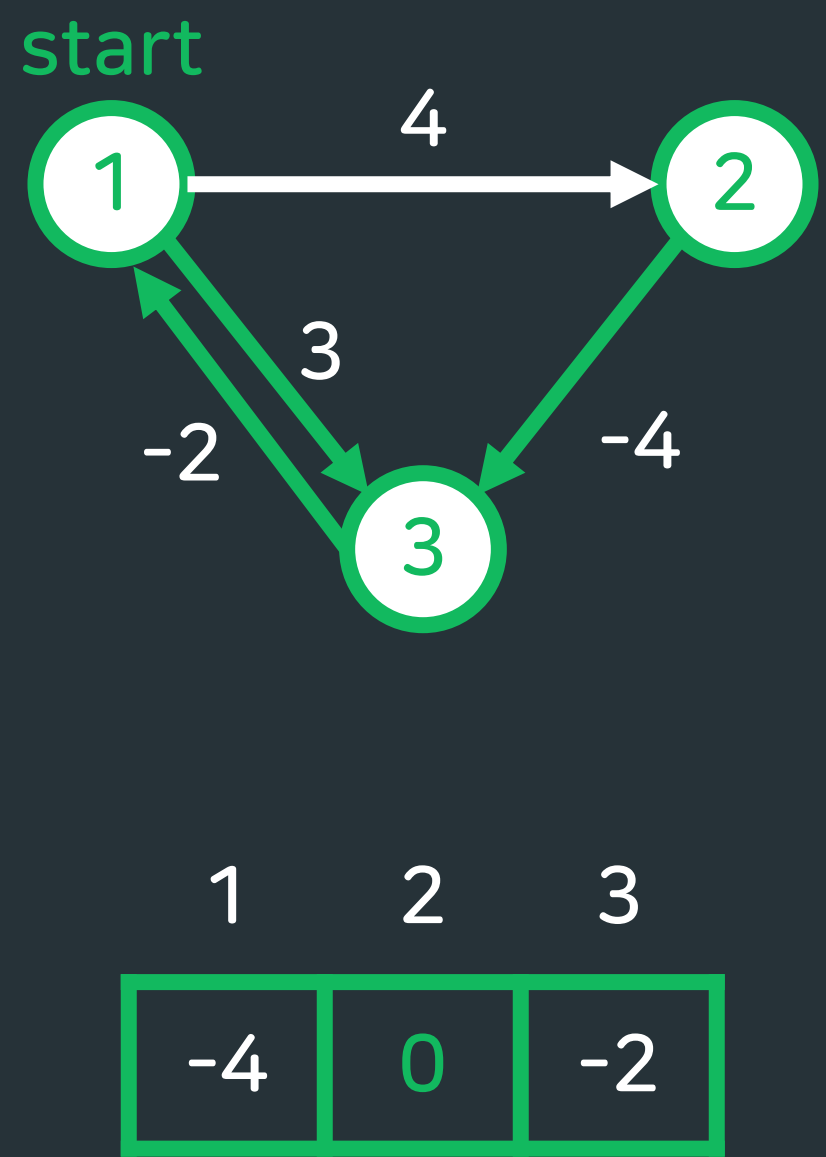


두번째 반복

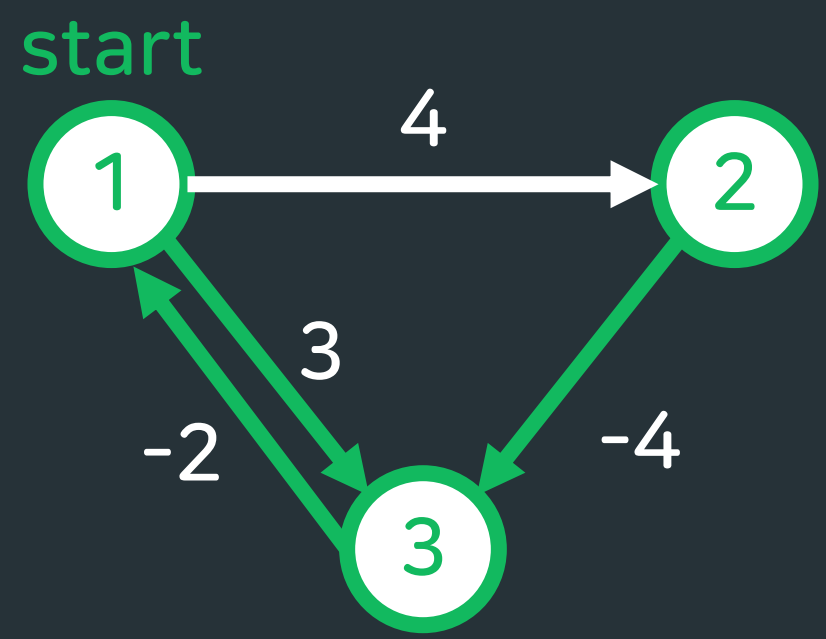


1	2	3
-4	2	-2

두번째 반복



세번째 반복 : 여기서 갱신이 또 일어나면 음의 사이클



1	2	3
-4	0	-2

갱신 확인
= 음의 사이클 있음

세번째 반복



```
for (V-1회 루프){  
    for (모든 간선에 대해)  
        간선을 사용하여 최단 경로 갱신  
}
```

← $(V-1) * E$

```
for (모든 간선에 대해){  
    if (간선을 사용하여 최단 경로가 갱신됨)  
        음의 사이클 존재!  
}
```

← E

$O(VE)$

/<> 11657번 : 타임머신 - Gold 4

문제

- 1번 도시에서 출발해 나머지 모든 도시로 가는 가장 빠른 시간은?
- 단, 순간이동과 타임머신으로 걸리는 시간이 음수인 경우가 있을 수 있음
- 어떠한 도시로 가는 시간을 무한히 오래 전으로 돌릴 수 있음 (음의 사이클)

제한 사항

- 도시의 개수 N 은 $1 \leq N \leq 500$
- 도시 사이를 오가는 버스의 수는 $1 \leq M \leq 6,000$
- 이동 비용 C 는 $-10,000 \leq C \leq 10,000$

예제 입력 1

```
3 4
1 2 4
1 3 3
2 3 -1
3 1 -2
```

예제 출력 1

```
4
3
```

예제 입력 3

```
3 2
1 2 4
1 2 3
```

예제 출력 3

```
3
-1
```

예제 입력 2

```
3 4
1 2 4
1 3 3
2 3 -4
3 1 -2
```

예제 출력 2

```
-1
```

정리

- 간선에 **가중치**가 있는 그래프의 최단 경로는 **BFS**를 사용할 수 없음
- 하나의 출발지, **모든** 도착지에 대한 최단 경로는 **다익스트라**, **벨만-포드**
- **모든** 출발지, **모든** 도착지에 대한 최단 경로는 **플로이드-워셜**
- **다익스트라** 구현시 **프림** 알고리즘(최소 신장 트리 알고리즘)과 헷갈리지 않도록 주의!
- 벨만-포드는 코테에서 한 번도 본 적 없어요. (개인 경험)


이것도 알아보세요

- 다익스트라의 시간 복잡도를 $O(V \log E + E \log E)$ 라고 하는 글도 있고, $O(E \log V)$ 라고 하는 글도 있어요. 사실 다 같은 얘기를 다르게 기술한 것이지만 그 **차이**를 이해하면 알고리즘에 대해 더 잘 이해할 수 있어요
- 가중치가 **두 가지 종류**로만 주어진다면 어떻게 될까요? 여기에도 그냥 **다익스트라**를 적용할까요?

필수

- /<> 15685번 : 드래곤 커브 - Gold 4
- /<> 1063번 : 킹 - Silver 4

3문제 이상 선택

- /<> 1238번 : 파티 - Gold 3
- /<> 1504번 : 특정한 최단 경로 - Gold 4
- /<> 1613번 : 역사 - Gold 3
- /<> 4485번 : 녹색 옷 입은 애가 젤다지? - Gold 4
- /<> 13549번 : 숨바꼭질 3 - Gold 5
-  2021 KAKAO BLIND RECRUITMENT : 합승 택시 요금 - Level 3