#4주차 알고리즘반

SHORTEST PATH

T. 강건

Graph

오늘도 역시나 그래프를 사용한다.

Graph

3주차 내용을 복습해주세요.

std::priority_queue

아직도 stl을 잘 모르시는 분들은 멘토를 통해 꼭 stl과 친해지길 바래요!

priority_queue_간단간단 완전 초간단 정리~~(야매 조심)~~

- **큐는 큐인데 front에 항상 가장 큰 값이 들어있다.**

  → 데이터를 push하면 알아서 정렬해주는 좋은친구!!!

- **#include <queue> 필요**

- **선언방법:**

  priority_queue<자료형> 이름;

priority_queue_추가적인 내용

https://www.google.com/search?ei=1nK2XrqDBIHU-QaFua6AAw&q=std%3A%3Apriority_queue&oq=std%3A%3Apriority_queue&gs_lcp=CgZwc3ktYWIQA1DWNFjWNGDhOGgAcAB4AIABb4gB1QGSAQMwLjKYAQCgAQGqAQdnd3Mtd2l6&sclient=psy-ab&ved=0ahUKEwi6xouOt6bpAhUBat4KHYWcCzAQ4dUDCAs&uact=5

더 알고싶은 학생들은
구글에 쳐보기만 자세히 나오니 참고해주세요!

2-1 자료구조 수업에서 배웁니다.(Heap)

ppt만으로 모든 것을 전달하기도 어렵고, 지금은
알고리즘 구현에서의 "활용"이 중요하기 때문에
강의자료에서는 자세히 다루지 않겠습니다.

예제

1927 – 최소 힙

1927_최소 힙

- 다음 슬라이드에 해답이 있습니다.

- 오늘 배우는 중요한 내용은 아니고 pq사용법을 위한 문제이니
  베끼는 것에 죄책감을 느끼지 마시고 맘껏 베껴요 ㅎㅎㅎ

1927_최소 힙

기본적으로 priority_queue는 MaxHeap입니다.

pq안에 내림차순으로 정렬되지요.

우리는 가장 작은 값을 찾으려고 합니다.

입력이 자연수 뿐이므로

pq에 -1을 곱한 수를 넣어주고
뺄때 (-)부호를 없애주면

정렬방법 변경 없이 편하게 할 수 있겠죠???

```cpp
#include <iostream>
#include <queue>
using namespace std;

priority_queue<int> pq;

int N, x;

int main(void)
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    cin >> N;
    for( int n = 1; n <= N; n++ )
    {
        cin >> x;
        if( x == 0)
        {
            if ( pq.empty() )
            {
                cout << "0\n";
            }
            else
            {
                cout << -pq.top()<<'\n';
                pq.pop();
            }
        }
        else
        {
            pq.push(-x);
        }
    }

    return 0;
}
```

Dijkstra's Algorithm

가장 기본이 되는 알고리즘

Dijkstra_개념

- 하나의 시작점에서 모든 정점의 최단경로를 구할 수 있다!
- 음수 가중치가 존재할 경우 사용할 수 없다!

Dijkstra_개념

0. 출발점으로부터 가장 가까운 정점을 구한다.

1. 그 정점까지의 최단 거리를 확정한다.

2. 최단거리가 확정된 정점과 가장 가까운 정점을 찾아 그 지점까지 최단거리를 확정한다.

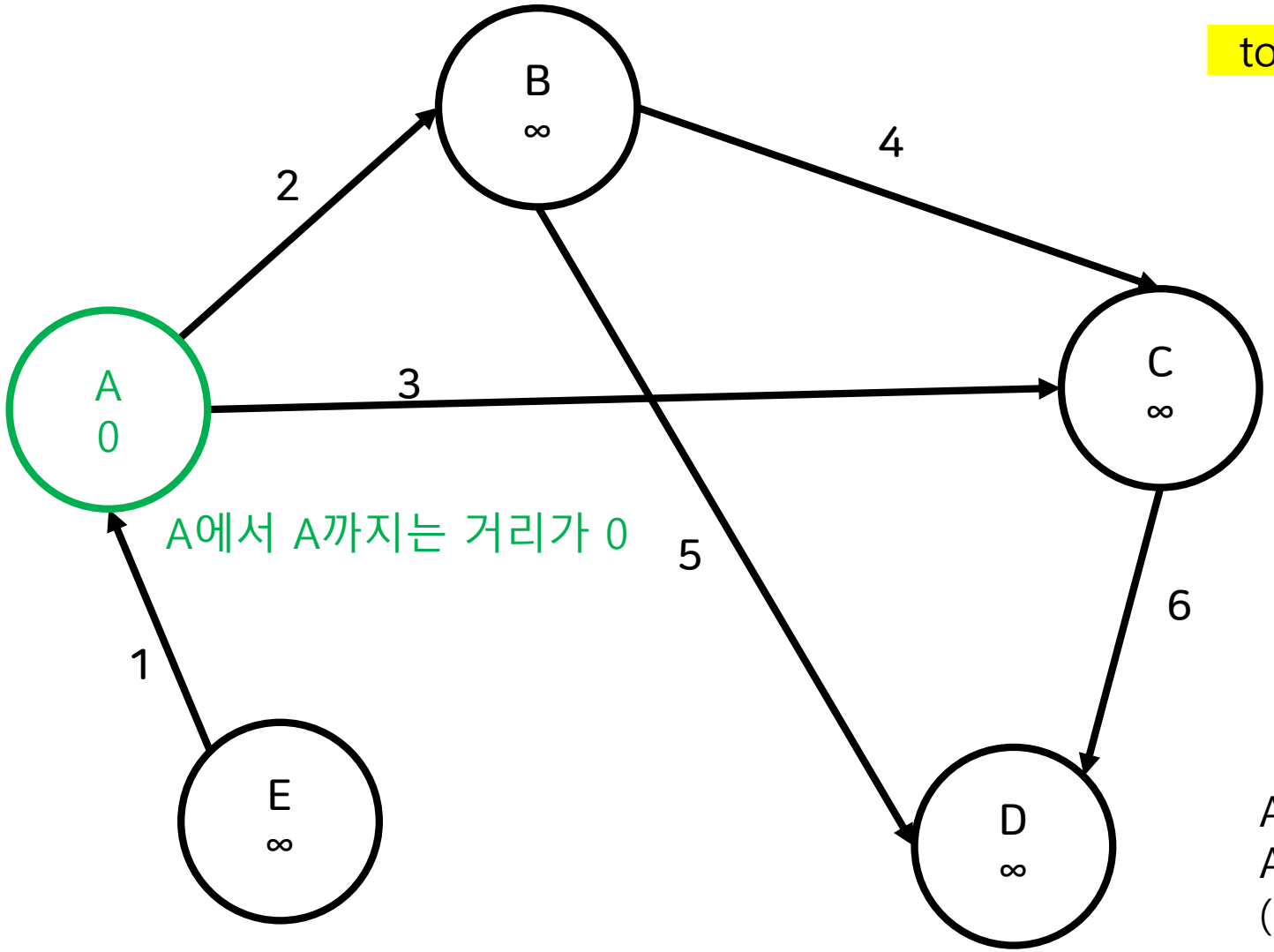
3. 2를 끝날때까지 반복한다.

모르겠죠??

Dijkstra_예시

시작점: A

∞ : 정보 없음,
도달 시간 무한대



priority_queue

top

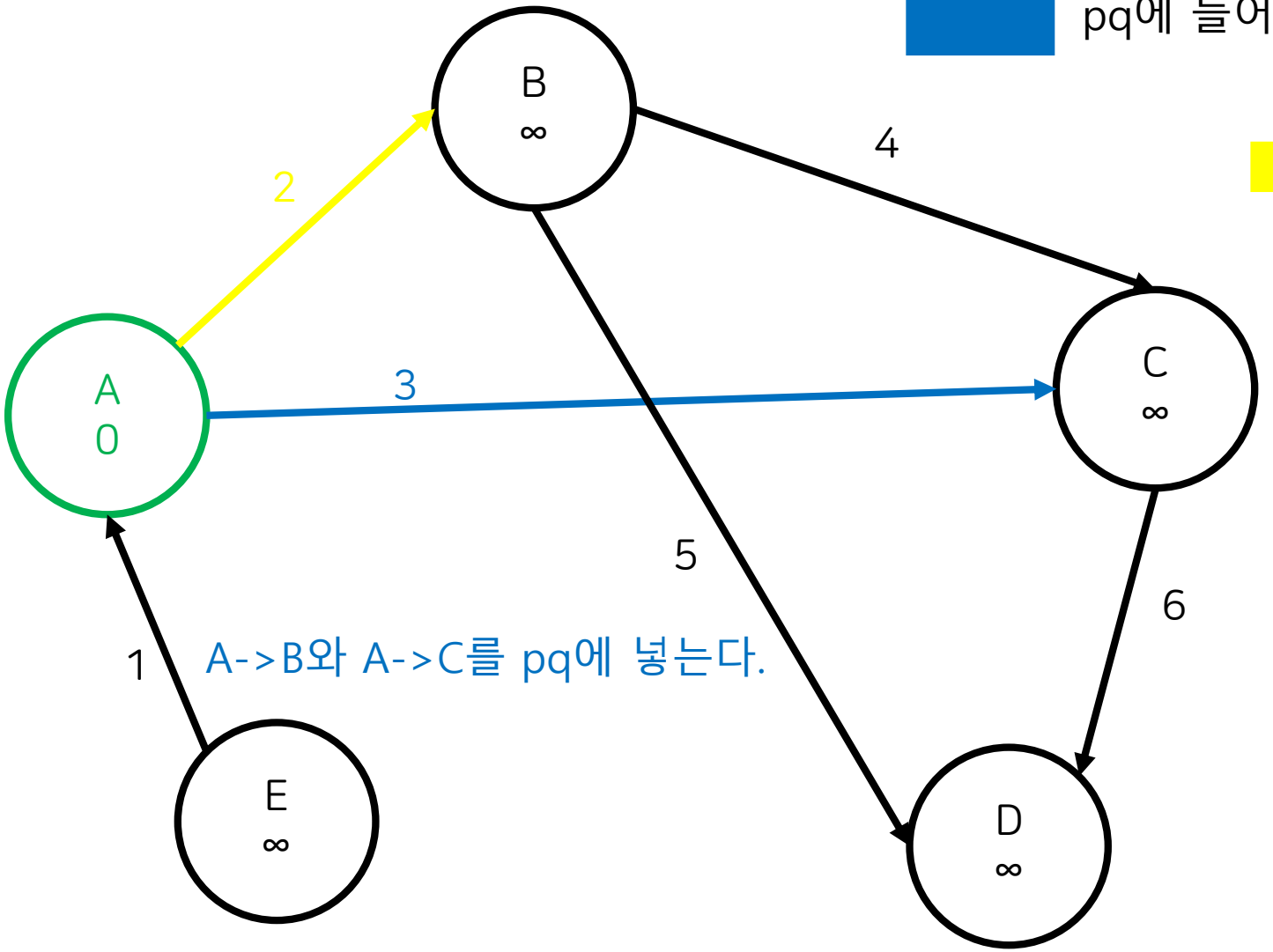
From	To	Weight

A까지의 최단 경로가 확정 되었으므로
A에서 갈 수 있는 길들을 pq에 넣어준다.
(다음 슬라이드)

Dijkstra_예시

pq: priority_queue

pq에 들어간 간선



priority_queue

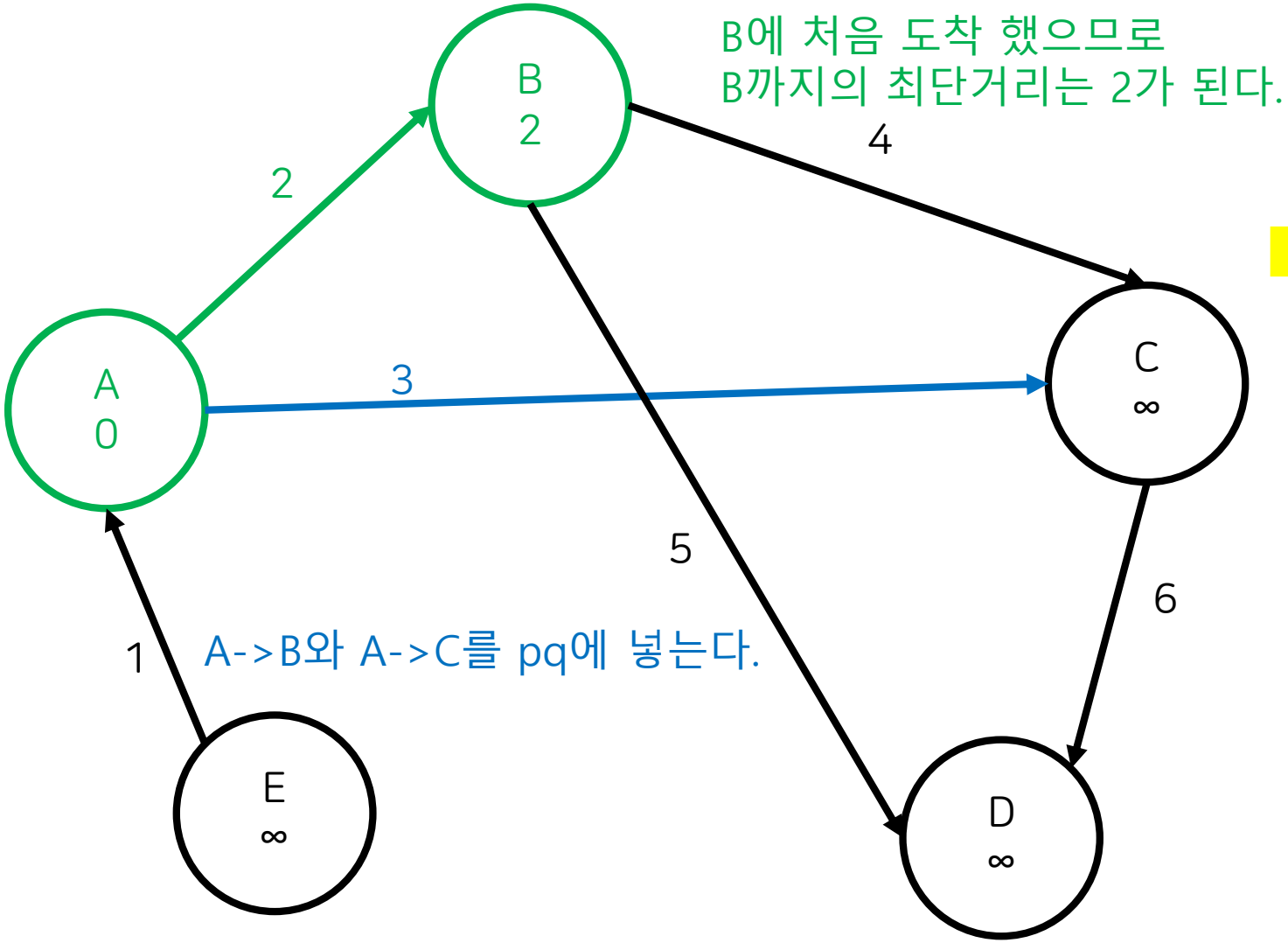
top

From	To	Weight
A	B	2
A	C	3

Weight 기준으로 정렬되어 있다.

Dijkstra_예시

top에 있는 경로를 이용해 최단경로를 갱신한다.



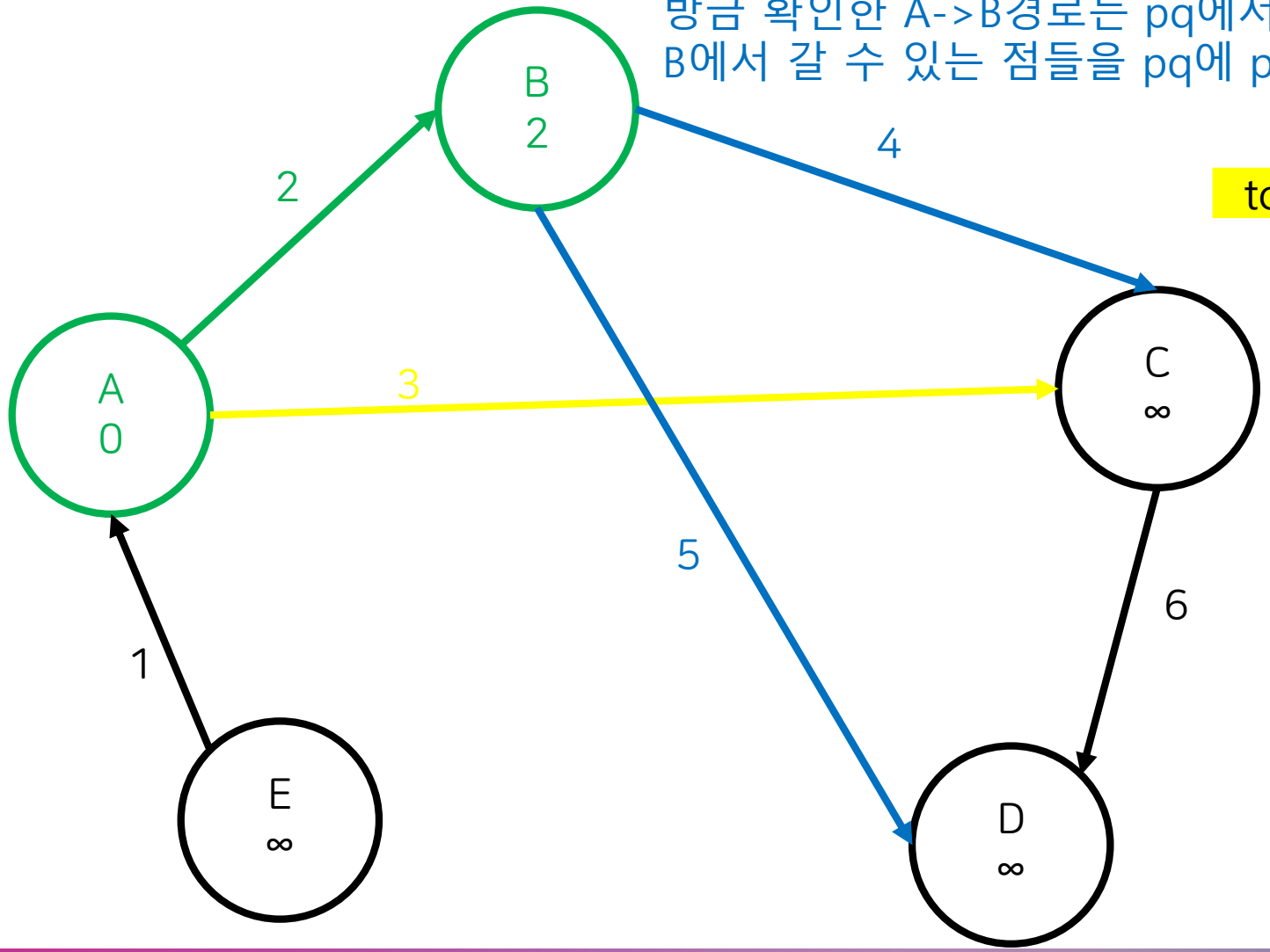
priority_queue

top

From	To	Weight
A	B	2
A	C	3

Dijkstra_예시

top에 있는 친구들을 하나씩 빼면서 최단경로를 확정한다.

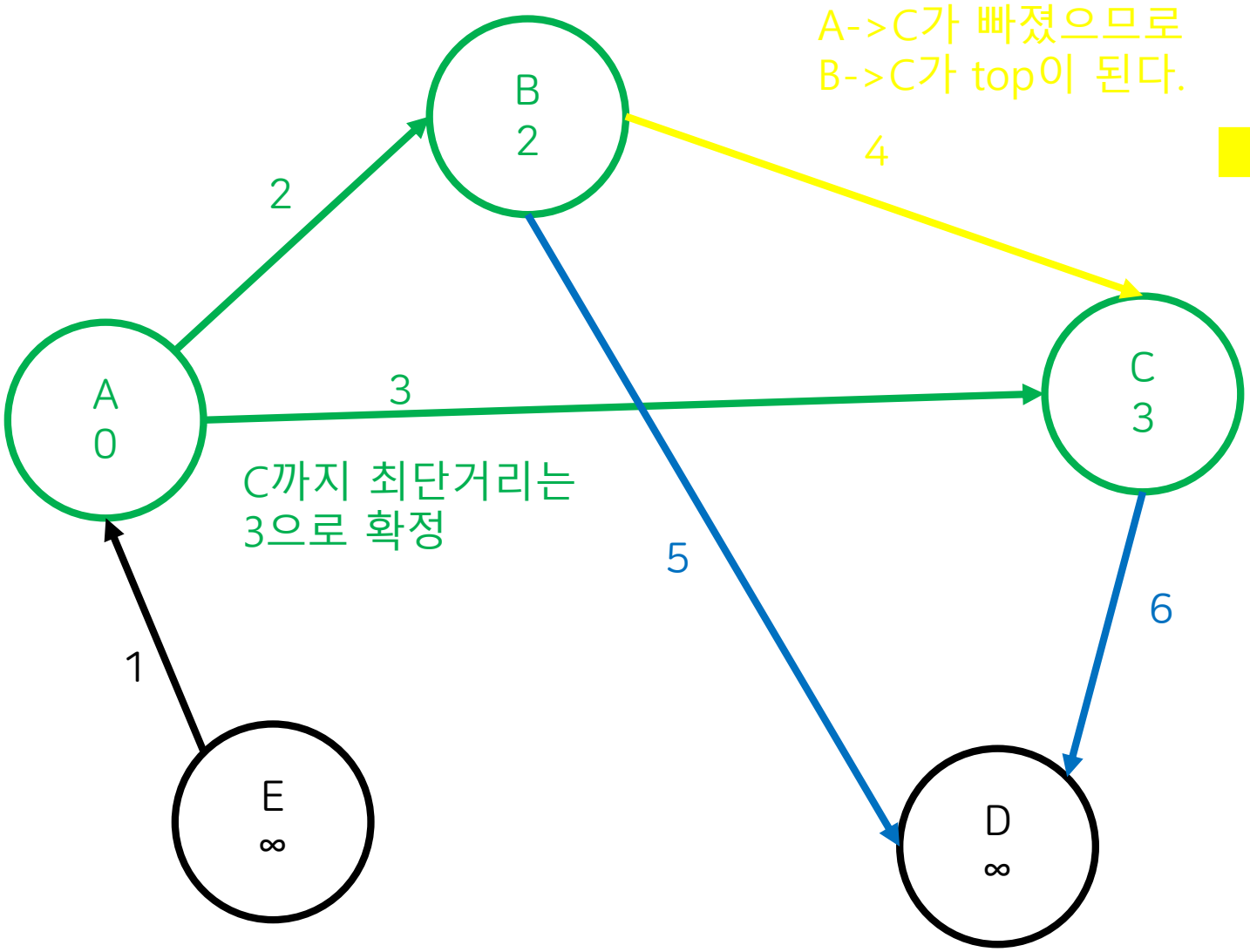


top

priority_queue

From	To	Weight
A	C	3
B	C	4
B	D	5

Dijkstra_예시

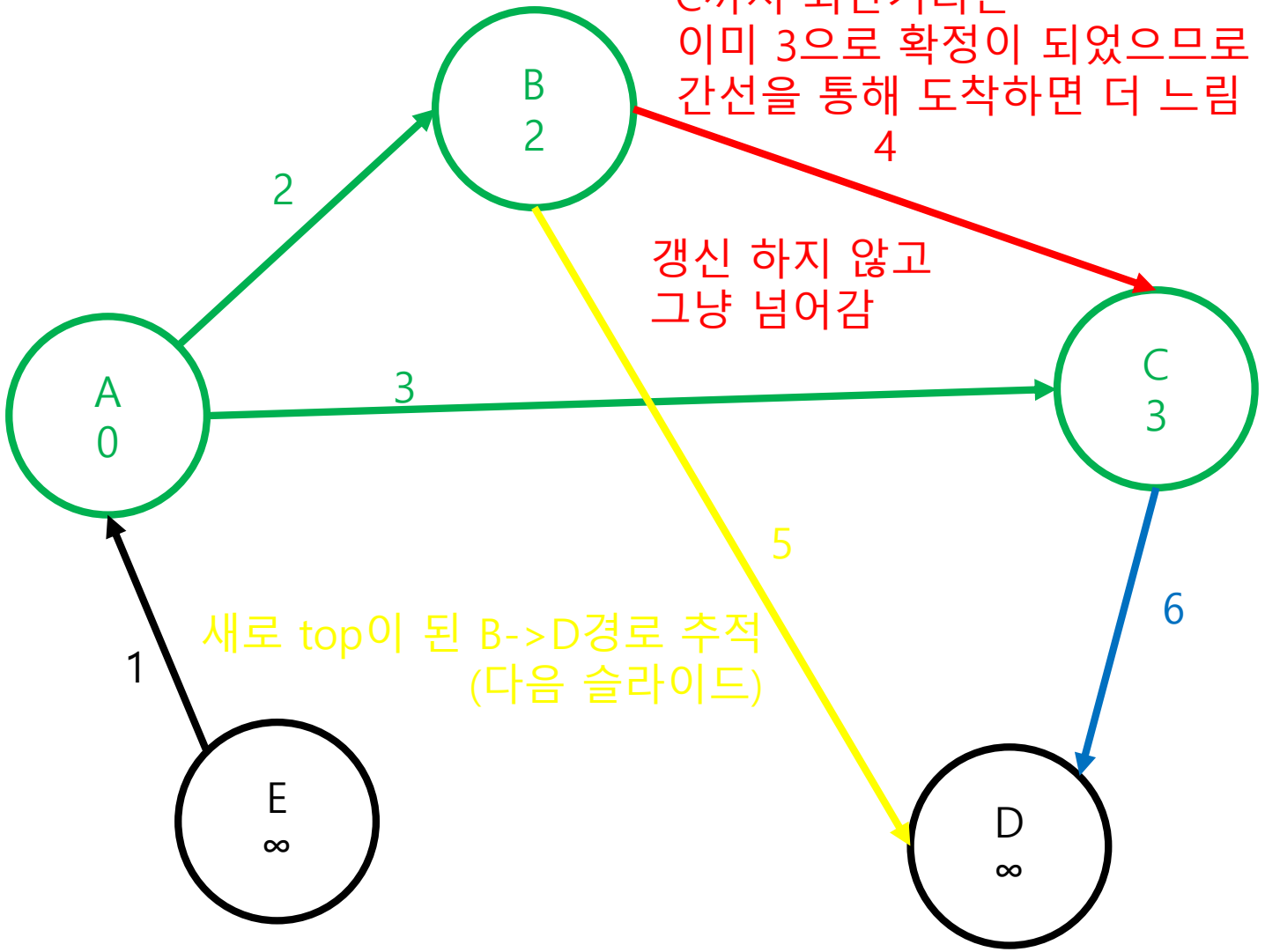


priority_queue

top

From	To	Weight
B	C	4
B	D	5
C	D	6

Dijkstra_예시

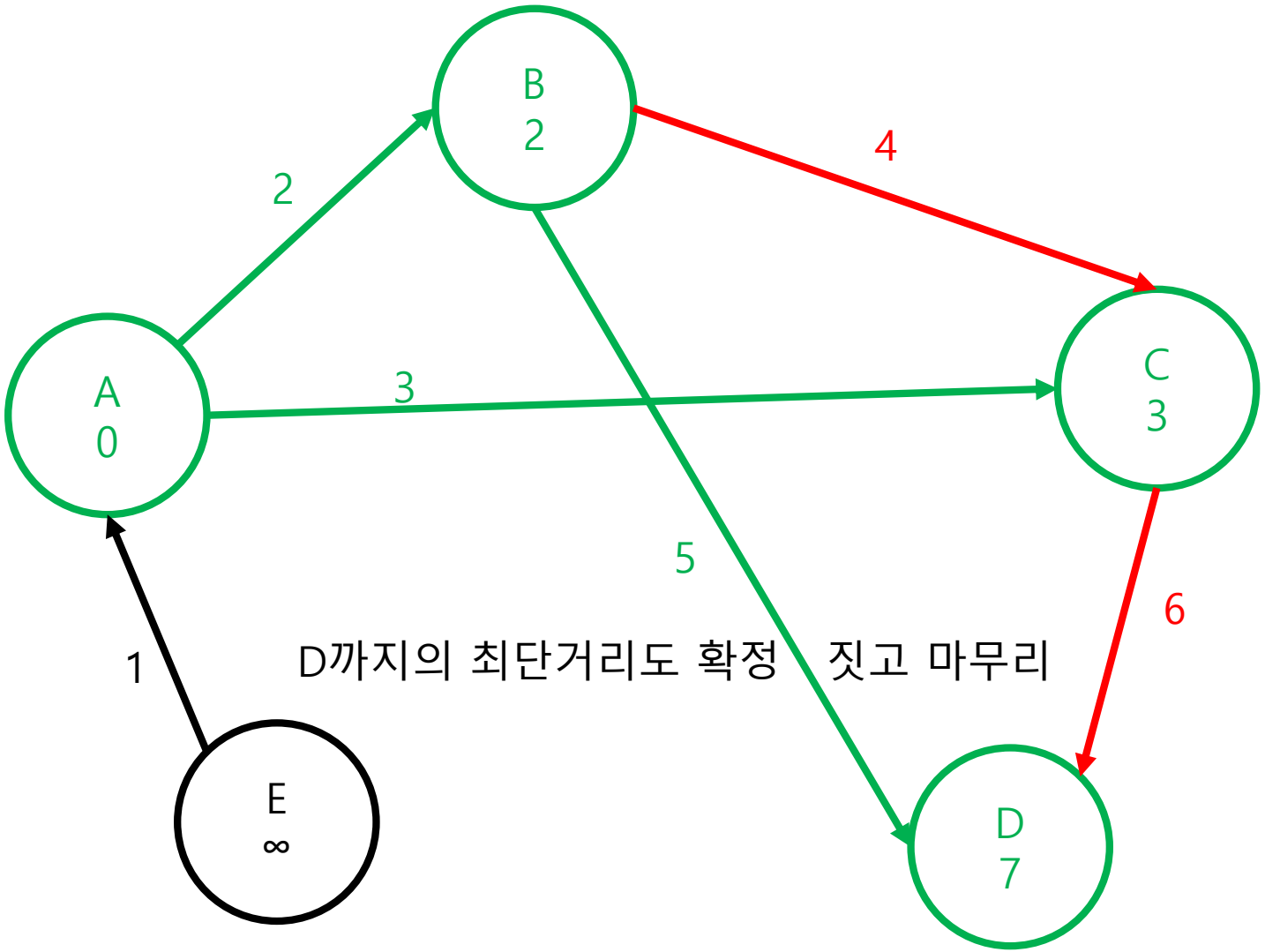


priority_queue

top

From	To	Weight
B	D	5
C	D	6

Dijkstra_예시

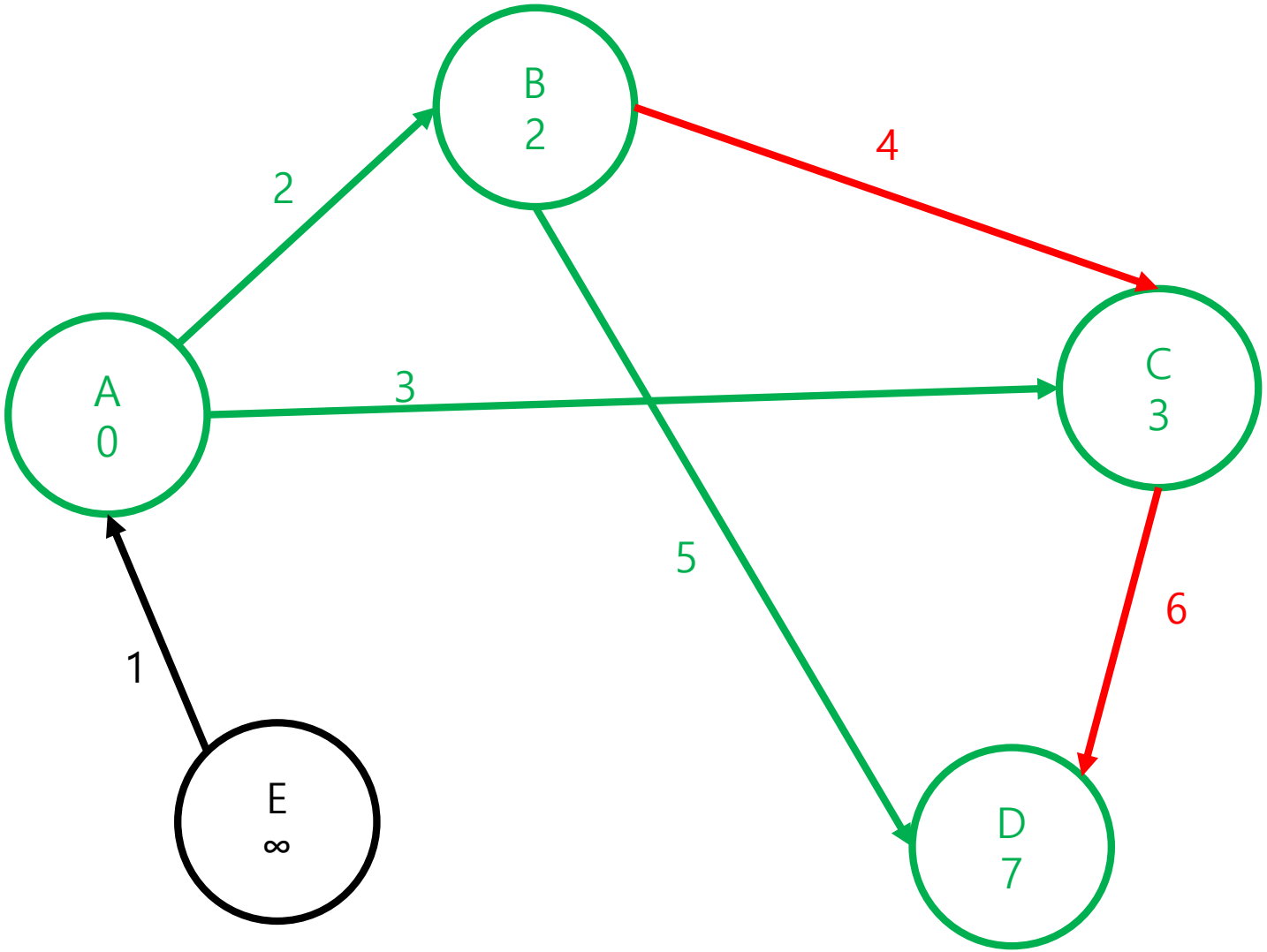


priority_queue

top

From	To	Weight

Dijkstra_예시



정점	최단거리
A	0
B	2
C	3
D	7
E	∞

E는 A에서 도달할 수 없다.
고로 최단거리는 ∞로 표현할 수 있다.

아직도... 모르겠나요...

Dijkstra_코드

```
void Dijkstra(int start)
{
    dist[start] = 0;
    for ( auto e : Edges[start] )
    {
        if ( dist[e.first] == INF )
        {
            pq.push({ -e.second, e.first });
        }
    }

    while ( !pq.empty() )
    {
        int now, nowdist;
        nowdist = -pq.top().first;
        now = pq.top().second;
        pq.pop();

        if ( dist[now] != INF )
            continue;

        dist[now] = nowdist;

        for ( auto e : Edges[now] )
        {
            if ( dist[e.first] == INF )
            {
                pq.push({ -(nowdist + e.second), e.first });
            }
        }
    }
}
```

- **dist: 최단거리를 저장하는 배열**
  → dist[1]: 시작지점부터 1번 정점까지의 최단거리
- **Edges: 인접리스트(3주차 참조, 똑같음)**
- **priorty_queue< pair<int, int> > pq;**
  priorty_queue<
  pair<시작지점부터 정점번호까지의 거리, 정점 번호> > pq;

  → pq에 pair가 들어갔을 때 기본 정렬이 어떻게 되냐면
  1. first값으로 정렬
  2. first가 같으면 second값으로 정렬

  (정렬기준은 내림차순)
- **INF: 무한대(엄청 큰 수)**
  → const int INF = 987654321;

Dijkstra_코드

```
void Dijkstra(int start)
{
    dist[start] = 0;
    for ( auto e : Edges[start] )
    {
        if ( dist[e.first] == INF )
        {
            pq.push({ -e.second, e.first });
        }
    }

    while ( !pq.empty() )
    {
        int now, nowdist;
        nowdist = -pq.top().first;
        now = pq.top().second;
        pq.pop();

        if ( dist[now] != INF )
            continue;

        dist[now] = nowdist;

        for ( auto e : Edges[now] )
        {
            if ( dist[e.first] == INF )
            {
                pq.push({ -(nowdist + e.second), e.first });
            }
        }
    }
}
```

- **pq에**
  <시작지점부터 정점번호까지의 거리, 정점 번호>
  이렇게 넣었을 경우
  ->top에 가장 큰 수가 온다.

- **우리가 원하는 정보는**
  가장 가까운 노드
  (시작지점부터 정점번호까지의 거리가 가장 작은 수)

- **pq에 넣을 때 거리에 -1을 곱해서 넣으면**
  -> 작은것부터 나오겠지??

~~이까 했지?~~

예제

1753번 - 최단경로

내가 진짜 1도 모르겠다. 그래도
저를 믿고 이 문제를 읽어주세요.
지금 읽어주세요.
제바ㅏㅏㅏㄹ~

1753_최단경로

- 문제 이해가 안되는 사람은 없겠죠?(제발)
- 그냥 문제 그대로 최단경로를 구해주면 됩니다.
- 다음 슬라이드에 해답이 있습니다.
- 혼자서 풀기 힘들거나 잘 안될 때는 강사의 코드를 베껴 보아요.

1753_최단경로

```cpp
#include <iostream>
#include <queue>

#include <climits>
using namespace std;

int V, E;
int K;
int u, v, w;

const int INF = INT_MAX;

const int MAX_V = 20'005;
vector<pair<int, int>> Edges[MAX_V];
vector<int> dist(MAX_V, INF);
priority_queue<pair<int, int>> pq;

void Dijkstra(int start);
```

```cpp
int main(void)
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    cin >> V >> E;
    cin >> K;
    for ( int e = 1; e <= E; e++ )
    {
        cin >> u >> v >> w;
        Edges[u].push_back({ v, w });
    }
    Dijkstra(K);

    for ( int v = 1; v <= V; v++ )
    {
        if ( dist[v] == INF )
        {
            cout << "INF\n";
        }
        else
        {
            cout << dist[v] << '\n';
        }
    }

    return 0;
}
```

Dijkstra함수만 작성하면 정답입니다.

~~(이 함수… 앞에서 본 것 같은데…)~~

예제

1753번 - 최단경로

이번엔 자신의 스타일로 코드를 바꿔서 다시 제출해보아요.

연습문제

1238 - 파티

1238_파티

- Dijkstra's Algorithm을 N번 돌려볼까?

연습문제

17396 - 백도어

17396_백도어

- 0번부터 시작하네요. 유의해주세요.

연습문제

5719 – 거의 최단 경로

5719_거의 최단 경로

- 최단경로를 구한다.
- 최단경로에 포함되는 모든 간선들을 지운다.
- 다시 최단 경로를 구한다.
- 최단경로가 1개가 아닐수도

연습문제

1854 – K번째 최단경로 찾기

1854_K번째 최단경로 찾기

- 알고리즘 자체를 변형해야 하는 문제
- pq에 더 많은 정보를 저장해야 할 것 같은데?

Floyd-Warshall Algorithm

이게 어쩌면 제일 쉬울 수도

Floyd-Warshall_개념

- **모든** 정점에 대한 최단경로를 알 수 있다.

- **모든** 경우의 수를 체크

Floyd-Warshall_개념

임의의 점을 경유해서 가는 경로가 더 빠르다면 갱신

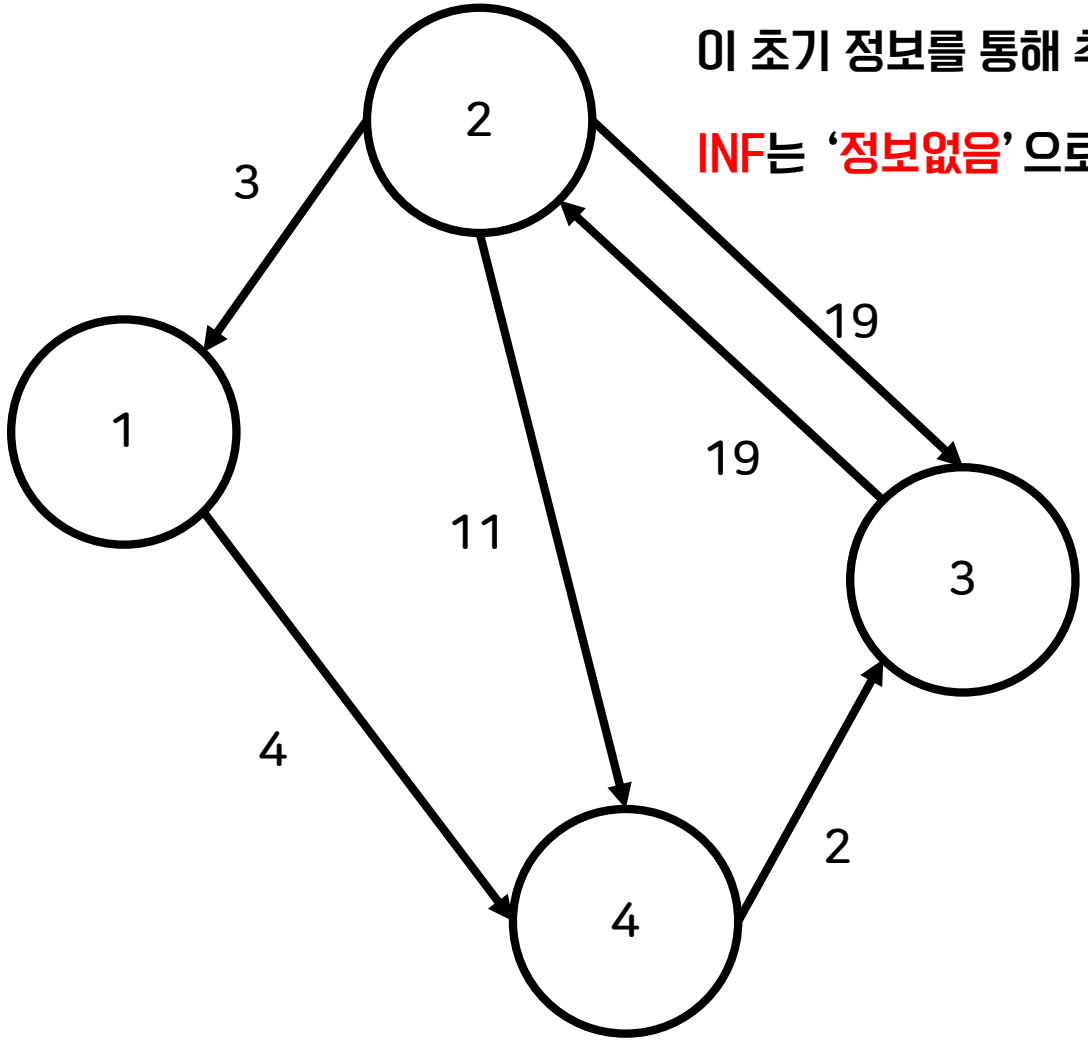
현재 알고 있는 i->j 의 거리보다 빠른 i->k->j가 존재한다면?

모든 i j k 에 대해 동작하면?

계산이 끝나면 **모든** 점에 대한 최단거리를 알 수 있다.

모르겠죠??

Floyd-Warshall_예시



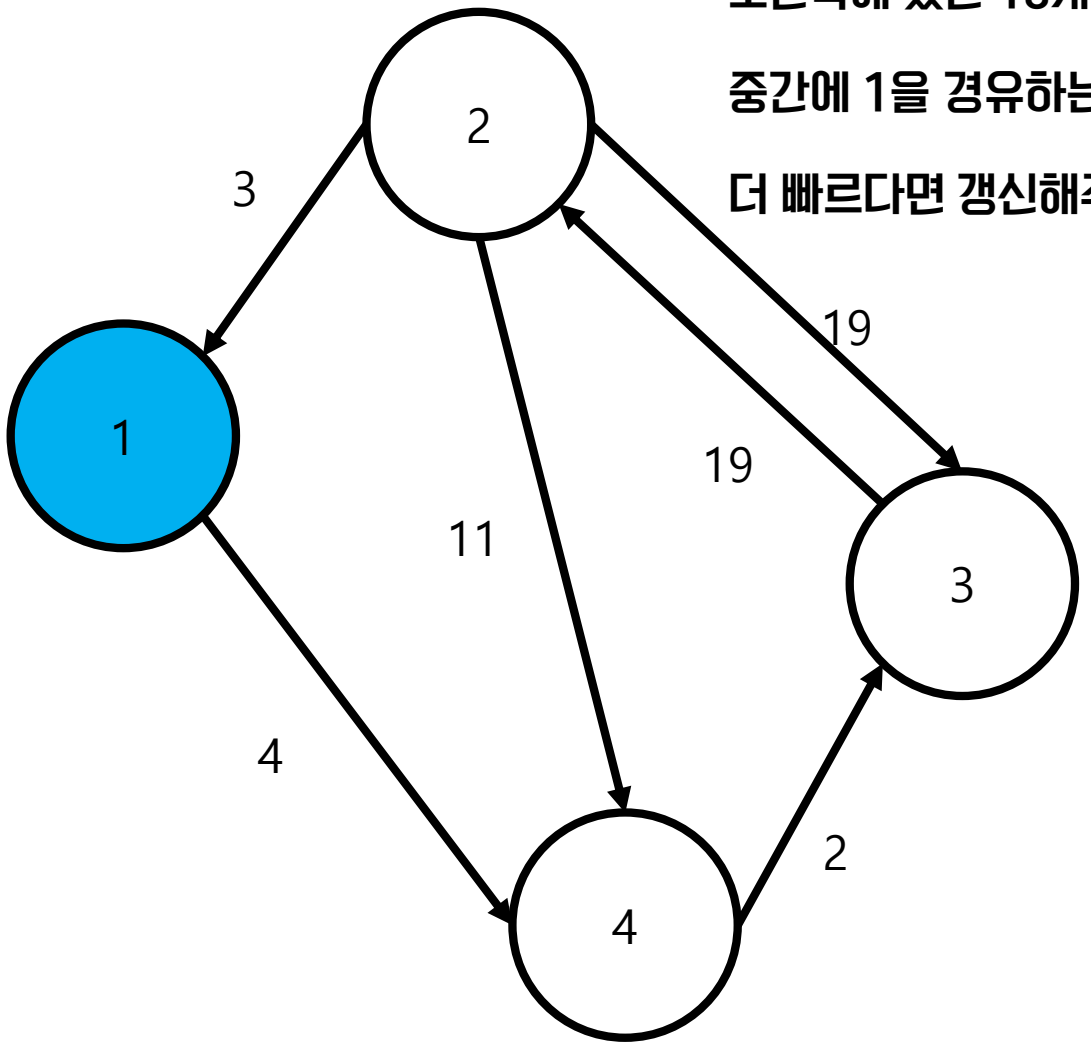
이 초기 정보를 통해 추론해 나갈거에요.

INF는 '정보없음' 으로 이해해도 됩니다.

출발부터 도착까지의
거리가 저장되어 있고요.

도착 / 출발	1	2	3	4
1	0	INF	INF	4
2	3	0	19	11
3	INF	19	0	INF
4	INF	INF	2	0

Floyd-Warshall_예시



오른쪽에 있는 16개의 모든 길을

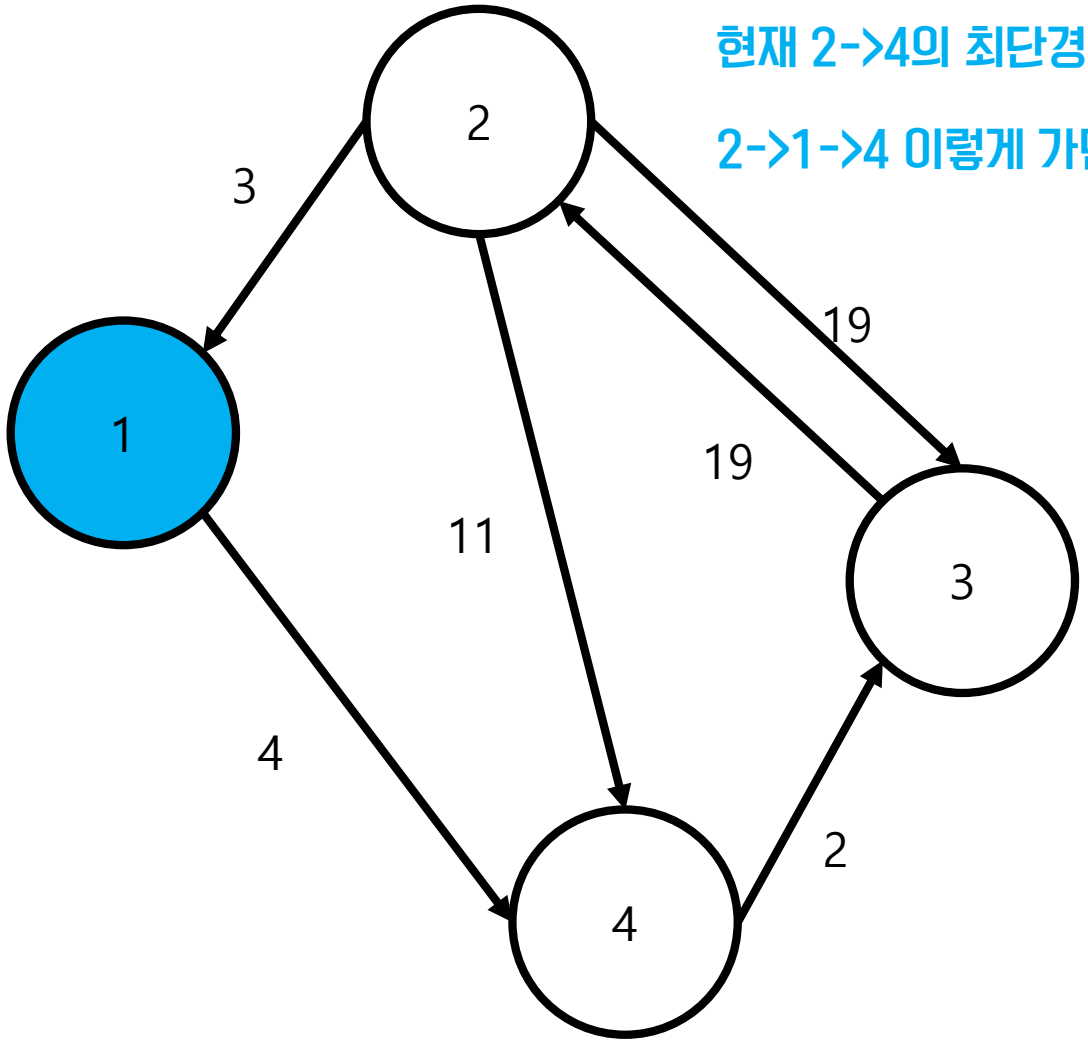
중간에 1을 경유하는 길로 돌아 갔을 때

더 빠르다면 갱신해주는 것입니다.

경유지 : 1

도착 / 출발	1	2	3	4
1	0	INF	INF	4
2	3	0	19	11
3	INF	19	0	INF
4	INF	INF	2	0

Floyd-Warshall_예시



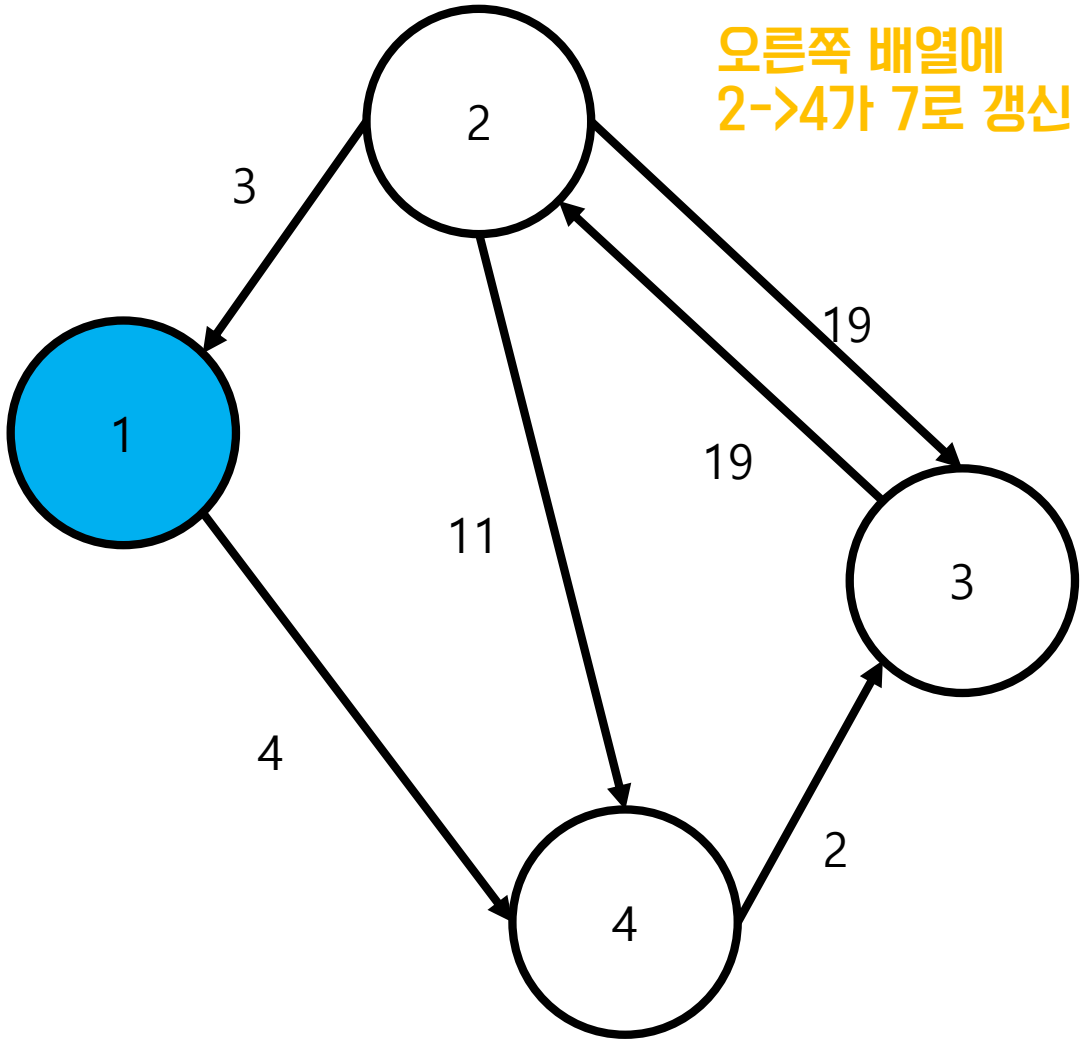
현재 2->4의 최단경로 11보다

2->1->4 이렇게 가면 더 빠르죠?

경유지 : 1

도착 / 출발	1	2	3	4
1	0	INF	INF	4
2	3	0	19	11
3	INF	19	0	INF
4	INF	INF	2	0

Floyd-Warshall_예시

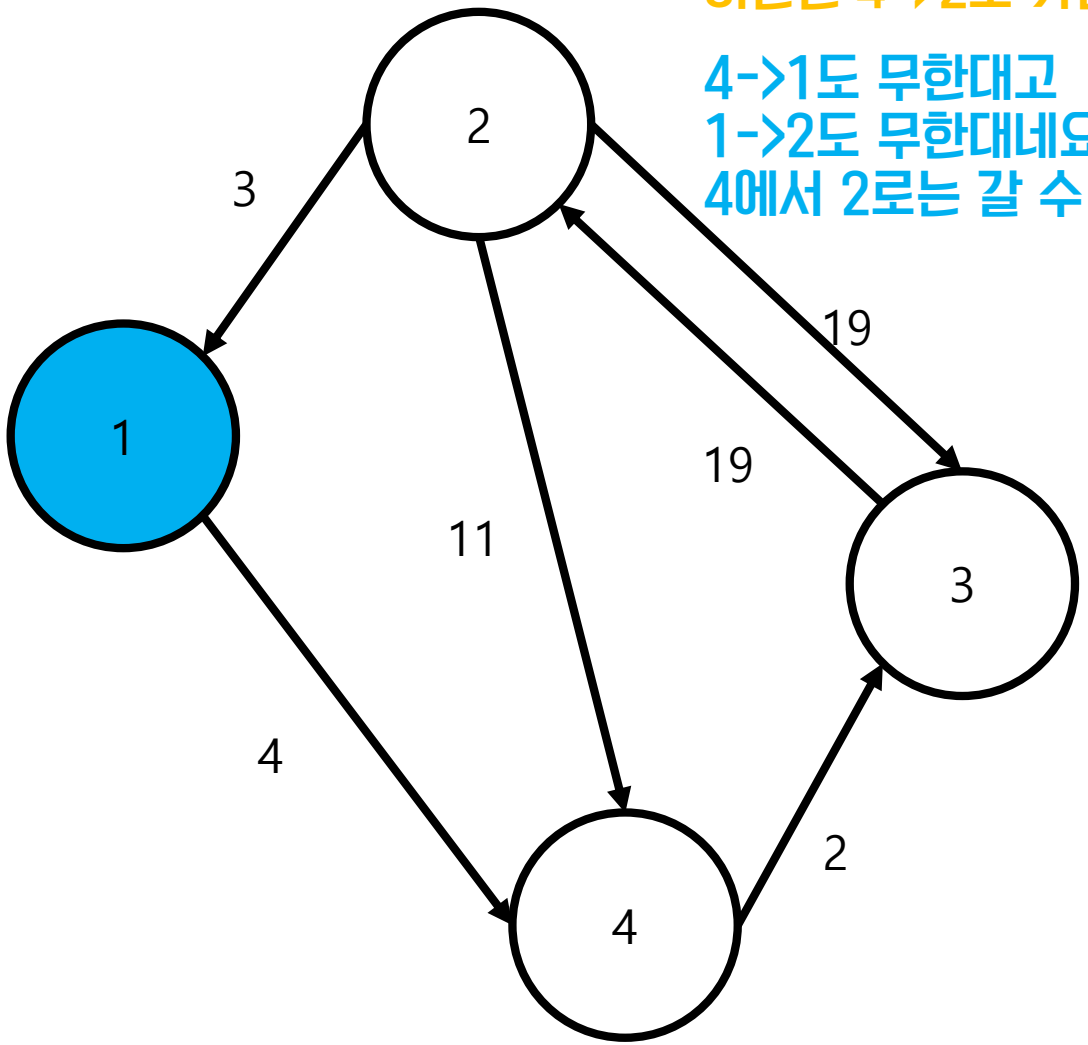


오른쪽 배열에
2->4가 7로 갱신 된것 보이시나요?

경유지 : 1

도착 / 출발	1	2	3	4
1	0	INF	INF	4
2	3	0	19	7
3	INF	19	0	INF
4	INF	INF	2	0

Floyd-Warshall_예시



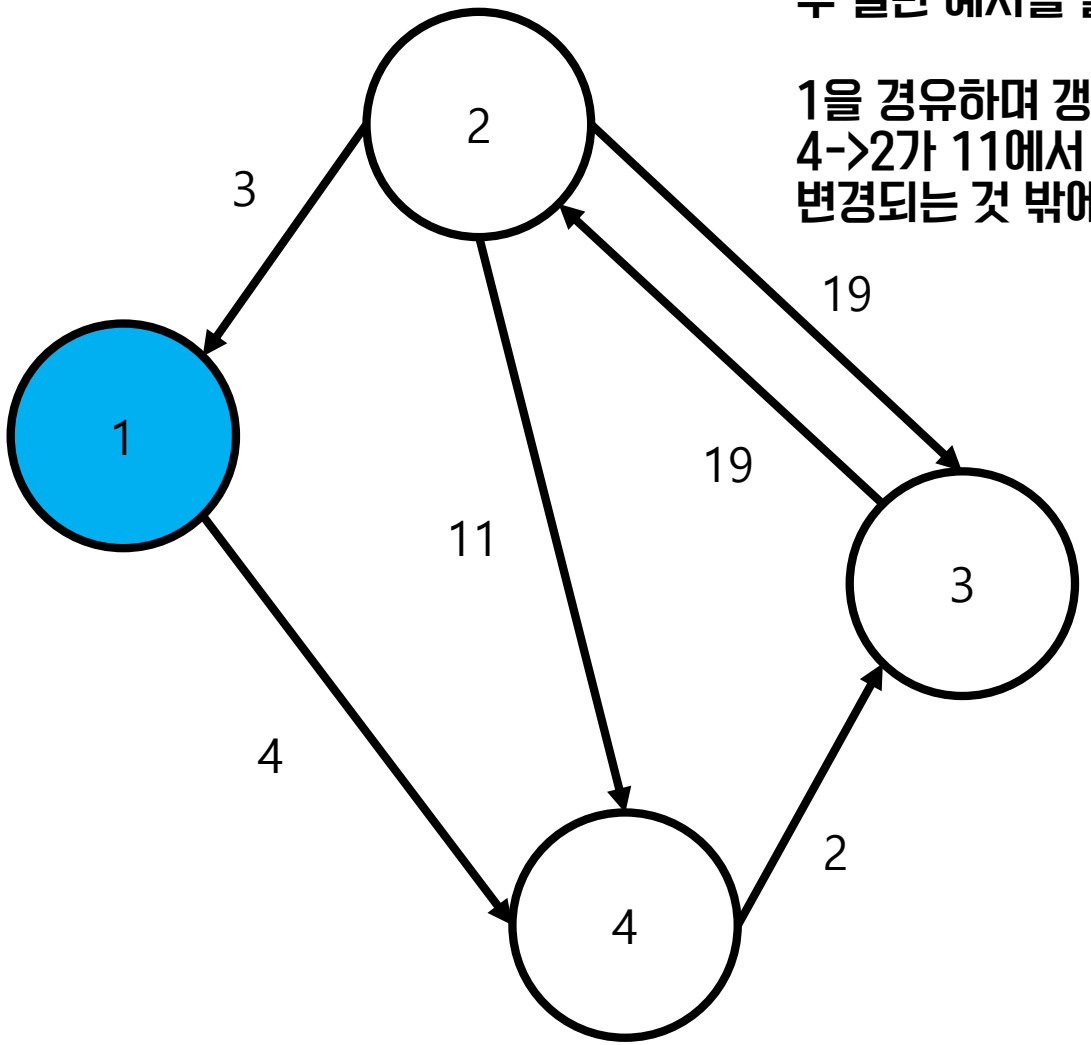
이번엔 4->2로 가는 경로를 볼까요?

4->1도 무한대고
1->2도 무한대네요. 1을 거쳐가도
4에서 2로는 갈 수 없네요.

경유지 : 1

도착 / 출발	1	2	3	4
1	0	INF	INF	4
2	3	0	19	7
3	INF	19	0	INF
4	INF	INF	2	0

Floyd-Warshall_예시



두 길만 예시를 들었는데요.

1을 경유하며 갱신되는 경로는
4->2가 11에서 7로
변경되는 것 밖에 없었네요.

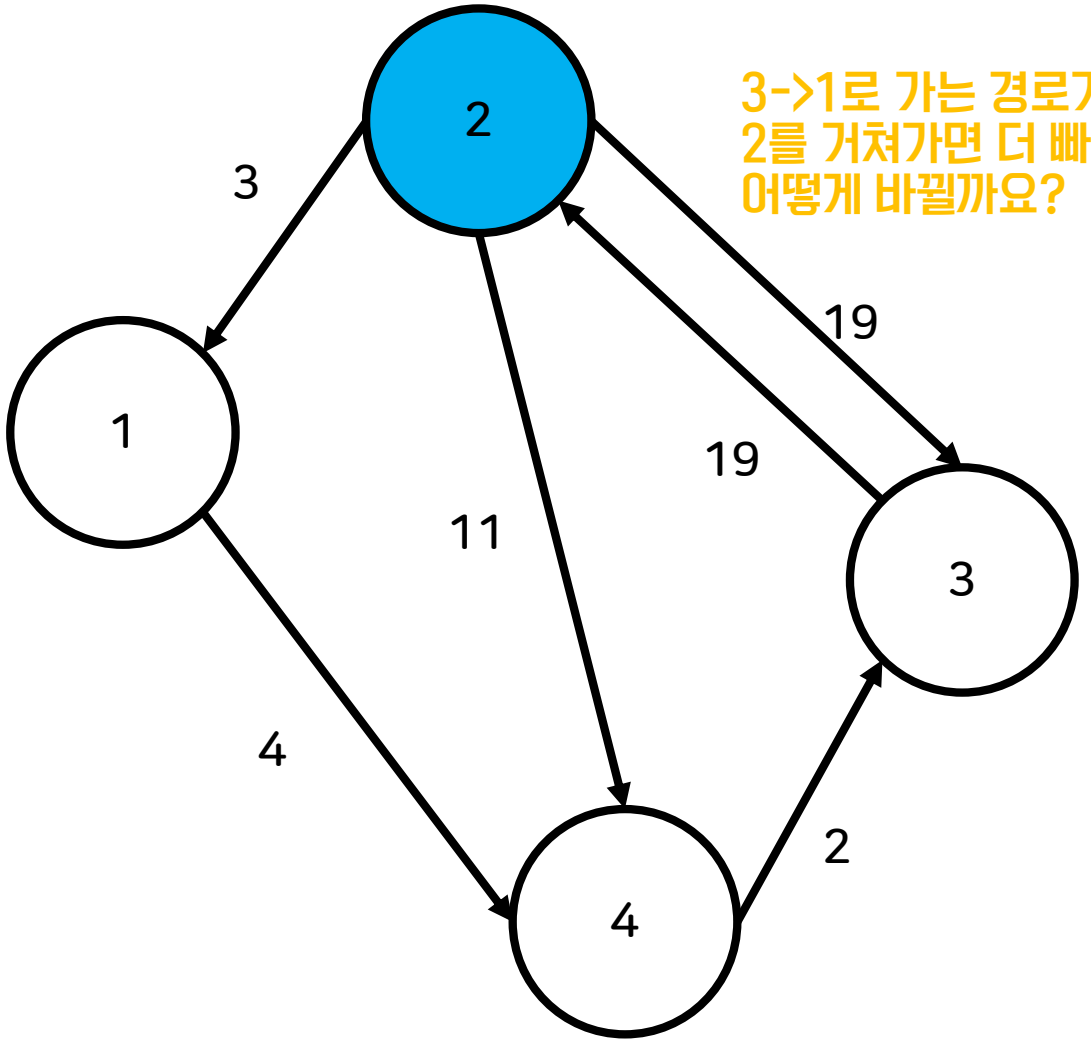
경유지 : 1

도착 / 출발	1	2	3	4
1	0	INF	INF	4
2	3	0	19	7
3	INF	19	0	INF
4	INF	INF	2	0

Floyd-Warshall_예시

경유지 : 2

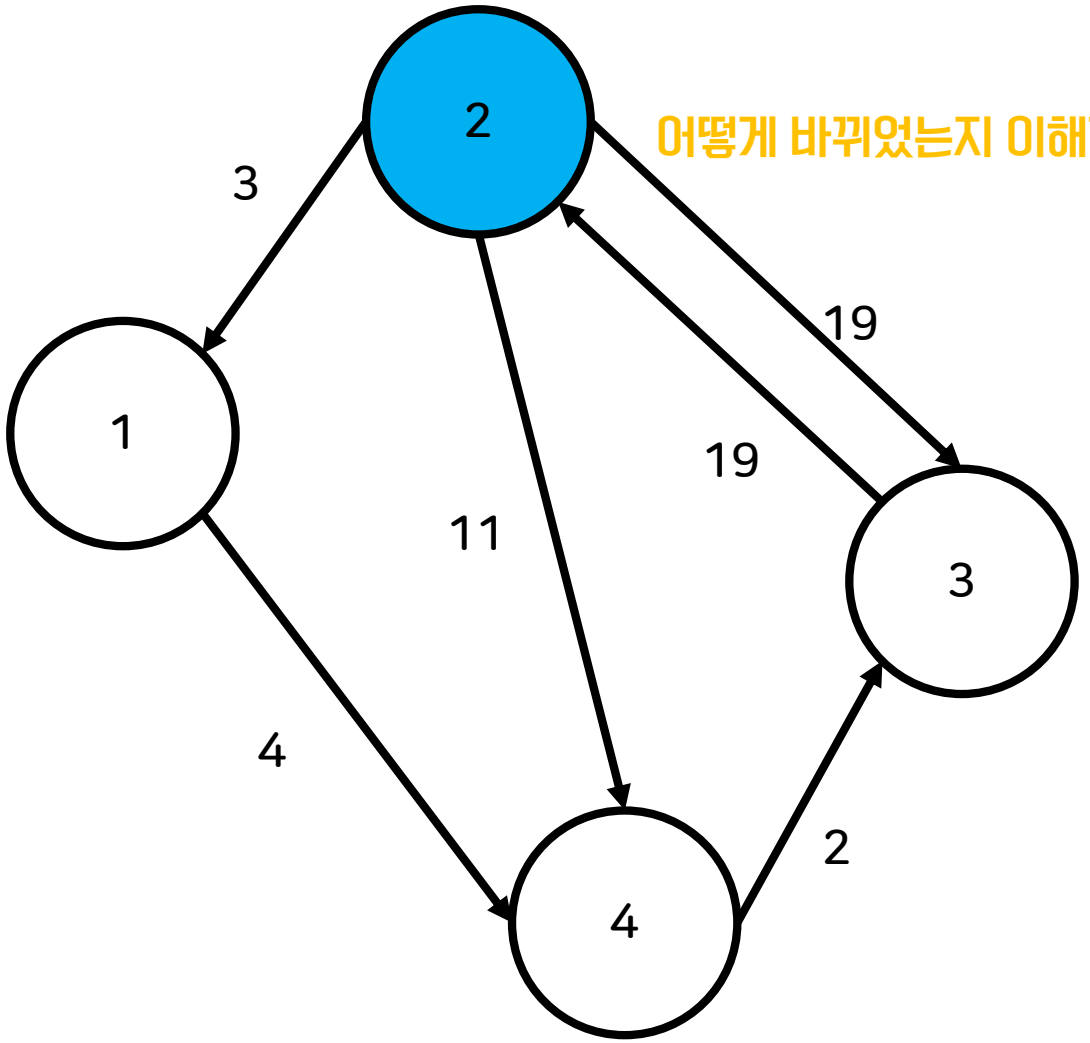
이제 2를 경유해서 갱신 되는 길을 찾아볼까요?



3->1로 가는 경로가
2를 거쳐가면 더 빠를 것 같은데요.
어떻게 바뀔까요?

도착 출발	1	2	3	4
1	0	INF	INF	4
2	3	0	19	7
3	INF	19	0	INF
4	INF	INF	2	0

Floyd-Warshall_예시



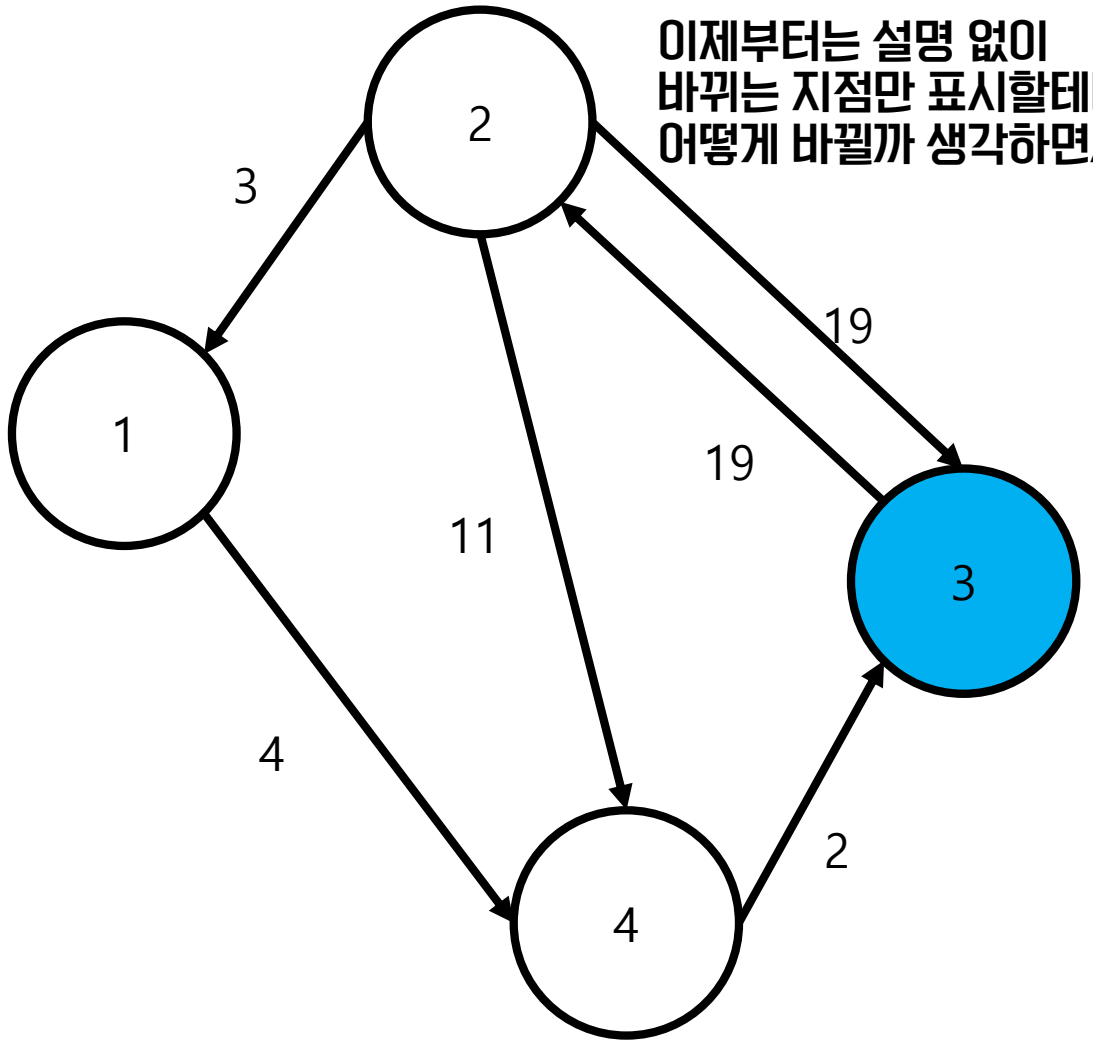
어떻게 바뀌었는지 이해가 되시나요?

경유지 : 2

도착 / 출발	1	2	3	4
1	0	INF	INF	4
2	3	0	19	7
3	22	19	0	INF
4	INF	INF	2	0

Floyd-Warshall_예시

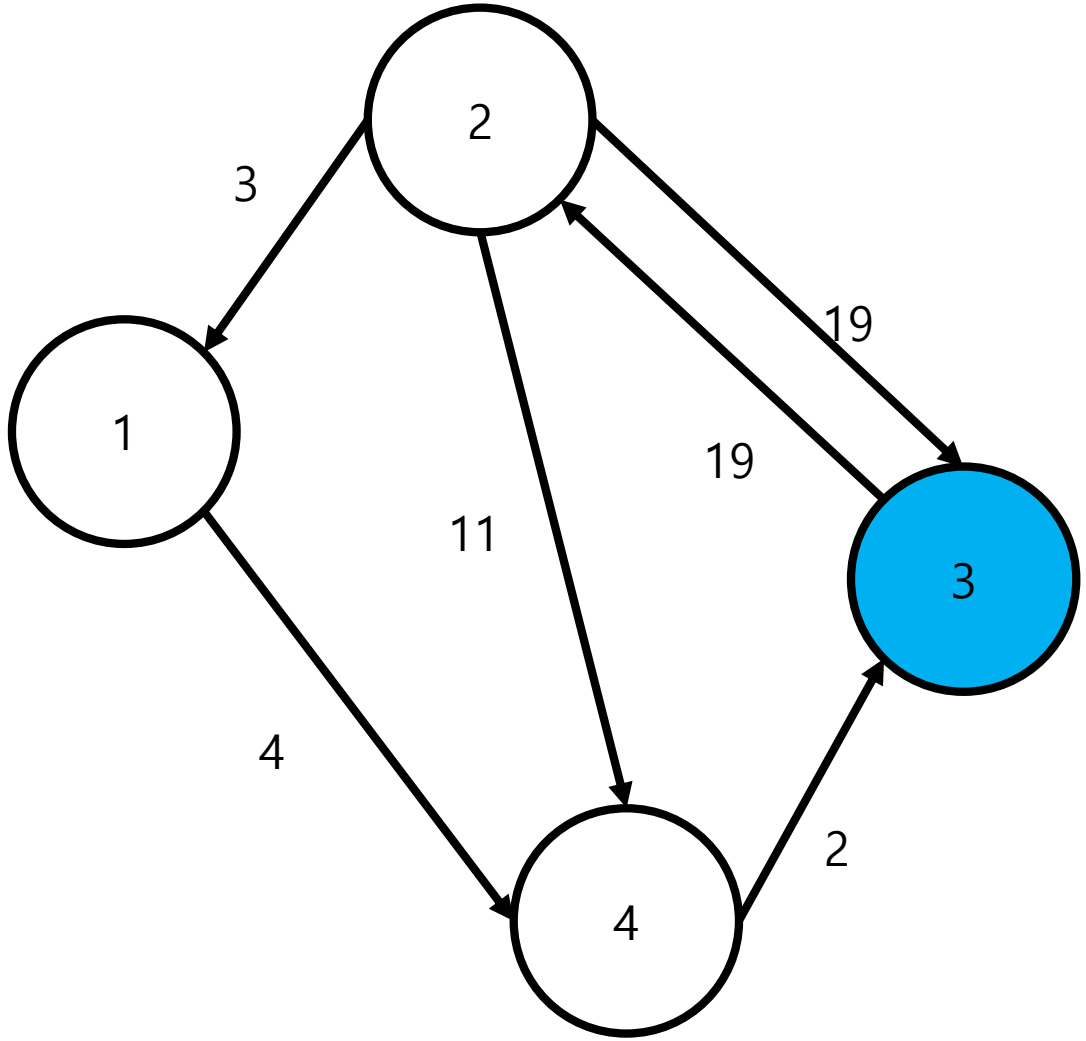
경유지 : 3



이제부터는 설명 없이
바뀌는 지점만 표시할테니
어떻게 바뀔까 생각하면서 넘겨주세요!

도착 / 출발	1	2	3	4
1	0	INF	INF	4
2	3	0	19	7
3	22	19	0	INF
4	INF	INF	2	0

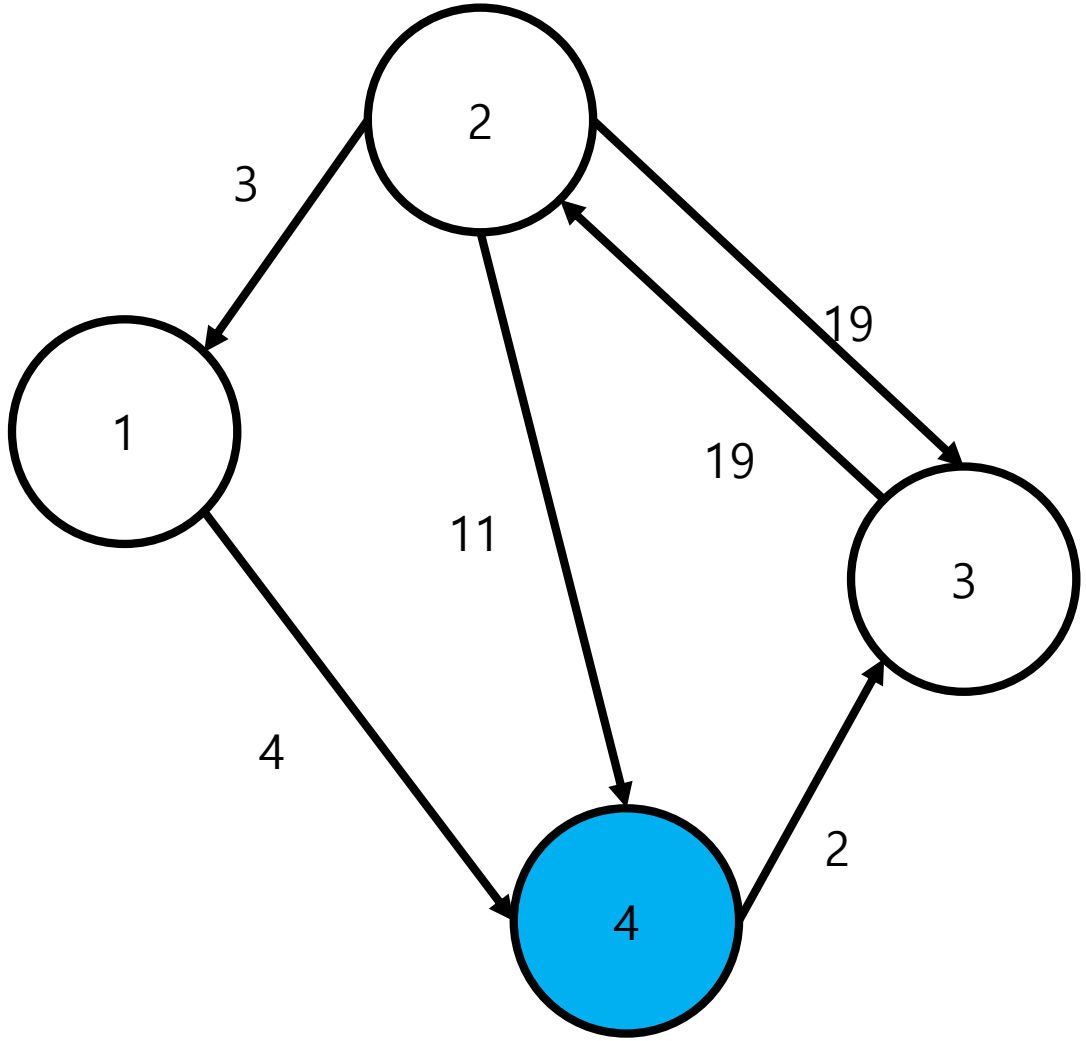
Floyd-Warshall_예시



경유지 : 3

도착 / 출발	1	2	3	4
1	0	INF	INF	4
2	3	0	19	7
3	22	19	0	INF
4	INF	21	2	0

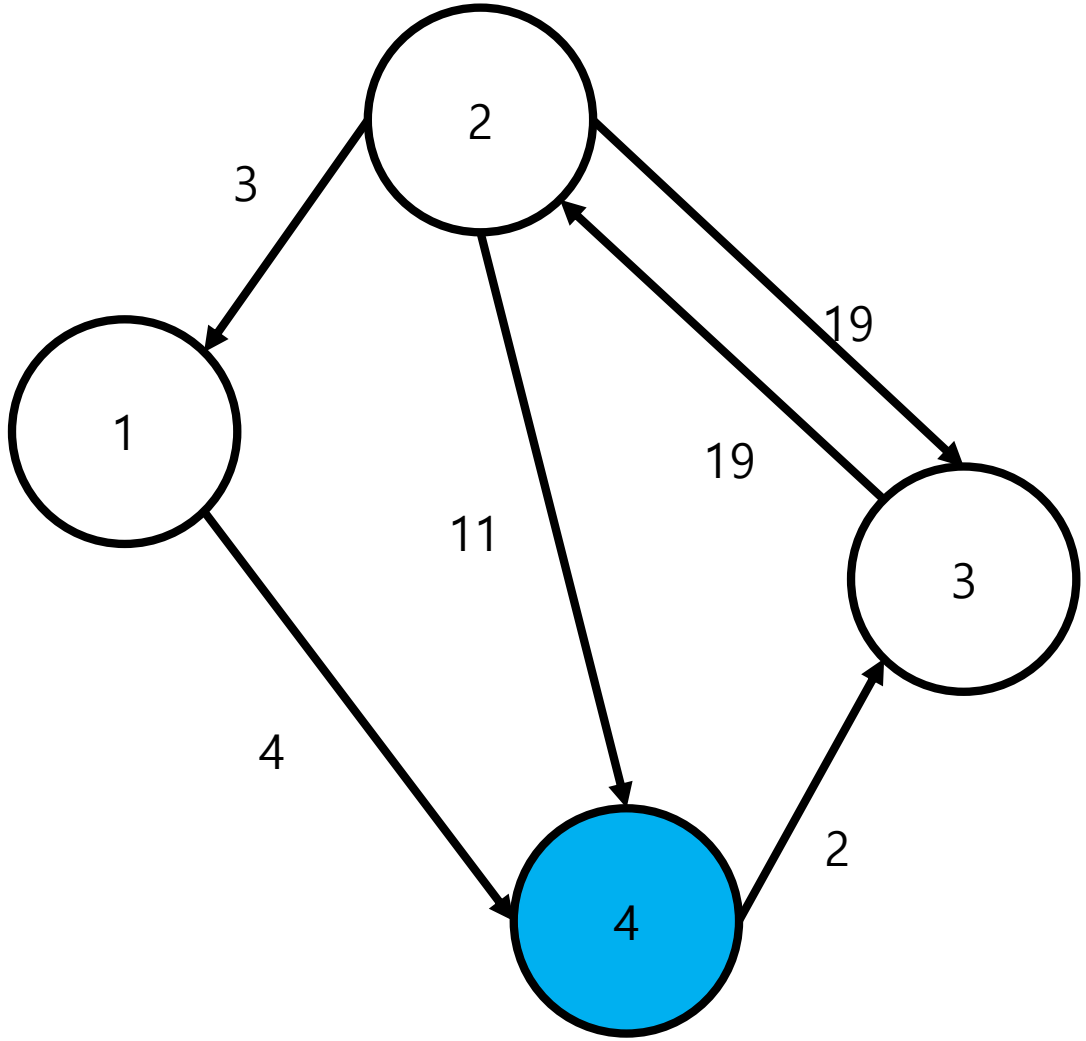
Floyd-Warshall_예시



경유지 : 4

도착 / 출발	1	2	3	4
1	0	INF	INF	4
2	3	0	19	7
3	22	19	0	INF
4	INF	21	2	0

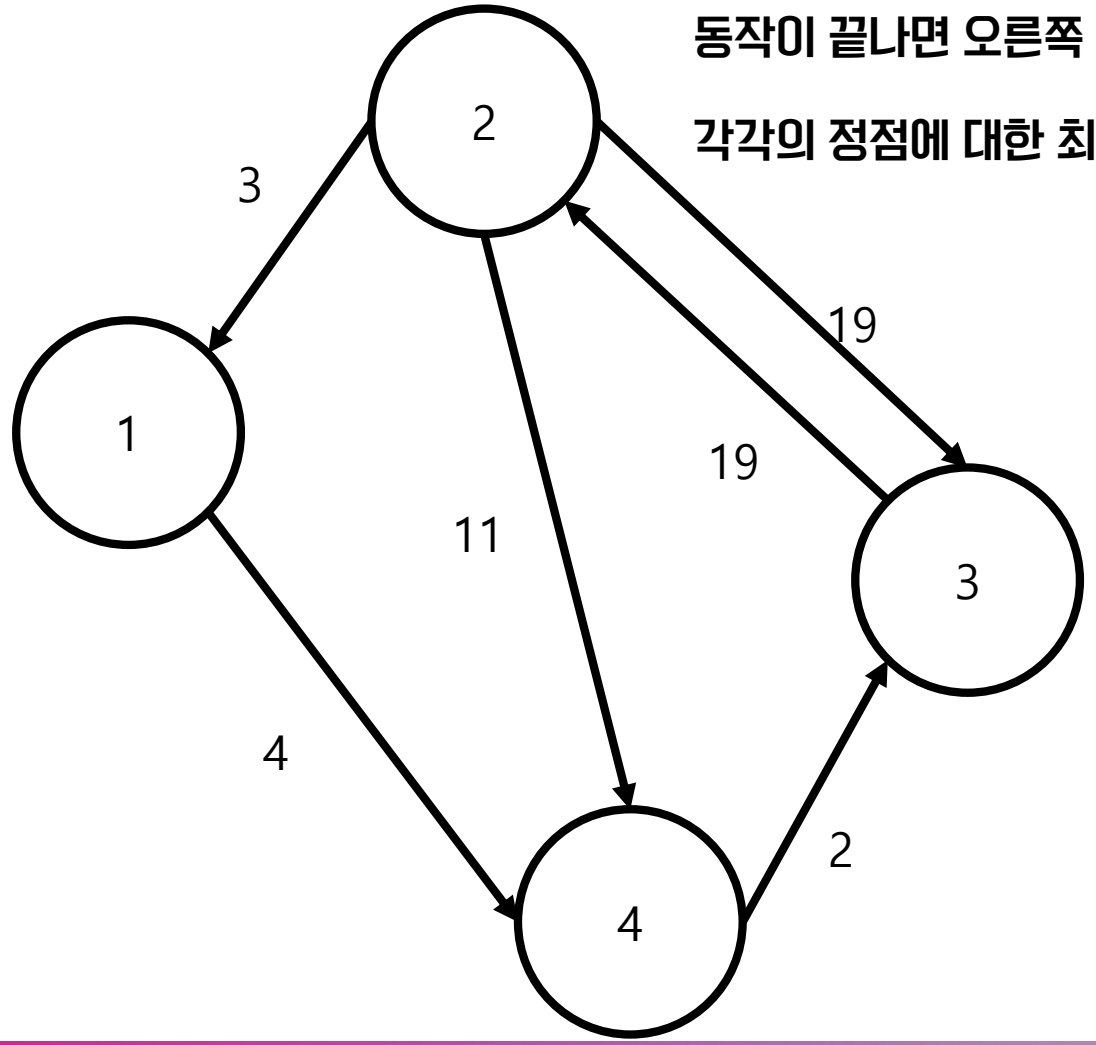
Floyd-Warshall_예시



경유지 : 4

도착 / 출발	1	2	3	4
1	0	INF	6	4
2	3	0	9	7
3	22	19	0	INF
4	INF	21	2	0

Floyd-Warshall_예시



동작이 끝나면 오른쪽 2차원 배열이

각각의 정점에 대한 최단경로가 됩니다.

도착 출발	1	2	3	4
1	0	INF	6	4
2	3	0	9	7
3	22	19	0	INF
4	INF	21	2	0

아직도... 모르겠나요...

Floyd-Warshall_코드

- DP[i][j] : i부터 j까지의 최단거리

진짜 이게 끝이에요.

```
for ( int k = 1; k <= N; k++ )
{
    for ( int i = 1; i <= N; i++ )
    {
        for ( int j = 1; j <= N; j++ )
        {
            min(DP[i][j], DP[i][k] + DP[k][j]);
        }
    }
}
```

서순주의!!!

예제

11404 - 플로이드

문제를 풀어보면 쉽습니다.
진짜

11404_플로이드

- 문제 이해가 안되는 사람은 없겠죠?(제발)
- 최단 '거리' 알고리즘이라고 거리만 생각하면 안되요!!
  거리, 시간, 돈 어떠한 개념이 와도 응용할 수 있어야 합니다.
- 다음 슬라이드에 해답이 있습니다.
- 혼자서 풀기 힘들거나 잘 안될 때는 강사의 코드를 베껴 보아요.

11404_플로이드

```cpp
#include <iostream>
#include <algorithm>

#include <climits>
using namespace std;

int N, M;
int a, b, c;
const int MAX_N = 102;
int DP[MAX_N][MAX_N];

const int INF = MAX_N * 100'000;

void Floyd_Warshall(void);
```

```cpp
int main(void)
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    for ( int i = 0; i < MAX_N; i++ )
    {
        for ( int j = 0; j < MAX_N; j++ )
        {
            if ( i == j )
                DP[i][j] = 0;
            else
                DP[i][j] = INF;
        }
    }

    cin >> N >> M;
    for ( int m = 1; m <= M; m++ )
    {
        cin >> a >> b >> c;
        DP[a][b] = min(DP[a][b], c);
    }

    Floyd_Warshall();

    for ( int i = 1; i <= N; i++ )
    {
        for ( int j = 1; j <= N; j++ )
        {
            if ( DP[i][j] == INF )
            {
                DP[i][j] = 0;
            }
            cout << DP[i][j] << ' ';
        }
        cout << '\n';
    }

    return 0;
}
```

Floyd_Warshall함수만 작성하면 정답입니다.

~~(이 함수 어디서 봤는데…)~~

예제

11404 - 플로이드

이번엔 자신의 스타일로 코드를 바꿔서 다시 제출해보아요.

연습문제

2610 - 회의준비

2610_회의준비

- 일단 Floyd-Warshall을 한번 돌리고 생각해보자.
- 의사전달시간을 구해보자(1~N)

연습문제

11780 – 플로이드 2

11780_플로이드 2

- Hint: 플로이드 트래킹
- 구글에 쳐봐도 돼요. 물론 '플로이드 트래킹'을

Bellman-Ford Algorithm

모르면 어떻게 하라고?

따라한다. 외운다. 제출한다. 맞았습니다~ 그럼 이해된다!

그래도 모르면? 멘토를 부른다~

Bellman-Ford_개념

- 하나의 시작점에서 모든 정점의 최단경로를 구할 수 있다!
- 음수 가중치가 있어도 OK!
- 음수사이클 검출에 좋음.

Bellman-Ford_개념

임의의 간선을 경유해서 가는 경로가 더 빠르다면 갱신

V개의 정점이 있을 때, 한 정점에서 다른 정점으로 가는 최단거리는
최대 (V – 1)개의 간선을 지난다.

→ Ex) 1번 정점에서 5번 정점까지 정점을 최대한 거쳐가는 최단경로는
1->2->3->4->5 이렇게 4개의 간선을 거쳐가는 것이 최대이다.
1->2->5 는 최단경로가 될 수는 있지만
1->2->3->2->4->5 는 될 수가 없다.

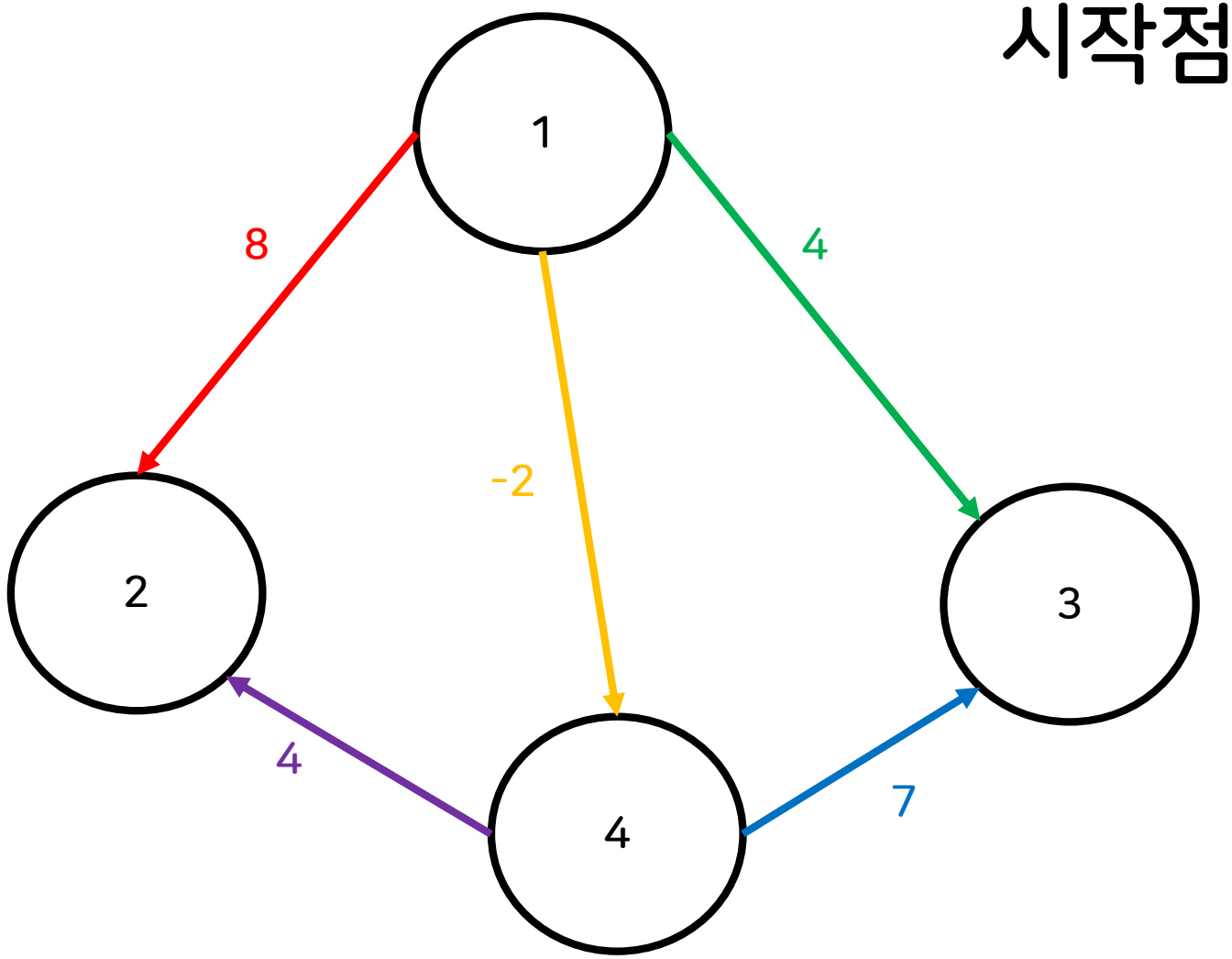
모든 간선에 대해 V-1번 갱신하면?

방문되지 않은 정점(값이 무한대)에서 출발하는 edge는 고려하지 않음.

이번에도
예시를 통해 알아보아요.

Bellman-Ford_예시

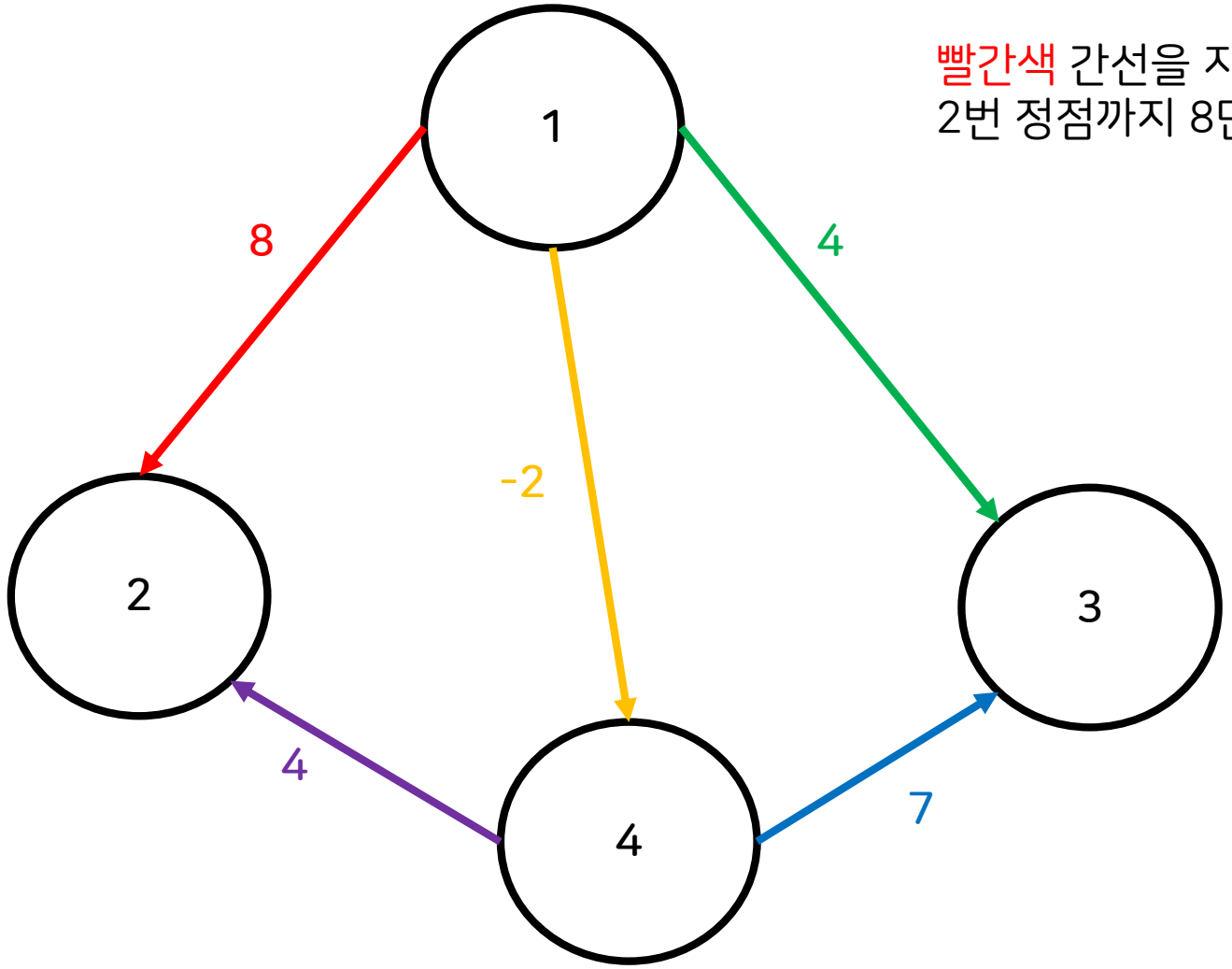
간선 탐색순서: 무지개



시작점: 1

정점번호	최단거리
1	0
2	INF
3	INF
4	INF

Bellman-Ford_예시

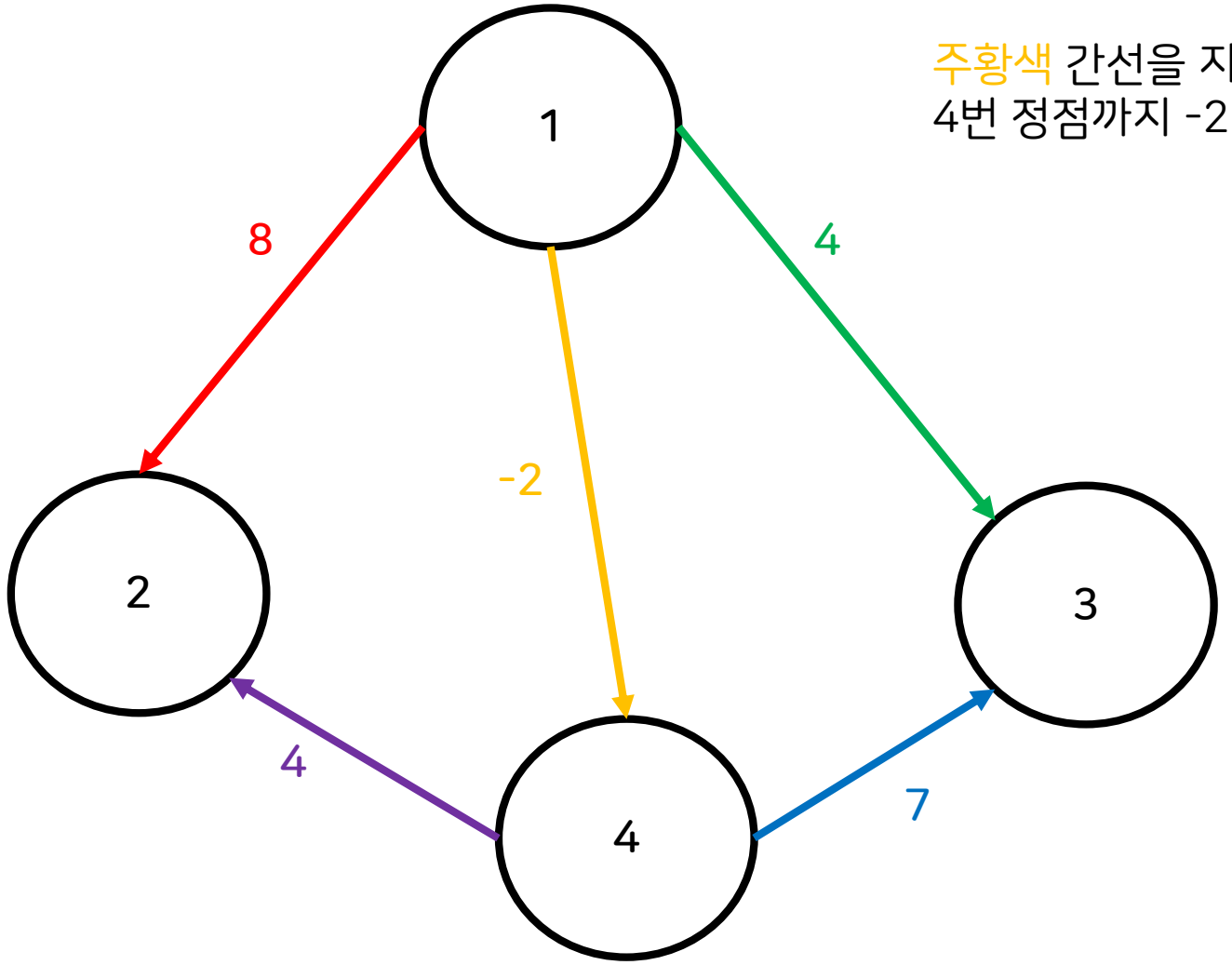


빨간색 간선을 지나면
2번 정점까지 8만큼 걸린다

1번째 방문

정점번호	최단거리
1	0
2	8
3	INF
4	INF

Bellman-Ford_예시

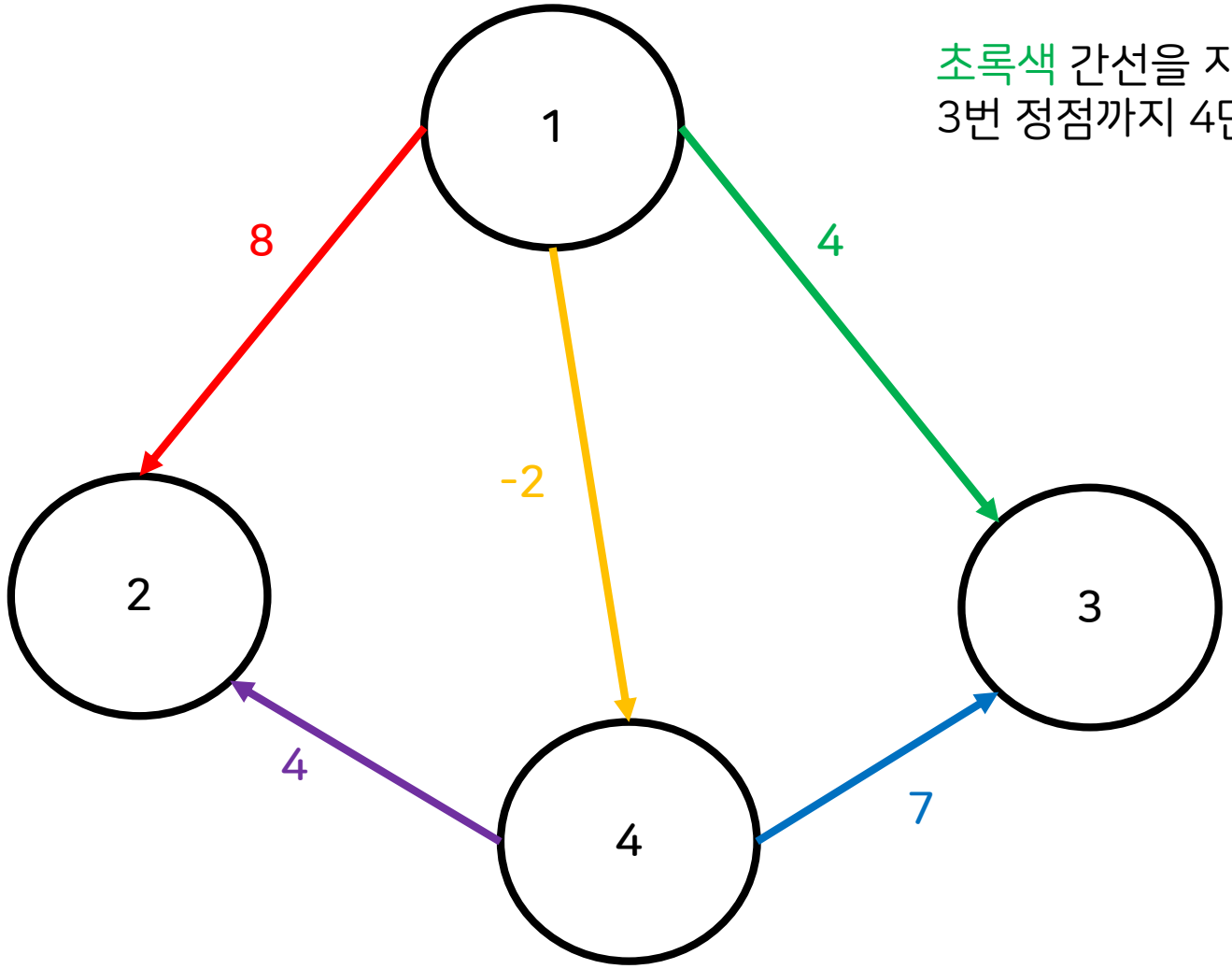


주황색 간선을 지나면
4번 정점까지 -2만큼 걸린다

1번째 방문

정점번호	최단거리
1	0
2	8
3	INF
4	-2

Bellman-Ford_예시

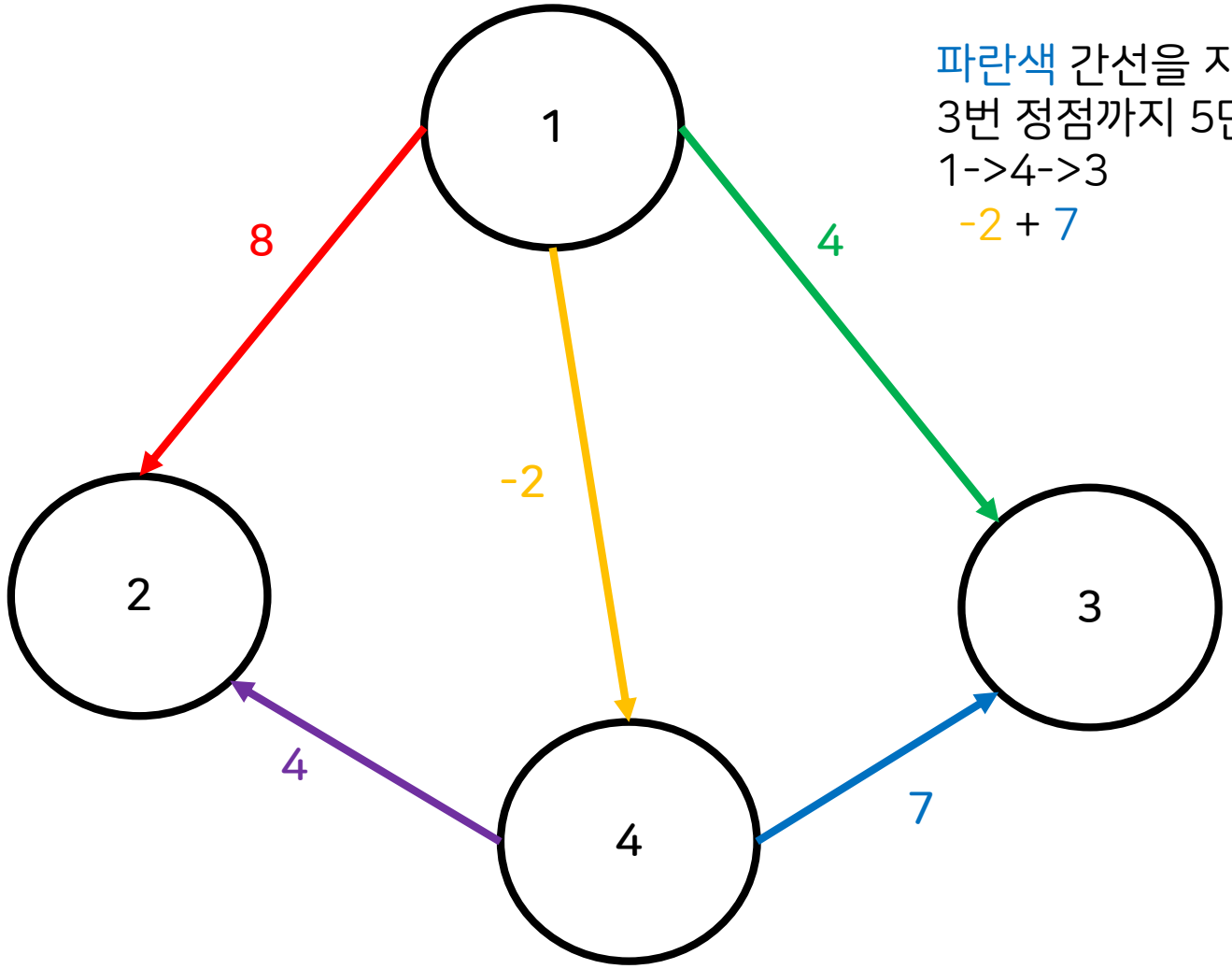


초록색 간선을 지나면
3번 정점까지 4만큼 걸린다

1번째 방문

정점번호	최단거리
1	0
2	8
3	4
4	-2

Bellman-Ford_예시

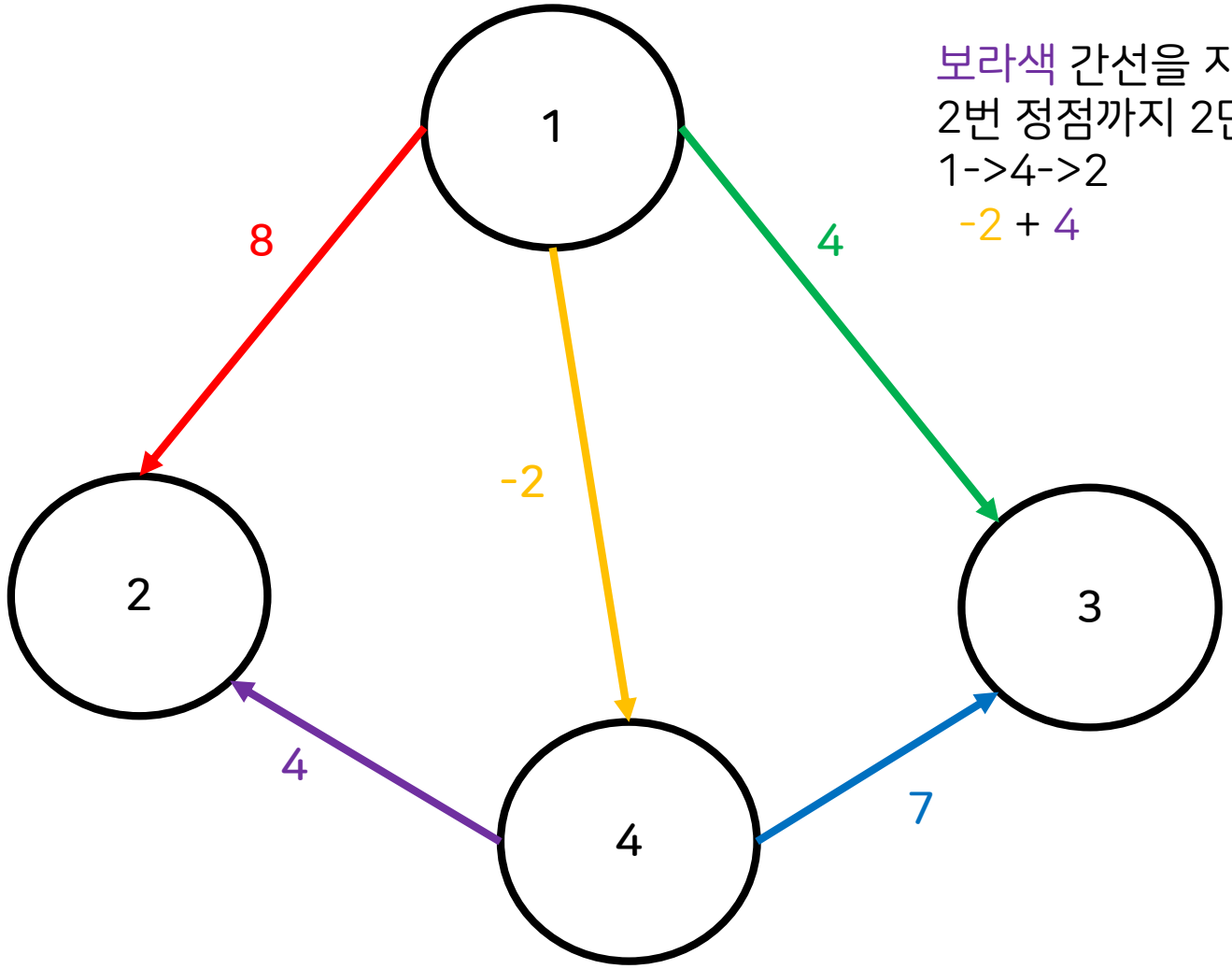


파란색 간선을 지나면
3번 정점까지 5만큼 걸린다.
1->4->3
-2 + 7

1번째 방문

정점번호	최단거리
1	0
2	8
3	5
4	-2

Bellman-Ford_예시

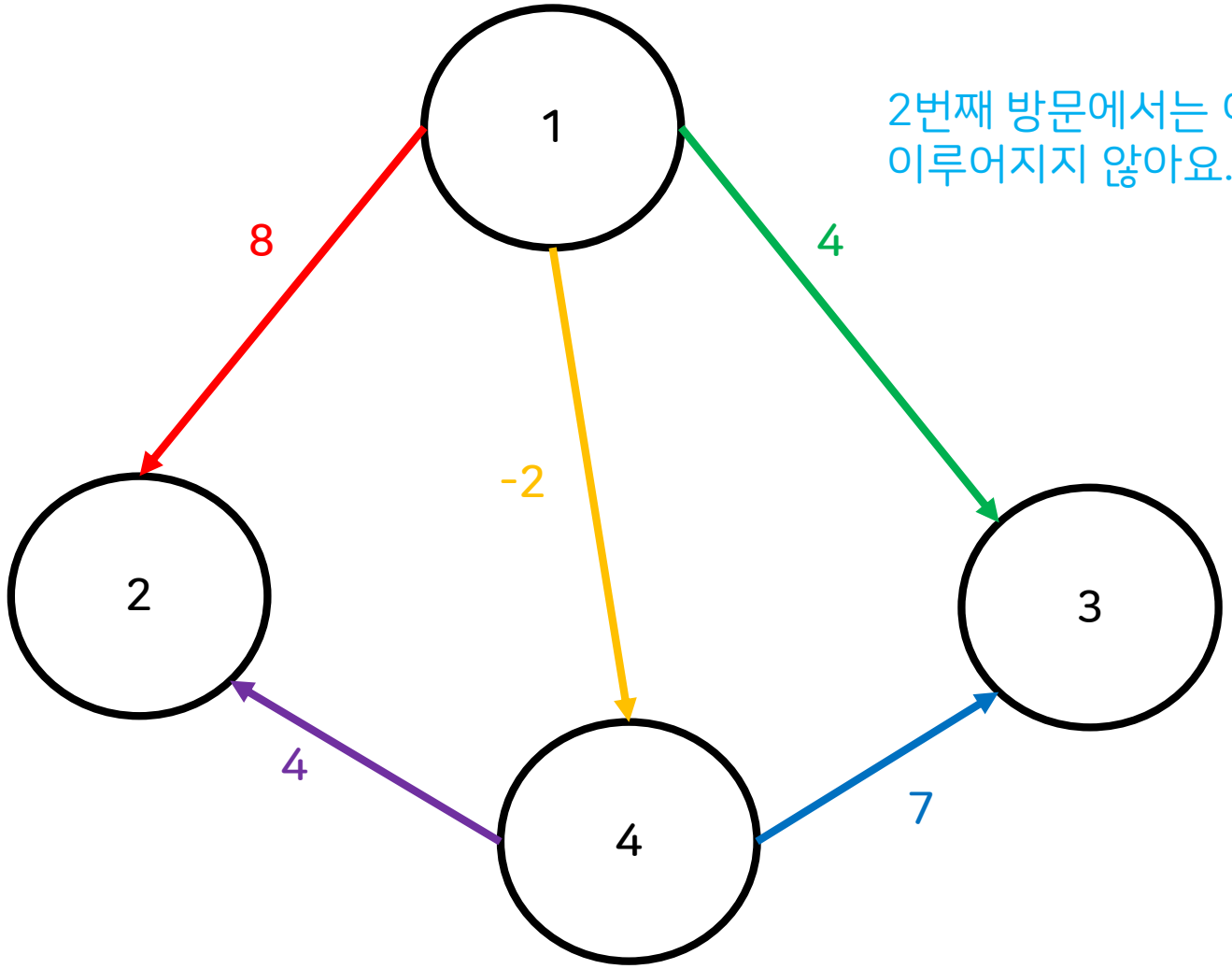


보라색 간선을 지나면
2번 정점까지 2만큼 걸린다.
1->4->2
-2 + 4

1번째 방문

정점번호	최단거리
1	0
2	2
3	5
4	-2

Bellman-Ford_예시

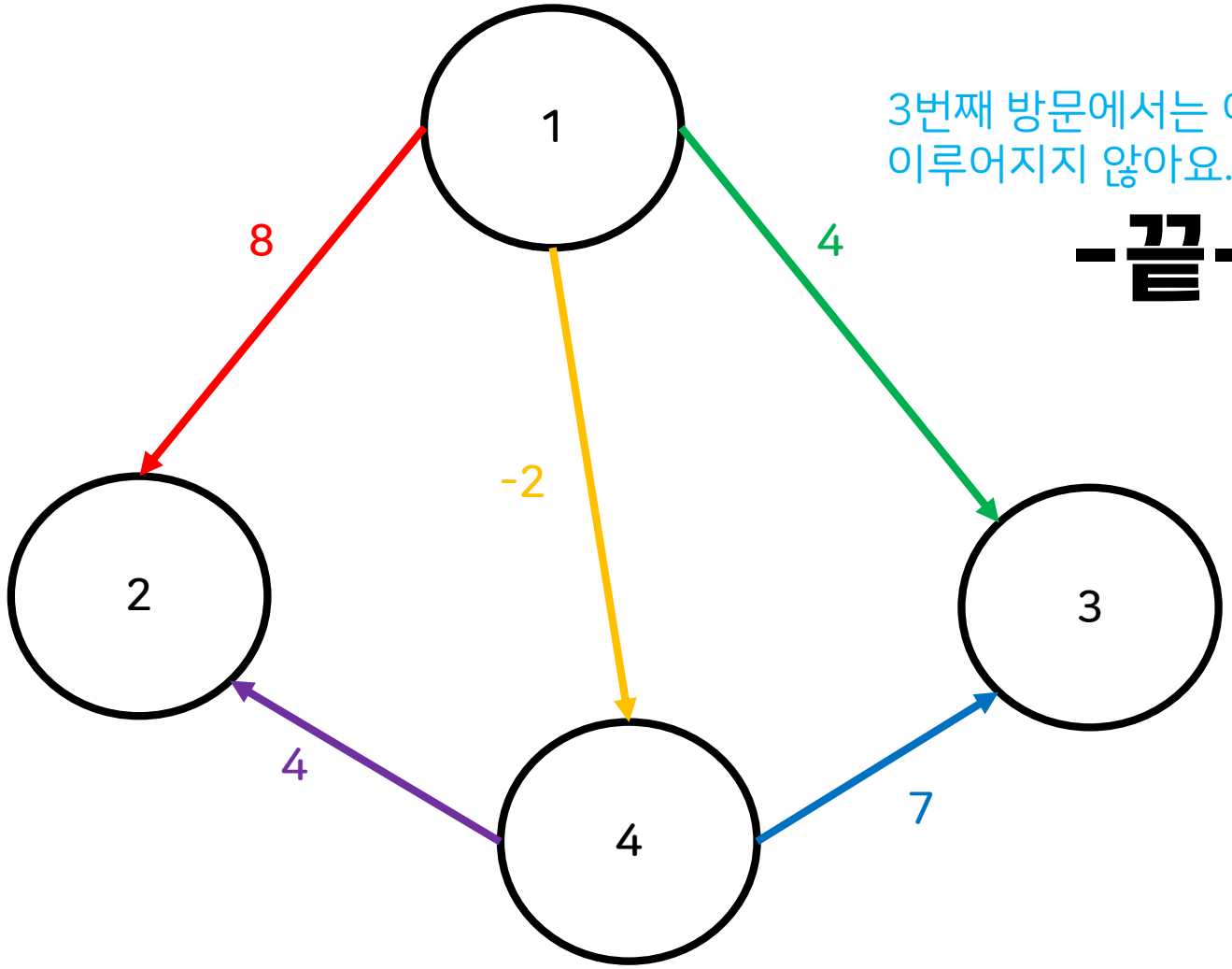


2번째 방문에서는 어떠한 갱신도 이루어지지 않아요.

2번째 방문

정점번호	최단거리
1	0
2	2
3	5
4	-2

Bellman-Ford_예시



3번째 방문에서는 어떠한 갱신도 이루어지지 않아요.

-끝-

3번째 방문

정점번호	최단거리
1	0
2	2
3	5
4	-2

Bellman-Ford_예시

만약 무지개 순서가 아니라 거꾸로 한다면?

→ 2번째에도 갱신이 일어남.

간선을 체크하는 순서에 따라 갱신되는 과정이 달라짐.

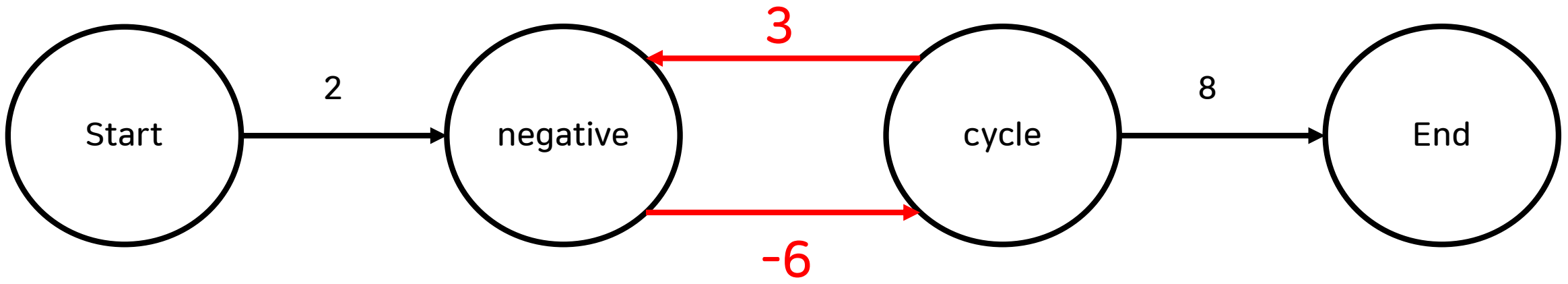
그러나 V-1번 이후에는 갱신이 일어나지 않음.

생각보다 쉬운데???

Bellman-Ford 음수 사이클

진짜가 나타났다.

Bellman-Ford_음수사이클



Negative랑 cycle을 계속 왔다 갔다 하면???

최단경로가 음의 무한대로 발산한다???

이걸 어떻게 찾지?

Bellman-Ford_개념

이말은??
다음 슬라이드에

임의의 간선을 경유해서 가는 경로가 더 빠르다면 갱신

V개의 정점이 있을 때, 한 정점에서 다른 정점으로 가는 최단거리는
최대 (V – 1)개의 간선을 지난다.

→ Ex) 1번 정점에서 5번 정점까지 정점을 최대한 거쳐가는 최단경로는
1->2->3->4->5 이렇게 4개의 간선을 거쳐가는 것이 최대이다.
1->2->5 는 최단경로가 될 수는 있지만
1->2->3->2->4->5 는 될 수가 없다.

모든 간선에 대해 V-1번 갱신하면?

방문되지 않은 정점(값이 무한대)에서 출발하는 edge는 고려하지 않음.

Bellman-Ford_음수사이클

V개의 정점이 있을 때, 한 정점에서 다른 정점으로 가는 최단거리는
최대 (V − 1)개의 간선을 지난다.

→ V-1번 돌린 이후에는 갱신이 일어나지 않는다.
→ V-1번만 갱신하면 된다.
(예시에서도 3번만 했어요.)

음수사이클이 있다면?

→ 몇 번을 돌리든 계속 갱신이 일어남.

→V번째 돌렸을 때도 갱신이 일어나면 음수사이클 존재!!!

Bellman-Ford_코드

- ShortPath[i] :
  시작점에서 i번 정점까지의 최단경로
- N: 정점의 개수
  M: 간선의 개수
- Edges: 간선들을 모아 놓은 vector
- ret: 음수사이클이 있으면 false
- V번(1번 더 돌린다고 큰일나지 않음)돌려서 최단경로 탐색
- V번(1번만 해도 됨) 또 돌려서 음수사이클 존재 파악

```
bool Bellman_Ford(void)
{
    ShortPath[1] = 0;
    for ( int n = 1; n <= N; n++ )
    {
        for ( int m = 0; m < M; m++ )
        {
            auto e = Edges[m];
            if ( ShortPath[e.from] == INF )
                continue;
            else if ( ShortPath[e.to] > ShortPath[e.from] + e.weight )
                ShortPath[e.to] = ShortPath[e.from] + e.weight;
        }
    }

    bool ret = true;
    for ( int n = 1; n <= N; n++ )
    {
        for ( int m = 0; m < M; m++ )
        {
            auto e = Edges[m];
            if ( ShortPath[e.from] == INF )
                continue;
            else if ( ShortPath[e.from] == -INF )
                ShortPath[e.to] = -INF;
            else if ( ShortPath[e.to] > ShortPath[e.from] + e.weight )
            {
                ShortPath[e.to] = -INF;
                ret = false;
            }
        }
    }

    return ret;
}
```

Shortest Path_총정리

Floyd-Warshall

모든 정점에 대하여~
모든 경로에 대해 다 구할 수 있음.
시간 복잡도는 $O(V^3)$

Bellman-Ford

모든 간선에 대하여~
시작점 정해져 있음
음수사이클 체크 가능
시간 복잡도는 $O(VE)$

Dijkstra

시작점 정해져 있음
음수가중치 불가
시간 복잡도는 $O(E \log V)$

E: 간선의 개수
V: 정점의 개수

참고: $V - 1 \leq E \leq V^2$

예제

11657 - 타임머신

11657_타임머신

- 문제 이해가 안되는 사람은 없겠죠?(제발제발제발제발)
- 역시나 첫 문제는 쉽습니다.
- 다음 슬라이드에 해답이 있습니다.
- 혼자서 풀기 힘들거나 잘 안될 때는 강사의 코드를 베껴 보아요.

11657_타임머신

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

#include <climits>
using namespace std;

using ll = long long;

const ll INF = LLONG_MAX;

int N, M;
int A, B, C;

class Edge
{
public:
    int from, to, weight;

    Edge(void)
    {
        from = 0;
        to = 0;
        weight = 0;
    }
    Edge(int u, int v, int w)
    {
        from = u;
        to = v;
        weight = w;
    }
    Edge(const Edge& edge)
    {
        from = edge.from;
        to = edge.to;
        weight = edge.weight;
    }
};

vector<Edge> Edges;
vector<ll> ShortPath;

bool Bellman_Ford(void);
```

```cpp
int main(void)
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    cin >> N >> M;
    ShortPath.resize(N + 1, INF);
    for ( int m = 1; m <= M; m++ )
    {
        cin >> A >> B >> C;
        Edges.push_back(Edge(A, B, C));
    }

    if ( Bellman_Ford() )
    {
        for ( int n = 2; n <= N; n++ )
        {
            if ( ShortPath[n] == INF )
                cout << "-1\n";
            else
                cout << ShortPath[n] << '\n';
        }
    }
    else
    {
        cout << "-1\n";
    }

    return 0;
}
```

Bellman_Ford함수만 작성하면 정답입니다.

~~(이 함수가 앞에 있었던것 같은데…)~~

예제

11657 - 타임머신

이번엔 자신의 스타일로 코드를 바꿔서 다시 제출해보아요.

연습문제

1865 - 웜홀

1866_웜홀

- 사실 타임머신 보다 쉬운 문제
- 모든 시작점에 대해 음수사이클이 하나라도 있으면 YES!
- 이 밑에 답이있어요.

음수사이클이 하나라도 있으면 YES가 답!

연습문제

1219 –오민식의 고민

1219_오민식의 고민

- 모델링을 신경 써 주세요. 어떠한 자료구조가 필요할지 어떤 방법으로 동작할지
- 돈을 많~~~이 버는 것이 목적이죠? 최대 돈 알고리즘이 필요합니다.
- 최단↔최대, 거리↔돈
- 음수 사이클이 있다고 해서 그 사이클이 목적지와 꼭 연결되라는 법은 없겠죠?

연습문제

3860 –할로윈 묘지

이 문제를 푼 선착순 10명(알고리즘반 학생만) 밥 사드립니다.
(~~수업 열심히 듣고 Bellman-Ford 열심히 공부 했다면 누구나 해결할 수 있어요.~~)

고생하셨습니다!!

"나도 고생했어……."