2021 겨울 신촌 연합 알고리즘 캠프
벨만 포드 & 플로이드 와샬

초급 알고리즘

HI-ARC 김기선

백준 11657 타임머신

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	256 MB	26637	3044	1882	17.540%

문제

N개의 도시가 있다. 그리고 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.

입력

첫째 줄에 도시의 개수 N (1 ≤ N ≤ 500), 버스 노선의 개수 M (1 ≤ M ≤ 6,000)이 주어진다. 둘째 줄부터 M개의 줄에는 버스 노선의 정보 A, B, C (1 ≤ A, B ≤ N, -10,000 ≤ C ≤ 10,000)가 주어진다.

출력

만약 1번 도시에서 출발해 어떤 도시로 가는 과정에서 시간을 무한히 오래 전으로 되돌릴 수 있다면 첫째 줄에 -1을 출력한다. 그렇지 않다면 N-1개 줄에 걸쳐 각 줄에 1번 도시에서 출발해 2번 도시, 3번 도시, ..., N번 도시로 가는 가장 빠른 시간을 순서대로 출력한다. 만약 해당 도시로 가는 경로가 없다면 대신 -1을 출력한다.

벨만 포드

한 개의 정점에서 나머지 정점까지의 최단 거리를 찾아주는 알고리즘

다익스트라 알고리즘과 매우 비슷하지만 음수 간선일때도 가능

벨만 포드

정점 A,B,C,D가 있을 때 A에서 C로 가는 최단 경로를 생각해보자

C를 갈 때 B를 경유하는 경우, D를 경유하는 경우, A에서 직접 가는 경우

가능한 경우의 수를 모두 확인하는 알고리즘

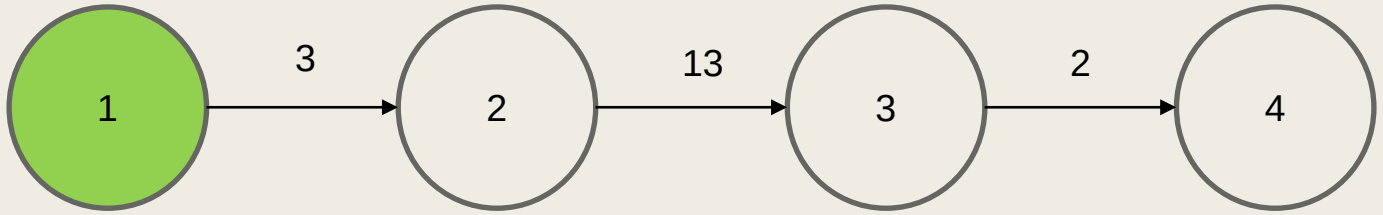
벨만 포드

최단 경로로 가려면 같은 지점을 한 번 이하로 지나야 함

따라서 최대 V-1개의 간선을 지나면 최단 경로를 형성할 수 있다는 것을 이용한다.

벨만 포드

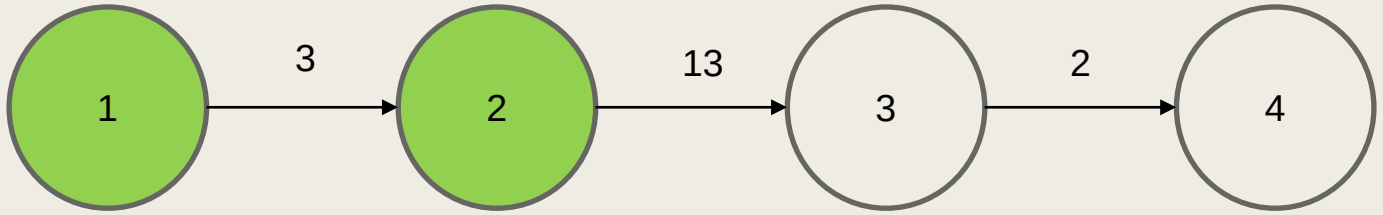
초기



	1	2	3	4
Dist[]	**0**	∞	∞	∞

벨만 포드

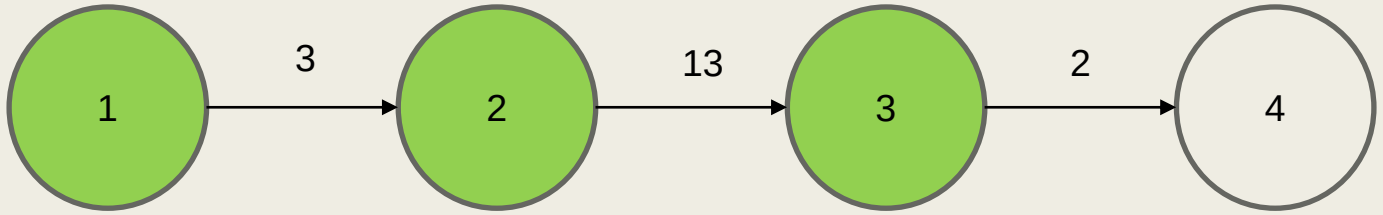
1회



	1	2	3	4
Dist[]	**0**	**3**	∞	∞

벨만 포드

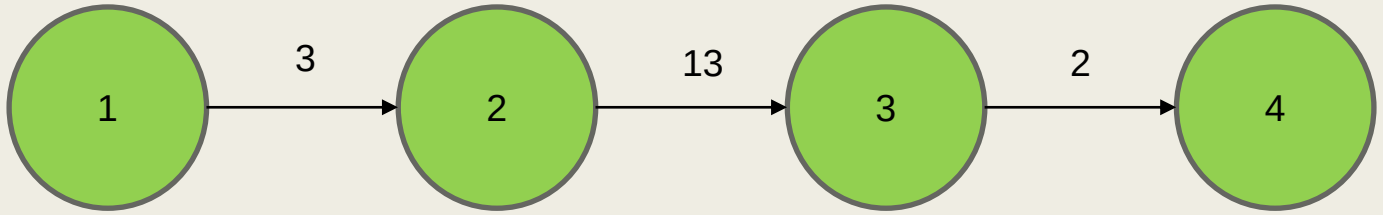
2회



	1	2	3	4
Dist[]	**0**	**3**	**16**	∞

벨만 포드

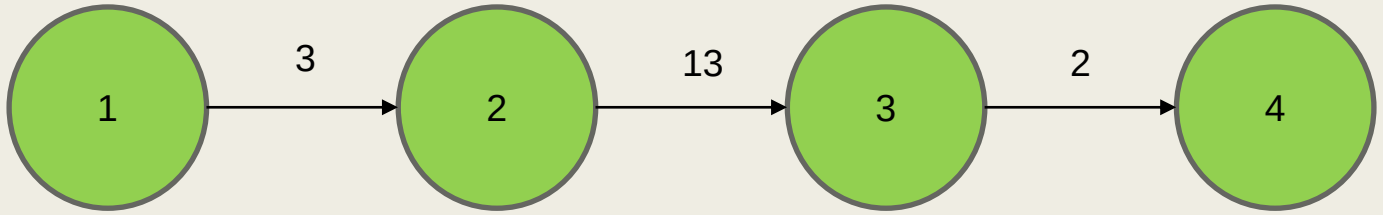
3회



	1	2	3	4
Dist[]	**0**	**3**	**16**	**18**

벨만 포드

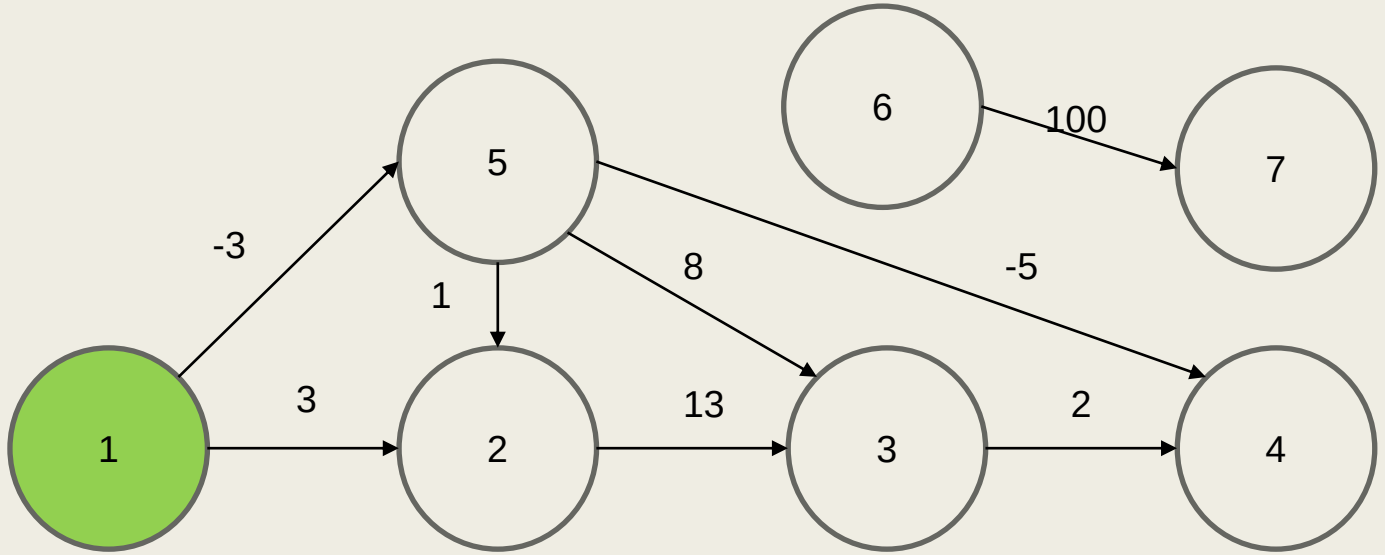
4회



	1	2	3	4
Dist[]	**0**	**3**	**16**	**18**

벨만 포드

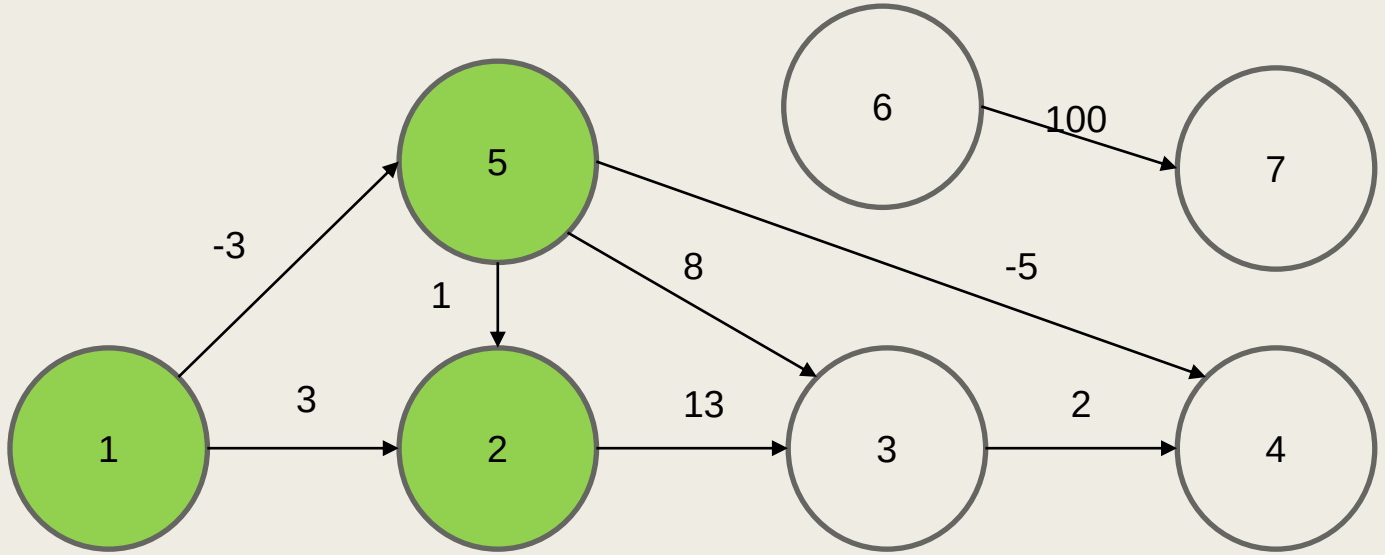
초기



	1	2	3	4	5	6	7
Dist[]	**0**	∞	∞	∞	∞	∞	∞

벨만 포드

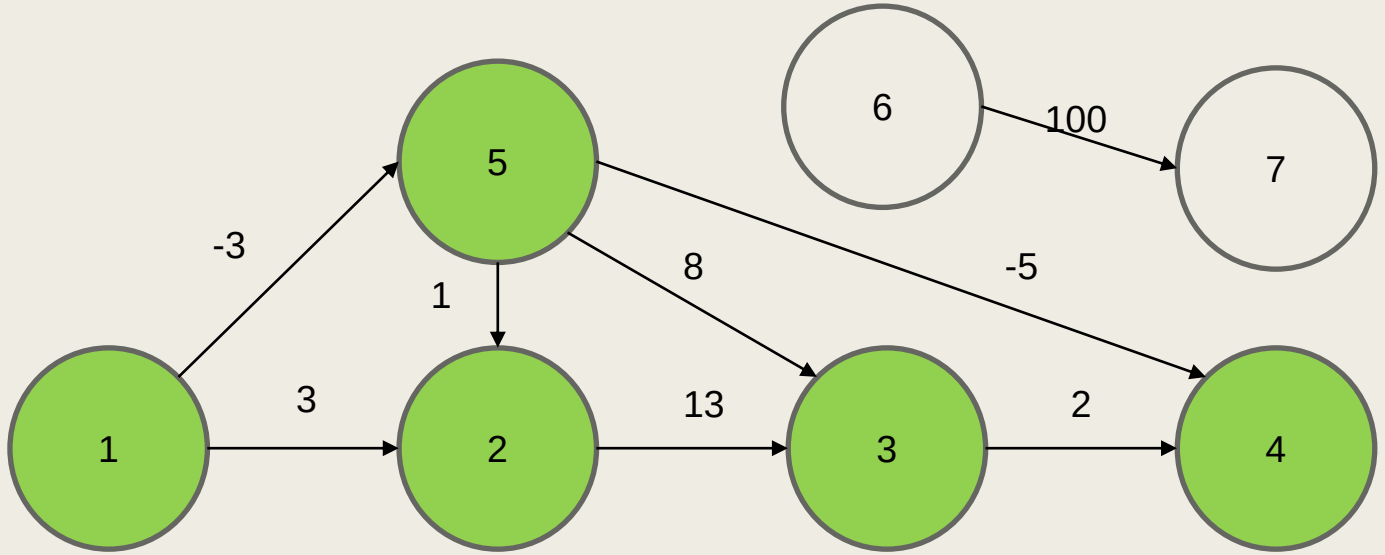
1회



	1	2	3	4	5	6	7
Dist[]	**0**	**3**	∞	∞	**-3**	∞	∞

벨만 포드

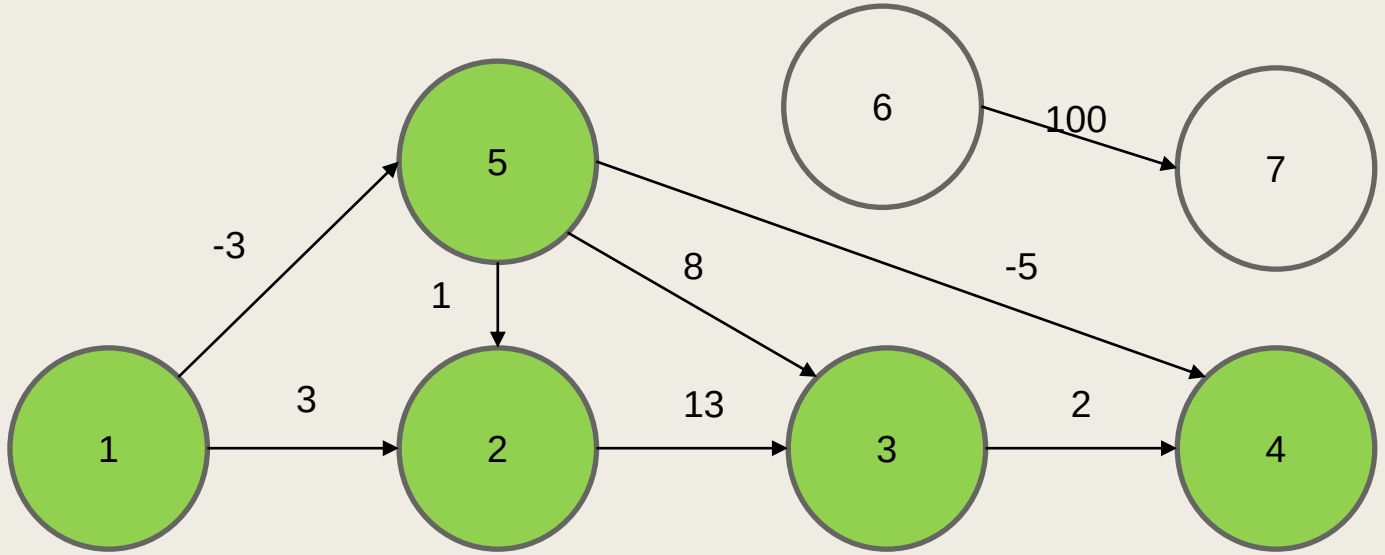
2회



	1	2	3	4	5	6	7
Dist[]	**0**	**-2**	**5**	**-8**	**-3**	∞	∞

벨만 포드

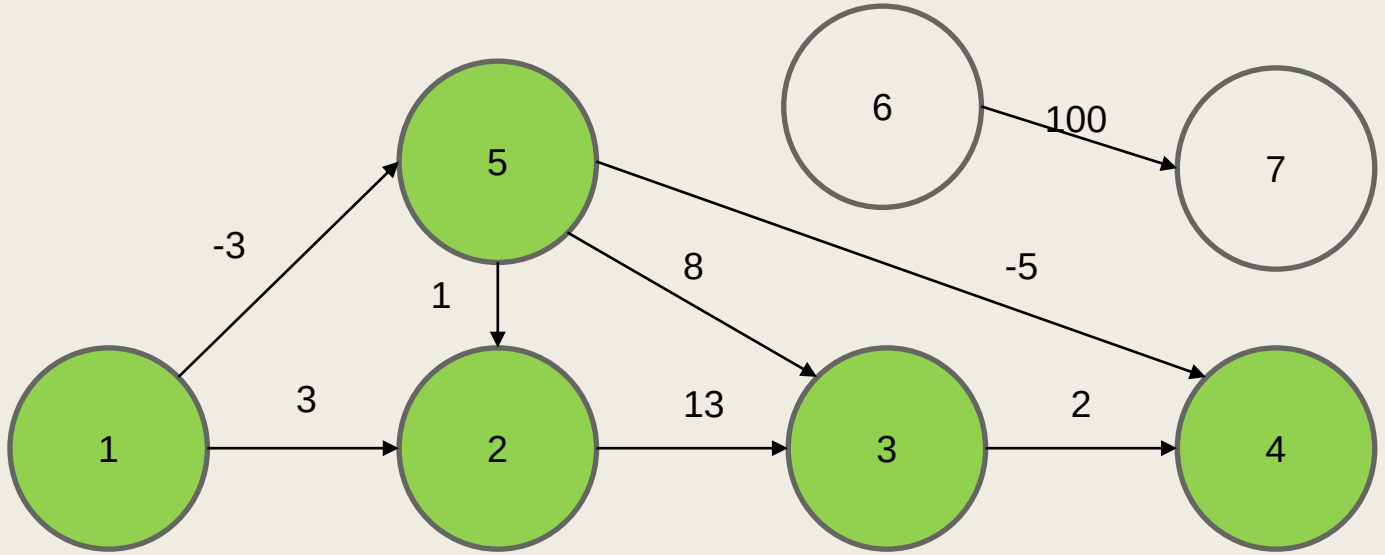
3회



	1	2	3	4	5	6	7
Dist[]	**0**	**-2**	**5**	**-8**	**-3**	∞	∞

벨만 포드

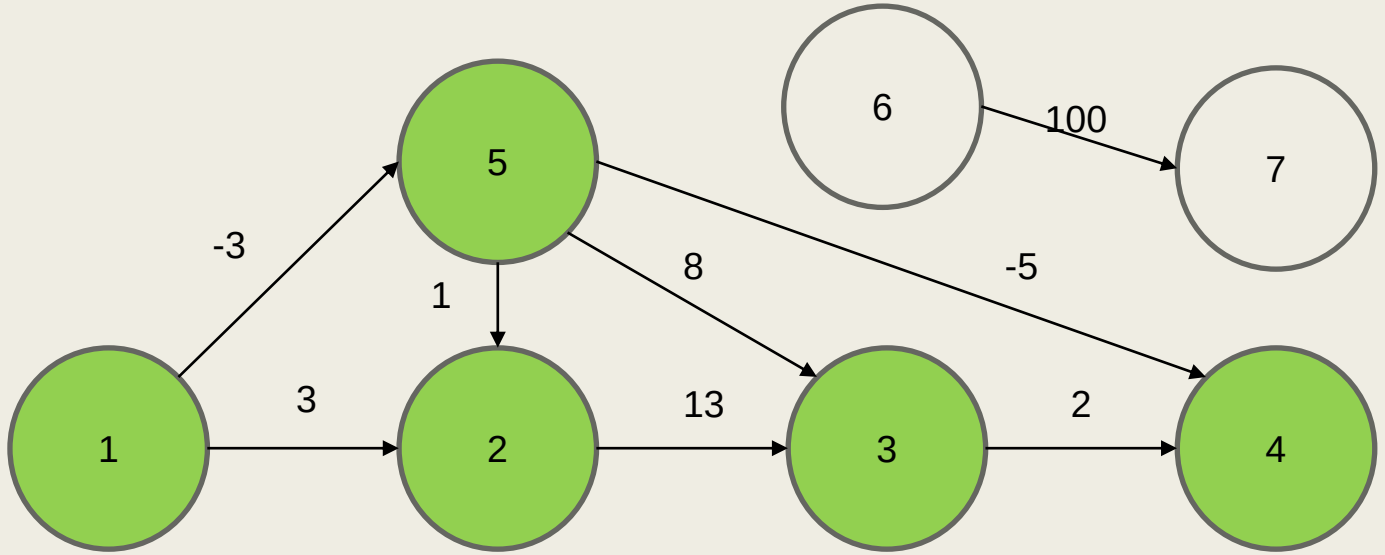
4회



	1	2	3	4	5	6	7
Dist[]	**0**	**-2**	**5**	**-8**	**-3**	∞	∞

벨만 포드

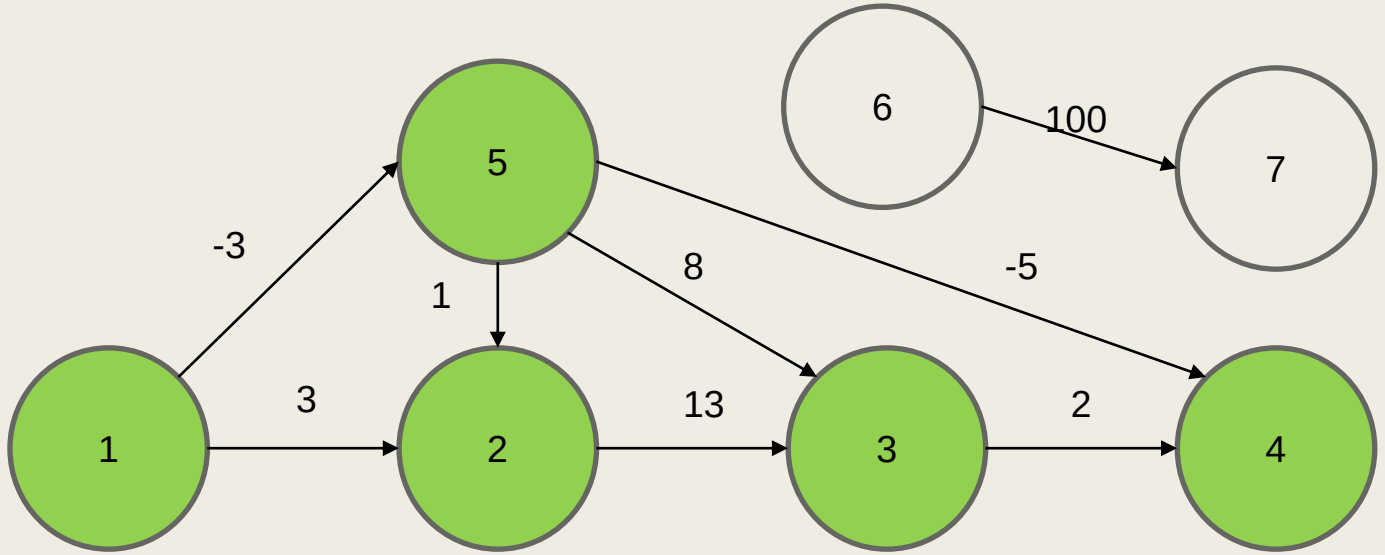
5회



	1	2	3	4	5	6	7
Dist[]	**0**	**-2**	**5**	**-8**	**-3**	∞	∞

벨만 포드

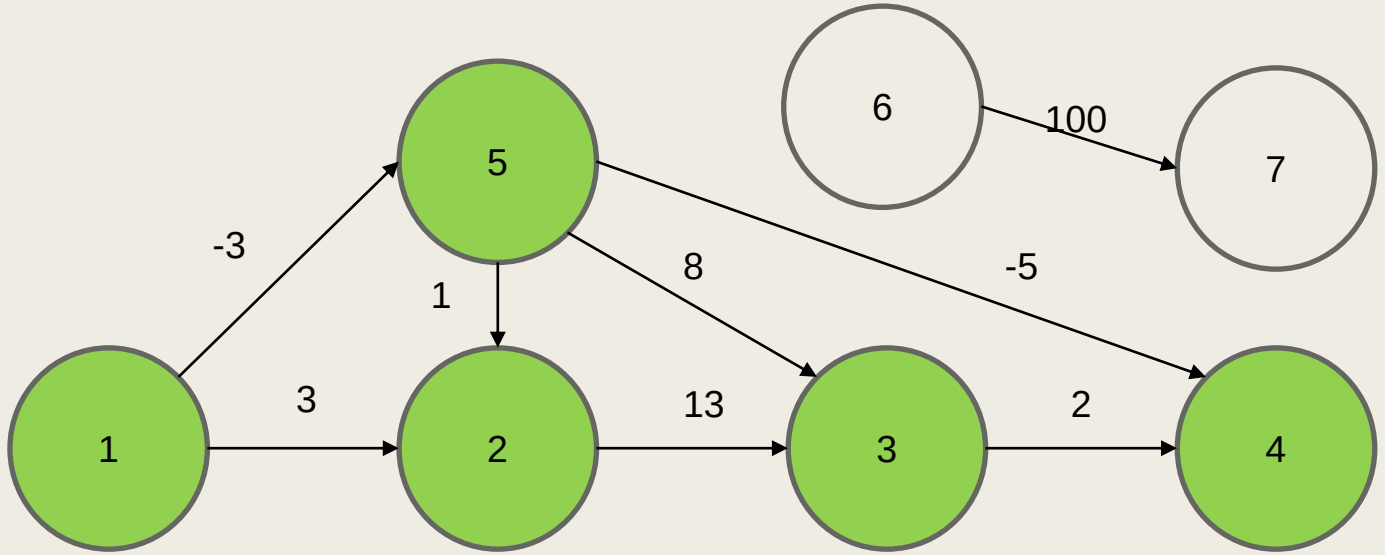
6회



	1	2	3	4	5	6	7
Dist[]	**0**	**-2**	**5**	**-8**	**-3**	∞	∞

벨만 포드

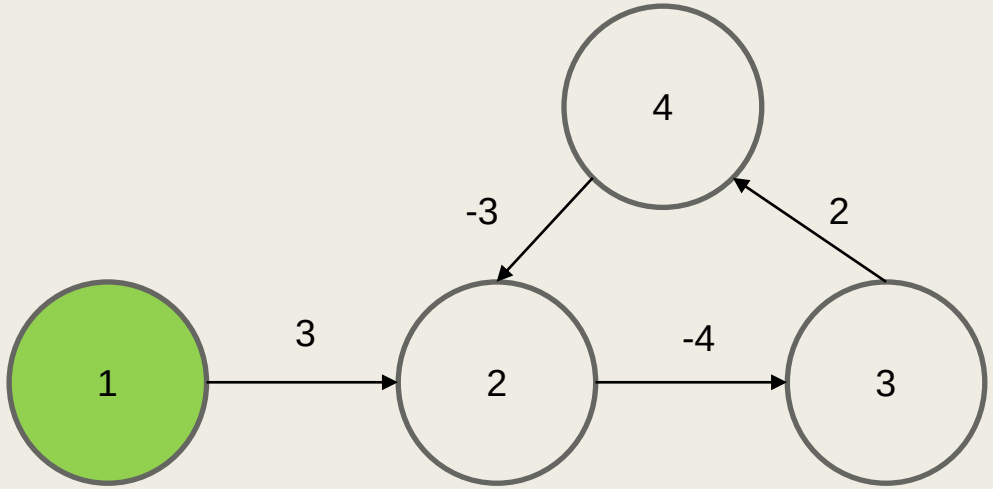
7회



	1	2	3	4	5	6	7
Dist[]	**0**	**-2**	**5**	**-8**	**-3**	∞	∞

벨만 포드

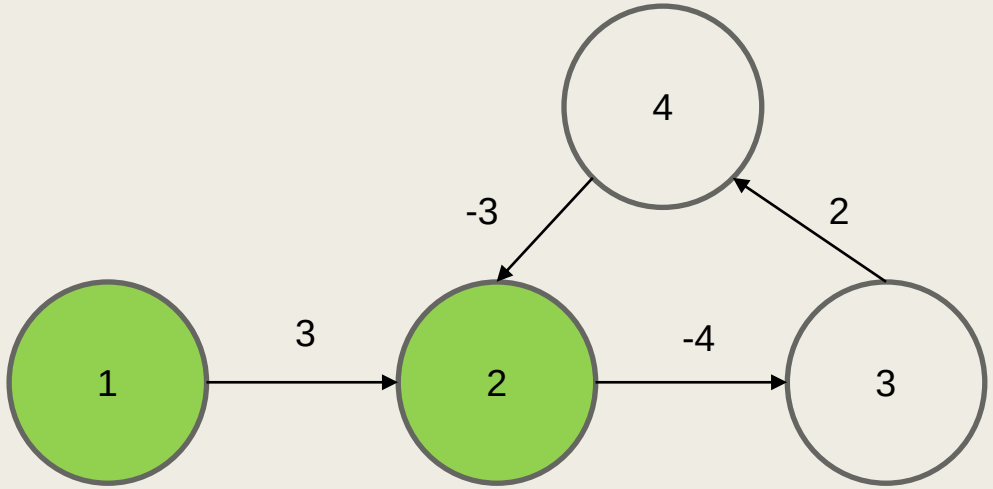
초기



	1	2	3	4
Dist[]	**0**	∞	∞	∞

벨만 포드

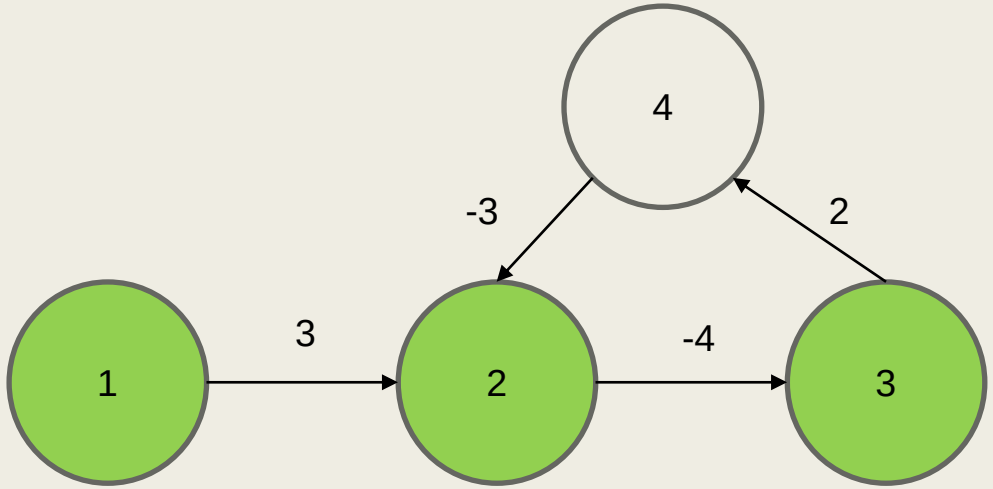
1회



	1	2	3	4
Dist[]	**0**	**3**	∞	∞

벨만 포드

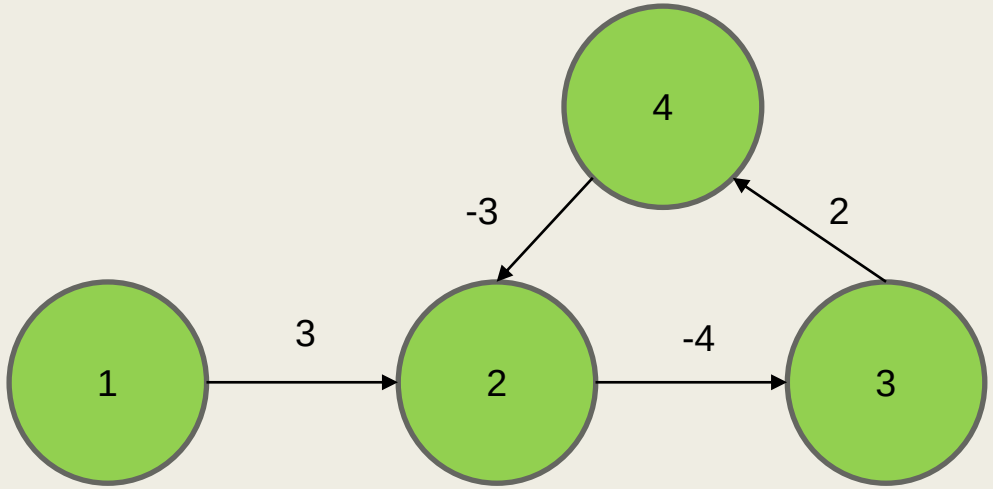
2회



	1	2	3	4
Dist[]	**0**	**3**	**-1**	∞

벨만 포드

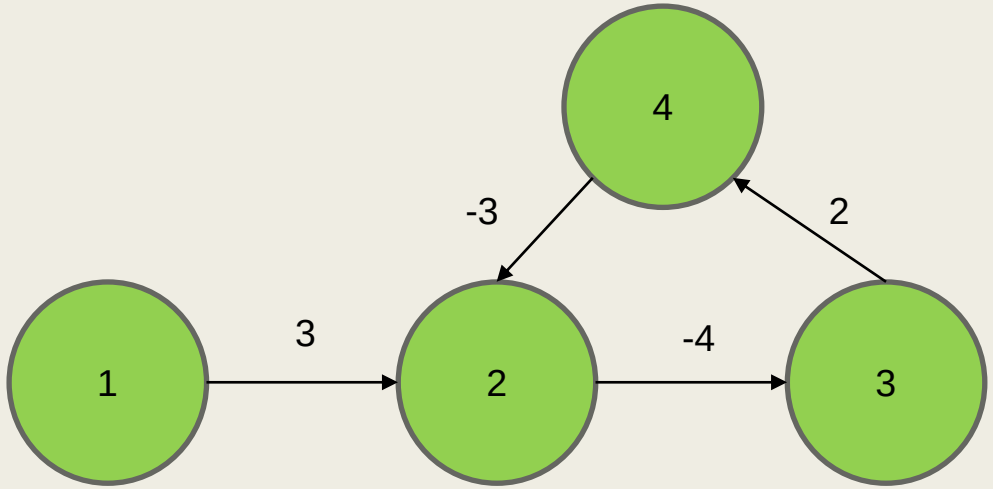
3회



	1	2	3	4
Dist[]	**0**	**3**	**-1**	**1**

벨만 포드

4회



	1	2	3	4
Dist[]	**0**	**-2**	**-1**	**1**

Cycle 발생

벨만 포드

- 한 점에서 출발하는 거리배열을 작성한다. (그 지점의 거리 = 0, 나머지 INF)
- 이어진 모든 간선에 대해서 V-1번 최솟값을 갱신해준다. (INF면 분리된 거)
- 만약 V번째에도 변화가 생긴다면 음의 사이클 존재

시간 복잡도 : O(VE) (모든 간선에 대해서 V번 체크)

벨만 포드 소스코드

```
int main() {
    int V, E;
    cin >> V >> E;
    for (int i = 0; i < E; i++) {
        // 간선을 입력받아 그래프를 구현
        long long a, b, c;
        cin >> a >> b >> c;
        adj[a].push_back({ b,c });
    }
    for (int i = 2; i <= V; i++)dist[i] = INF;
    // 절대 불가능한 값으로 설정

    bool negative_cycle = false;
    for (int i = 1; i <= V; i++) {
        // 이어진 모든 간선에 대해서 V번 반복하면서 거리배열의 최솟값을 갱신
        // a,b,c에서 a->c로 간다고 했을 때, b를 경유, a에서 직접 등 모두 확인
        // 최단 경로로 가려면 한 지점은 최대 한번만 들리기 때문에
        // 최대 V-1의 간선이면 최단경로로 갈 수 있다. (단 음의 사이클이 존재하지 않는다는 가정하에)
        // 음의 사이클이 존재한다면 특정 점에 갈 때는 무한히 빠르게 갈 수 있음
        for (int from = 1; from <= V; from++) {
            if (dist[from] == INF)continue;
            // 이어지지 않은 정점은 패스
            for (int a = 0; a < adj[from].size(); a++) {
                int to = adj[from][a].first;
                int cost = adj[from][a].second;
                if (dist[to] > dist[from] + cost) {
                    if (i == V)negative_cycle = true;
                    // 모든 간선을 V번째 보았을때도 변화가 있다면 음수 사이클 존재
                    dist[to] = dist[from] + cost;
                }
            }
        }
    }
}
```

백준 11657 타임머신

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	256 MB	26637	3044	1882	17.540%

문제

N개의 도시가 있다. 그리고 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.

입력

첫째 줄에 도시의 개수 N (1 ≤ N ≤ 500), 버스 노선의 개수 M (1 ≤ M ≤ 6,000)이 주어진다. 둘째 줄부터 M개의 줄에는 버스 노선의 정보 A, B, C (1 ≤ A, B ≤ N, -10,000 ≤ C ≤ 10,000)가 주어진다.

출력

만약 1번 도시에서 출발해 어떤 도시로 가는 과정에서 시간을 무한히 오래 전으로 되돌릴 수 있다면 첫째 줄에 -1을 출력한다. 그렇지 않다면 N-1개 줄에 걸쳐 각 줄에 1번 도시에서 출발해 2번 도시, 3번 도시, ..., N번 도시로 가는 가장 빠른 시간을 순서대로 출력한다. 만약 해당 도시로 가는 경로가 없다면 대신 -1을 출력한다.

백준 1865 웜홀

문제

때는 2020년, 백준이는 월드나라의 한 국민이다. 월드나라에는 N개의 지점이 있고 N개의 지점 사이에는 M개의 도로와 W개의 웜홀이 있다. (단 도로는 방향이 없으며 웜홀은 방향이 있다.) 웜홀은 시작 위치에서 도착 위치로 가는 하나의 경로인데, 특이하게도 도착을 하게 되면 시작을 하였을 때보다 시간이 뒤로 가게 된다. 웜홀 내에서는 시계가 거꾸로 간다고 생각하여도 좋다.

시간 여행을 매우 좋아하는 백준이는 한 가지 궁금증에 빠졌다. 한 지점에서 출발을 하여서 시간여행을 하기 시작하여 다시 출발을 하였던 위치로 돌아왔을 때, 출발을 하였을 때보다 시간이 되돌아가 있는 경우가 있는지 없는지 궁금해졌다. 여러분은 백준이를 도와 이런 일이 가능한지 불가능한지 구하는 프로그램을 작성하여라.

입력

첫 번째 줄에는 테스트케이스의 개수 TC($1 \le TC \le 5$)가 주어진다. 그리고 두 번째 줄부터 TC개의 테스트케이스가 차례로 주어지는데 각 테스트케이스의 첫 번째 줄에는 지점의 수 N($1 \le N \le 500$), 도로의 개수 M($1 \le M \le 2500$), 웜홀의 개수 W($1 \le W \le 200$)이 주어진다. 그리고 두 번째 줄부터 M+1번째 줄에 도로의 정보가 주어지는데 각 도로의 정보는 S, E, T 세 정수로 주어진다. S와 E는 연결된 지점의 번호, T는 이 도로를 통해 이동하는데 걸리는 시간을 의미한다. 그리고 M+2번째 줄부터 M+W+1번째 줄까지 웜홀의 정보가 S, E, T 세 정수로 주어지는데 S는 시작 지점, E는 도착 지점, T는 줄어드는 시간을 의미한다. T는 10,000보다 작거나 같은 자연수 또는 0이다.

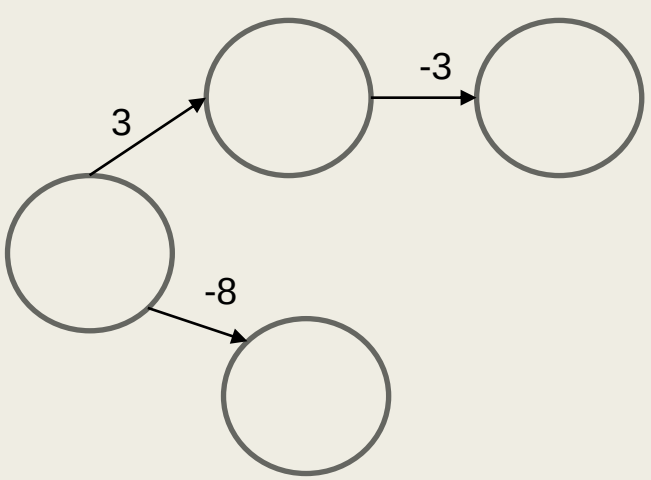
두 지점을 연결하는 도로가 한 개보다 많을 수도 있다. 지점의 번호는 1부터 N까지 자연수로 중복 없이 매겨져 있다.

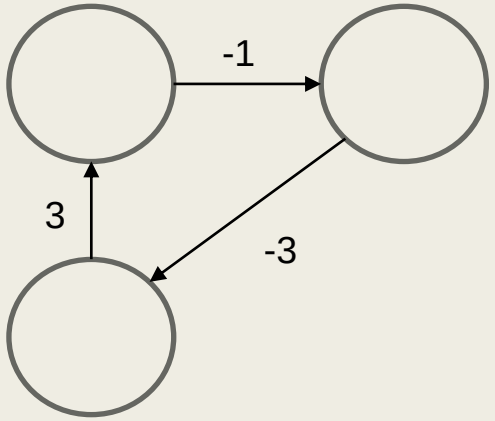
출력

TC개의 줄에 걸쳐서 만약에 시간이 줄어들면서 출발 위치로 돌아오는 것이 가능하면 YES, 불가능하면 NO를 출력한다.

백준 1865 웜홀

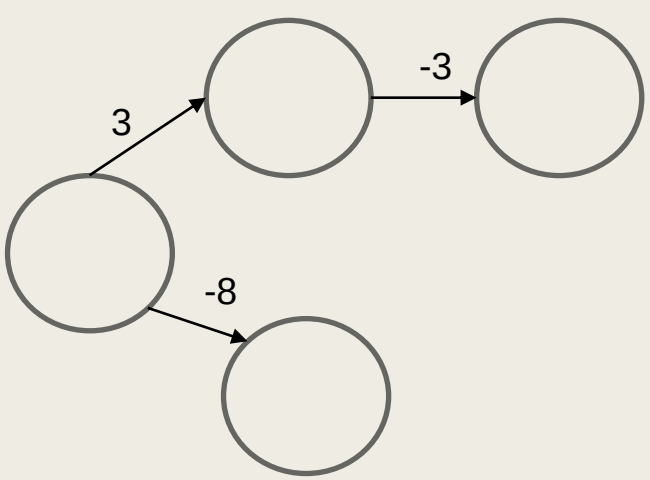
백준 1865 웜홀

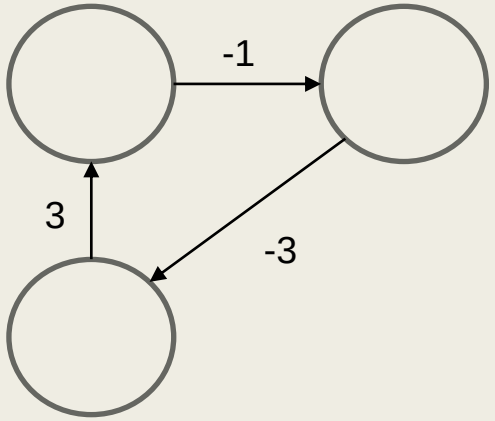




백준 1865 웜홀

다른 방법은?





플로이드 와샬

모든 정점 -> 모든 정점으로의 최단 거리를 구하는 알고리즘

A,B,C,D 의 정점이 있고, A->D의 최단 경로를 구하려면

B를 경유, C를 경유, A에서 직접 가는 경우들을 모두 확인하면서 최솟값을 갱신

플로이드 와샬

i -> j까지의 최단거리(i,j,k) = i 에서 j 까지 가는데 k개의 정점을 사용한 최단 거리

dist[i][j] 와 dist[i][k] + dist[k][j] 를 비교 최솟값으로 갱신

최단 거리(i,j,k) => 최단 거리(i,j,k-1)을 가지고 구함

따라서 dp형태의 방식이 됨

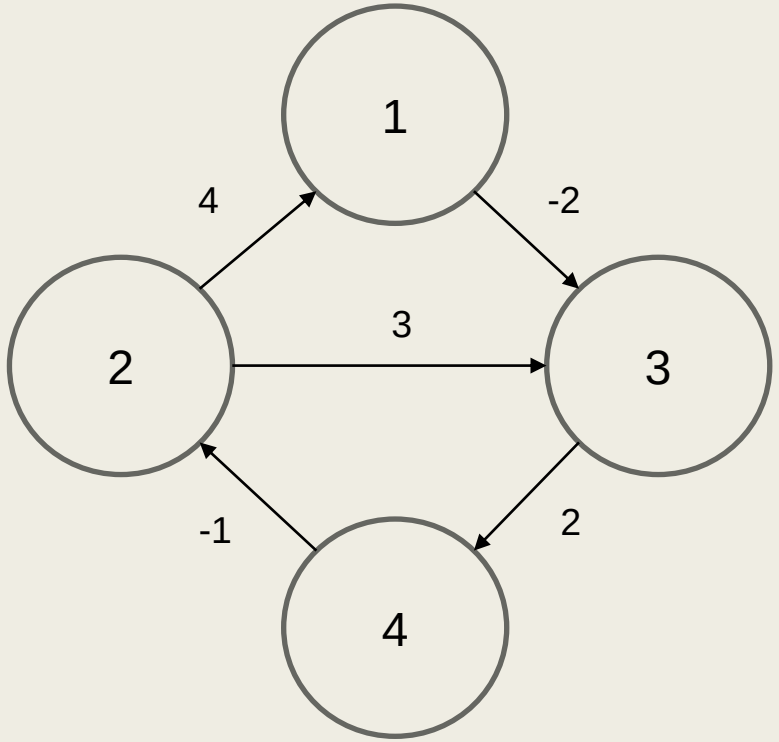
플로이드 와샬

A,B,C,D 의 정점들로 구현된 그래프가 있을 때 C를 뺐다고 생각해보자

C를 제외했을 때의 최단 거리를 구하고, 이제 모든 경로에 C를 끼어 넣어보면 그 전에 A,B,C,D에서의 최단 거리와 동일하다.

플로이드 와샬

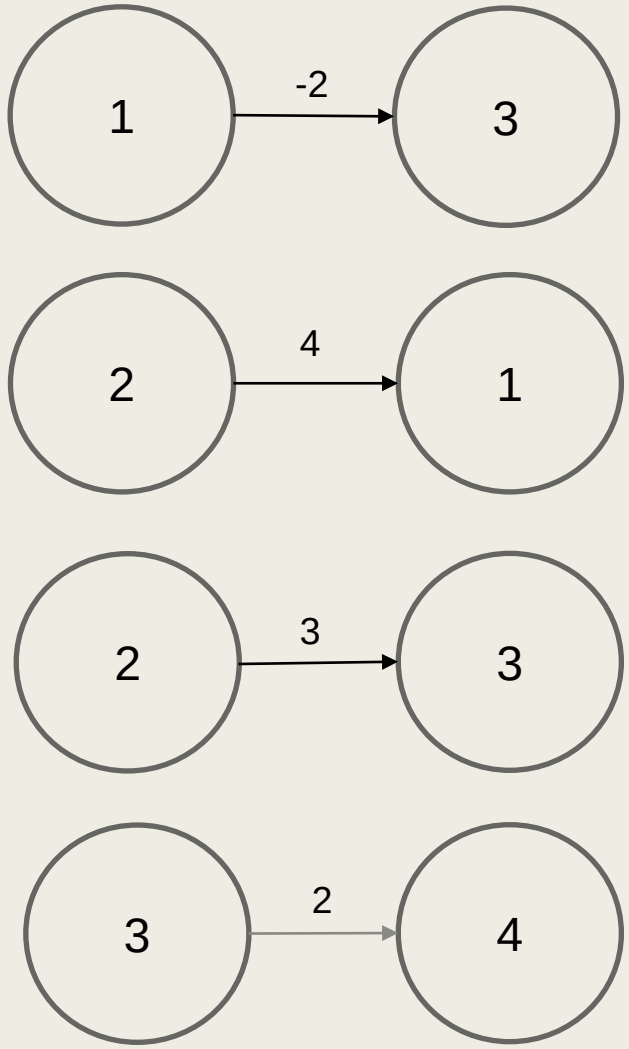
단방향 간선 그래프

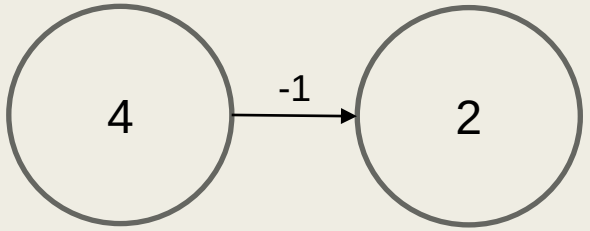


플로이드 와샬

K = 0

i \ j	1	2	3	4
1	0	∞	-2	∞
2	4	0	3	∞
3	∞	∞	0	2
4	∞	-1	∞	0

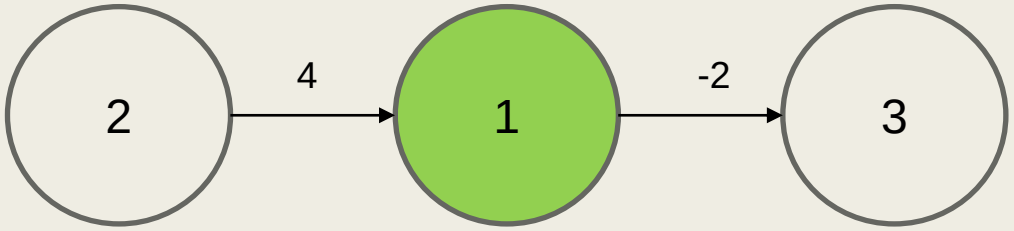




플로이드 와샬

K = 1
(1을 경유하는 경우)

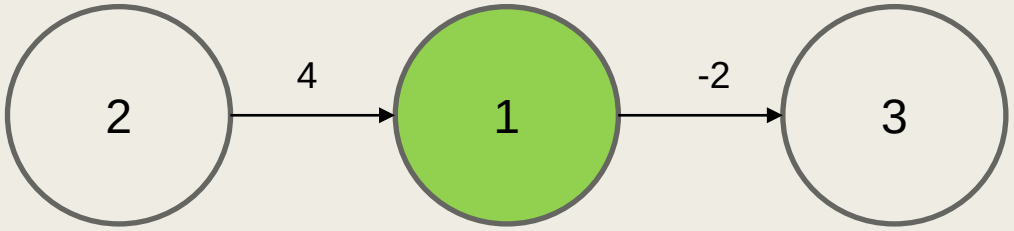
i \ j	1	2	3	4
1	0	∞	-2	∞
2	4	0	3	∞
3	∞	∞	0	2
4	∞	-1	∞	0



플로이드 와샬

K = 1
(1을 경유하는 경우)

i \ j	1	2	3	4
1	0	∞	-2	∞
2	4	0	2	∞
3	∞	∞	0	2
4	∞	-1	∞	0

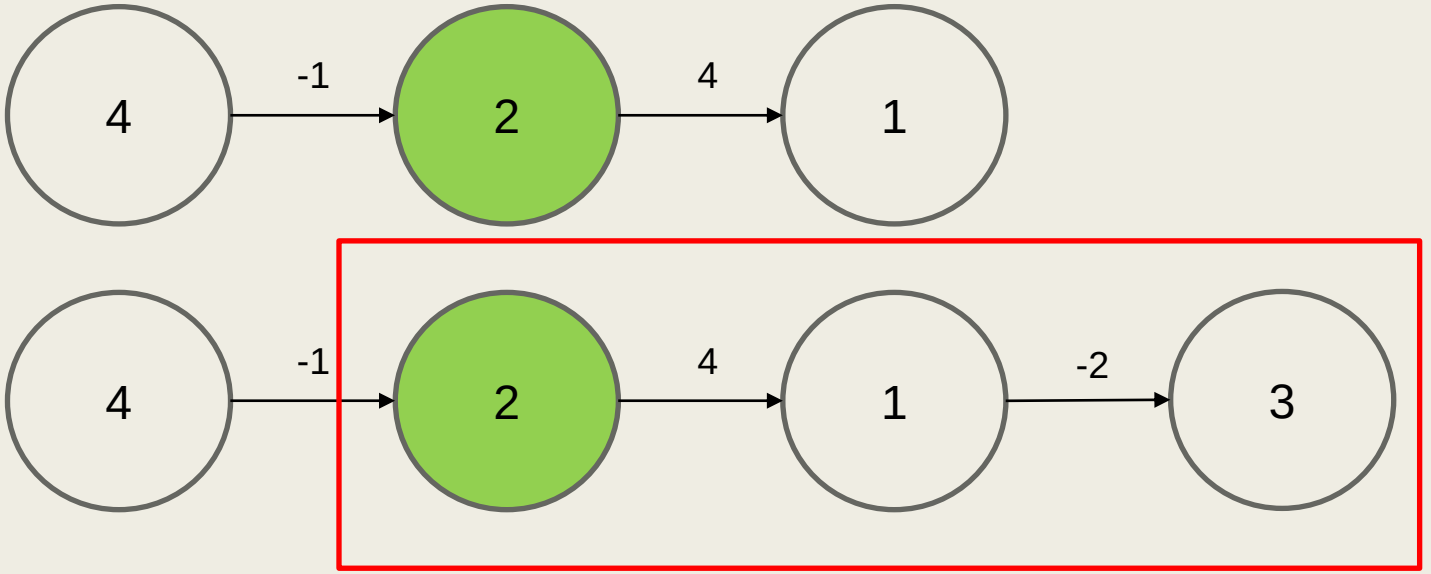


dist[2][3] > dist[2][1] + dist[1][3] 이므로 갱신 O

플로이드 와샬

K = 2
(2를 경유하는 경우)

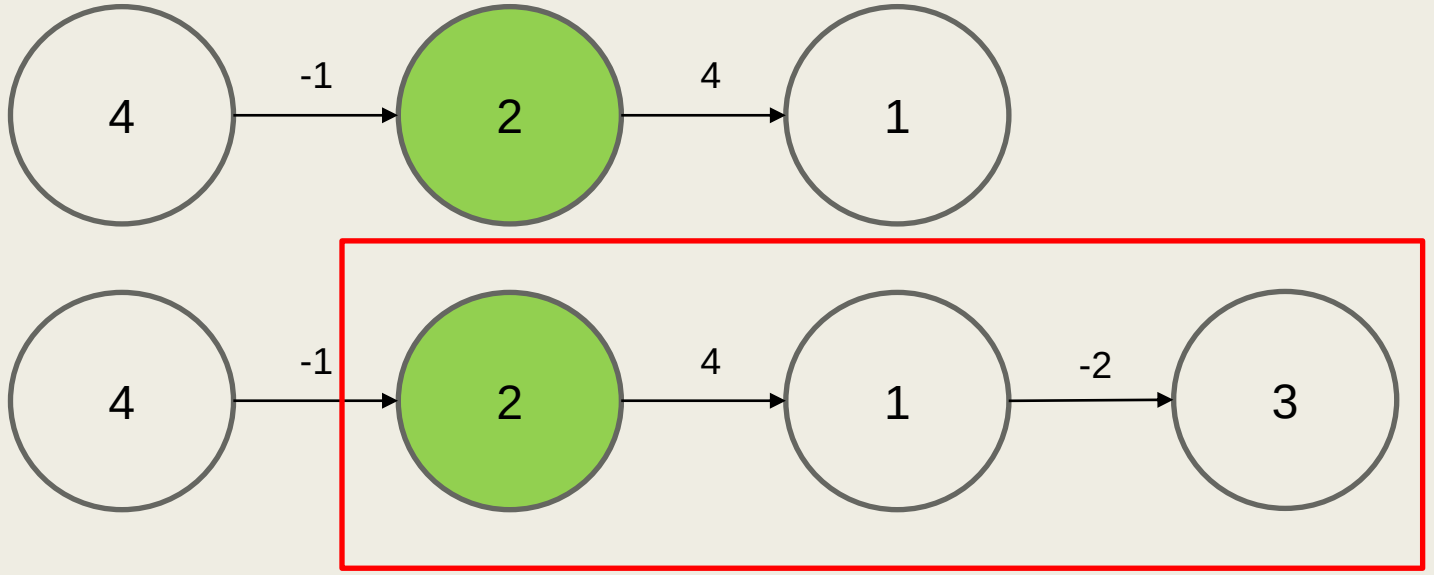
i \ j	1	2	3	4
1	0	∞	-2	∞
2	4	0	2	∞
3	∞	∞	0	2
4	∞	-1	∞	0



플로이드 와샬

K = 2
(2를 경유하는 경우)

i \ j	1	2	3	4
1	0	∞	-2	∞
2	4	0	2	∞
3	∞	∞	0	2
4	3	-1	1	0

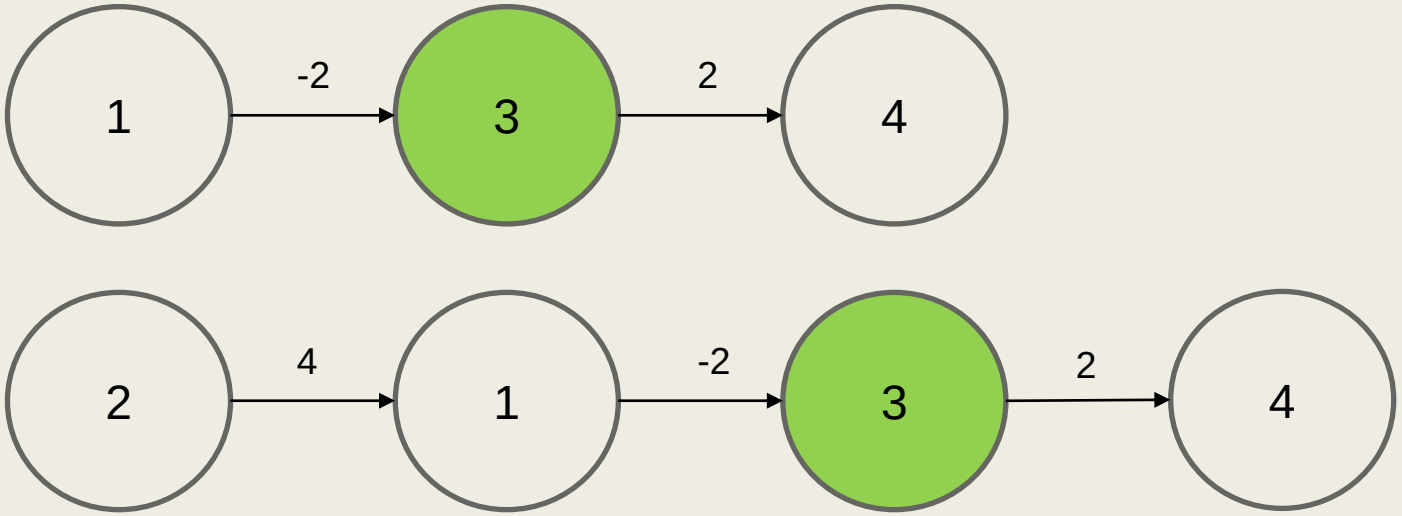


dist[4][1] > dist[4][2] + dist[2][1] 이므로 갱신 O
dist[4][3] > dist[4][2] + dist[2][3] 이므로 갱신 O

플로이드 와샬

K = 3
(3을 경유하는 경우)

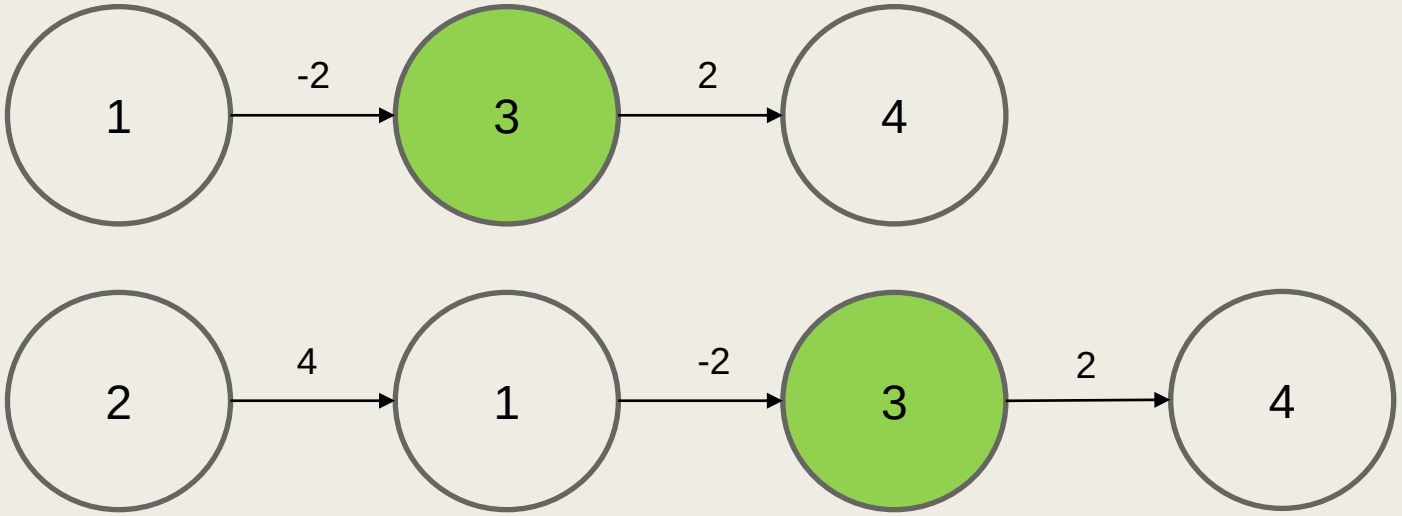
i \ j	1	2	3	4
1	0	∞	-2	∞
2	4	0	2	∞
3	∞	∞	0	2
4	3	-1	1	0



플로이드 와샬

K = 3
(3을 경유하는 경우)

i \ j	1	2	3	4
1	0	∞	-2	0
2	4	0	2	4
3	∞	∞	0	2
4	3	-1	1	0

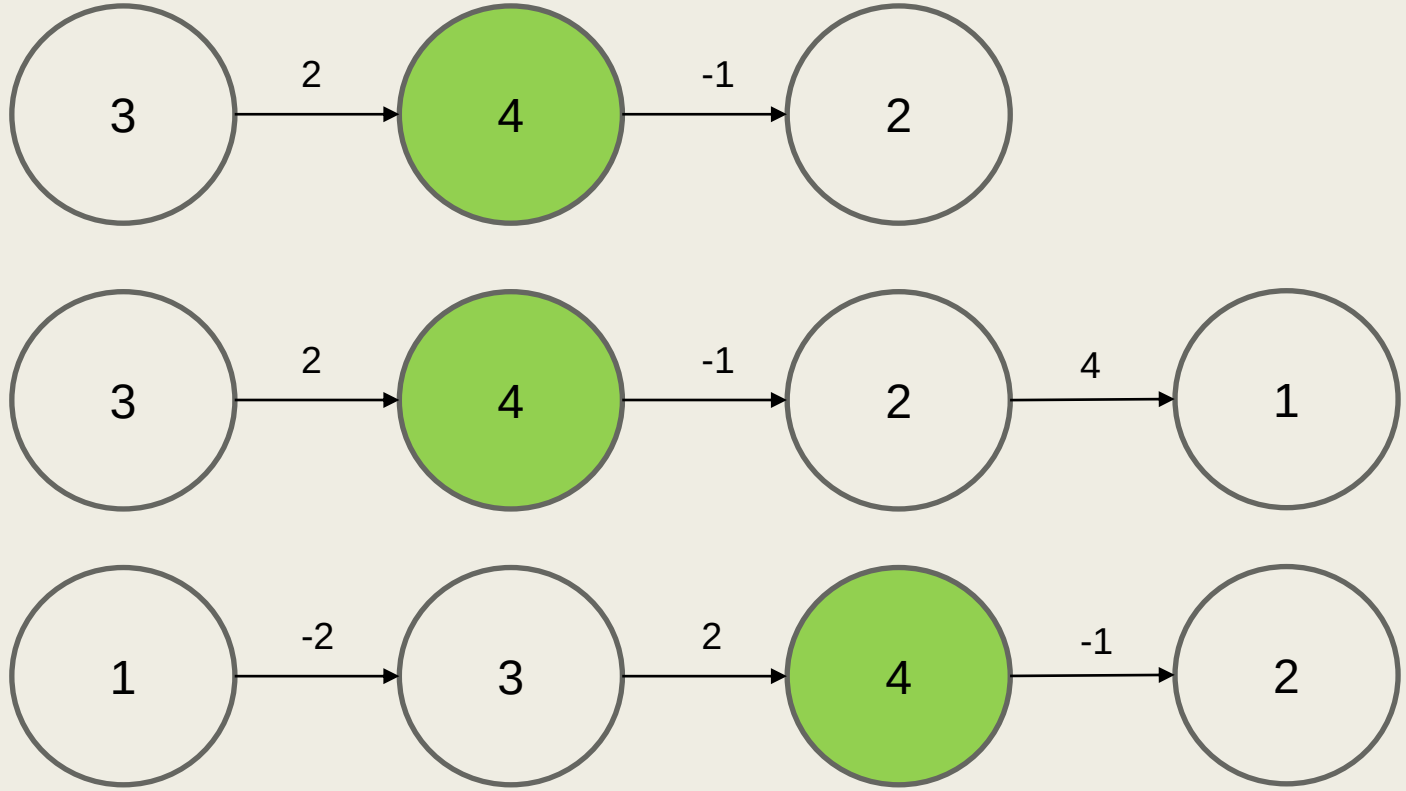


dist[1][4] > dist[1][3] + dist[3][4] 이므로 갱신 O
dist[2][4] > dist[2][3] + dist[3][4] 이므로 갱신 O

플로이드 와샬

K = 4
(4를 경유하는 경우)

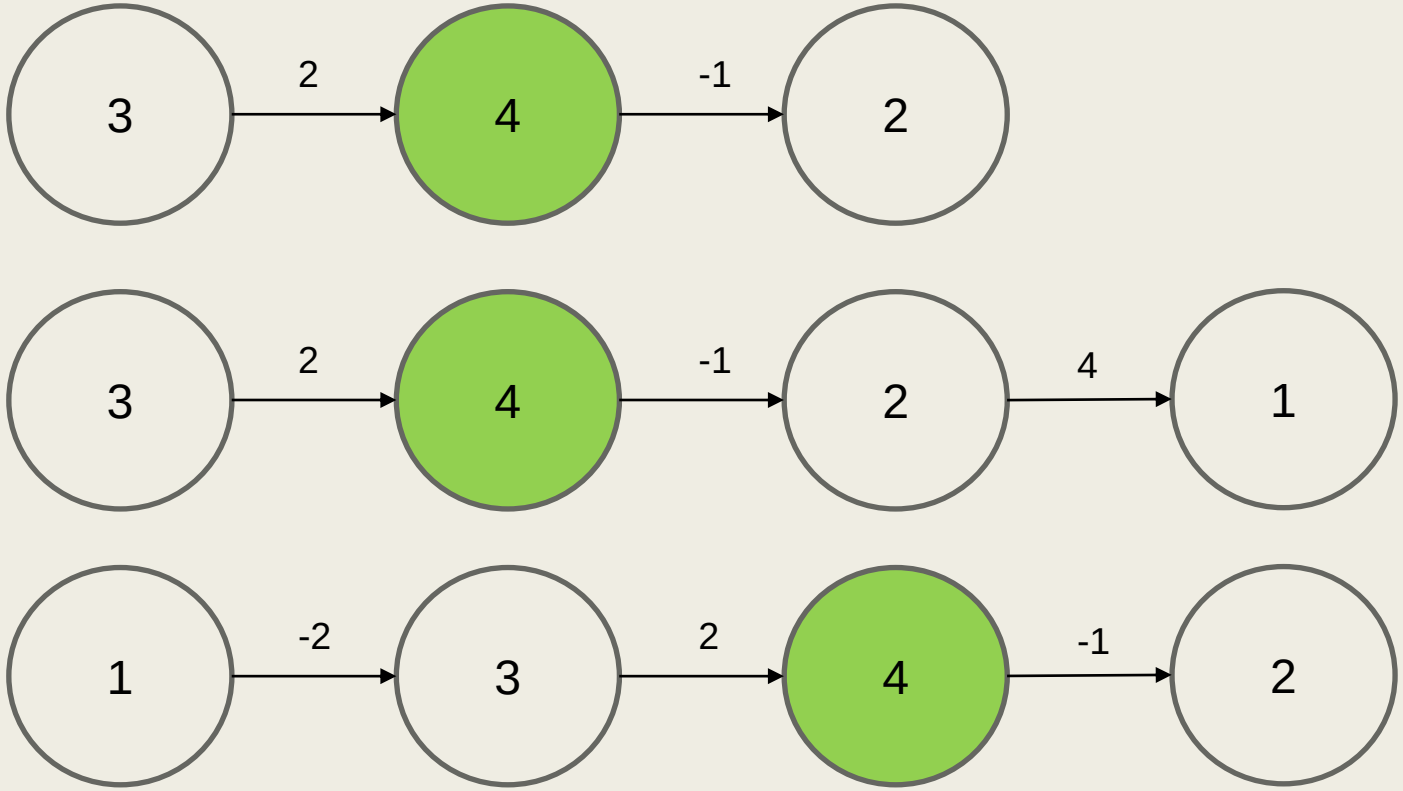
i \ j	1	2	3	4
1	0	∞	-2	0
2	4	0	2	4
3	∞	∞	0	2
4	3	-1	1	0



플로이드 와샬

K = 4
(4를 경유하는 경우)

i \ j	1	2	3	4
1	0	-1	-2	0
2	4	0	2	4
3	5	1	0	2
4	3	-1	1	0

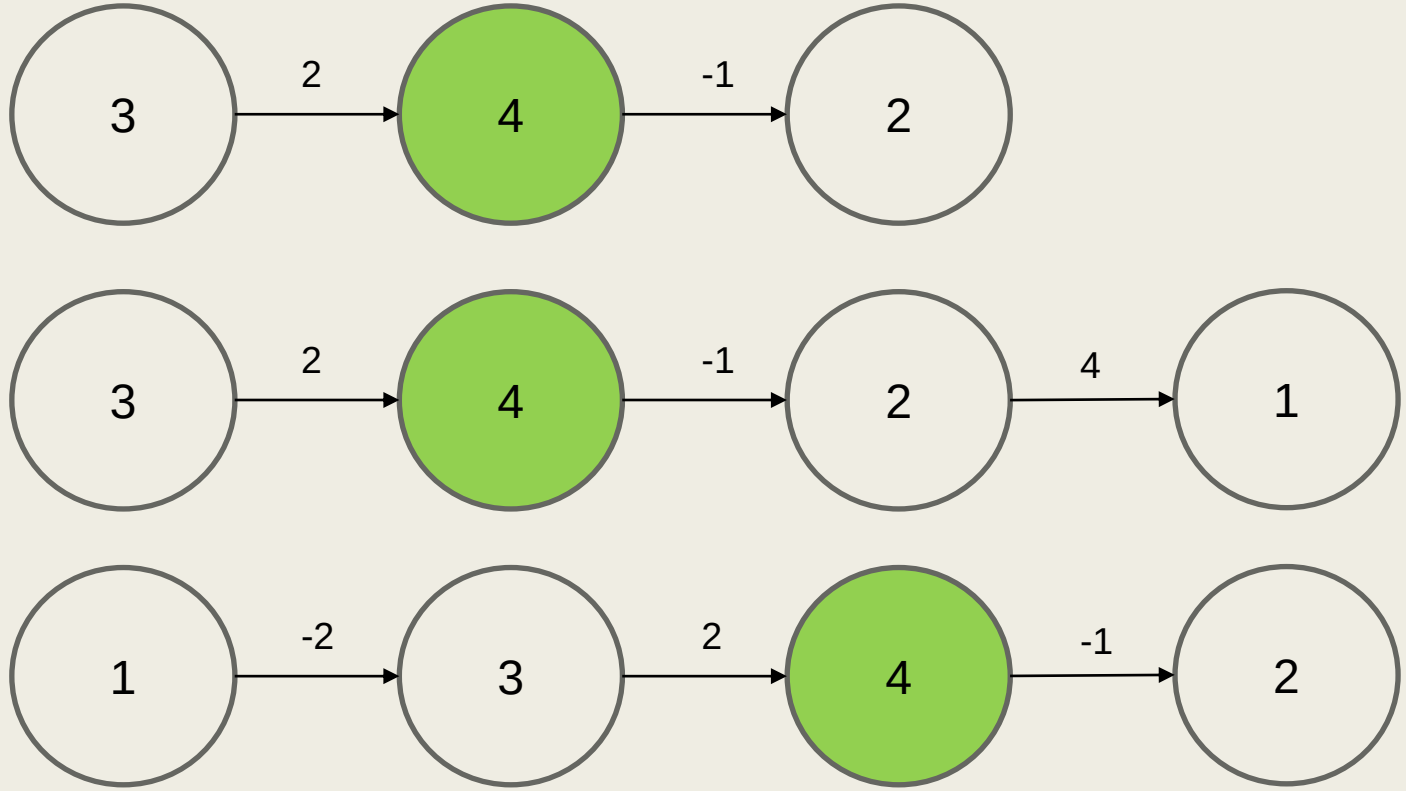


dist[3][2] > dist[3][4] + dist[4][2] 이므로 갱신 O
dist[3][1] > dist[3][4] + dist[4][1] 이므로 갱신 O
dist[1][2] > dist[1][4] + dist[4][2] 이므로 갱신 O

플로이드 와샬

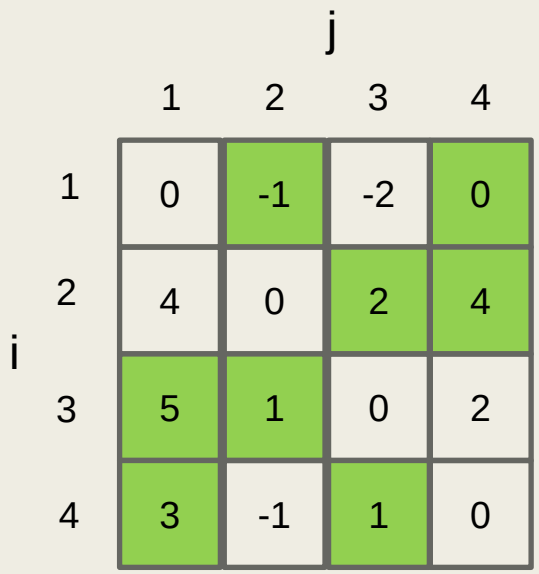
K = 4
(4를 경유하는 경우)

i \ j	1	2	3	4
1	0	-1	-2	0
2	4	0	2	4
3	5	1	0	2
4	3	-1	1	0



dist[3][2] > dist[3][4] + dist[4][2] 이므로 갱신 O
dist[3][1] > dist[3][4] + dist[4][1] 이므로 갱신 O
dist[1][2] > dist[1][4] + dist[4][2] 이므로 갱신 O

플로이드 와샬



만약 모든 연산후에도 ∞로 남아 있다면 i->j로 가는 경로 없다고 보면 됨

그렇다면 플로이드 와샬에서는 음의 사이클을 어떻게 찾을까?

주 대각선에 있는 (i,i)의 값들 중 하나라도 0이 아닌 다른 수라면
음의 사이클 존재한다고 보면 됨

플로이드 와샬 소스코드

```cpp
int main() {
    int V, E;
    cin >> V >> E;
    for (int i = 1; i <= V; i++) {
        for (int j = 1; j <= V; j++) {
            if (i == j)continue;
            // 자기 자신까지의 거리는 0 그대로 (전역변수 초기값 0)
            dist[i][j] = INF;
            // 절대 불가능한 값으로 설정
        }
    }
    for (int i = 0; i < E; i++) {
        int a, b, c;
        cin >> a >> b >> c;
        dist[a][b] = min(dist[a][b], c);
        //간선도 가장 최솟값으로
    }

    for (int k = 1; k <= V; k++) {
        // 정점 k를 경유하는 경우
        for (int i = 1; i <= V; i++) {
            for (int j = 1; j <= V; j++) {
                dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);
                // 최솟값을 갱신해줘야 한다면 갱신
            }
        }
    }
}
```

필수문제

- A - 타임머신
- B - 웜홀
- C - 케빈 베이컨의 6단계 법칙
- D - 경로 찾기
- E - 플로이드

연습문제

- A - 오민식의 고민
- B - 골목길
- C - 할로윈 묘지
- D - 택배
- E - 운동
- F - 플로이드 2
- G - 저울
- H - 도망자 원숭이

피드백 및 질의응답

감사합니다!