#12-3. 벨만 포드 알고리즘

나정휘

https://justiceHui.github.io/

벨만 포드 알고리즘

Bellman-Ford Algorithm

- SSSP를 푸는 알고리즘
- 시간 복잡도 : O(VE)

- 몇 가지 관찰을 해보자.
  - 관찰 1) 가중치의 합이 음수인 사이클이 있으면 최단 거리를 구할 수 없음 (음의 무한대로 발산)
  - 관찰 2) 음수 사이클을 지나지 않는다면 모든 최단 경로는 최대 V-1개의 간선을 지남

- Dijkstra's Algorithm처럼 relaxation을 통해 최단 거리를 구함
  - 시작 정점 S의 거리는 0, 다른 모든 정점까지의 거리는 INF로 초기화
  - 모든 간선 (s, e, w)에 대해 D[e] = min(D[e], D[s] + w)를 수행하는 것을 V-1번 반복
    - i번째 iteration이 끝나면 간선 i개를 거쳐서 가는 최단 거리를 정확하게 구할 수 있음
    - 귀납법으로 증명 가능

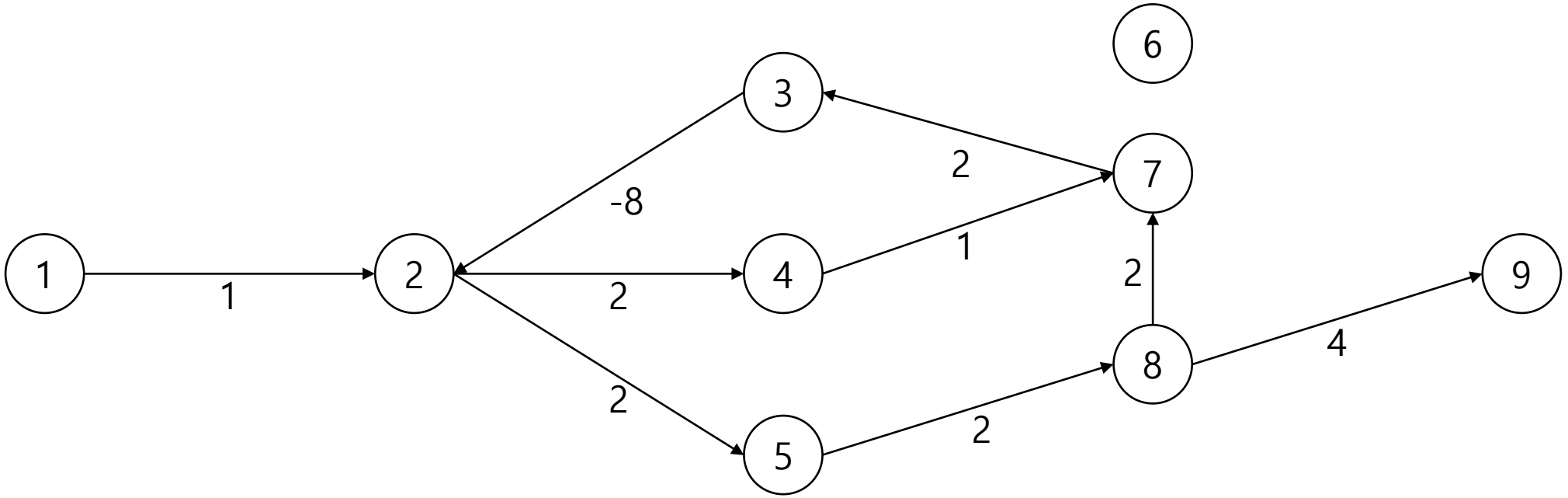
벨만 포드 알고리즘

Bellman-Ford Algorithm

- 음수 사이클이 존재하는지 판별하는 것도 가능

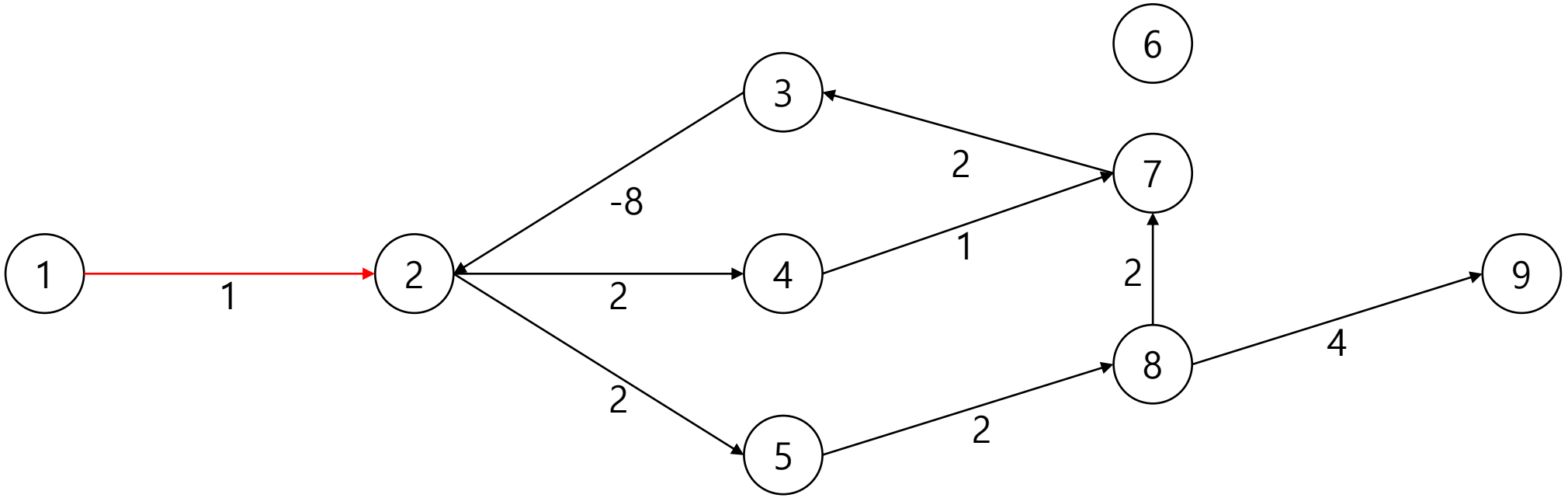
- 음수 사이클이 없다면 V-1번의 iteration으로 항상 최단 거리를 구할 수 있음
- 만약 V번째 iteration에서 relaxation이 발생하면? 음수 사이클 존재

벨만 포드 알고리즘



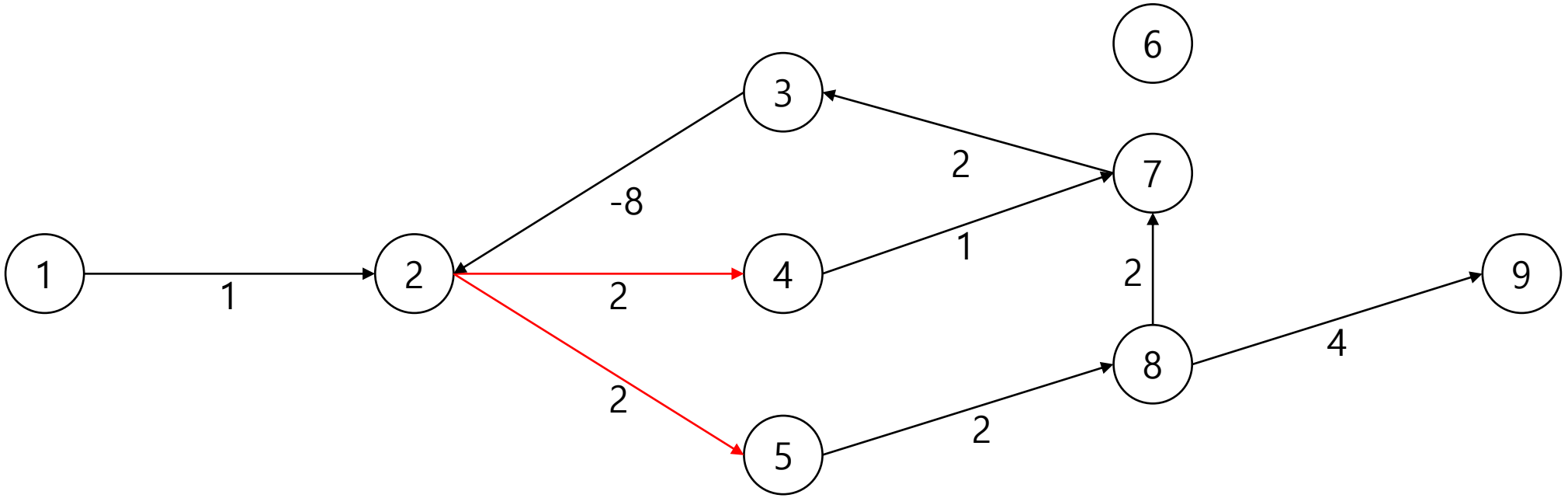
1	2	3	4	5	6	7	8	9
0	inf	inf	inf	inf	inf	inf	inf	inf

벨만 포드 알고리즘 - 1



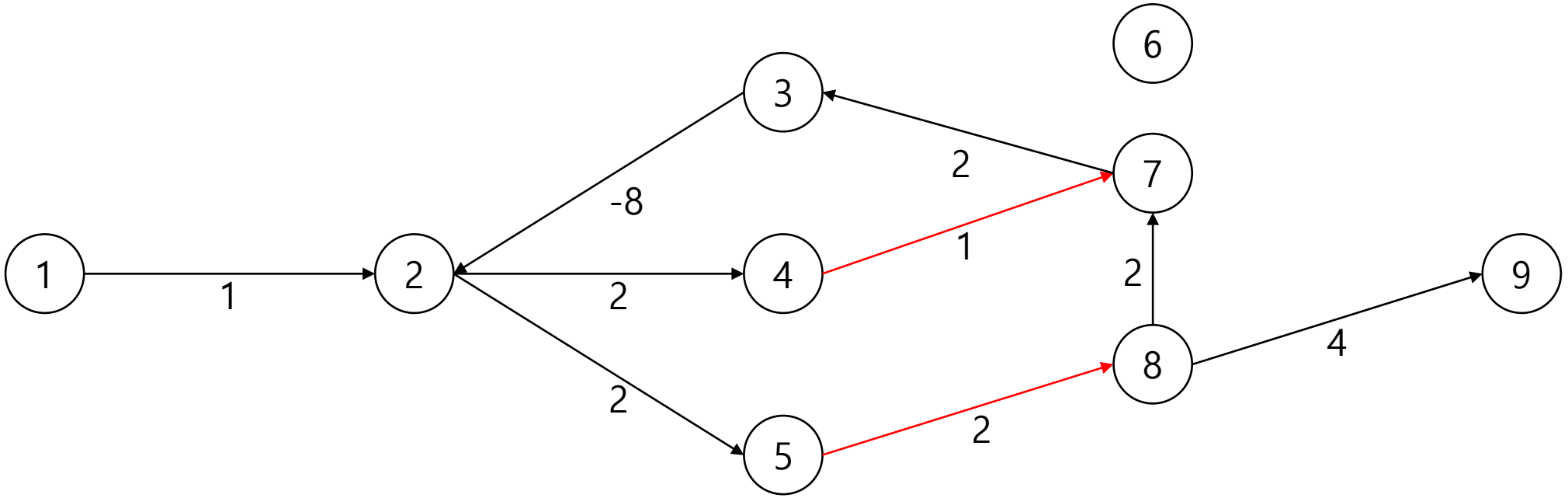
1	2	3	4	5	6	7	8	9
0	1	inf	inf	inf	inf	inf	inf	inf

벨만 포드 알고리즘 - 2



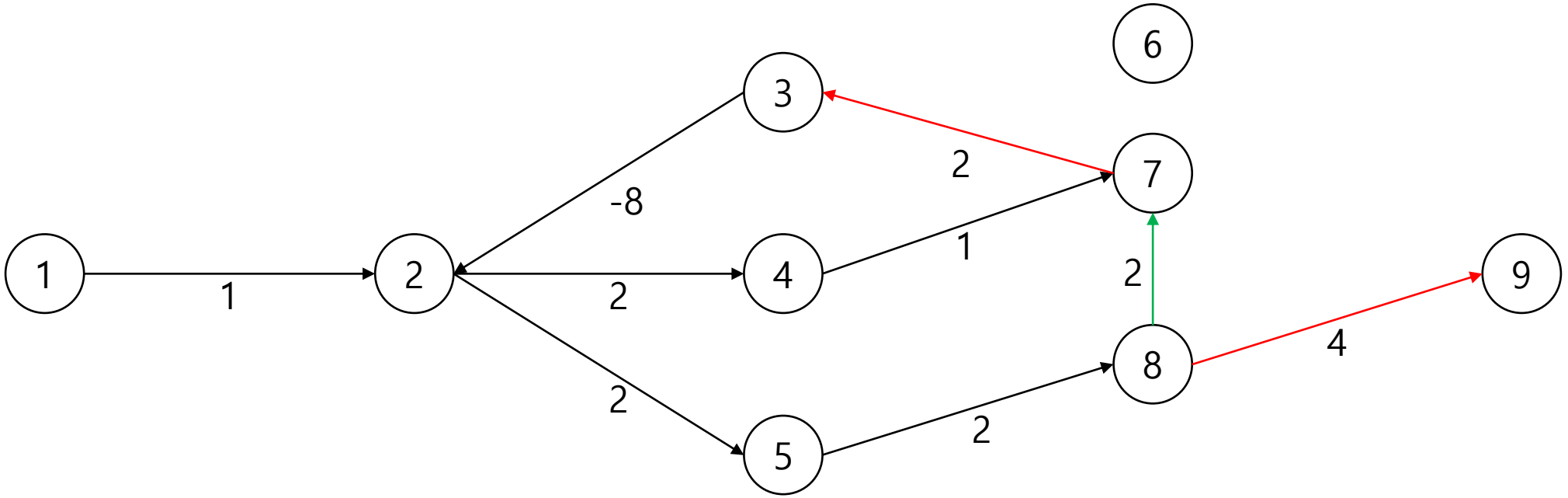
1	2	3	4	5	6	7	8	9
0	1	inf	3	3	inf	inf	inf	inf

벨만 포드 알고리즘 - 3



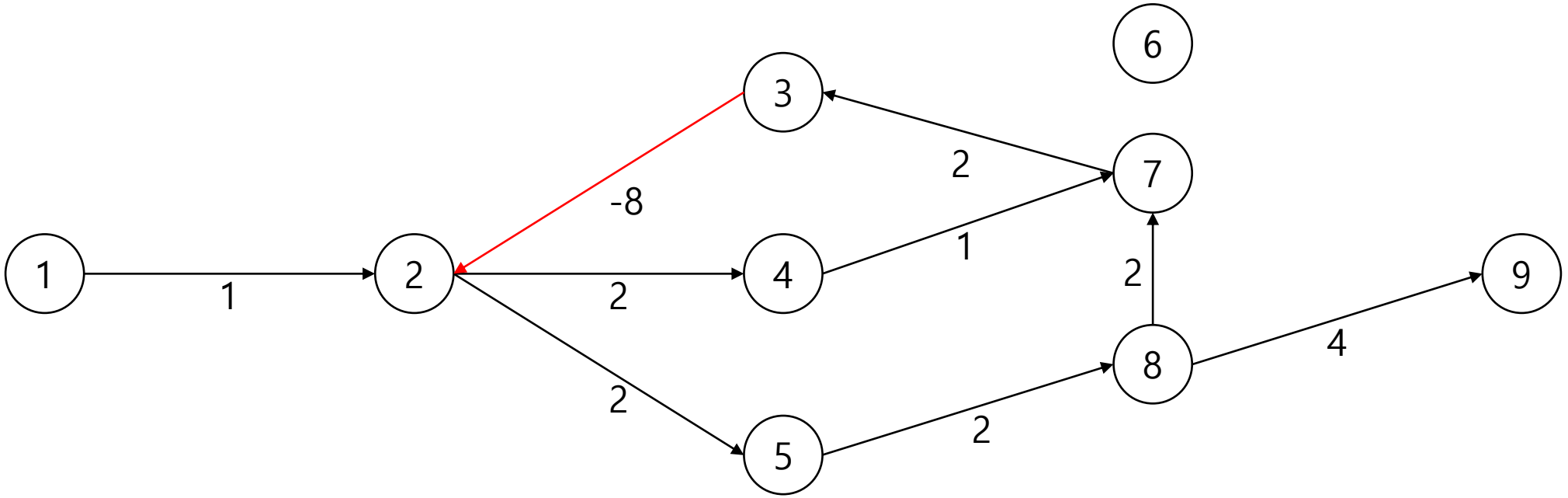
1	2	3	4	5	6	7	8	9
0	1	inf	3	3	inf	4	5	inf

벨만 포드 알고리즘 - 4



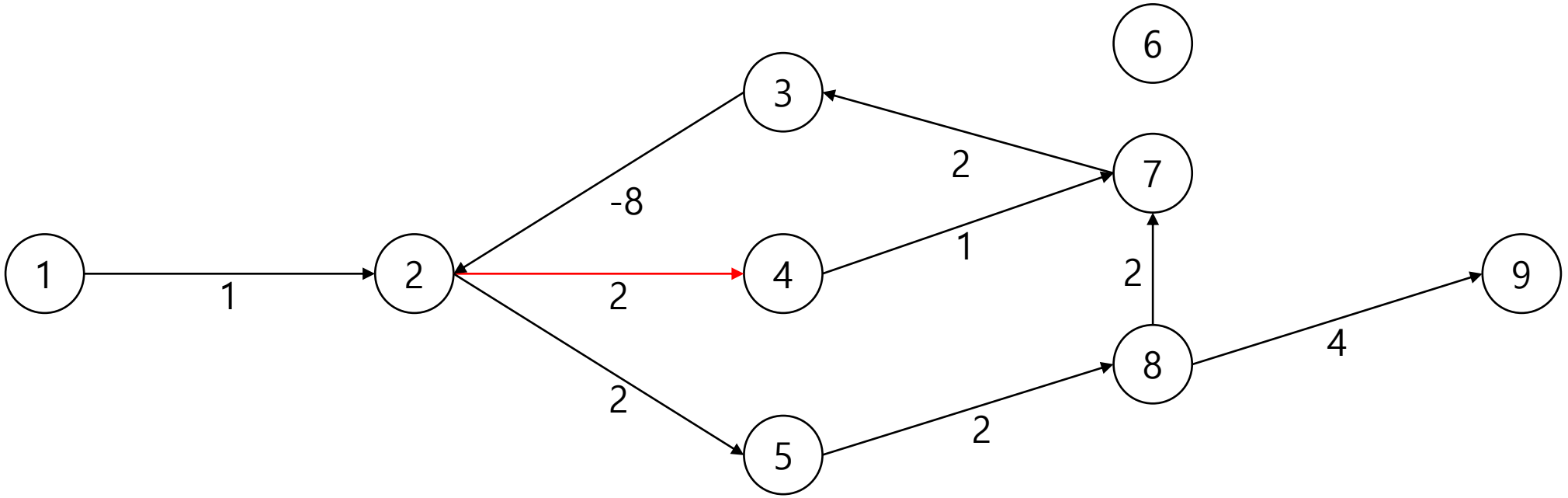
1	2	3	4	5	6	7	8	9
0	1	6	3	3	inf	4	5	9

벨만 포드 알고리즘 - 5



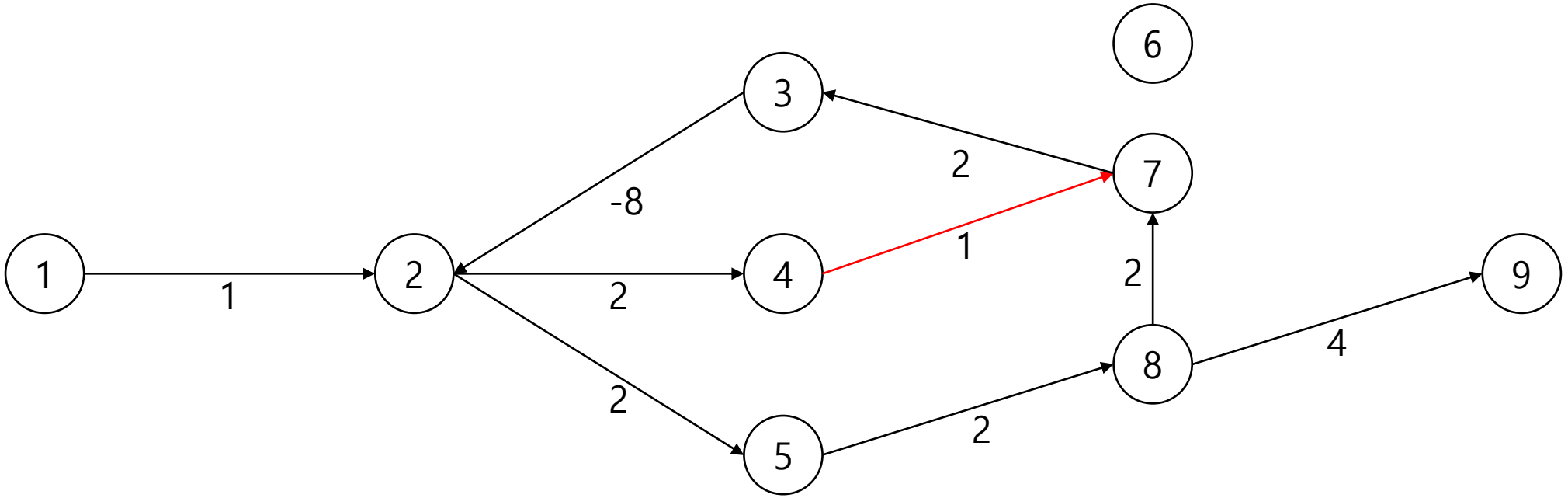
1	2	3	4	5	6	7	8	9
0	-2	6	3	3	inf	4	5	9

벨만 포드 알고리즘 - 6



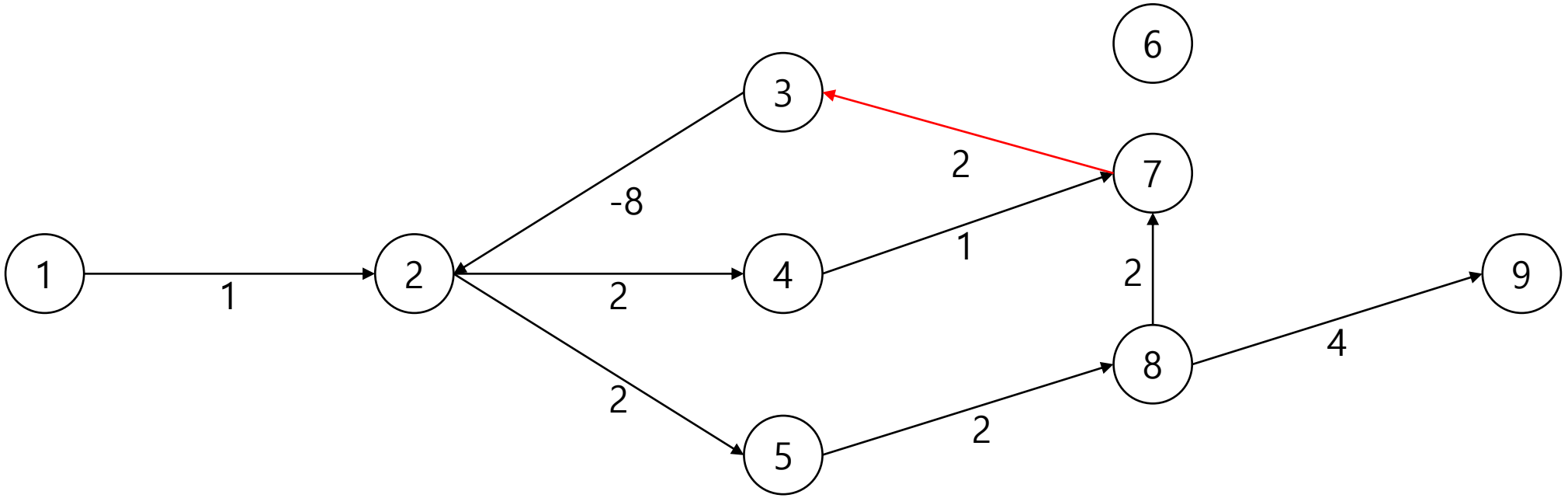
1	2	3	4	5	6	7	8	9
0	-2	6	0	3	inf	4	5	9

벨만 포드 알고리즘 - 7



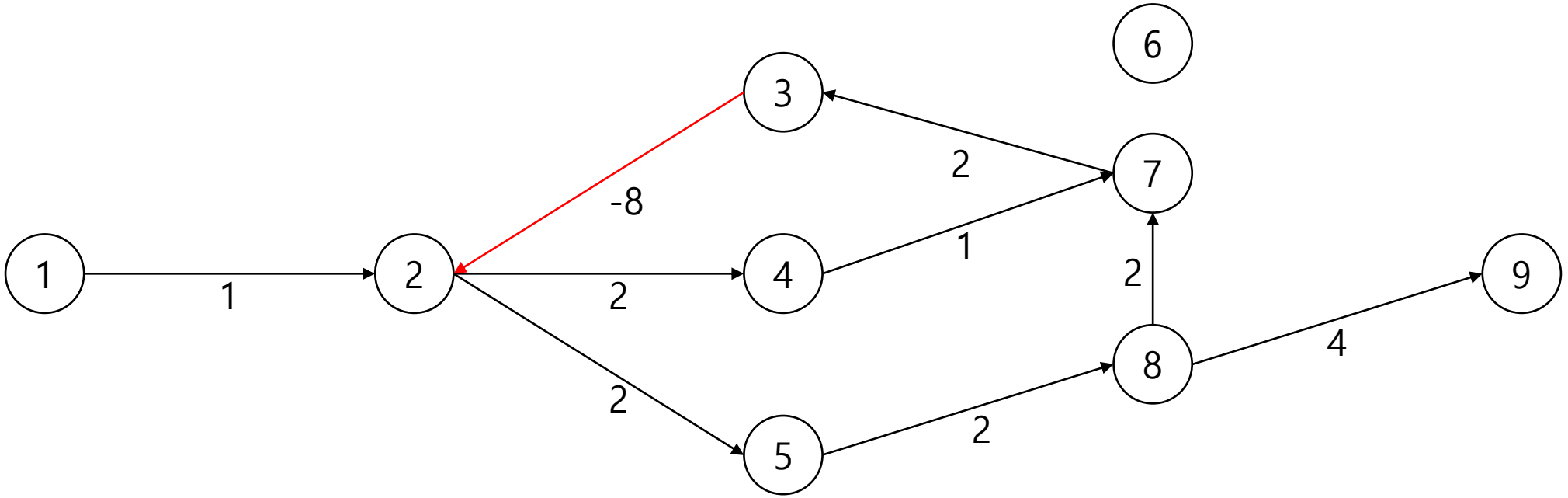
1	2	3	4	5	6	7	8	9
0	-2	6	0	3	inf	1	5	9

벨만 포드 알고리즘 - 8



1	2	3	4	5	6	7	8	9
0	-2	3	0	3	inf	1	5	9

벨만 포드 알고리즘 - 9



1	2	3	4	5	6	7	8	9
0	-5	3	0	3	inf	1	5	9

벨만 포드 알고리즘

```cpp
struct Edge{
    int s, e, w;
    Edge() = default;
    Edge(int s, int e, int w) : s(s), e(e), w(w) {}
};

int N, M;
vector<Edge> E;
ll D[555], INF;

bool BellmanFord(int S){
    INF = N * 10000LL;
    fill(D, D+555, INF);
    D[S] = 0;
    for(int iter=1; iter<=N; iter++){
        bool isChanged = false;
        for(const auto &[s,e,w] : E){
            if(D[s] == INF) continue; // 간선의 출발점이 INF라면 넘어감
            if(D[e] > D[s] + w){      // relaxation
                D[e] = D[s] + w;
                isChanged = true;
            }
        }
        // N번째 iteration에서 relaxation이 발생하면 음수 사이클 존재
        if(isChanged && iter == N) return false;
    }
    return true;
}
```

벨만 포드 알고리즘

시간 복잡도

- V번의 iteration * E개의 간선 탐색 : O(VE)

주의 사항

- 거리 배열은 V * (간선 가중치 최댓값)으로 초기화 : 모든 경로는 최대 V-1개의 간선으로 구성
- 실행 도중에 값이 약 V * E * (간선 가중치 최솟값)까지 내려갈 수 있음 : 오버 플로우 주의
  - 간선이 (1,2,-x), (2,1,-x), (1,2,-x), (2,1,-x), ... , 순으로 주어진다고 하자.
  - 한 번의 iteration에서
    - D[2] = -x, D[1] = -2x, D[2] = -3x, ...
    - iteration 종료되면 D[1] = D[2] ~= -xE/2
  - V번 반복하므로 D[1] = D[2] ~= -xVE/2

벨만 포드 알고리즘

응용

- $X_b - X_a \le w(a, b)$ 꼴의 부등식이 여러 개 주어졌을 때 X_i의 값을 구하는 문제
  - $X_b \le X_a + w(a, b)$
  - 가상의 정점 S에서 1, 2, ... , N으로 가는 가중치 0 간선 만들고
  - a에서 b로 가는 가중치 w(a, b) 간선 만들면
  - S에서 시작하는 SSSP를 이용해 부등식을 만족하는 X_i를 구할 수 있음

- $X_b - X_a = w(a, b)$ 꼴의 부등식도 처리 가능
  - $X_b - X_a = w(a, b)$ 는 $X_b - X_a \le w(a, b)$ 와 $X_b - X_a \ge w(a, b)$ 의 교집합이라고 생각할 수 있음
  - $X_b - X_a \le w(a, b)$ 는 $X_b \le X_a + w(a, b)$ 이므로 a에서 b로 가는 가중치 w(a, b) 간선
  - $X_b - X_a \ge w(a, b)$ 는 $X_b - w(a, b) \ge X_a$ 이므로 b에서 a로 가는 가중치 $-w(a, b)$ 간선

질문?

벨만 포드 알고리즘

BOJ 1219 오민식의 고민

- 정점과 간선에 가중치가 있는 유향 그래프가 주어짐
- 간선을 따라 이동할 때마다 가중치 만큼 돈을 지불해야 하고, 정점을 방문할 때마다 가중치 만큼 돈을 얻음
- S번 정점에서 T번 정점으로 갈 때 얻을 수 있는 최대 이익을 구하는 문제
- 같은 정점을 여러 번 방문할 수 있고, 방문할 때마다 돈을 얻음
- 도착이 불가능하면 “gg” 출력, 돈을 무한히 많이 얻을 수 있다면 “Gee” 출력

- 돈을 A[x] 만큼 얻는 것을 -A[x] 만큼 잃는 것이라고 생각하자.
- S에서 출발해서 음수 사이클을 지난 다음 T로 갈 수 있으면 “Gee”를 출력하는 문제
  - 정점 가중치는 x*2번 정점에서 x*2+1번 정점으로 가는 가중치 -A[x] 간선으로 생각할 수 있음

- 음수 사이클을 지난 다음 도착점으로 이동
  - 음수 사이클을 판별할 때 S에서 도달 가능한 사이클인지, T까지 도달할 수 있는 사이클인지 확인해야 함
  - S에서 도달할 수 없거나, T로 도달할 수 없는 정점을 모두 제거한 다음 벨만 포드 알고리즘 수행

- 실수할 여지가 많은 문제

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
constexpr ll INF = 0x3f3f3f3f3f3f3f3f;

ll N, M, S, T, D[111], U[111];
vector<pair<ll,ll>> G[111], R[111];
void AddEdge(int s, int e, int w){
    G[s].emplace_back(e, w);
    R[e].emplace_back(s, w);
}

void DFS(vector<pair<ll,ll>> *gph, int v, int flag){
    U[v] |= flag;
    for(auto [i,w] : gph[v]) if(~U[i] & flag) DFS(gph, i, flag);
}

bool BellmanFord(){
    memset(D, 0x3f, sizeof D); D[S*2] = 0;
    for(int iter=1; iter<=N+N; iter++){
        bool flag = false;
        for(int i=0; i<N+N; i++){
            if(U[i] != 3 || D[i] == INF) continue;
            for(auto [j,w] : G[i]){
                if(U[j] == 3 && D[j] > D[i] + w){
                    D[j] = D[i] + w; flag = true;
                }
            }
        }
        if(flag && iter == N+N) return false;
    }
    return true;
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> S >> T >> M;
    for(int i=1,s,e,w; i<=M; i++) cin >> s >> e >> w, AddEdge(s*2+1, e*2, w);
    for(int i=0,t; i<N; i++) cin >> t, AddEdge(i*2, i*2+1, -t);
    DFS(G, S*2, 1); DFS(R, T*2+1, 2);
    if(U[T*2+1] != 3) cout << "gg";
    else if(!BellmanFord()) cout << "Gee";
    else cout << -D[T*2+1];
}
```

벨만 포드 알고리즘

BOJ 7577 탐사

- 길이가 N인 배열이 있고, 각 칸의 값은 0 또는 1임
- 각 칸의 정확한 값은 모르지만, x부터 y까지의 합이 r이라는 정보는 알고 있음
- 가능한 배열의 값 중 하나를 구하는 문제

- 누적 합 배열을 생각하면 S[y] - S[x-1] = r 이라는 정보가 주어지는 것
  - x-1 에서 y로 가는 가중치 r 간선
  - y에서 x-1로 가는 가중치 -r 간선
- 몇 가지 조건이 더 필요함
  - 모든 칸의 값은 0 이상: N+1에서 i로 가는 가중치 0 간선
  - 모든 칸의 값은 0 또는 1: $S[i-1] \le S[i] \le S[i-1] + 1$
    - i-1 에서 i로 가는 가중치 1 간선
    - i에서 i-1 로 가는 가중치 0 간선
- N+1에서 시작하는 벨만 포드 알고리즘을 이용해 해결 가능
  - 배치가 불가능하면 음수 가중치

벨만 포드 알고리즘

BOJ 7577 탐사

```cpp
int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M;
    for(int i=1; i<=M; i++){
        int s, e, w; cin >> s >> e >> w;
        E.emplace_back(s-1, e, w);
        E.emplace_back(e, s-1, -w);
    }
    for(int i=0; i<=N; i++) E.emplace_back(N+1, i, 0);
    for(int i=1; i<=N; i++){
        E.emplace_back(i-1, i, 1);
        E.emplace_back(i, i-1, 0);
    }
    if(!BellmanFord(N+1)){ cout << "NONE"; return 0; }
    for(int i=1; i<=N; i++){
        if(D[i-1] + 1 == D[i]) cout << "#";
        else cout << "-";
    }
}
```

질문?