Shortest path algorithms

20120302 김우진
20130870 이건희

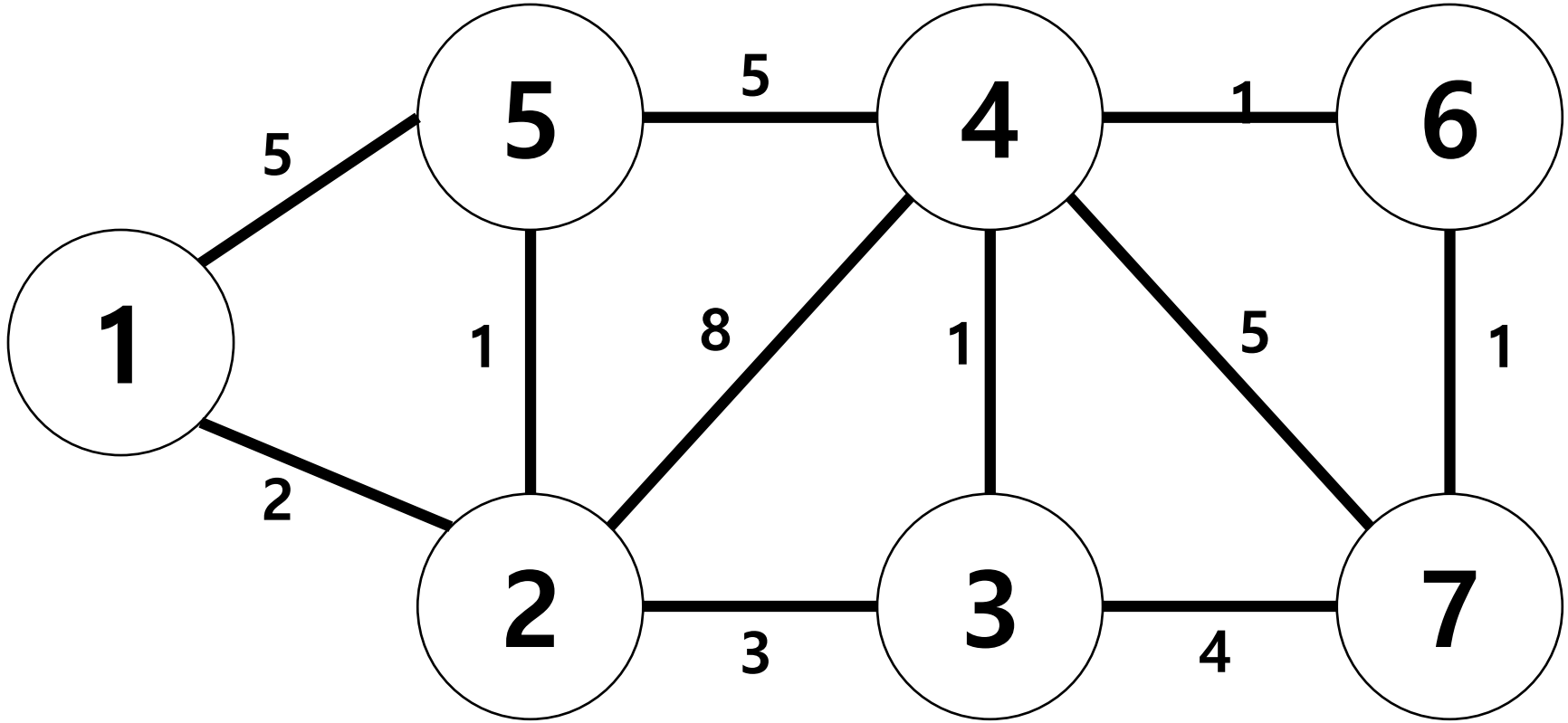
조사한 3가지 알고리즘

1. Dijkstra's Algorithm
2. Bellman-Ford's Algorithm
3. Floyd-Warshall's Algorithm

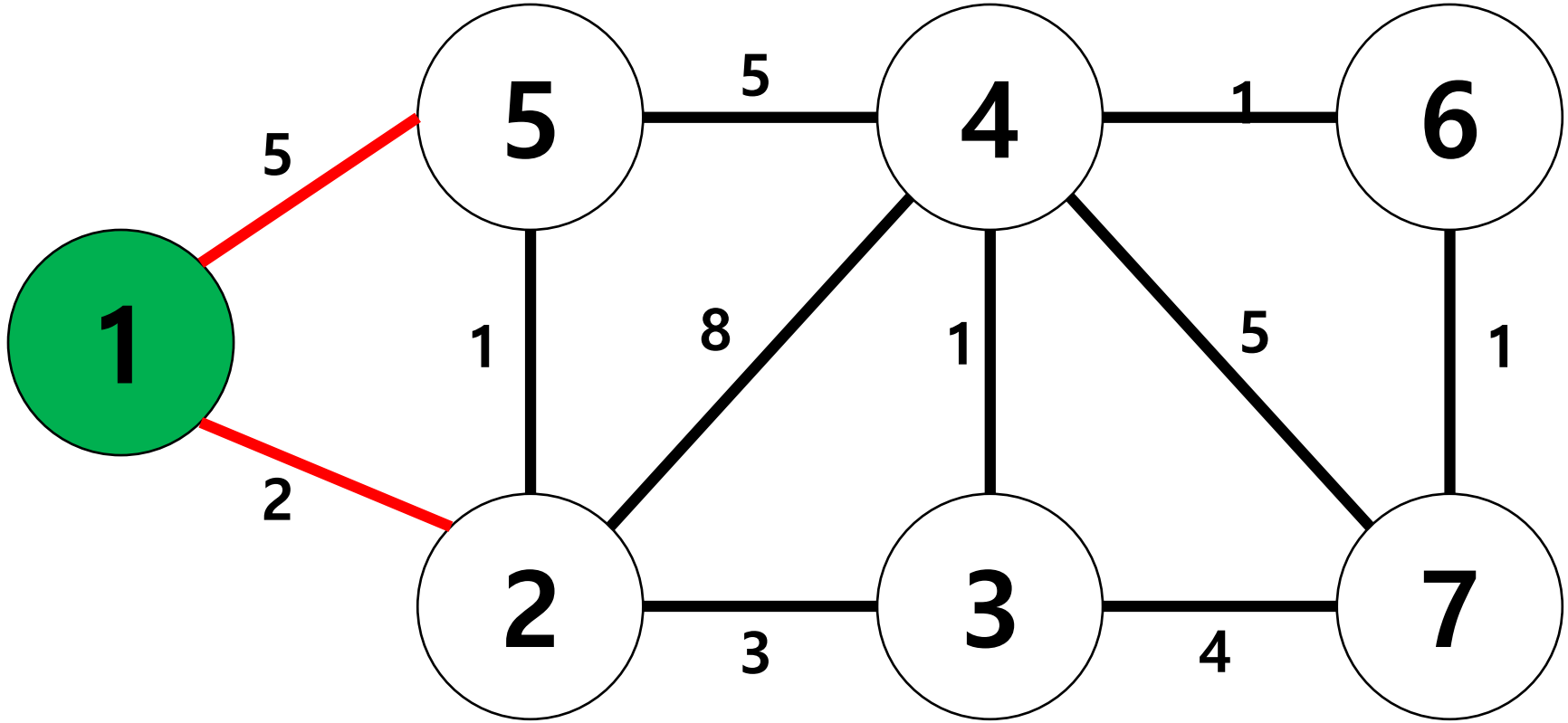
Dijkstra's Algorithm

- **개요**
  - Edsger W. Dijkstra가 고안한 알고리즘
  - 한 정점에서 다른 모든 정점으로의 최단 경로를 구하는 알고리즘

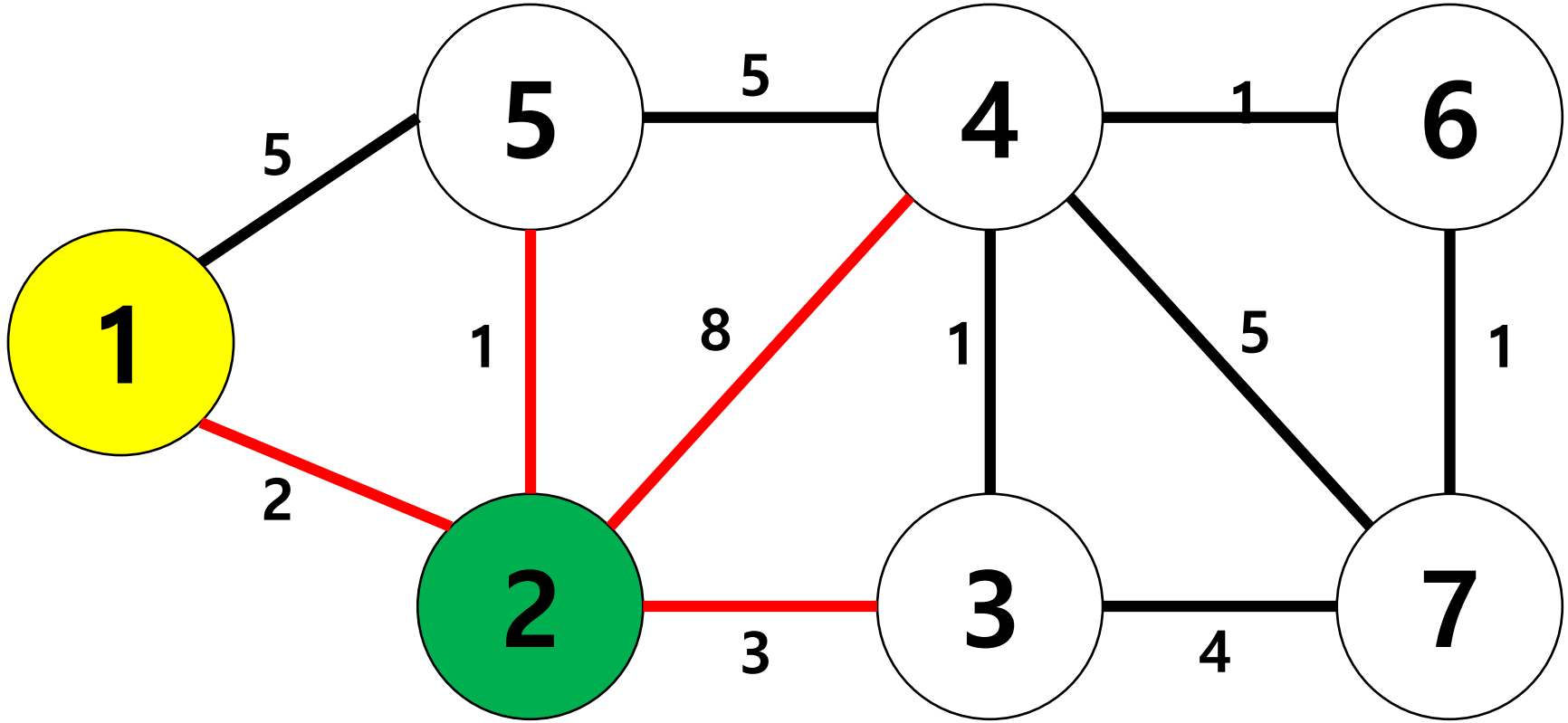
- **조건**
  - 간선들은 모두 양의 간선을 가져야한다. (음의 간선을 가져선 안된다.)
  - 첫 정점을 기준으로 연결된 정점들을 추가하며 최단 거리를 갱신
  - 정점 연결 전까지는 시작점을 제외하고 모두 무한대 값



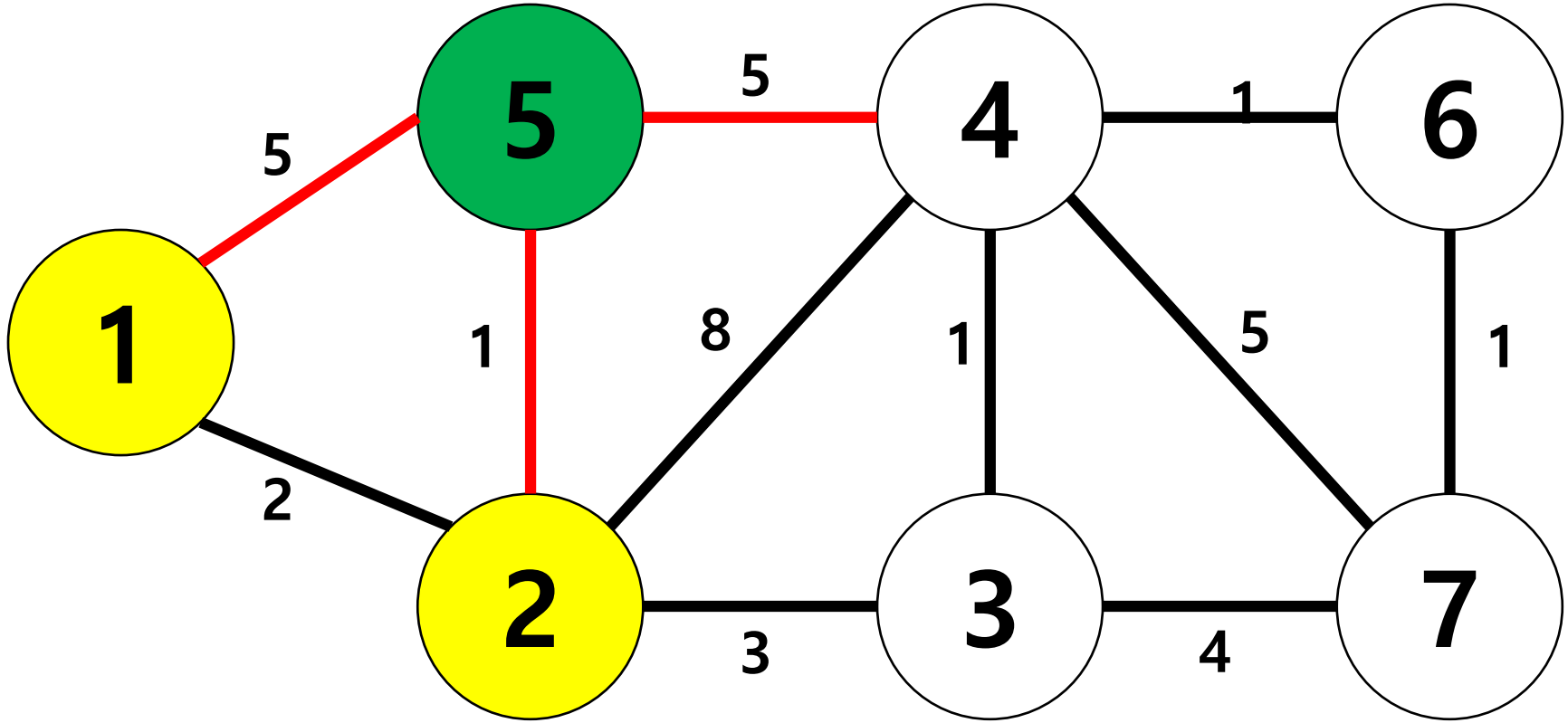
i	1	2	3	4	5	6	7
dist	0	MAX	MAX	MAX	MAX	MAX	MAX
visit							



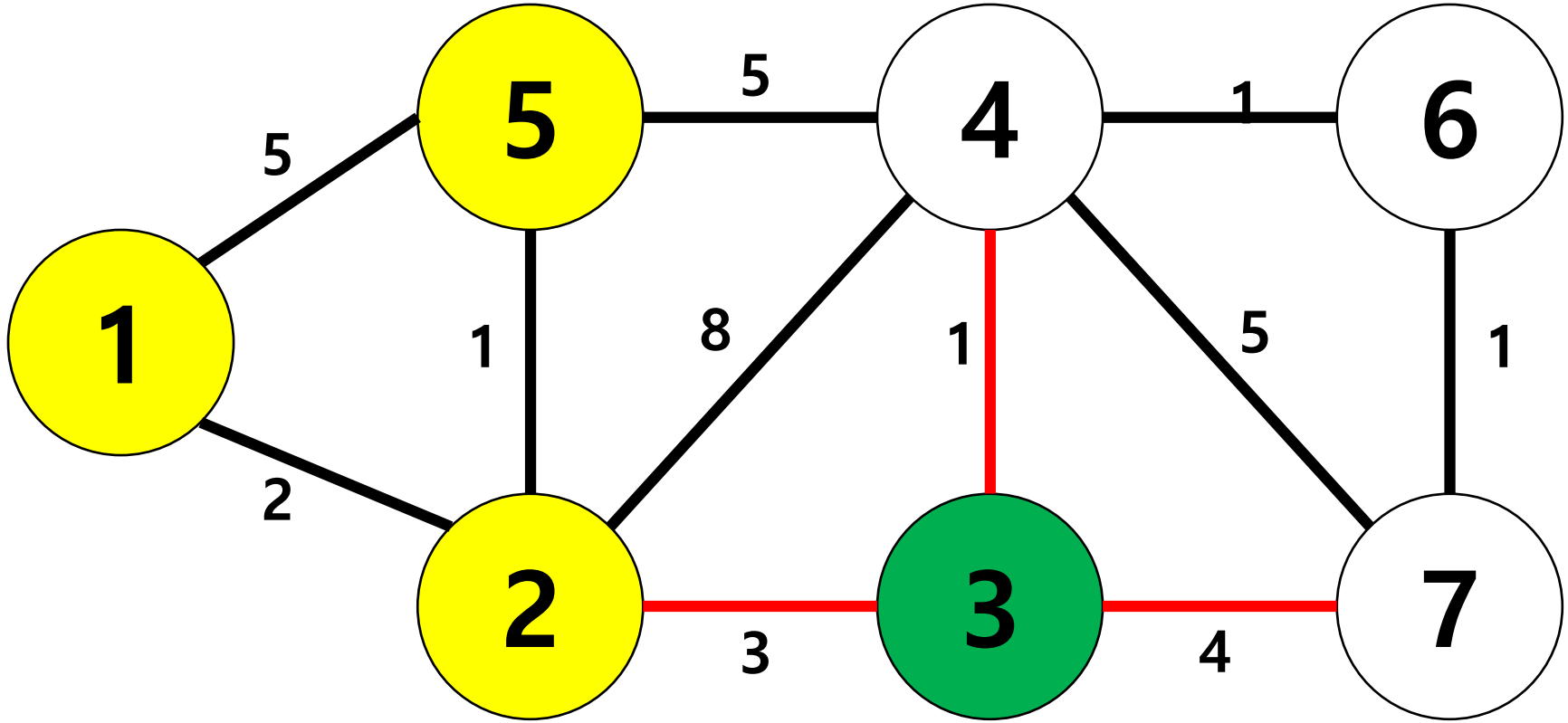
i	1	2	3	4	5	6	7
dist	0	2	MAX	MAX	5	MAX	MAX
visit	1						



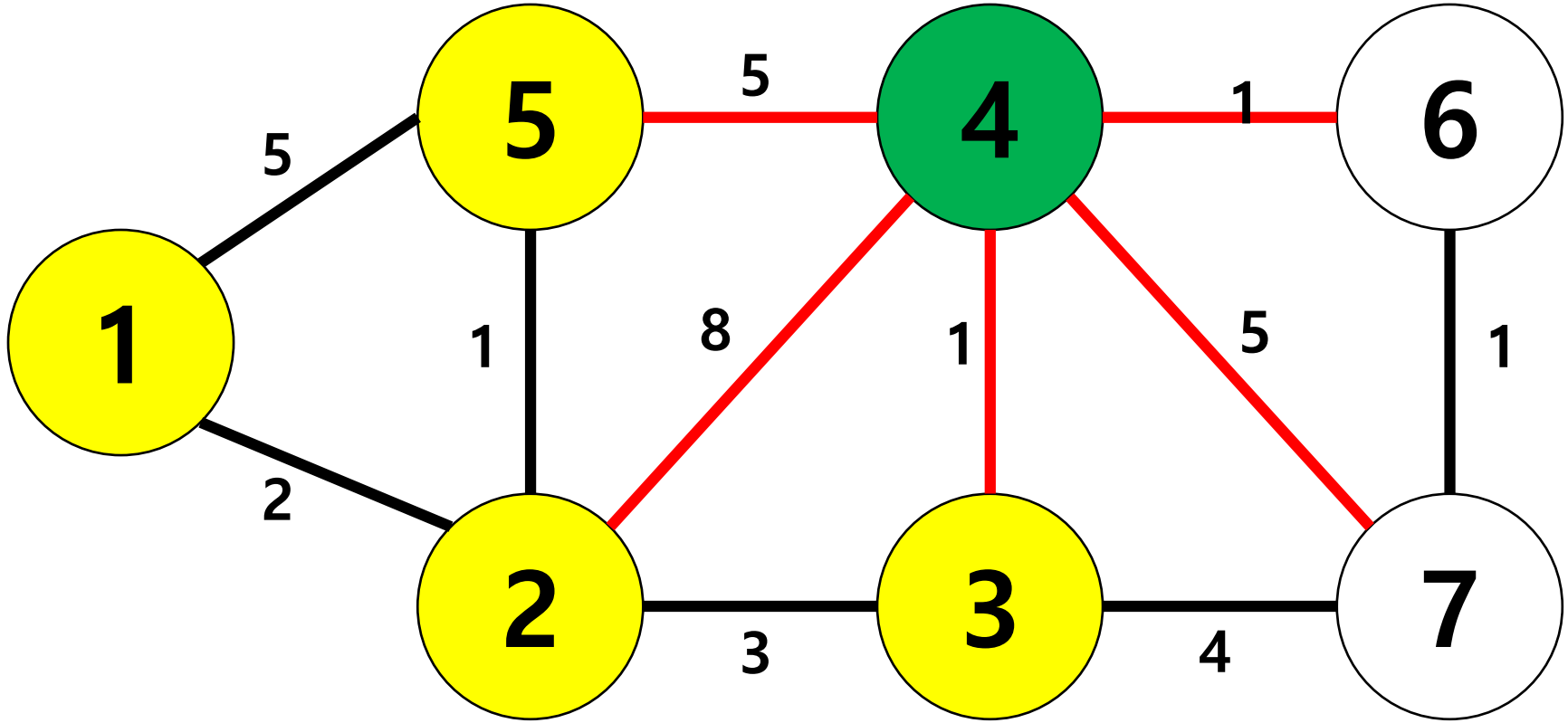
i	1	2	3	4	5	6	7
dist	0	2	**5**	**10**	**3**	MAX	MAX
visit	1	**1**					



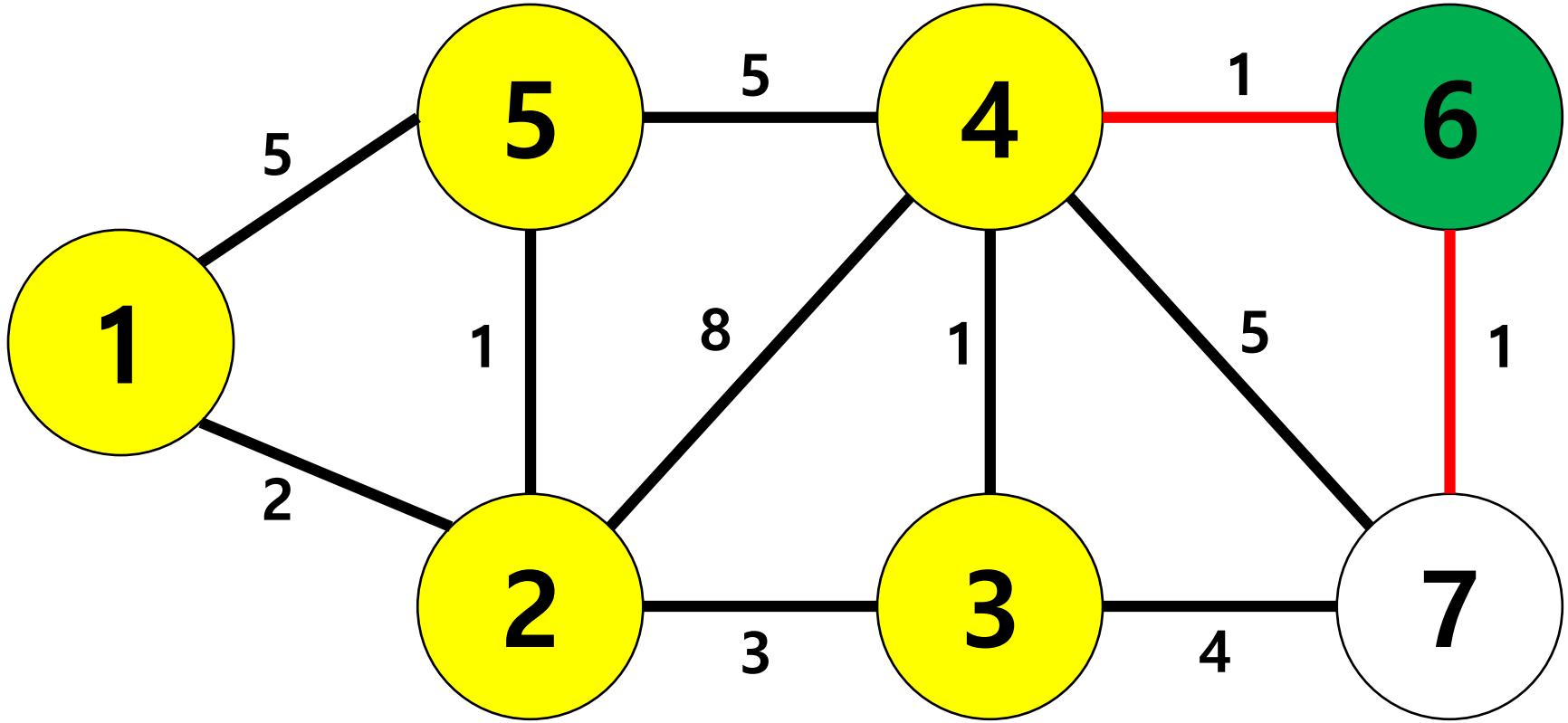
i	1	2	3	4	5	6	7
dist	0	2	5	**8**	3	MAX	MAX
visit	1	**1**	**1**		**1**		



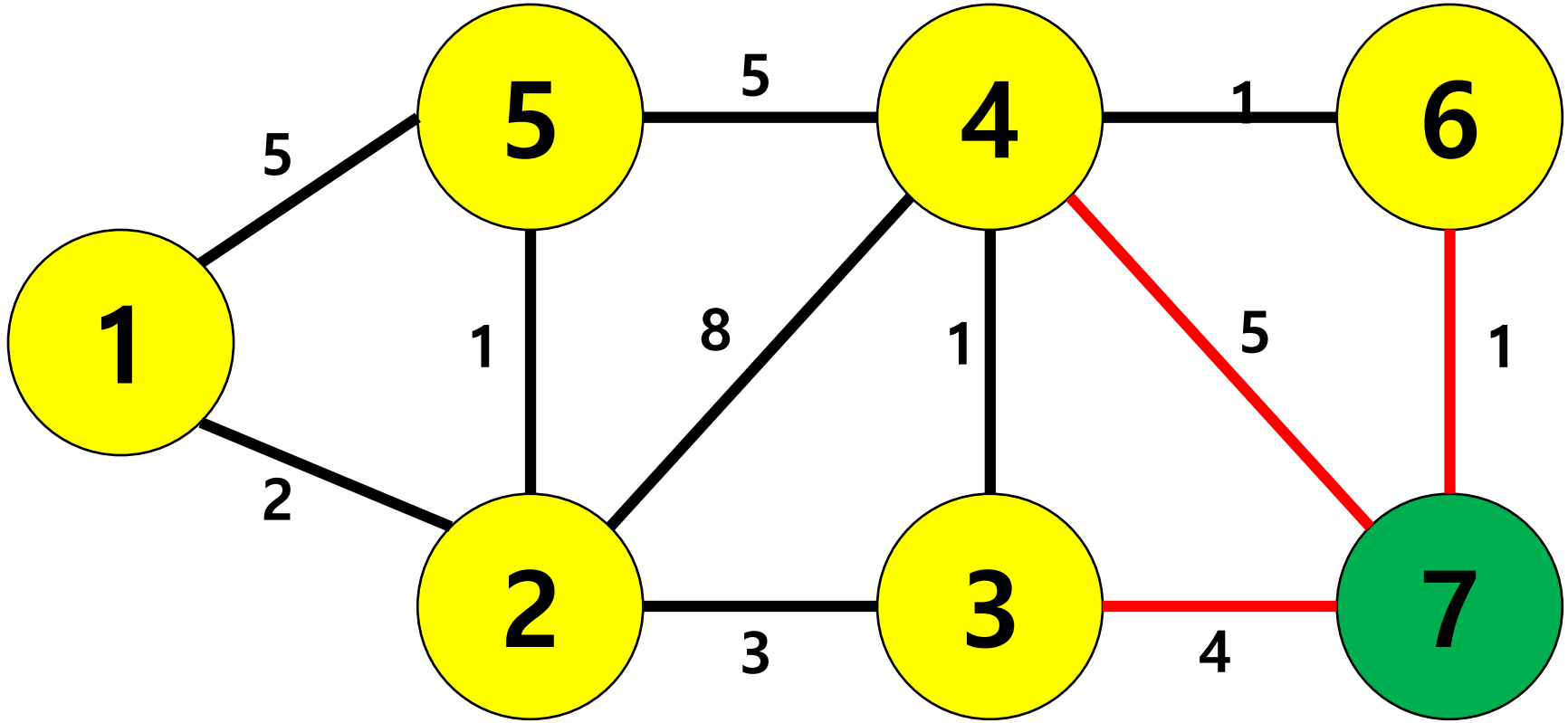
i	1	2	3	4	5	6	7
dist	0	2	5	**6**	3	MAX	**9**
visit	1	1	1		1		



i	1	2	3	4	5	6	7
dist	0	2	5	6	3	7	9
visit	1	1	1	1	1		



i	1	2	3	4	5	6	7
dist	0	2	5	6	3	7	8
visit	1	1	1	1	1	1	



i	1	2	3	4	5	6	7
dist	0	2	5	6	5	7	8
visit	1	1	1	1	1	1	1

Dijkstra's Algorithm

- **시간 복잡도**
  - 배열 : O(V^2)
  - 이진 힙 : O(E*logV)
  - 피보나치 힙 : O(E+V*logV)

- **실질적 사용**
  - 경로 찾기
  - 네비게이션
  - 미로탐색 알고리즘

Dijkstra’s Algorithm 관련 논문

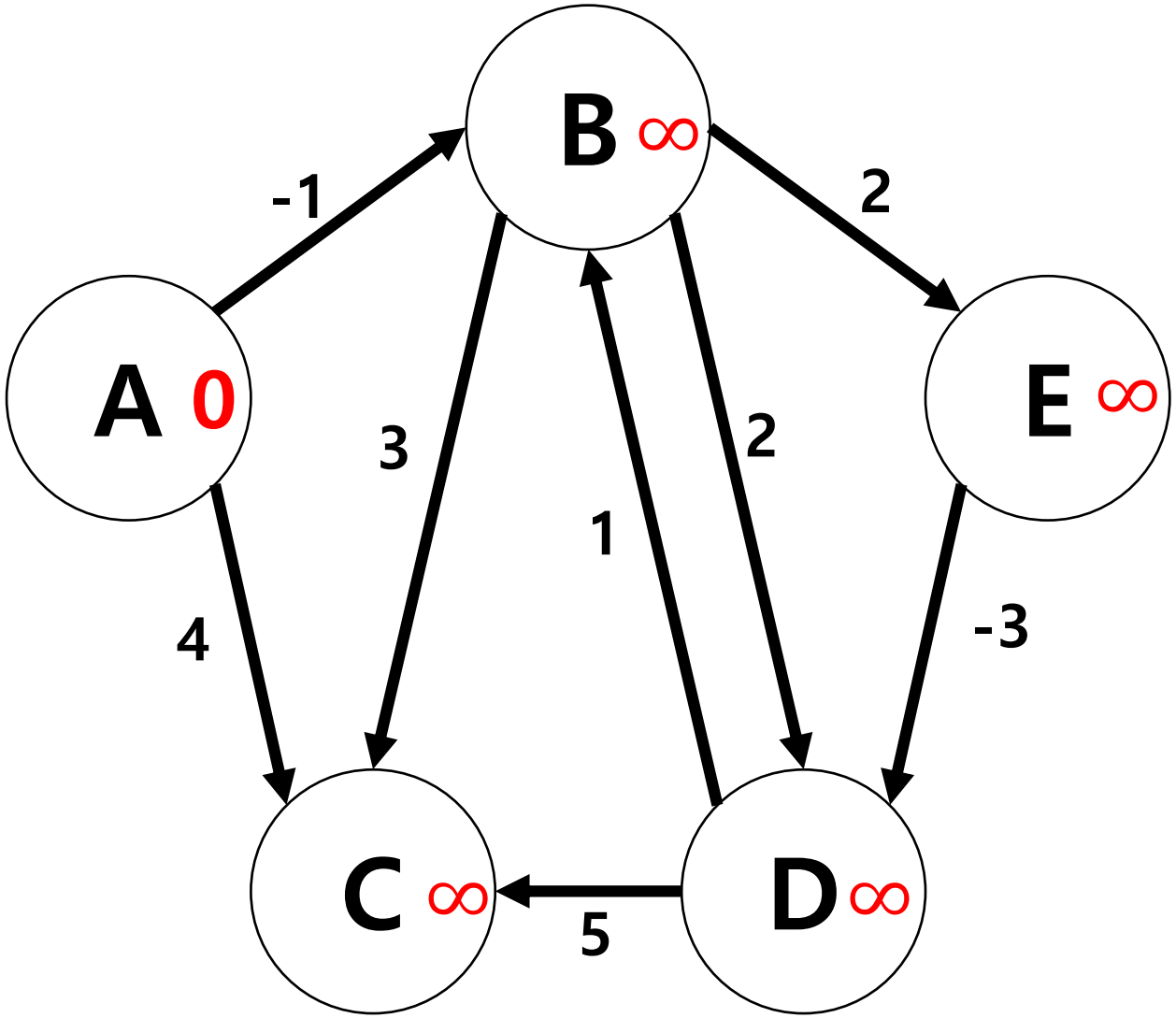
- 윤창민 외 3명. (2008). **연산시간 최적화를 위한 P-Dijkstra 알고리즘에 관한 연구(A Study on the P-Dijkstra Algorithms for Optimization of Computation Time)**. 한국통신학회 2017년도 추계종합학술발표회, 177~178

- 기존의 Dijkstra 알고리즘은 모든 경로를 탐색함 -> 속도 저하
- 모든 노드를 구획(partition)화하고 Dijkstra 알고리즘을 적용

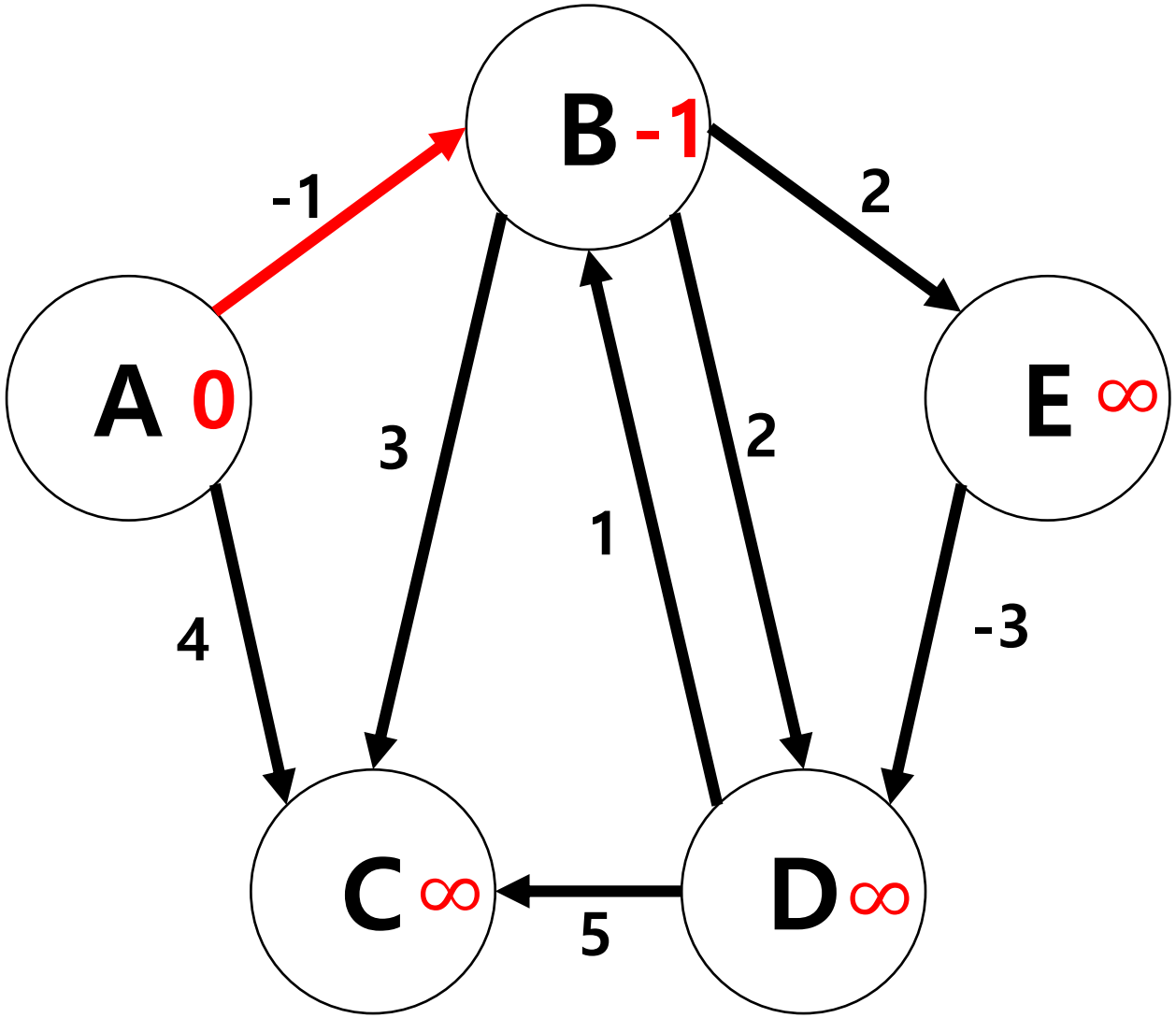
Bellman-Ford Algorithm

- **개요**
  - Dijkstra 알고리즘에 비해 느린 알고리즘이다.
  - 음의 가중치까지 더할 수 있다.

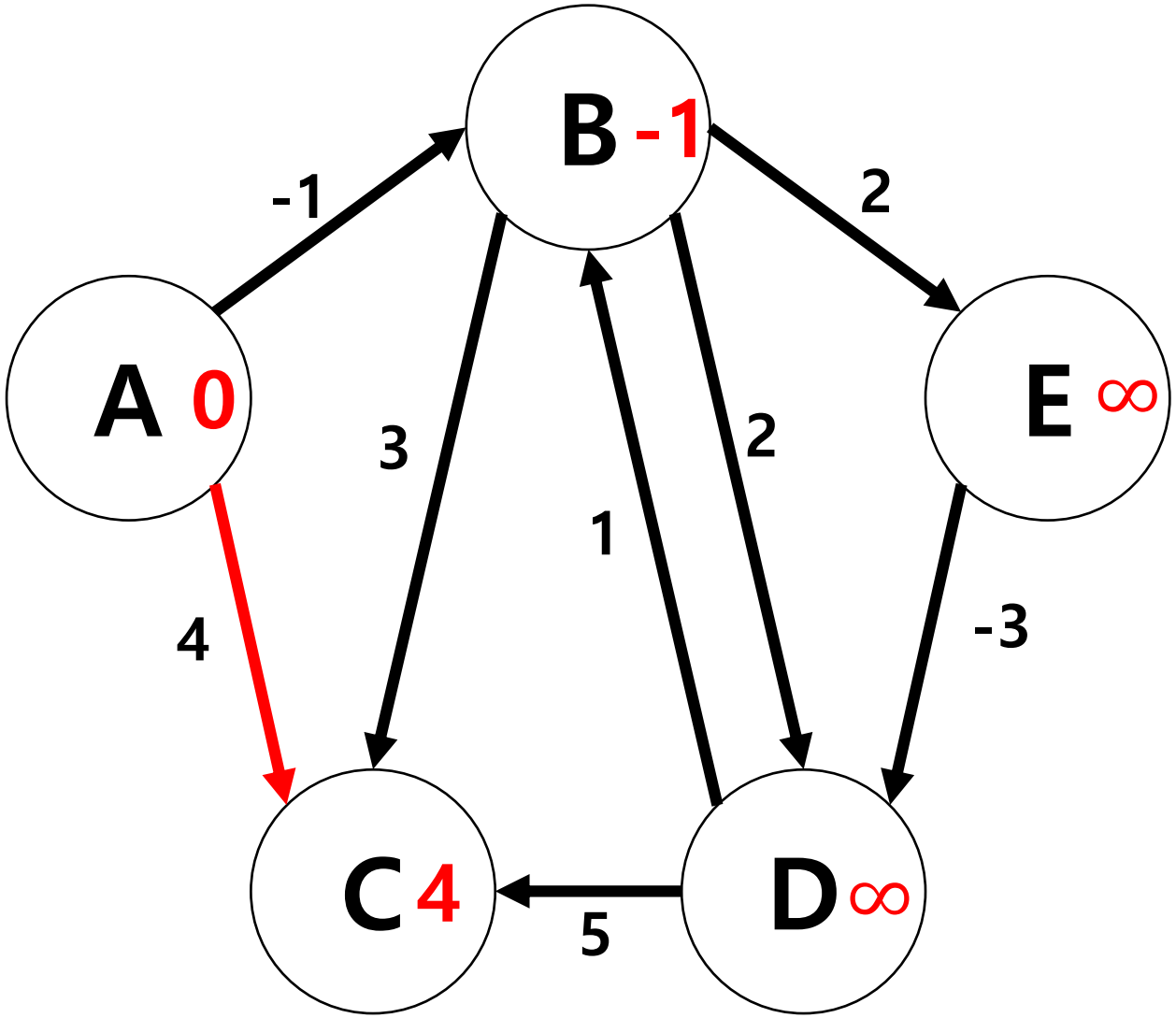
- **조건**
  - 최단 경로는 사이클을 포함하면 안되므로, 최대 **|V|-1**개의 간선만 사용
  - 최단 거리가 업데이트되는 노드가 없어질 때까지 반복해서 구해줌.



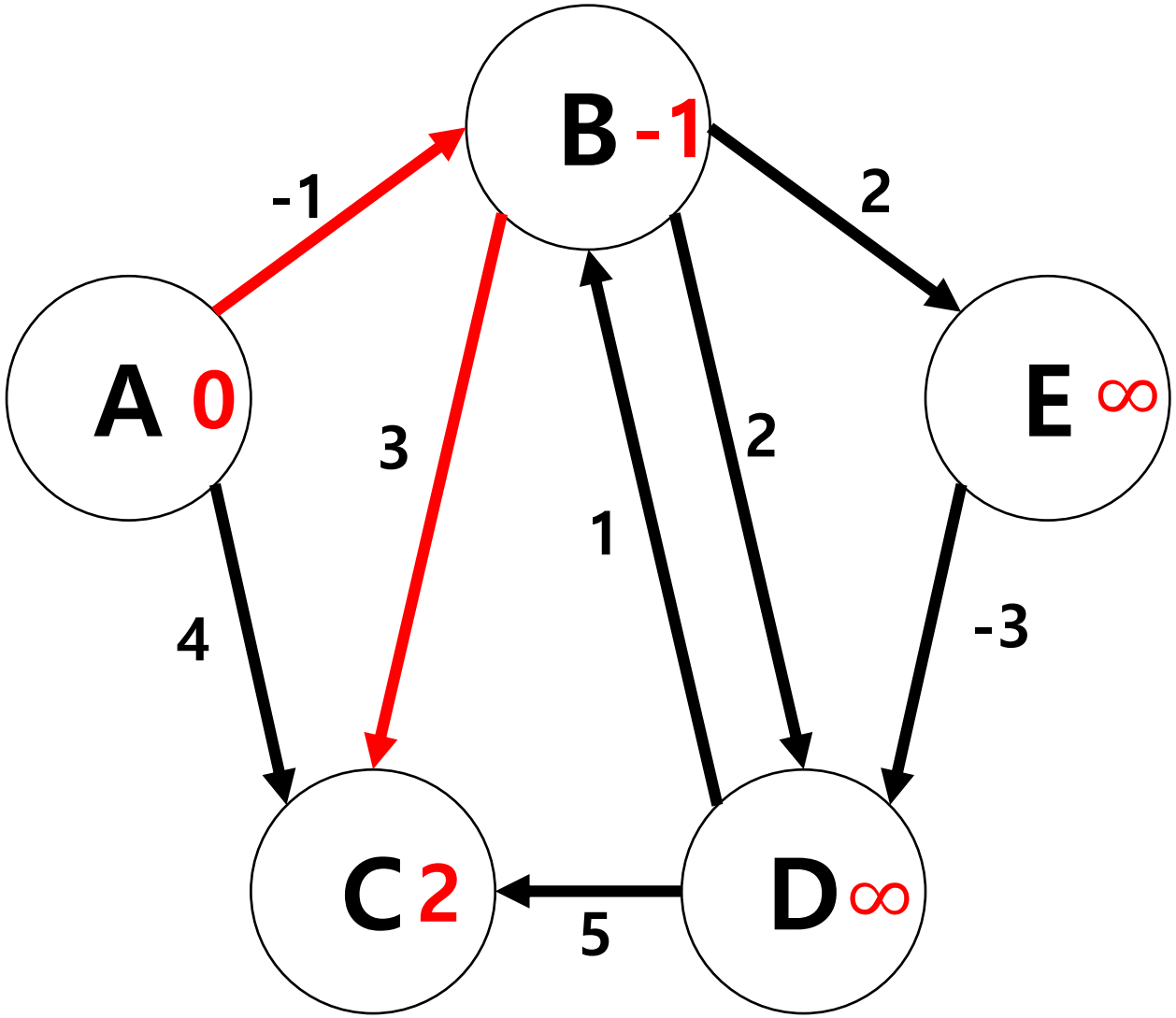
A	B	C	D	E
0	∞	∞	∞	∞



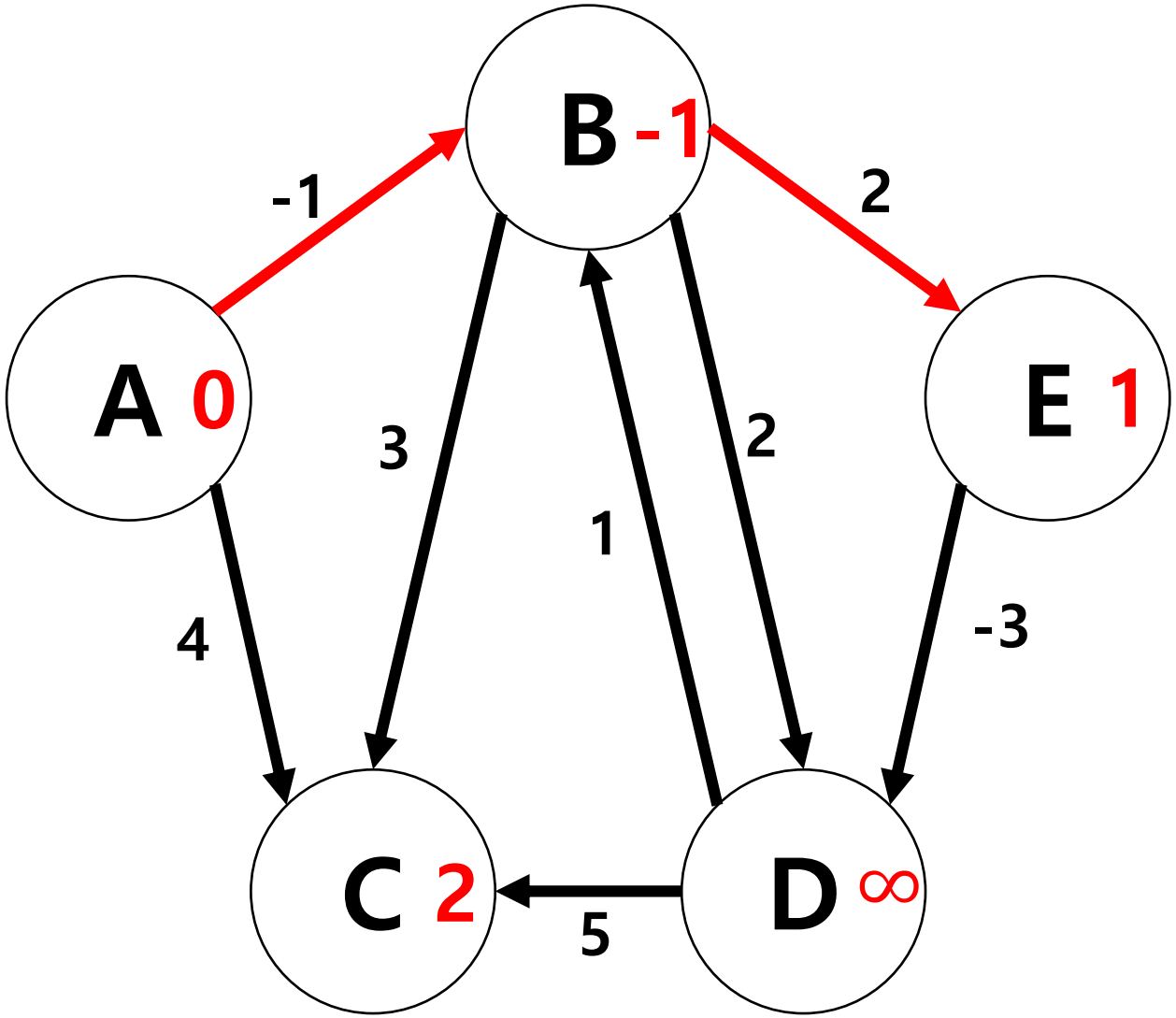
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞



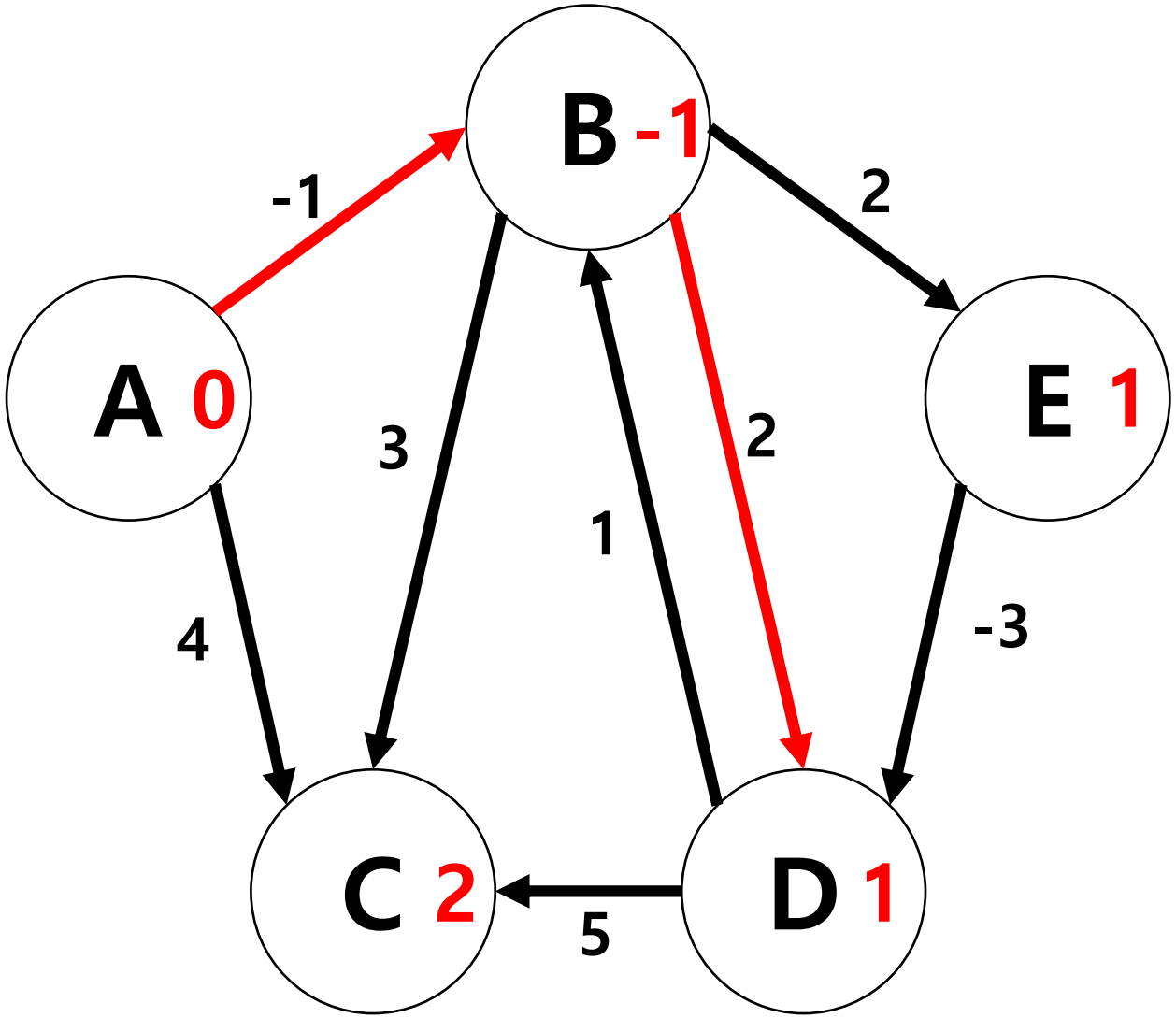
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞



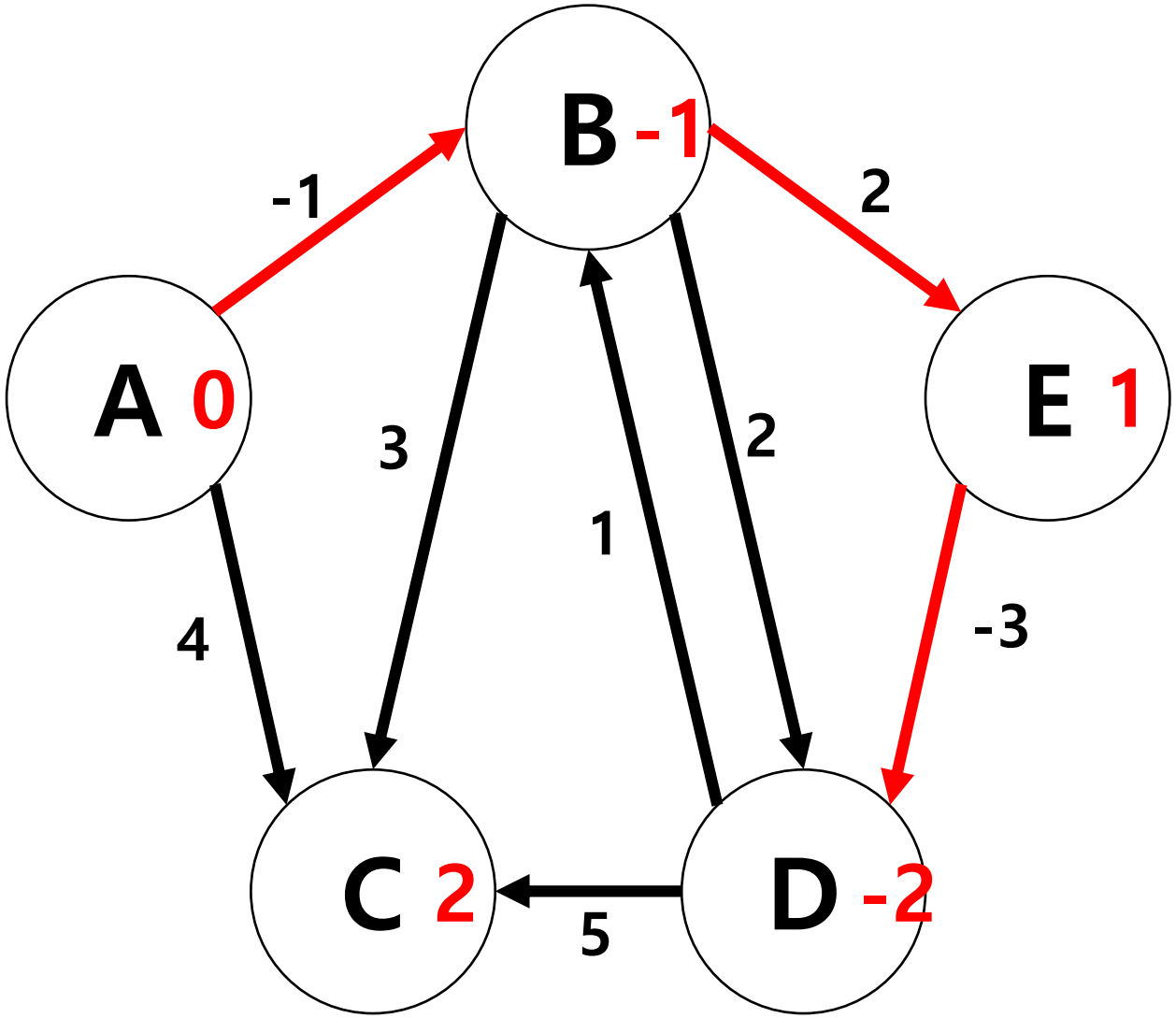
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞



A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1



A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1



A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1

Bellman-Ford Algorithm

- **시간 복잡도**
  - O(|V|*|E|) (단, |E| <= |V|^2)
- **공간 복잡도**
  - O(|V|)
- **실질적 사용**
  - MCMF(Minimum Cost Maximum Flow) 문제 해결

Bellman-Ford Algorithm 관련 논문

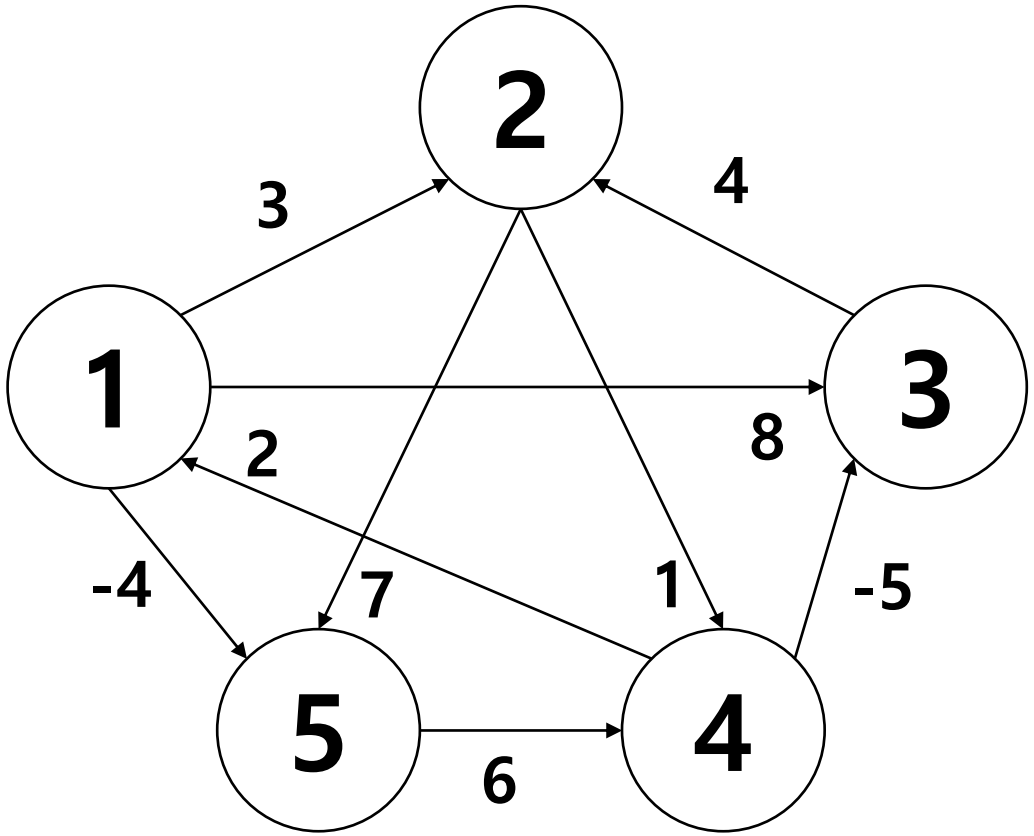
- 진호,서희종. (2012). **SPFA를 기반으로 개선된 벨만-포드 알고리듬(An improved Bellman-Ford algorithm based on SPFA)**. 대한전자통신학회 논문지 제7권 제4호, 721~726

- SPFA : Shortest Path Faster Algorithm
- Bellman-Ford 알고리즘은 모든 노드를 순회함으로 인한 문제점을 개선
- 인접 리스트를 용해서 각 표의 노드를 저장, 대기열을 통해 데이터를 저장
- 새로운 점에 계속 relaxtion을 통해 최적의 경로를 얻을 수 있음

Floyd-Warshall Algorithm

- **개요**
  - 그래프에서 모든 꼭짓점 사이의 최단 경로의 거리를 구하는 알고리즘
  - 모든 경로의 최소 비용을 구함
  - 음수 가중치를 갖는 간선도, 순환만 없다면 처리 가능

- **조건**
  - 3중 반복문(for)문을 사용: **거쳐가는 노드, 출발노드, 도착노드**
  - 2가지 테이블 : **모든 경로의 비용 관련 테이블, 각 정점까지 가기 직전의 정점 관련 테이블**

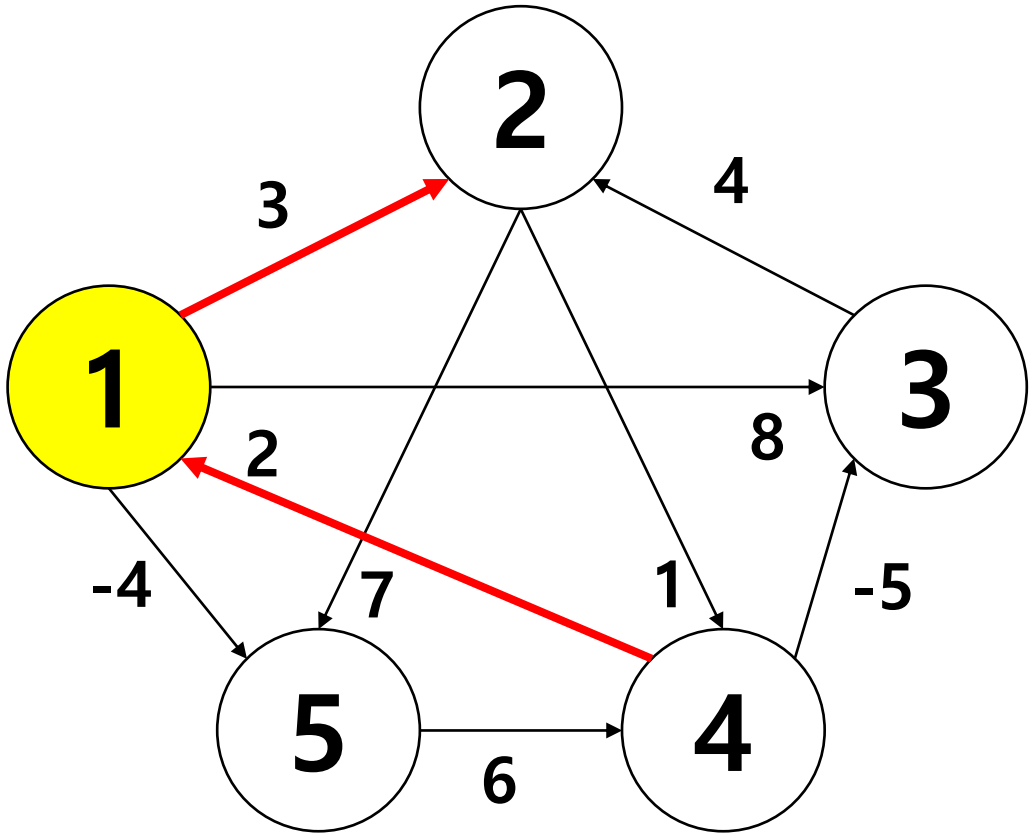


테이블1 : 거리를 저장한 테이블 D

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

테이블2 : 직전 정점을 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	-	1	1	-	1
2	-	-	-	2	2
3	-	3	-	-	-
4	4	-	4	-	-
5	-	-	-	5	-

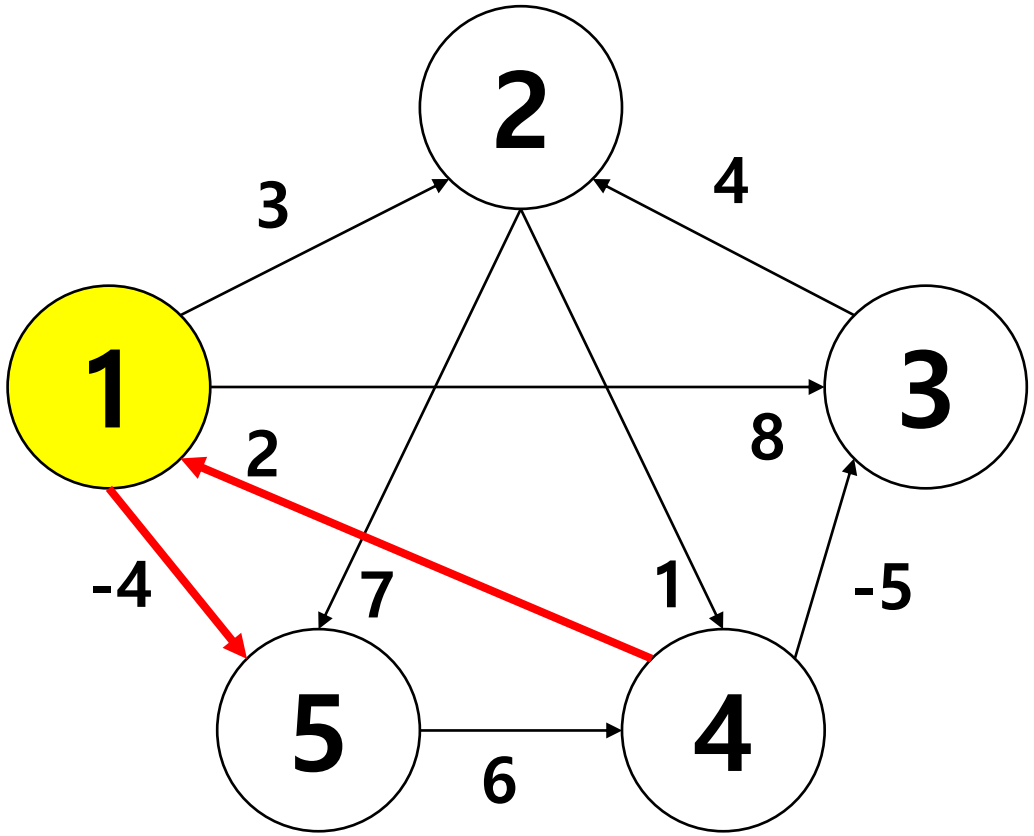


테이블1 : 거리를 저장한 테이블 D

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

테이블2 : 직전 정점을 저장한 테이블 D

	1	2	3	4	5
1	-	1	1	-	1
2	-	-	-	2	2
3	-	3	-	-	-
4	4	**1**	4	-	-
5	-	-	-	5	-

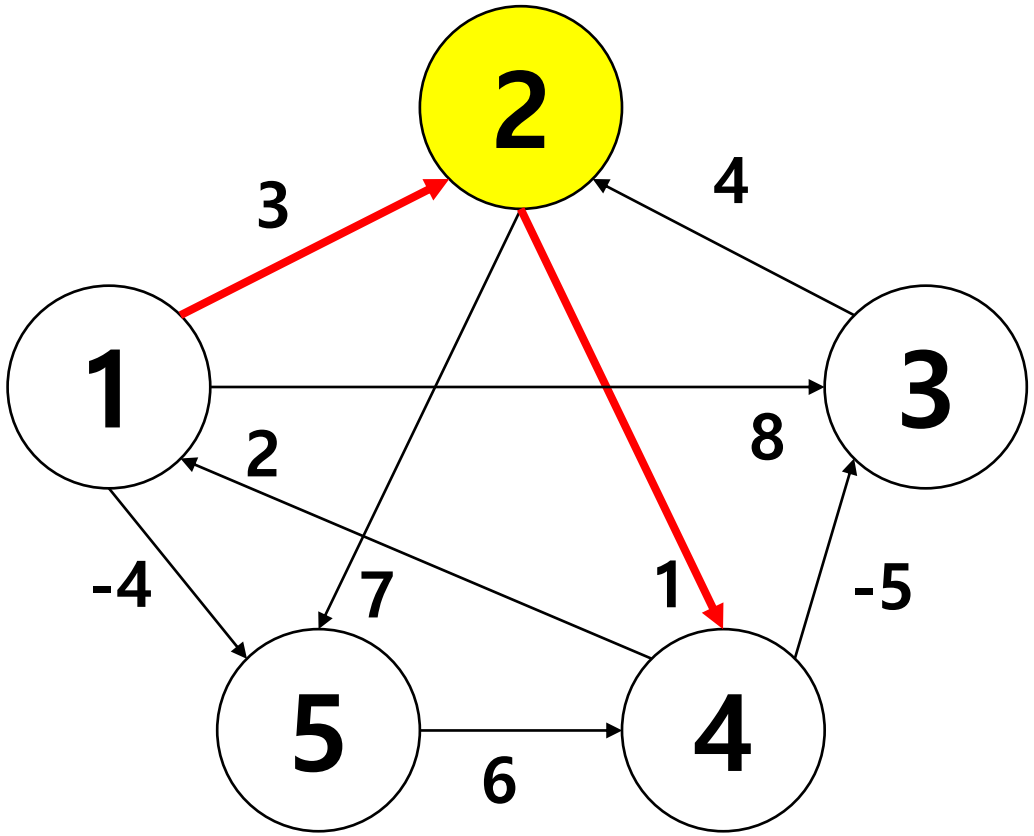


테이블1 : 거리를 저장한 테이블 D

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

테이블2 : 직전 정점을 저장한 테이블 D

	1	2	3	4	5
1	-	1	1	-	1
2	-	-	-	2	2
3	-	3	-	-	-
4	4	1	4	-	**1**
5	-	-	-	5	-

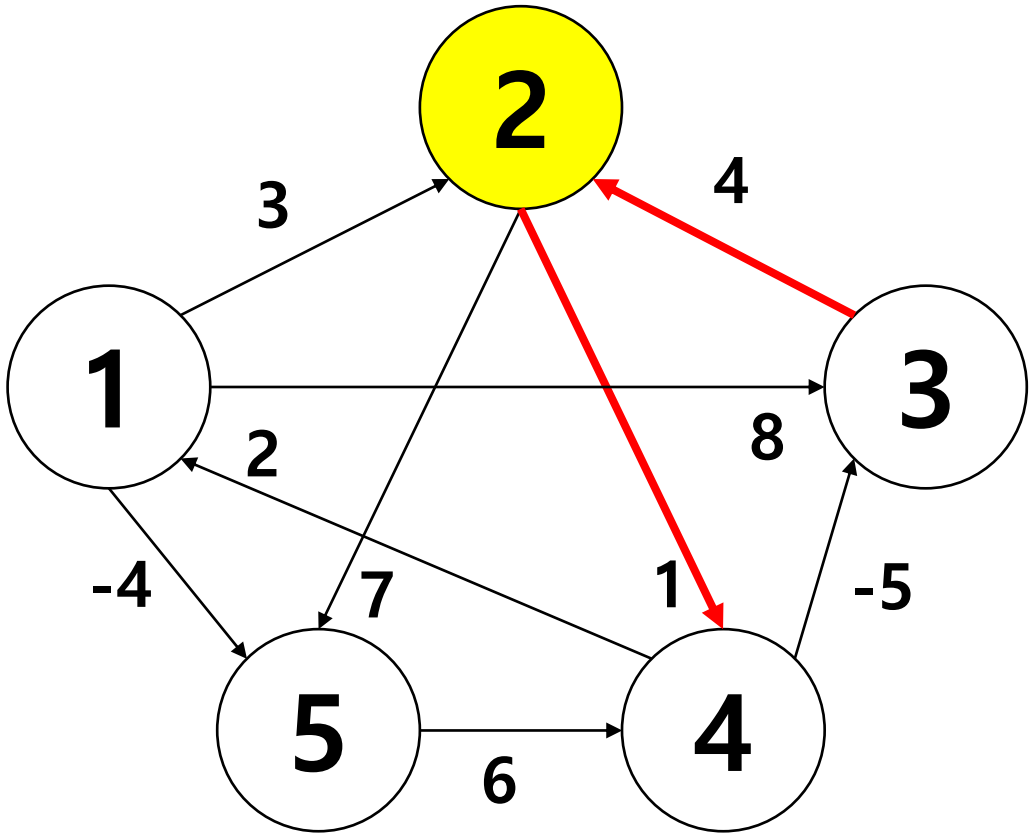


테이블1 : 거리를 저장한 테이블 D

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

테이블2 : 직전 정점을 저장한 테이블 D

	1	2	3	4	5
1	-	1	1	**2**	1
2	-	-	-	2	2
3	-	3	-	-	-
4	4	1	4	-	1
5	-	-	-	5	-

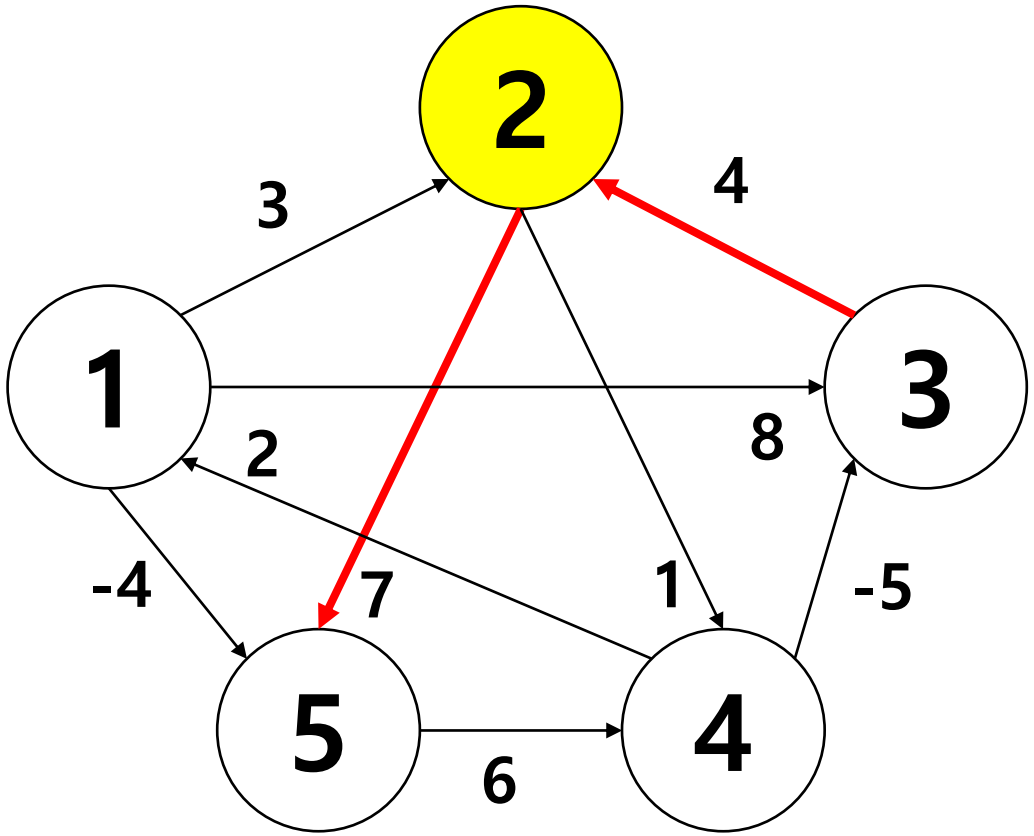


테이블1 : 거리를 저장한 테이블 D

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

테이블2 : 직전 정점을 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	-	1	1	2	1
2	-	-	-	2	2
3	-	3	-	**2**	-
4	4	1	4	-	1
5	-	-	-	5	-

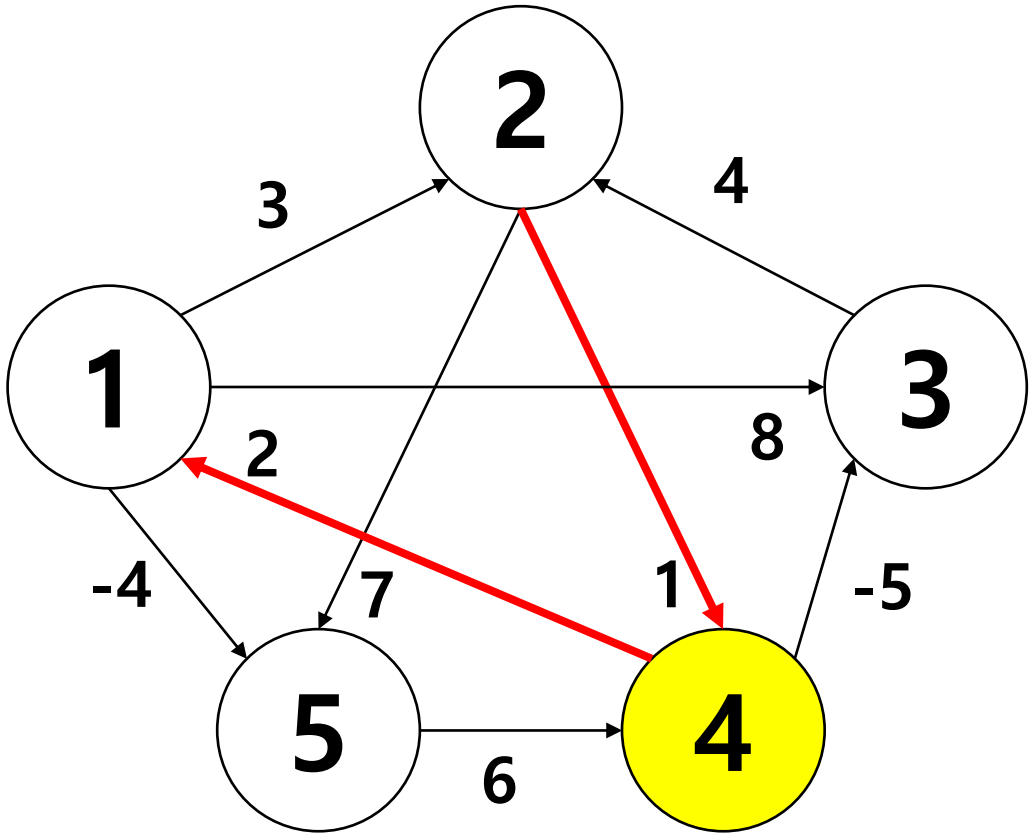


테이블1 : 거리를 저장한 테이블 D

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	**11**
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

테이블2 : 직전 정점을 저장한 테이블 D

	1	2	3	4	5
1	-	1	1	2	1
2	-	-	-	2	2
3	-	3	-	2	**2**
4	4	1	4	-	1
5	-	-	-	5	-

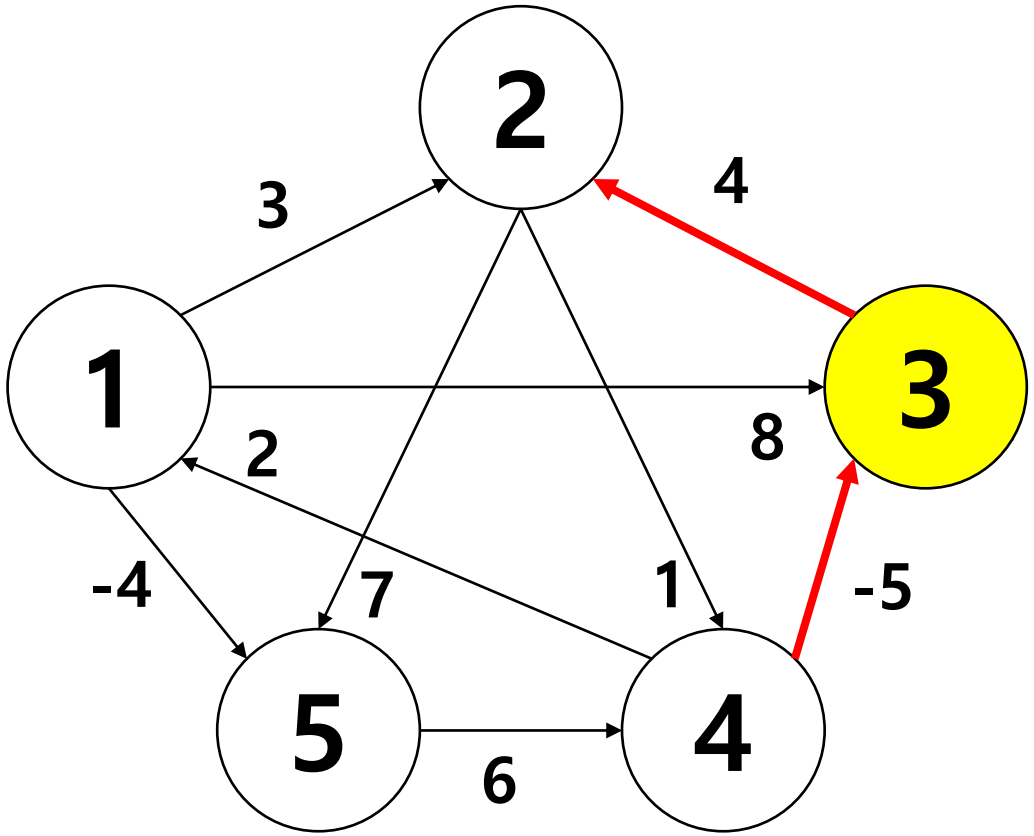


테이블1 : 거리를 저장한 테이블 D

	1	2	3	4	5
1	0	3	8	4	-4
2	3	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

테이블2 : 직전 정점을 저장한 테이블 D

	1	2	3	4	5
1	-	1	1	2	1
2	4	-	-	2	2
3	-	3	-	2	2
4	4	1	4	-	1
5	-	-	-	5	-

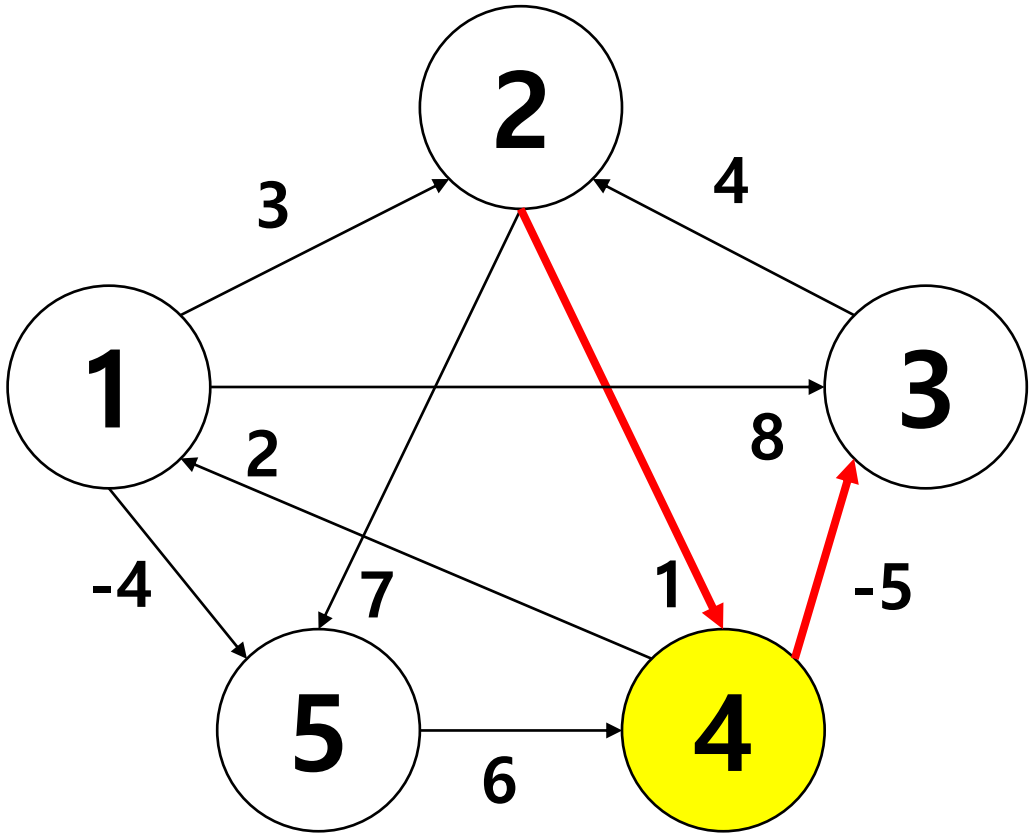


테이블1 : 거리를 저장한 테이블 D

	1	2	3	4	5
1	0	3	8	4	-4
2	3	0	∞	1	7
3	∞	4	0	5	11
4	2	**-1**	-5	0	-2
5	∞	∞	∞	6	0

테이블2 : 직전 정점을 저장한 테이블 D

	1	2	3	4	5
1	-	1	1	2	1
2	4	-	-	2	2
3	-	3	-	2	2
4	4	**3**	4	-	1
5	-	-	-	5	-

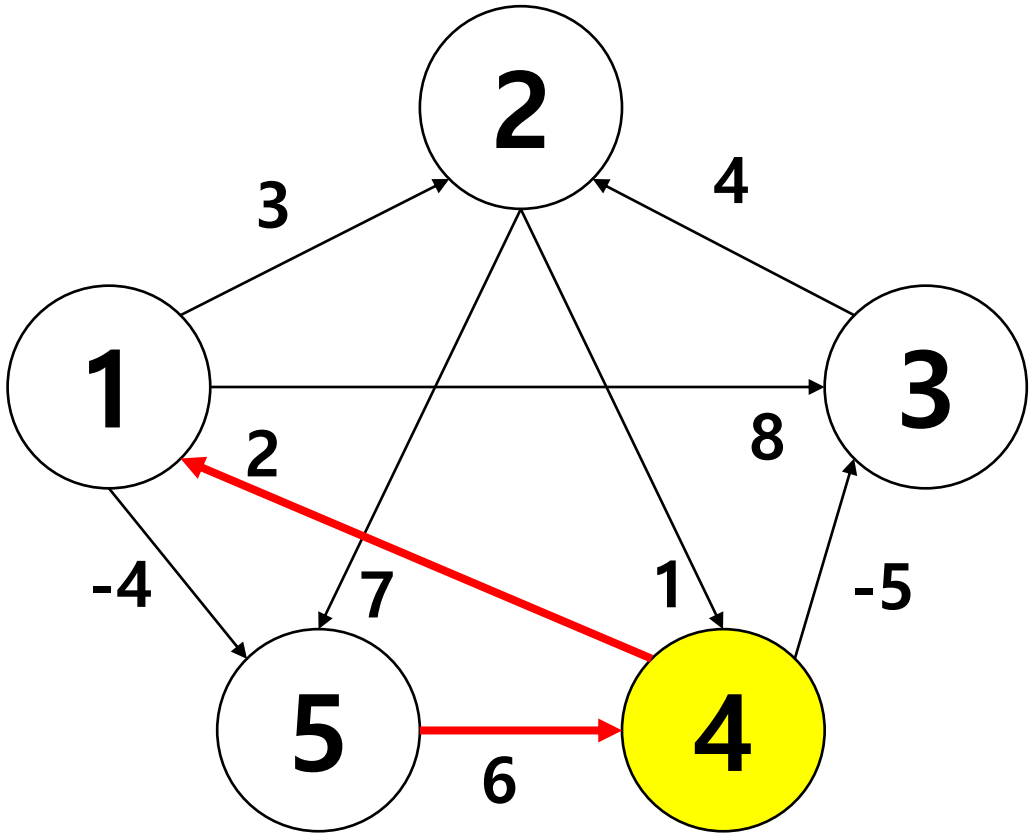


테이블1 : 거리를 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	0	3	8	4	-4
2	3	0	**-4**	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0

테이블2 : 직전 정점을 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	-	1	1	2	1
2	4	-	**4**	2	2
3	-	3	-	2	2
4	4	3	4	-	1
5	-	-	-	5	-

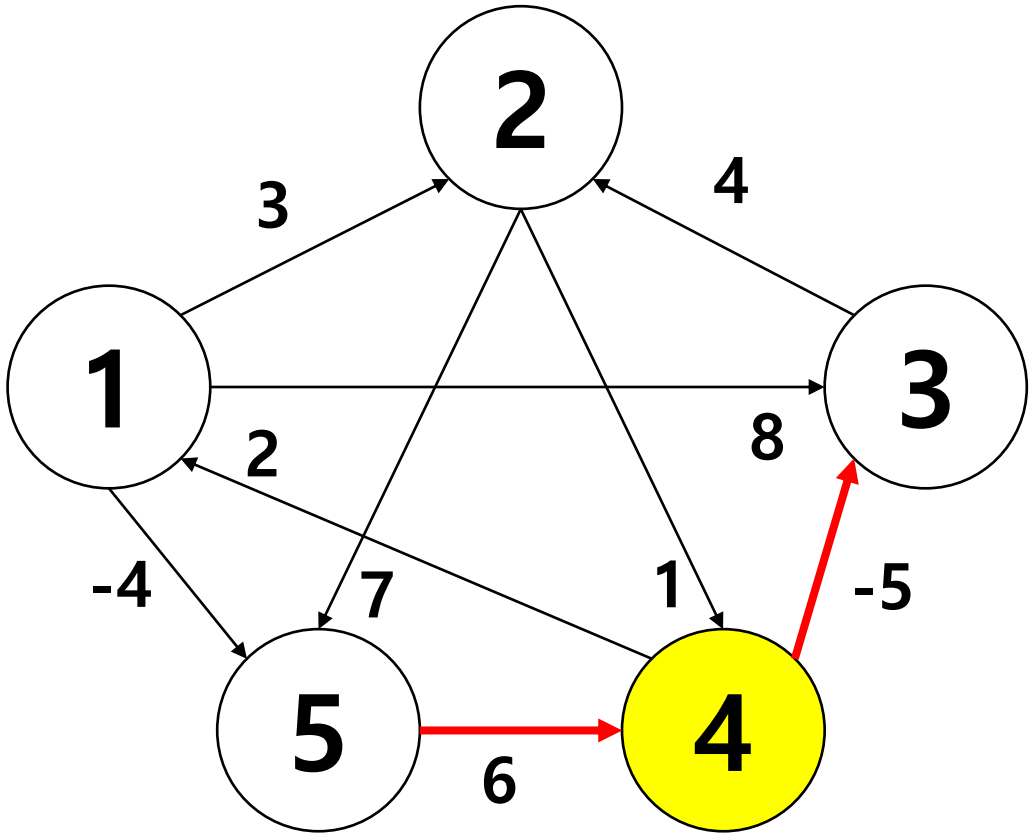


테이블1 : 거리를 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	0	3	8	4	-4
2	3	0	-4	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	**8**	∞	∞	6	0

테이블2 : 직전 정점을 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	-	1	1	2	1
2	4	-	4	2	2
3	-	3	-	2	2
4	4	3	4	-	1
5	**4**	-	-	5	-

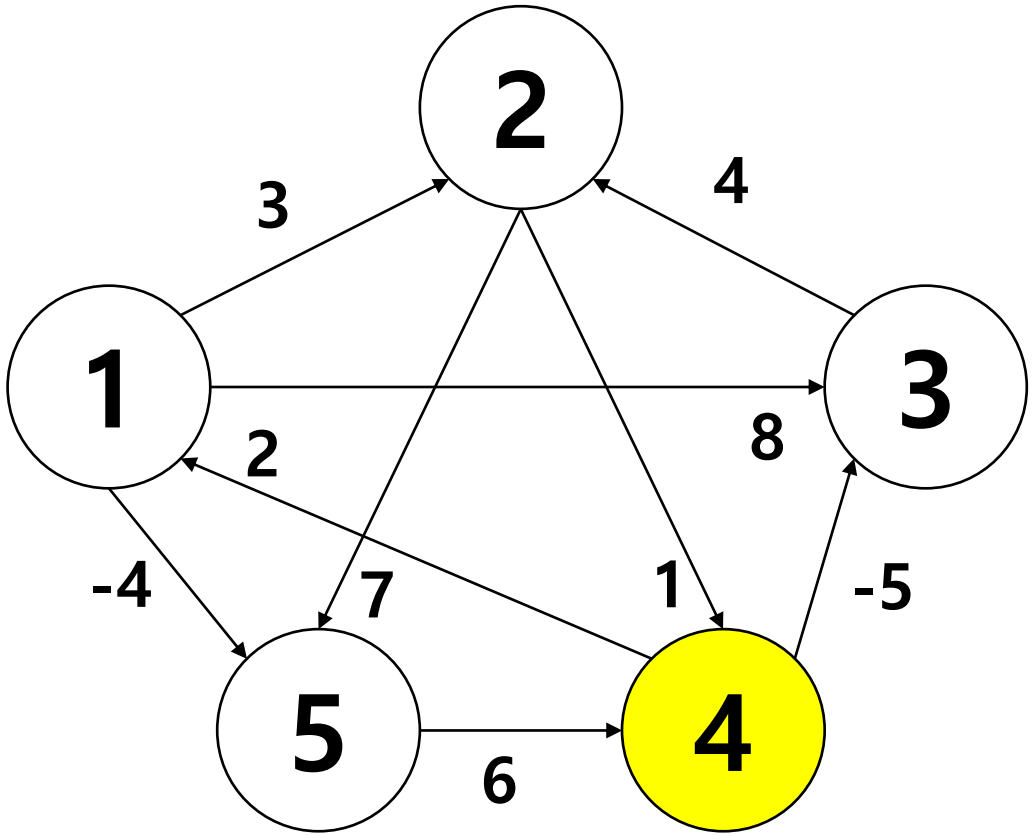


테이블1 : 거리를 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	0	3	8	4	-4
2	3	0	-4	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	8	∞	**1**	6	0

테이블2 : 직전 정점을 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	-	1	1	2	1
2	4	-	4	2	2
3	-	3	-	2	2
4	4	3	4	-	1
5	4	-	**4**	5	-

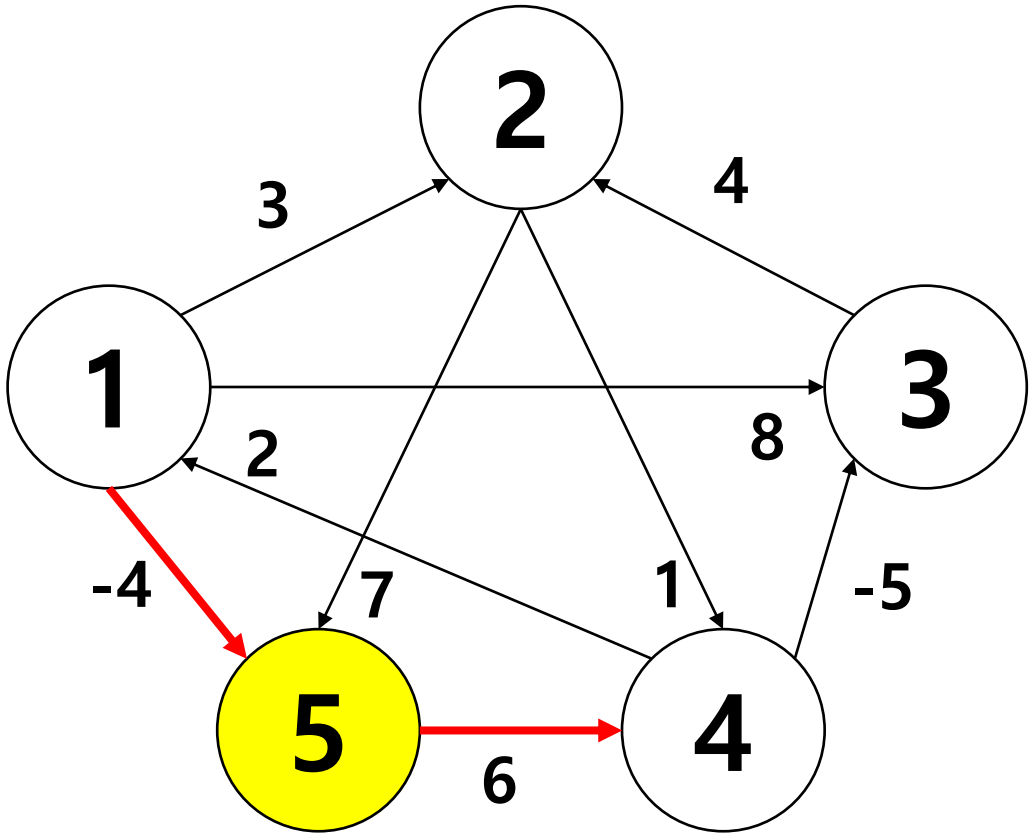


테이블1 : 거리를 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	0	3	8	4	-4
2	3	0	-4	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	8	∞	1	6	0

테이블2 : 직전 정점을 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	-	1	1	2	1
2	4	-	4	2	2
3	-	3	-	2	2
4	4	3	4	-	1
5	4	-	4	5	-



테이블1 : 거리를 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	0	3	8	**2**	-4
2	3	0	-4	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	8	∞	1	6	0

테이블2 : 직전 정점을 저장한 테이블 D

	1	**2**	**3**	**4**	**5**
1	-	1	1	**5**	1
2	4	-	4	2	2
3	-	3	-	2	2
4	4	3	4	-	1
5	4	-	4	5	-

Floyd-Warshall Algorithm

- **시간 복잡도 :** O(|V|^3)
  - (노드의 수) * (노드의 수) * (노드의 수) => |V| * |V| * |V|
- **공간 복잡도 :** O(2 * |V|^2)
  - |V| * |V| 사이즈의 테이블 2개

각 알고리즘 간 비교

	Dijkstra	Bellman-Ford	Floyd-warshall
시간 복잡도	O(|V|^2) (Array) O(E*logV) (heap)	O(|V|*|E|)	O(|V|^3)
공간 복잡도	O(|V|) (Array)	O(|V|)	O(|V|^2)
특징	하나의 노드에서부터 최단 경로를 구함	최단 경로를 구할 때, 모든 edge를 고려함	모든 노드 사이의 최단 경로를 구함

참고자료

- 다익스트라 알고리즘 :
  - http://manducku.tistory.com/29
- P-Dijkstra 알고리즘 논문 :
  - https://fall.kics.or.kr/storage/paper/event/20171110_workshop/publish/9A-4.pdf
- 벨만-포드 알고리즘:
  - https://www.geeksforgeeks.org/dynamic-programming-set-23-bellman-ford-algorithm/
- SPFA를 기반으로 한 개선된 벨만-보드 알고리즘
  - http://dcollection.jnu.ac.kr/public_resource/pdf/000000034527_20180413044750.pdf
- 플로이드 워셜 알고리즘
  - https://ko.wikipedia.org/wiki/%ED%94%8C%EB%A1%9C%EC%9D%B4%EB%93%9C-%EC%9B%8C%EC%85%9C_%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98