BOJ. 1005 – ACM Craft

0. 의의

- 위상 정렬(Topological Sorting)
  - 개념 학습, 코드 구현

1. 개념

- 비선형 자료 구조 **그래프**에서
  - **정점** 간 논리적인 관계에 기반하여 순서를 매기는 작업
  - V1 → V2
    - V1은 V2보다 앞에 존재한다.
- 조건
  - 위상 정렬이 가능하기 위해, 그래프는 특정 성질을 만족해야 한다.

2. DAG(Directional Acyclic Graph)

- 간선에 **방향**이 존재하며, **사이클**이 존재하지 않는 그래프
  - a.k.a 방향성 비순환 그래프, 유향 비순환 그래프

[입력]
첫 줄에 테스트케이스의 개수 T가 주어진다. (1 ≤ T ≤ 15)

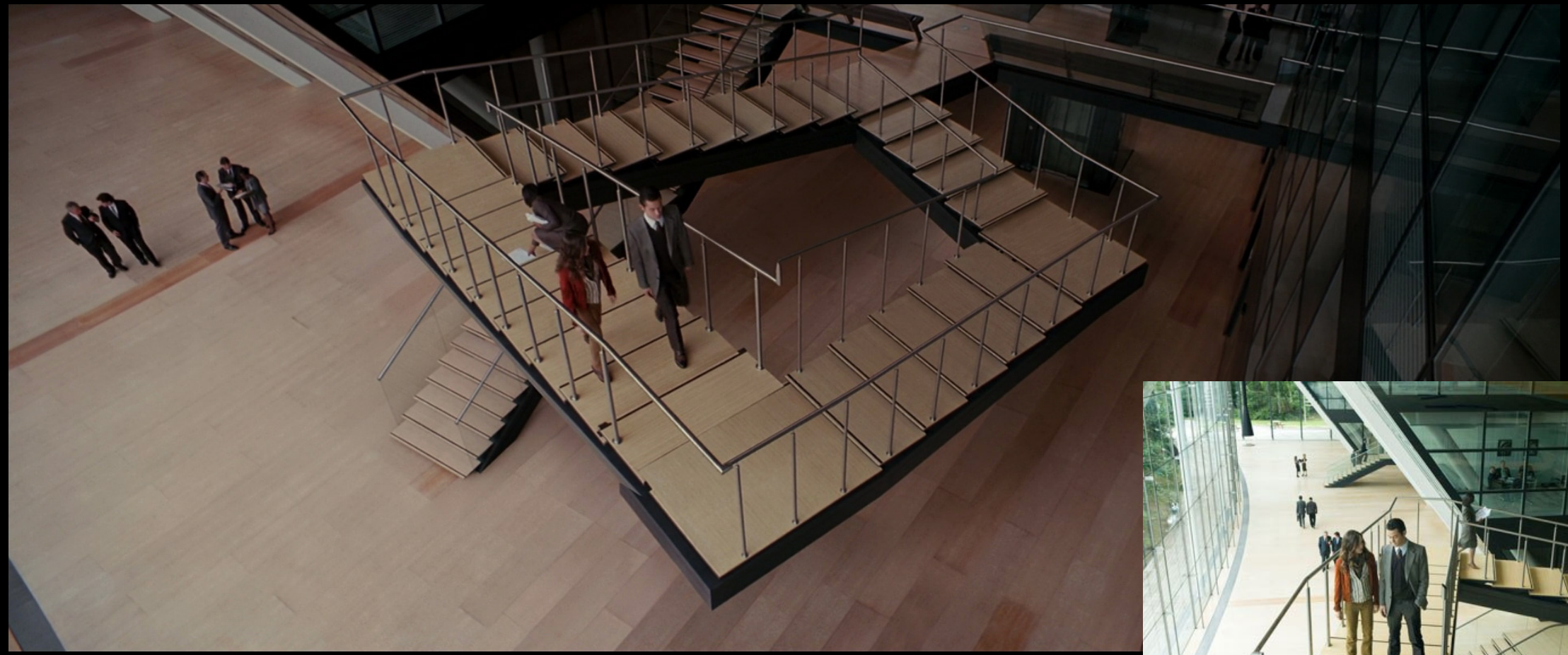
각 테스트 케이스의 첫 번째 줄에 학생들의 수 N이 주어진다. (2 ≤ N ≤ 500)

각 테스트 케이스의 두 번째 줄에 두 학생의 키를 비교한 횟수 M이 주어진다. (0 ≤ M ≤ N*(N-1)/2)

각 테스트 케이스의 세 번째 줄부터 M개의 줄에 걸쳐 두 학생의 키를 비교한 결과를 나타내는 두 양의 정수 a, b가 주어진다.

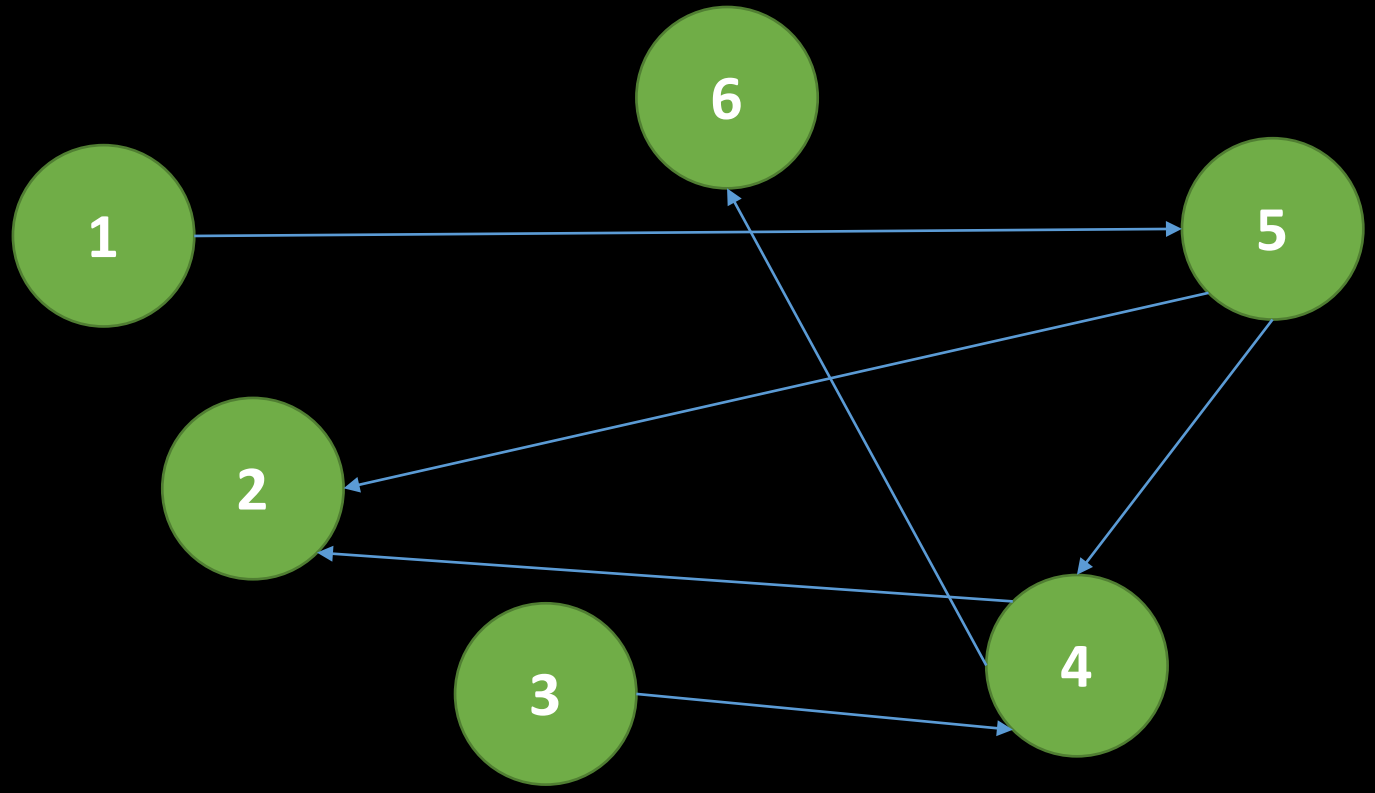
이는 번호가 a인 학생이 번호가 b인 학생보다 키가 작은 것을 의미한다. 이 때, 입력은 항상 모순이 없도록 주어진다.

3. 인셉션 – 펜로즈의 계단



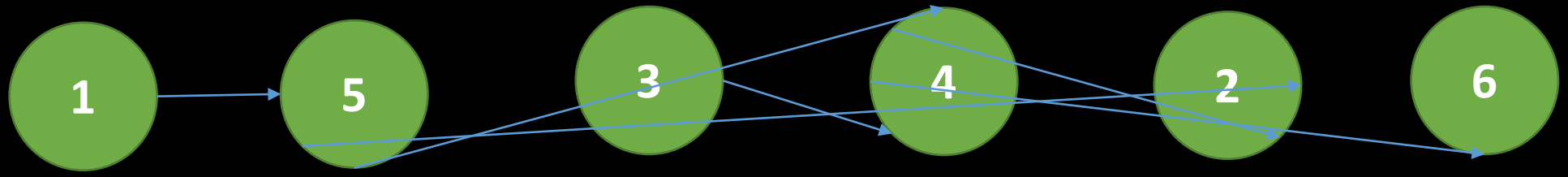
4. 위상 정렬(Topological Sorting)

- DAG에서, 정점 간 순서에 모순이 존재하지 않도록 정렬하는 알고리즘



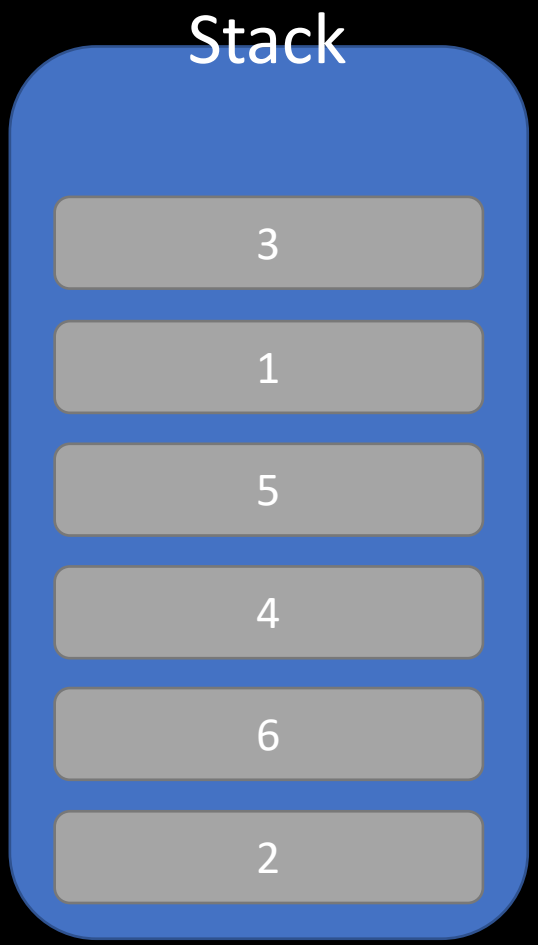
4. 위상 정렬(Topological Sorting)

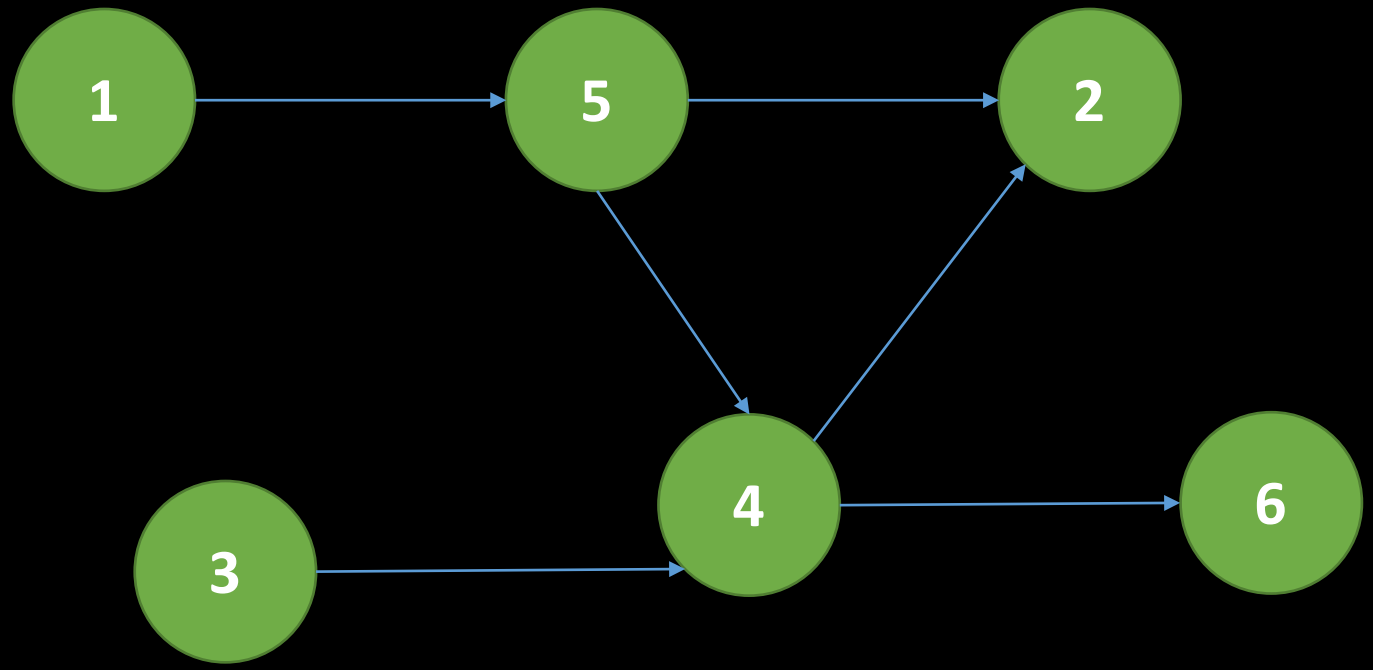
- DAG에서, 정점 간 순서에 모순이 존재하지 않도록 정렬하는 알고리즘



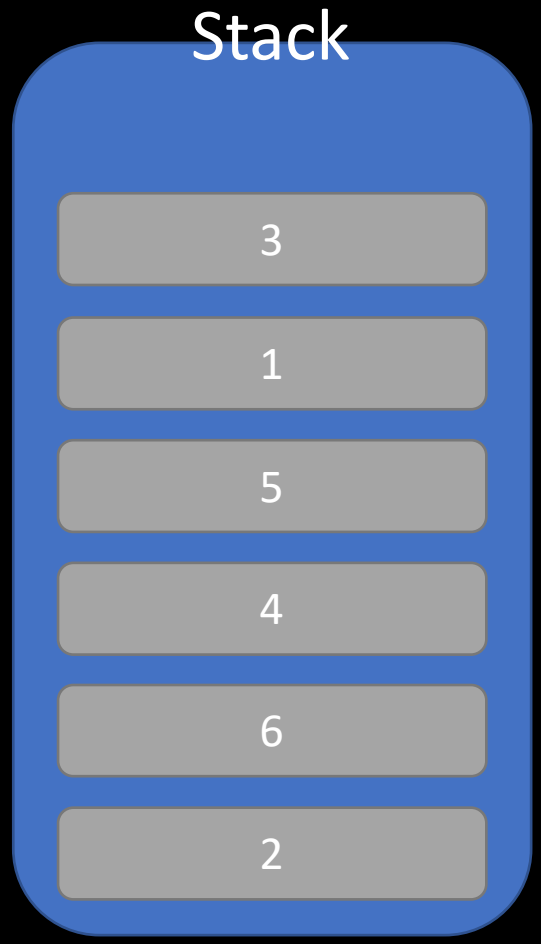
다만, 순서가 상관 없는 정점들이 존재할 수 있다

5. DFS / Stack





6. 한계



- 정렬 순서에는 모순이 존재하지 않는다.
- **정확한 관계**를 알기 어렵다.
  - 같은 높이에 있는 정점을 알아낼 수 없다.
- 기타
  - DAG가 보장되지 않는다면, 사이클을 판별해야 한다.

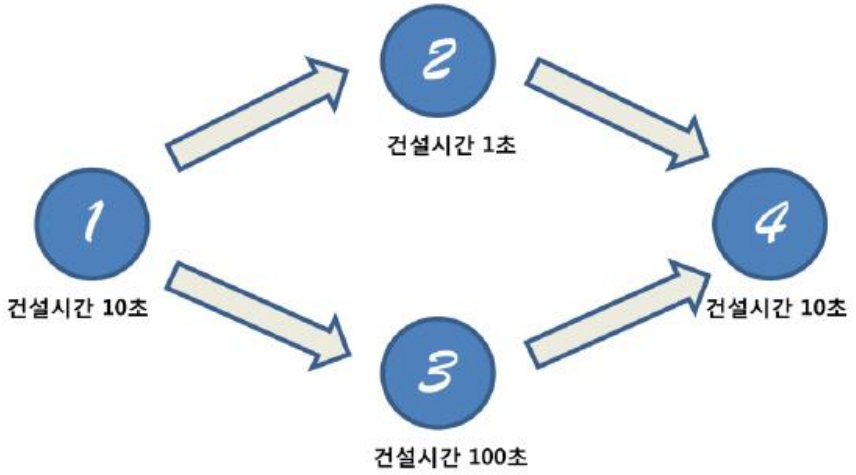
7. 문제 해석(1)

문제

서기 2012년! 드디어 2년간 수많은 국민들을 기다리게 한 게임 ACM Craft (Association of Construction Manager Craft)가 발매되었다.

이 게임은 지금까지 나온 게임들과는 다르게 ACM크래프트는 다이나믹한 게임 진행을 위해 건물을 짓는 순서가 정해져 있지 않다. 즉, 첫 번째 게임과 두 번째 게임이 건물을 짓는 순서가 다를 수도 있다. 매 게임시작 시 건물을 짓는 순서가 주어진다. 또한 모든 건물은 각각 건설을 시작하여 완성이 될 때까지 Delay가 존재한다.

각 정점의 무게가 다르다.



7. 문제 해석(2)

위의 예시를 보자.

이번 게임에서는 다음과 같이 건설 순서 규칙이 주어졌다. 1번 건물의 건설이 완료된다면 2번과 3번의 건설을 시작할수 있다. (동시에 진행이 가능하다) 그리고 4번 건물을 짓기 위해서는 2번과 3번 건물이 모두 건설 완료되어야지만 4번건물의 건설을 시작할수 있다.

따라서 4번건물의 건설을 완료하기 위해서는 우선 처음 1번 건물을 건설하는데 10초가 소요된다. 그리고 2번 건물과 3번 건물을 동시에 건설하기 시작하면 2번은 1초뒤에 건설이 완료되지만 아직 3번 건물이 완료되지 않았으므로 4번 건물을 건설할 수 없다. 3번 건물이 완성되고 나면 그때 4번 건물을 지을수 있으므로 4번 건물이 완성되기까지는 총 120초가 소요된다.

프로게이머 최백준은 애인과의 데이트 비용을 마련하기 위해 서강대학교배 ACM크래프트 대회에 참가했다! 최백준은 화려한 컨트롤 실력을 가지고 있기 때문에 모든 경기에서 특정 건물만 짓는다면 무조건 게임에서 이길 수 있다. 그러나 매 게임마다 특정건물을 짓기 위한 순서가 달라지므로 최백준은 좌절하고 있었다. 백준이를 위해 특정건물을 가장 빨리 지을 때까지 걸리는 최소시간을 알아내는 프로그램을 작성해주자.

입력

첫째 줄에는 테스트케이스의 개수 T가 주어진다. 각 테스트 케이스는 다음과 같이 주어진다. 첫째 줄에 건물의 개수 N과 건물간의 건설순서 규칙의 총 개수 K이 주어진다. (건물의 번호는 1번부터 N번까지 존재한다)

둘째 줄에는 각 건물당 건설에 걸리는 시간 D_1, D_2, ..., D_N이 공백을 사이로 주어진다. 셋째 줄부터 K+2줄까지 건설순서 X Y가 주어진다. (이는 건물 X를 지은 다음에 건물 Y를 짓는 것이 가능하다는 의미이다)

마지막 줄에는 백준이가 승리하기 위해 건설해야 할 건물의 번호 W가 주어진다.

출력

건물 W를 건설완료 하는데 드는 최소 시간을 출력한다. 편의상 건물을 짓는 명령을 내리는 데는 시간이 소요되지 않는다고 가정한다.

건설순서는 모든 건물이 건설 가능하도록 주어진다.

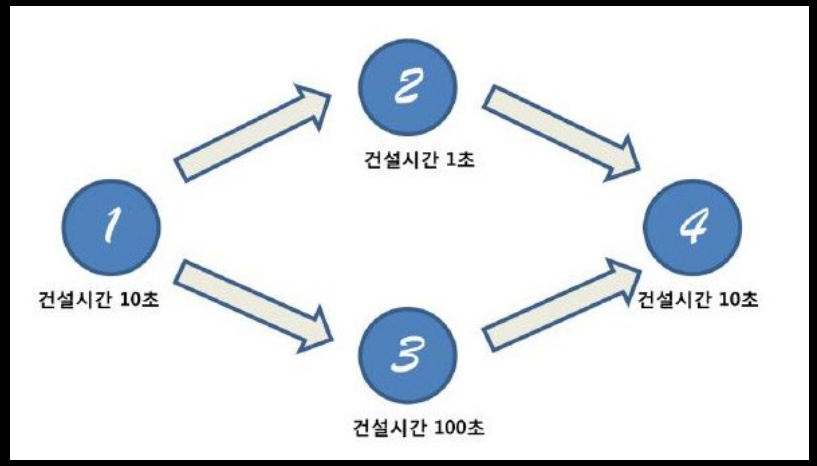
위상 정렬

DAG

방향이 존재한다.

사이클이 발생하지 않는다.

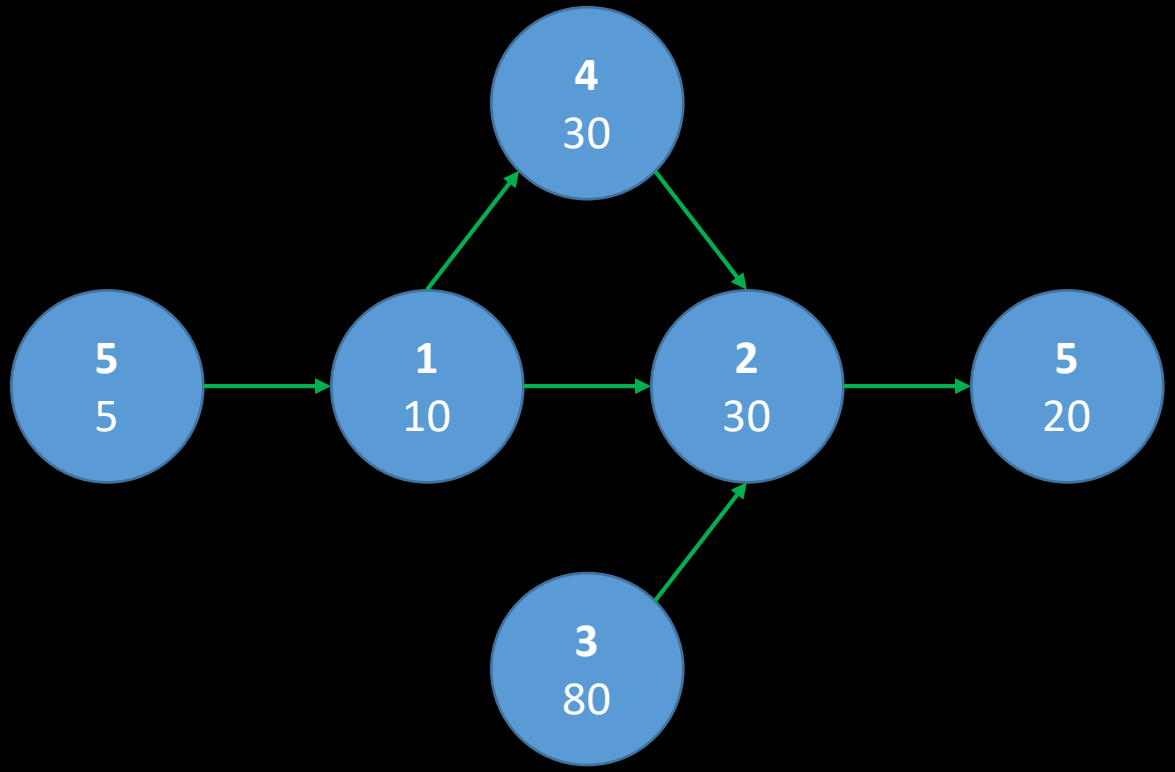
8. 차이점



- 정렬만으로는, 건물 W의 최소 건설 시간을 알아낼 수 없다.

9. 과정

	건설 시간
rank[0]	
rank[1]	
rank[2]	
rank[3]	
rank[4]	
rank[5]	



10. Pseudo-code(1)

- DFS(int v)
  - 건설 시간이 0보다 크다면, rank[v]를 반환
  - 건설 시간(v) =
    - v를 짓는 데 걸리는 시간 +
    - v의 이전 노드들을 짓는 데 걸리는 시간 중 **최댓값**
  - rank[v] = 건설 시간(v)
  - rank[v] 반환

rank[]

건물을 짓기 위해 필요한 최소 시간

d[]

각 건물을 짓기 위해 필요한 시간

10. Pseudo-code(2)

- DFS(int v)
  - 건설 시간이 0보다 크다면, rank[v]를 반환
  - tempVisited[v] = true
  - 건설 시간(v) =
    - V를 짓는 데 걸리는 시간 +
    - V의 이전 노드들을 짓는 데 걸리는 시간 중 **최댓값**
  - tempVisited[v] = false
  - Rank[v] = 건설 시간(v)
  - Rank[v] 반환

방문한 적 있는 노드라면,
사이클이 발생!!

다음 순회에서는 방문 가능!!

11. Reference

- https://www.intjournal.com/0717/inception
- https://ko.wikipedia.org/wiki/위상정렬
- BOJ – 1005. ACM Craft
- SWEA – 5643. 키 순서