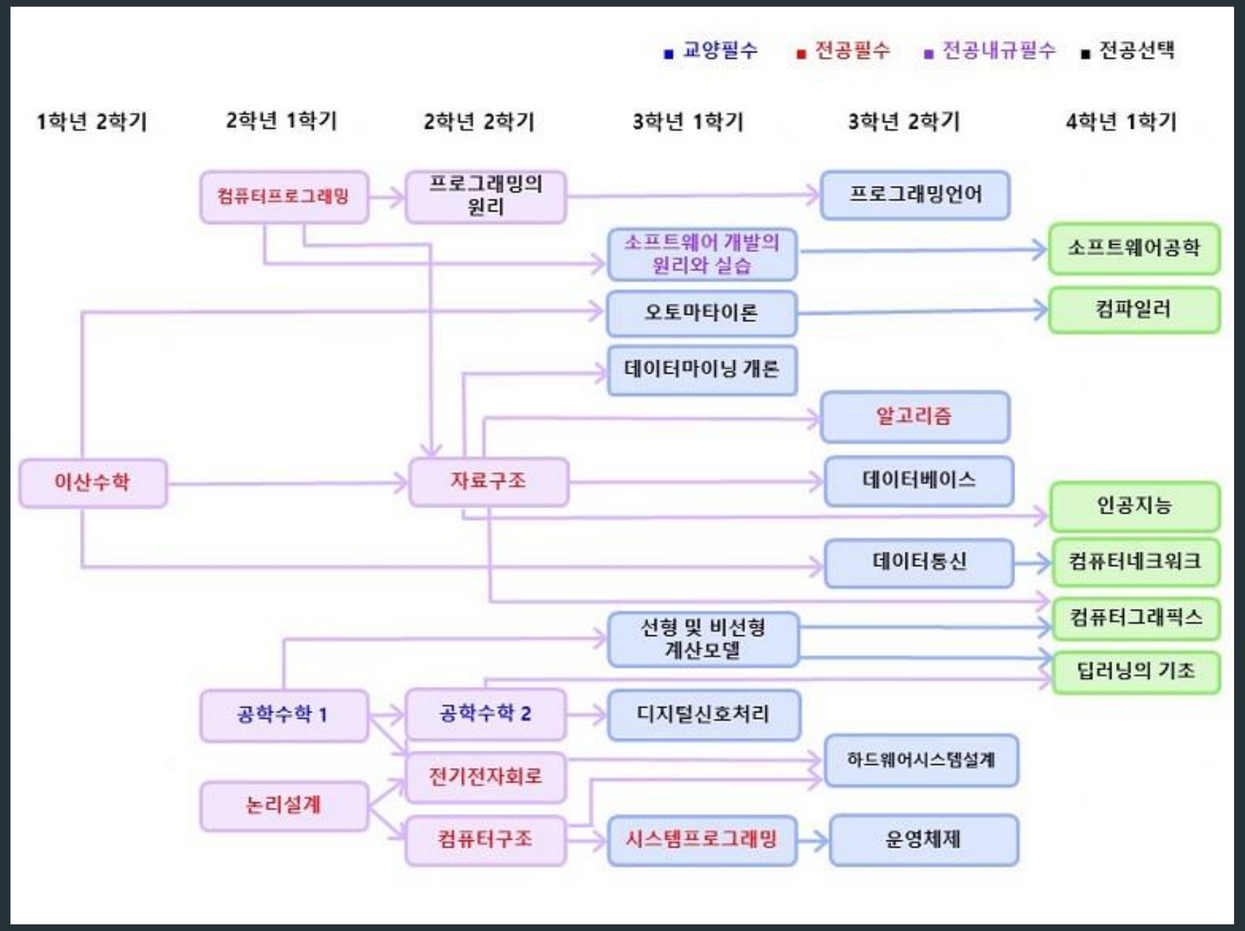
알튜비튜

위상 정렬

비순환 방향 그래프의 모든 정점을 선후 관계를 지키며 나열하는 위상 정렬을 배웁니다.
주로 단독으로 나오진 않고, 정말 가끔 다른 그래프 알고리즘과 함께 나오곤 해요.

일상 속 위상 정렬



- 이수 교과목 순서?

→ 그 전에 꼭 이수하고 와야 할 교과목만 지킨다면, 듣는 순서는 자유

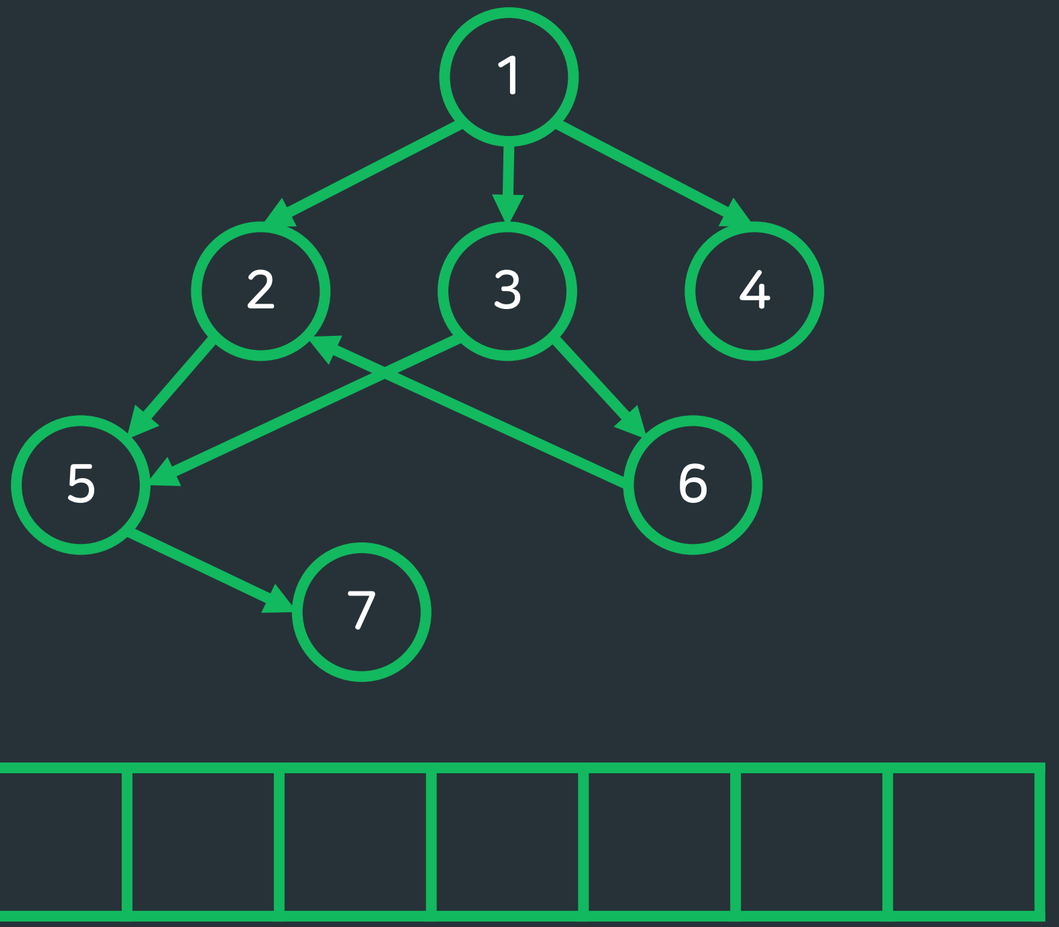
위상 정렬

#알튜비튜

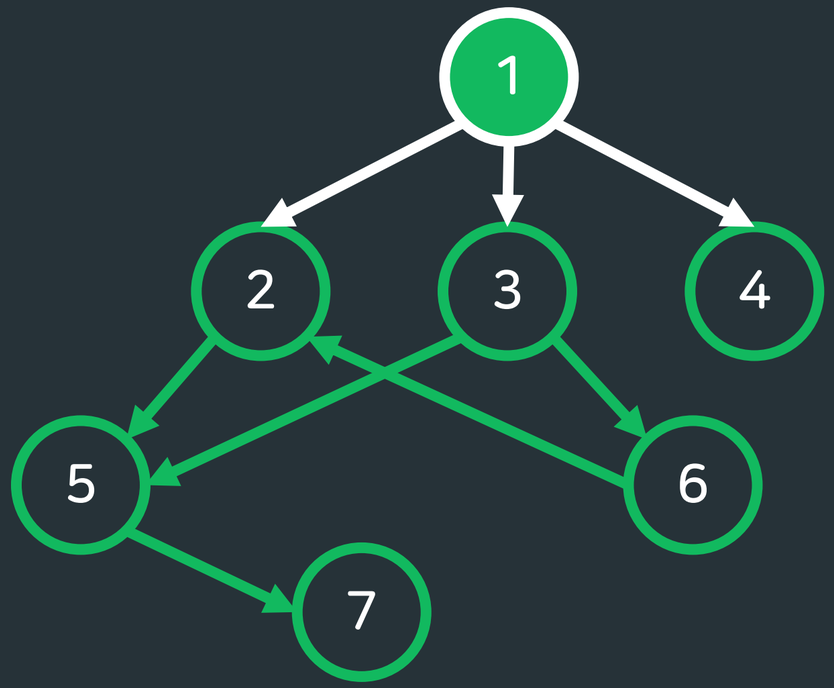
Topological Sort

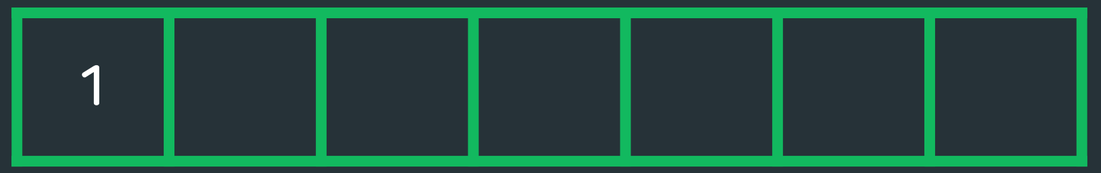
- 그래프의 선후 관계를 지키며 모든 정점을 일렬로 나열하는 알고리즘
- 순서가 정해져 있는 작업을 차례로 수행해야 할 때, 작업의 순서를 결정함
- 위상 정렬의 결과는 여러가지가 가능
- 사이클이 없는 방향 그래프(DAG)에서 사용
- 일상 생활에서는 보통 스케줄을 짤 때 많이 쓰임

위상 정렬 과정

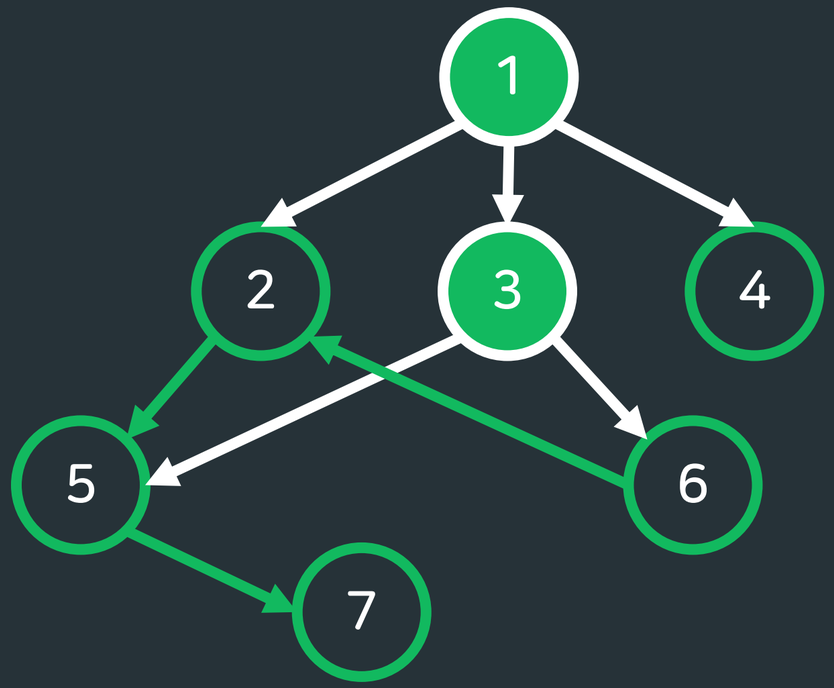


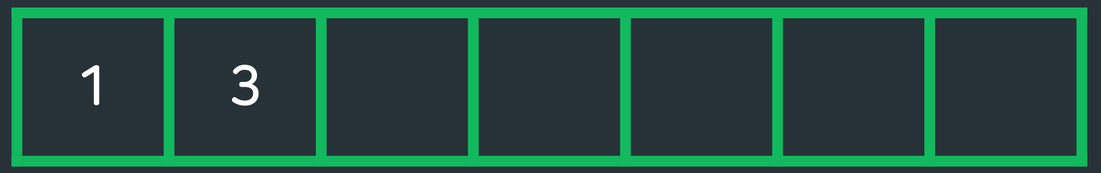
위상 정렬 과정



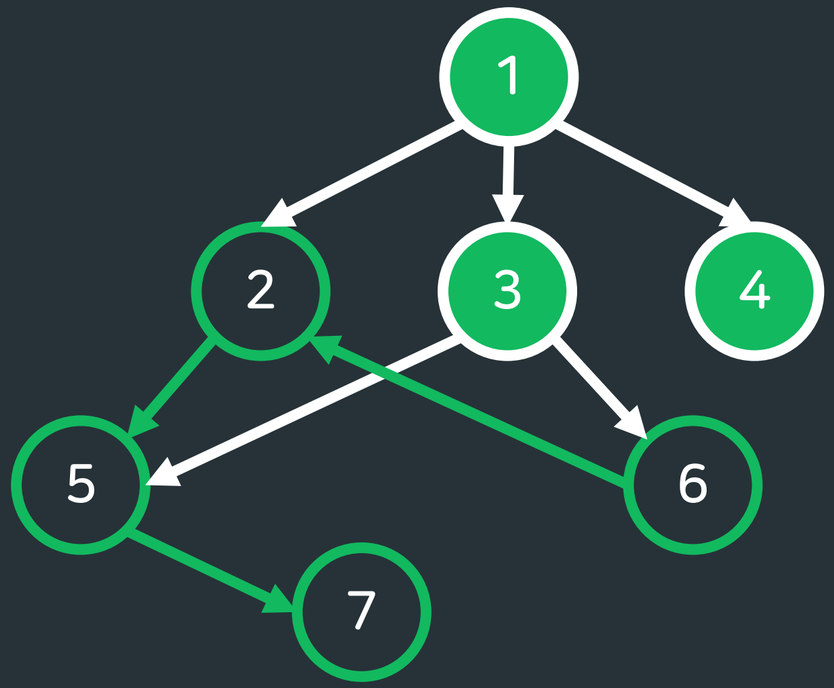


위상 정렬 과정



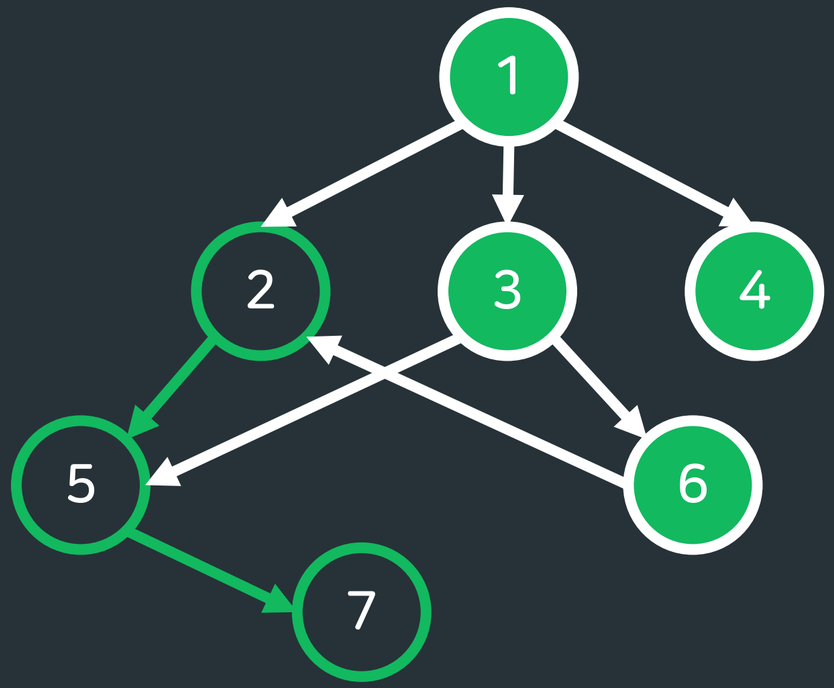


위상 정렬 과정



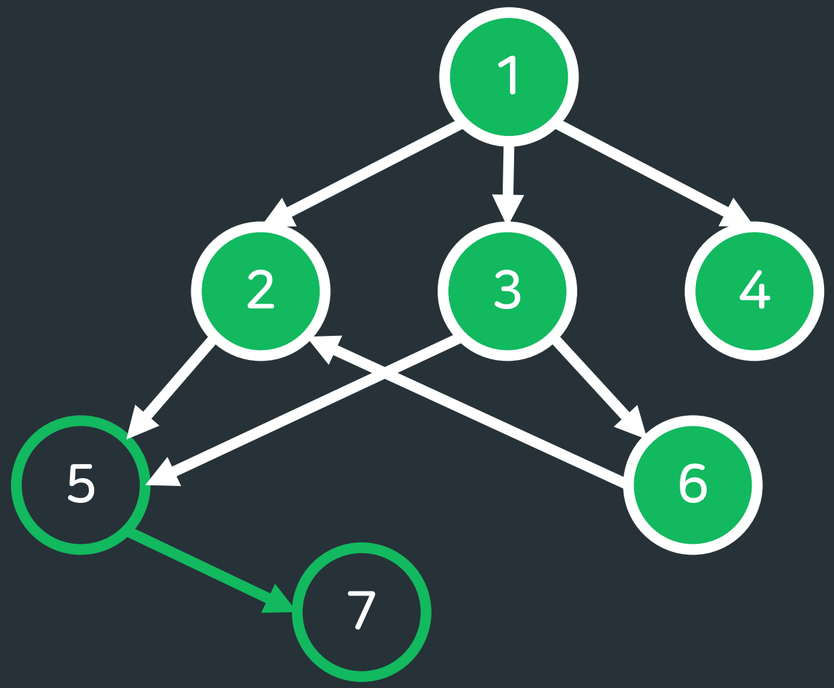
1	3	4				

위상 정렬 과정



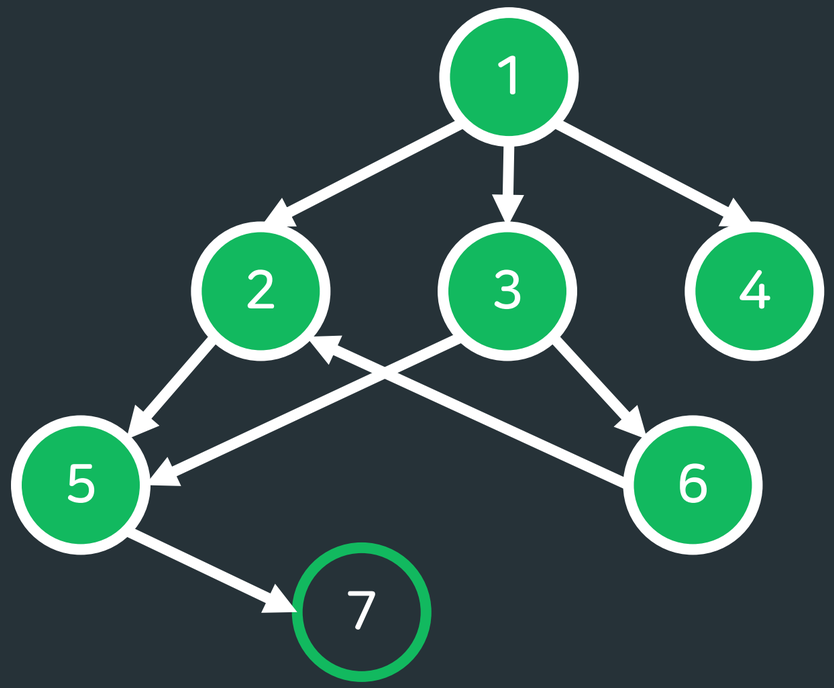
1	3	4	6			

위상 정렬 과정



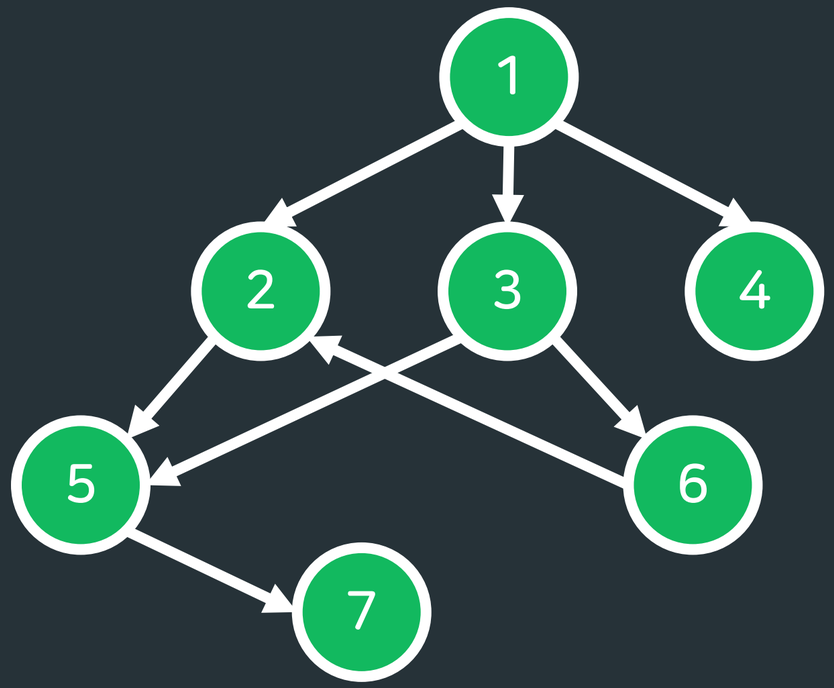
1	3	4	6	2		

위상 정렬 과정



1	3	4	6	2	5	

위상 정렬 과정

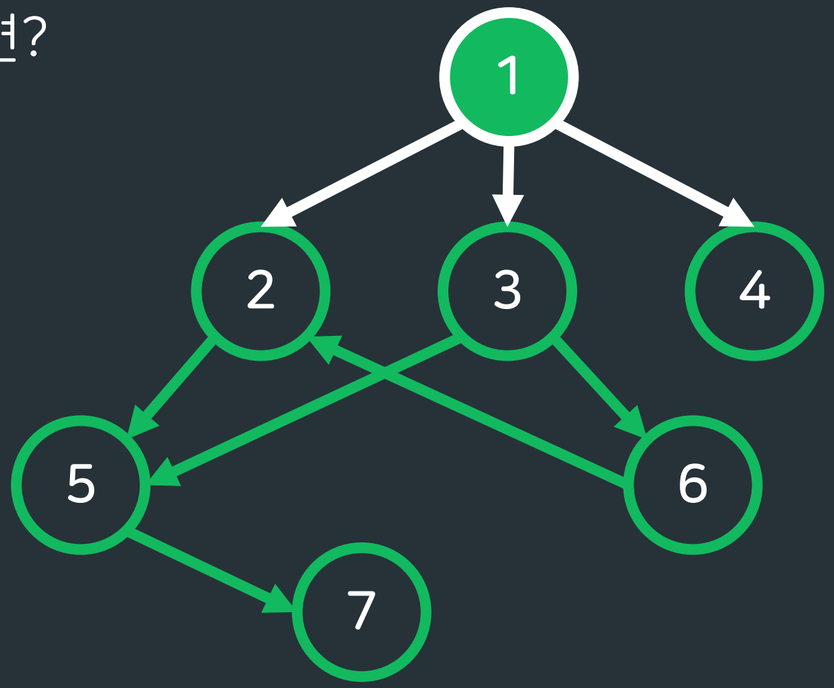


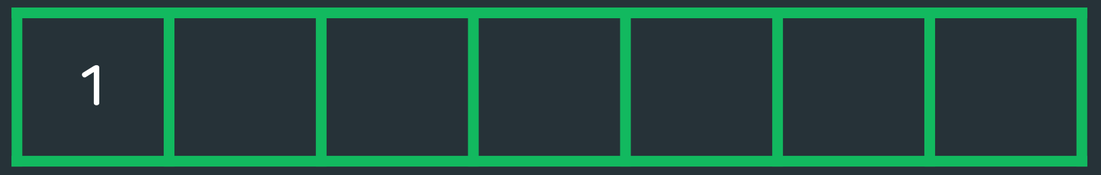
1	3	4	6	2	5	7

위상 정렬의 결과는 여러가지 가능

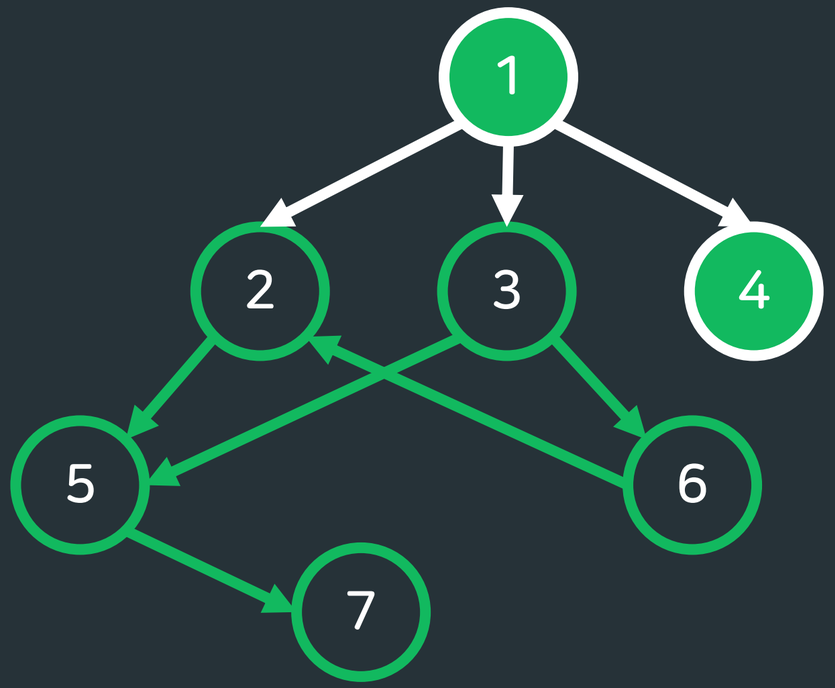
#알튜비튜

- 만약 4를 먼저 검사한다면?

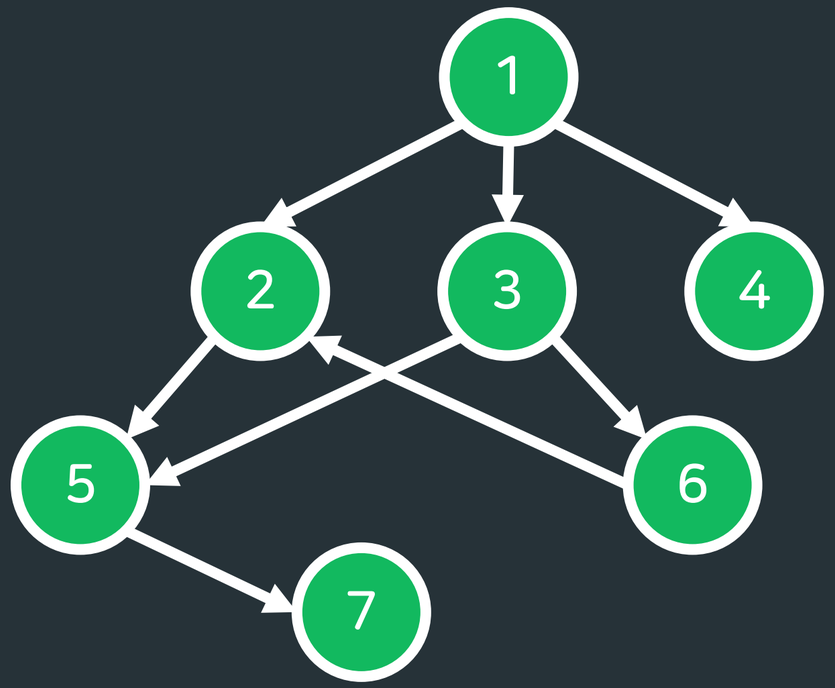




위상 정렬의 결과는 여러가지 가능

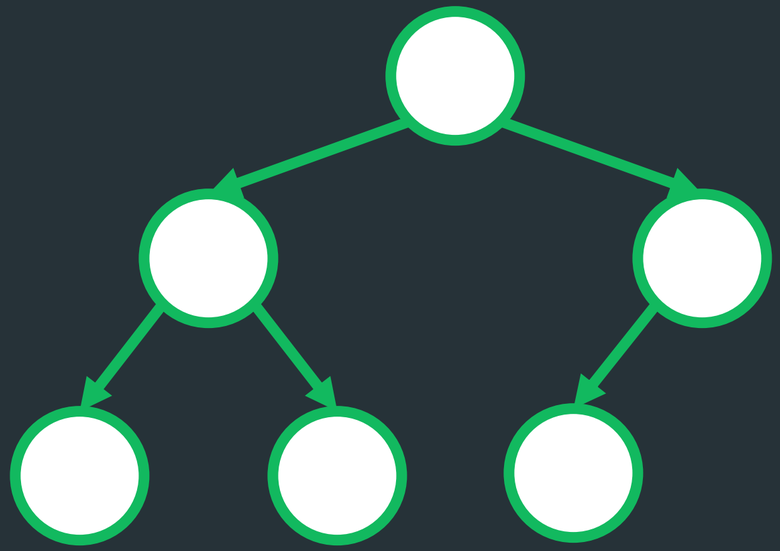


1	4	3	6	2	5	7



1	3	4	6	2	5	7

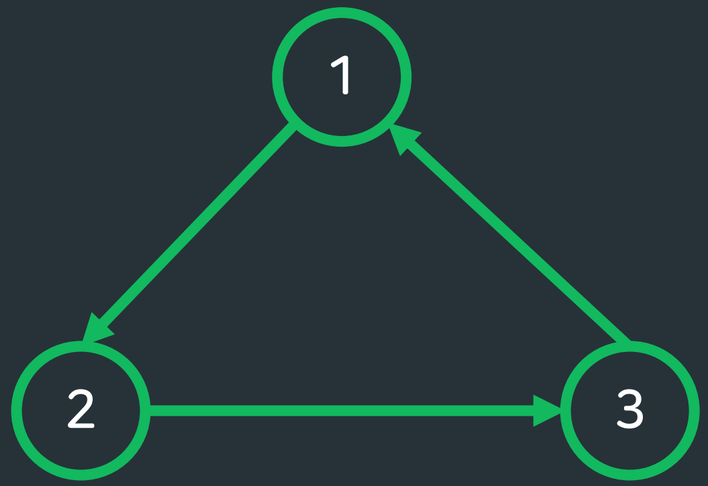
그래프 용어 복습



차수

- Indegree(진입차수): 방향 그래프에서 해당 정점으로 들어오는 간선의 수
- Outdegree(진출차수): 방향 그래프에서 해당 정점에서 나가는 간선의 수

만약 사이클이 존재한다면?



- 진입차수가 0인 정점이 존재하지 않음
- 다른 연결된 정점이 있다 해도, 사이클 내의 정점은 진입차수가 0이 절대 될 수 없음

과정

위상 정렬

1. 진입차수가 0인 정점을 찾아서 나열함
2. 1에서 찾은 정점과 그 정점으로 부터 나오는 간선을 그래프에서 삭제함
3. 1-2 과정을 반복
4. 그래프가 모두 삭제되면 종료

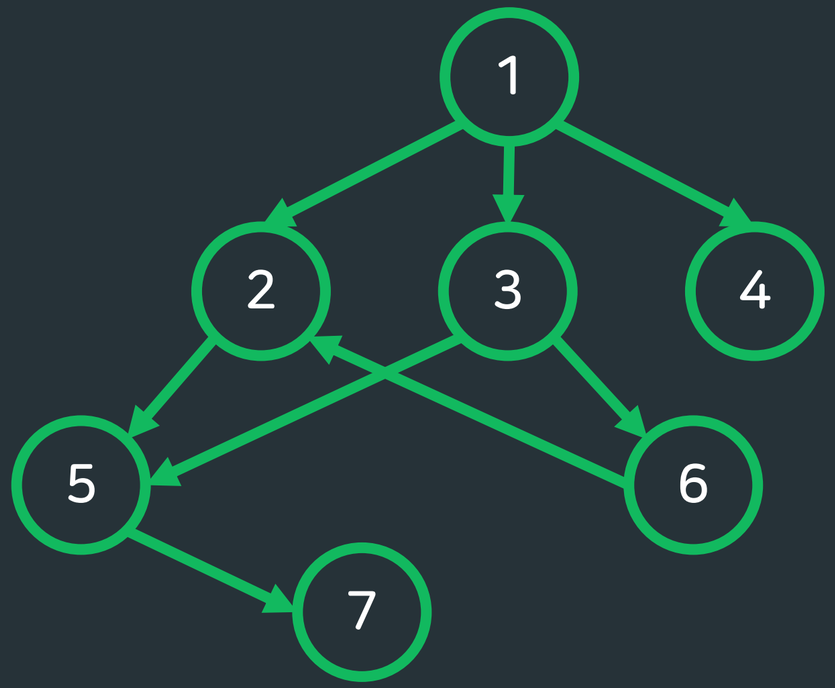
위상 정렬

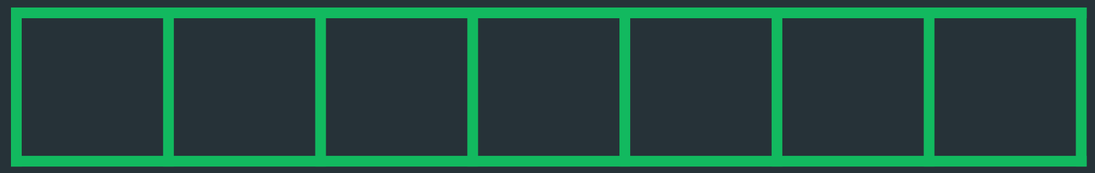
1. 진입차수가 0인 정점을 찾아서 컨테이너에 저장
2. 1에서 찾은 정점과 연결된 정점의 진입차수를 1씩 감소
3. 1-2 과정을 반복
4. 모든 정점이 선택되었다면 종료

위상 정렬

- 각 정점마다 Indegree를 저장할 배열을 만듦
- 진입차수가 0인 정점이 여러 개일 수 있으므로, 이를 저장해 둘 큐(queue) 필요

위상 정렬 과정



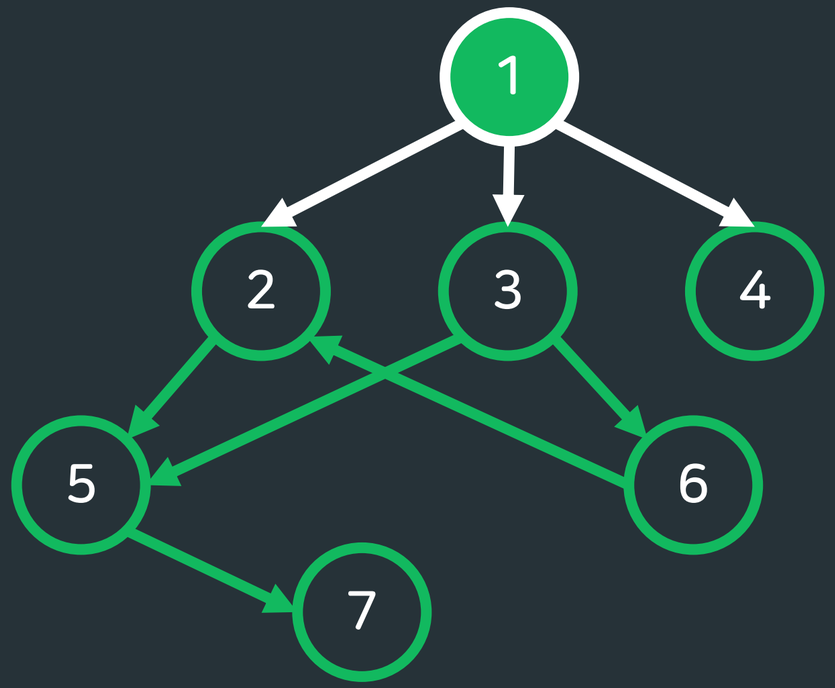


Indegree

1	2	3	4	5	6	7
0	2	1	1	2	1	1

queue

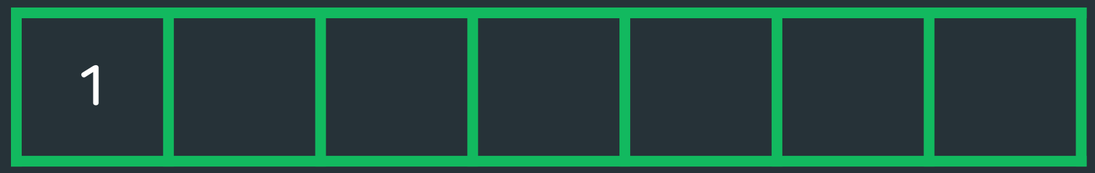
위상 정렬 과정



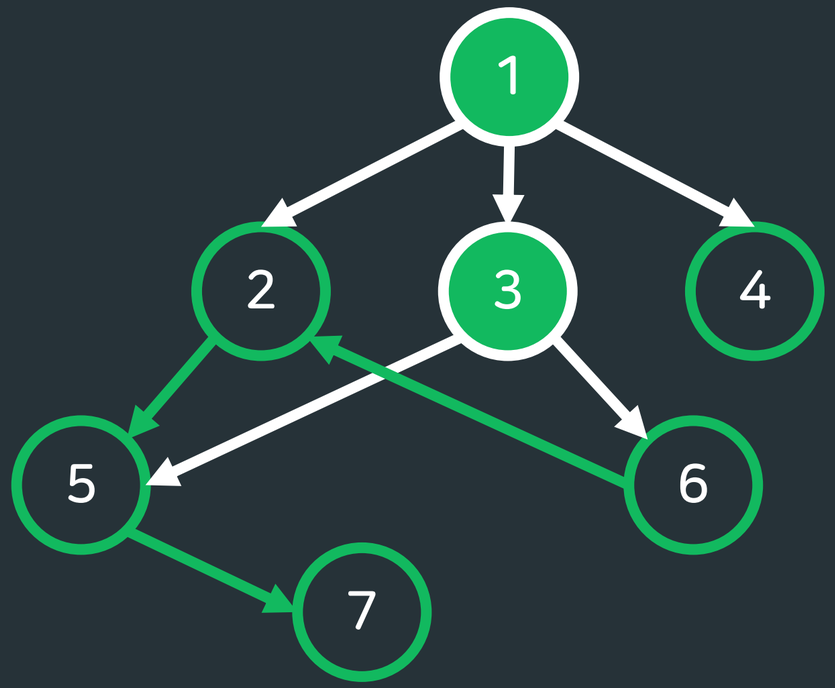
Indegree

1	2	3	4	5	6	7
0	1	0	0	2	1	1

queue



위상 정렬 과정



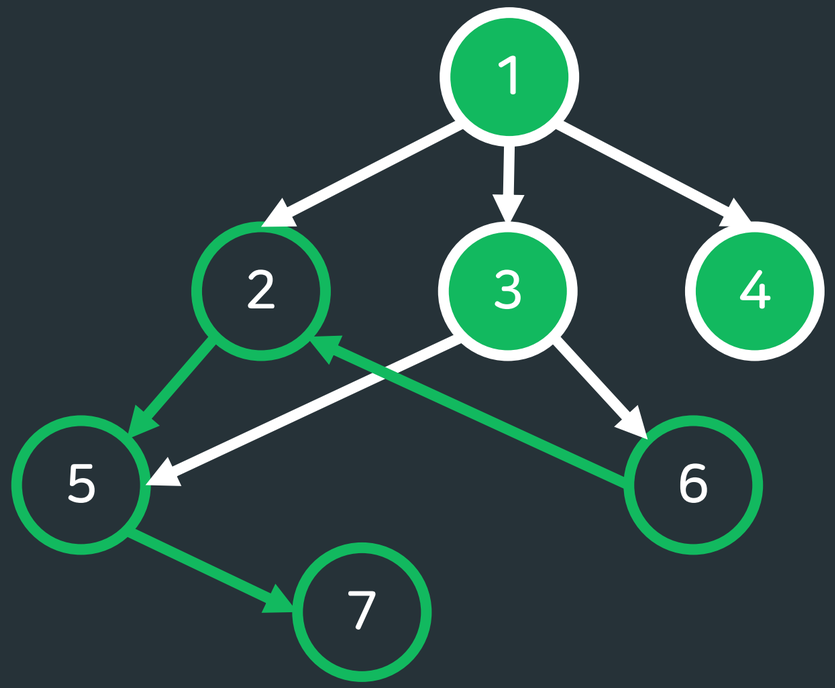
Indegree

1	2	3	4	5	6	7
0	1	0	0	1	0	1

queue

1	3					

위상 정렬 과정



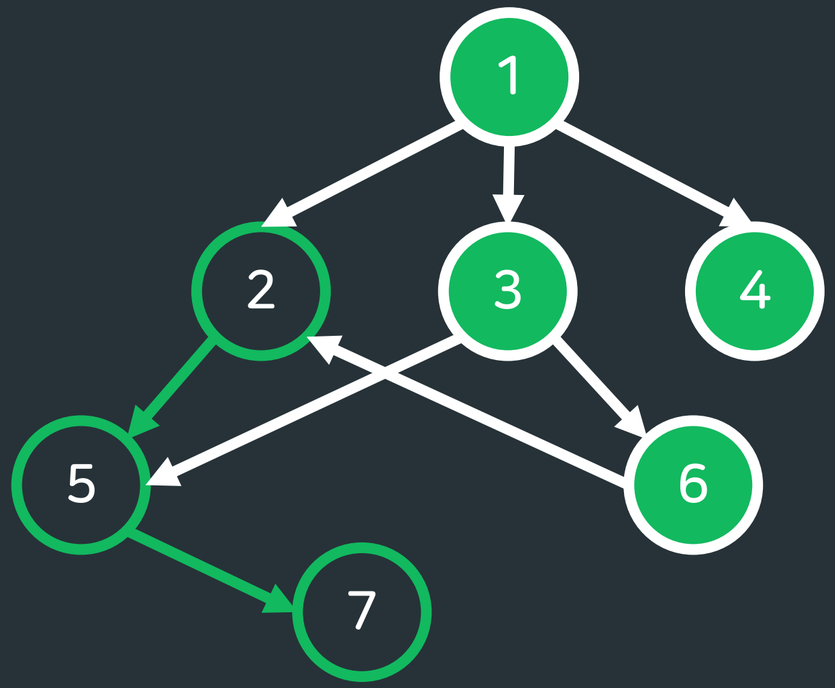
Indegree

1	2	3	4	5	6	7
0	1	0	0	1	0	1

queue

1	3	4				

위상 정렬 과정



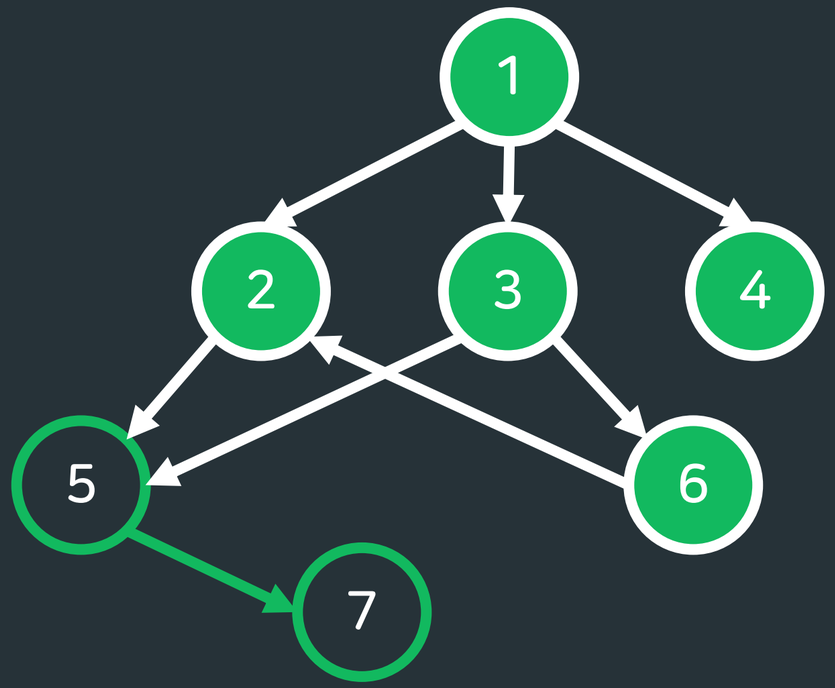
Indegree

1	2	3	4	5	6	7
0	0	0	0	1	0	1

queue

1	3	4	6			

위상 정렬 과정



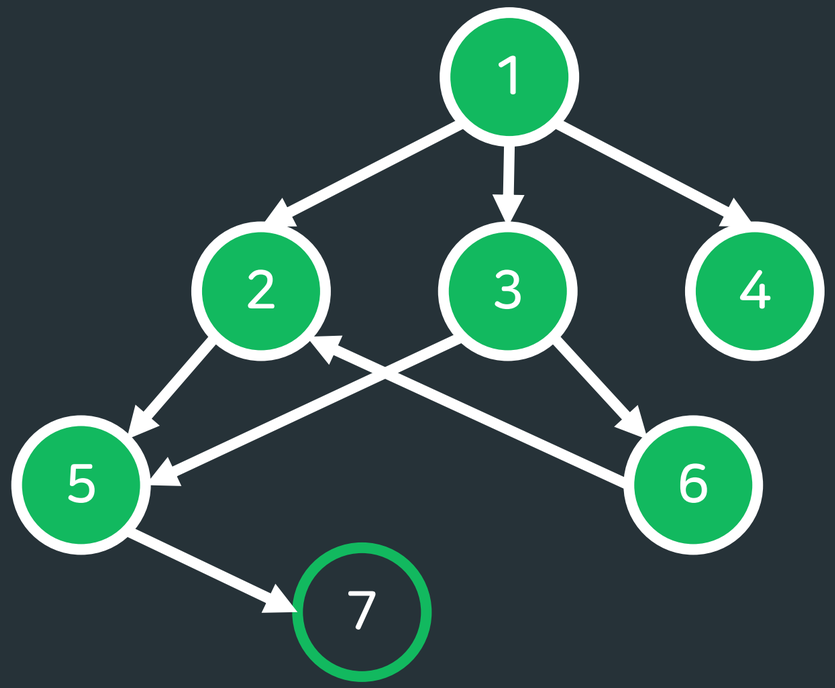
Indegree

1	2	3	4	5	6	7
0	0	0	0	0	0	1

queue

1	3	4	6	2		

위상 정렬 과정



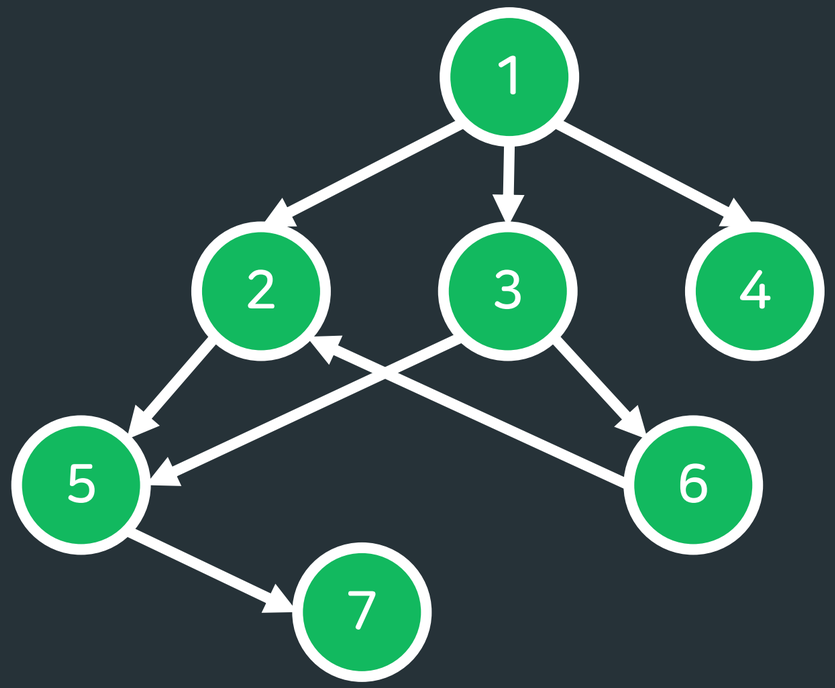
Indegree

1	2	3	4	5	6	7
0	0	0	0	0	0	0

queue

1	3	4	6	2	5	

위상 정렬 과정



Indegree

1	2	3	4	5	6	7
0	0	0	0	0	0	0

queue

1	3	4	6	2	5	7

사실 DFS로도 구현할 수 있어요

#알튜비튜

위상 정렬과 DFS

- DAG에서 DFS를 실행하면서 한 정점의 탐색이 끝났을 때를 저장하면, 이 순서의 역순이 위상 정렬한 결과와 같음
- DFS가 끝났다는 것은 더 이상 깊게 파고들 정점이 없다는 것을 의미하기 때문
- 스택을 이용하여 DFS가 끝나고 리턴 시 현재 정점을 삽입해줌 (DFS를 다시 시작하기 전까지)
- 증명은 수학적 귀류법을 사용해야 하니까 직관적으로 이해해요..😂

사실 DFS로도 구현할 수 있어요

#알튜비튜

```
//위상정렬
void dfs(int node) {
    if (visited[node]) //이미 방문한 정점이라면
        return;

    visited[node] = true; //방문 체크
    for (int i = 0; i < graph[node].size(); i++)
        dfs(graph[node][i]);
    st.push(node); //더 이상 탐색할 정점이 없다면 현재 정점 삽입
}
```

2252번 : 줄 세우기 - Gold 3

문제

- 일부 학생들의 키를 비교한 결과가 주어졌을 때, 키 순서대로 줄을 세우자

제한 사항

- 학생의 수 N은 1 <= N <= 32,000
- 비교 관계 수 M은 1 <= M <= 100,000
- 학생의 번호는 1 ~ N

예제

#알튜비튜

예제 입력 1

```
3 2
1 3
2 3
```

예제 출력 1

```
1 2 3
```

예제 입력 2

```
4 2
4 2
3 1
```

예제 출력 2

```
4 2 3 1
```

/<> 1005번 : ACM Craft - Gold 3

문제

- 매 게임시작 시 건물을 짓는 순서가 주어진다
- 건물은 각각 건설 시간이 존재한다
- 건물을 짓는 순서에 따라 전 건물이 다 지어진 후에 다음 건물을 지을 수 있다
- 건물은 동시에 지을 수 있다
- 특정 건물을 가장 빨리 지을 때까지 걸리는 최소 시간을 알아내자

제한 사항

- 건물의 개수 N은 2 <= N <= 1000
- 건물 순서 규칙 개수 K는 1 <= K <= 100,000
- 건물 건설 시간 Di는 0 <= Di <= 100,000 (Di는 정수)
- 건물 순서 X, Y, 지어야 할 특정 건물 W는 1 <= X, Y, W <= N

예제

#알튜비튜

예제 입력 1

```
4 4
10 1 100 10
1 2
1 3
2 4
3 4
4
```

예제 출력 1

```
120
```

예제 입력 1

```
8 8
10 20 1 5 8 7 1 43
1 2
1 3
2 4
2 5
3 6
5 7
6 7
7 8
7
```

예제 출력 1

```
39
```

* 테스트케이스 여러 개로 같이 주어지는 데 보기 편하게 임의로 나눴습니다.

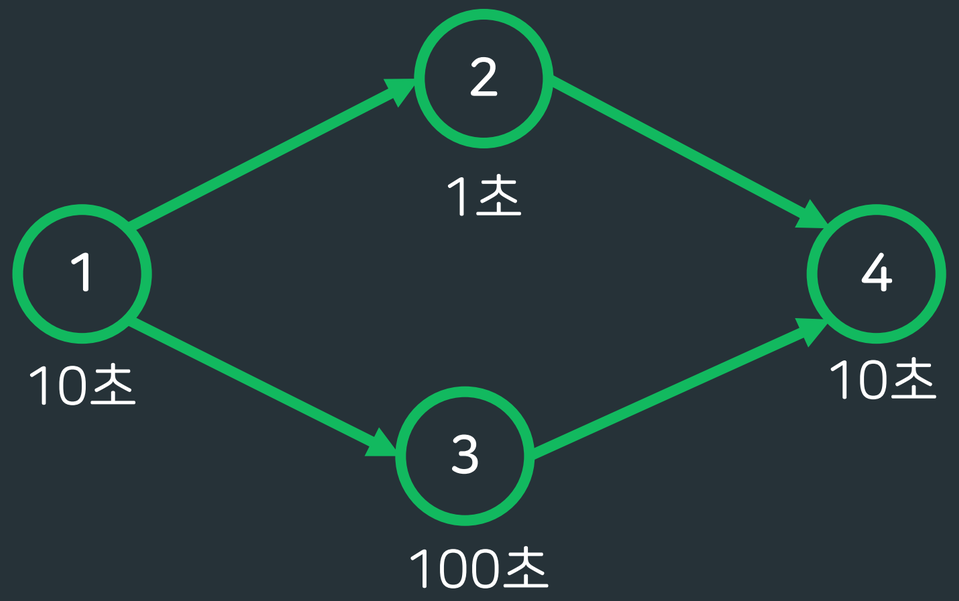
몰래 보세요

#알튜비튜

Hint

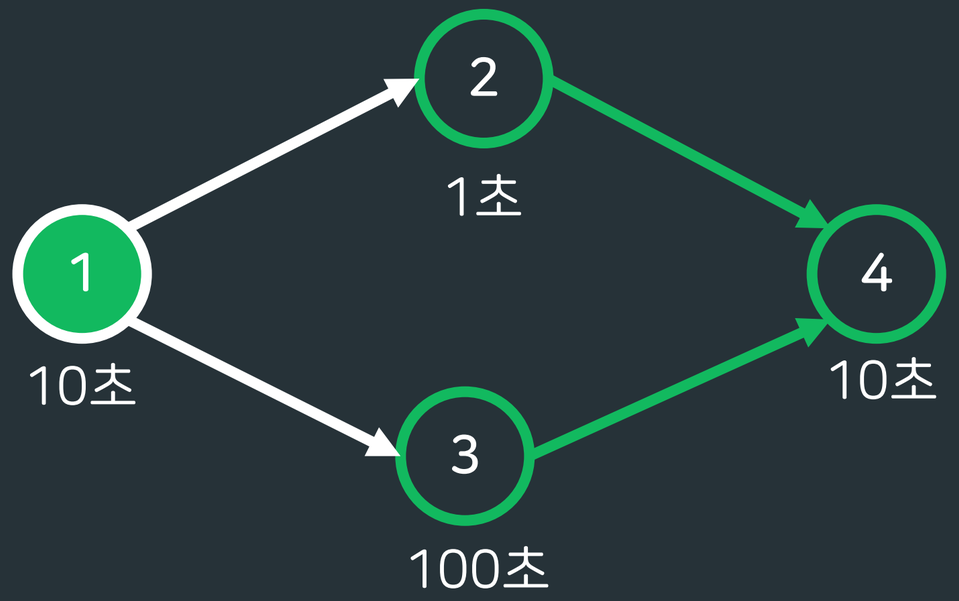
1. 선후 관계가 주어지고, 모든 건물의 차례를 알아야 하는 문제네요!
2. 건물을 동시에 지을 수도 있으니, 동시에 지을 수 있는 건물은 최대한 동시에 지어야겠네요! 그렇다면 현재 지을 수 있는 건물을 모두 동시에 짓는다 할 때, 드는 건설 시간은 어떻게 될까요?

ACM Craft



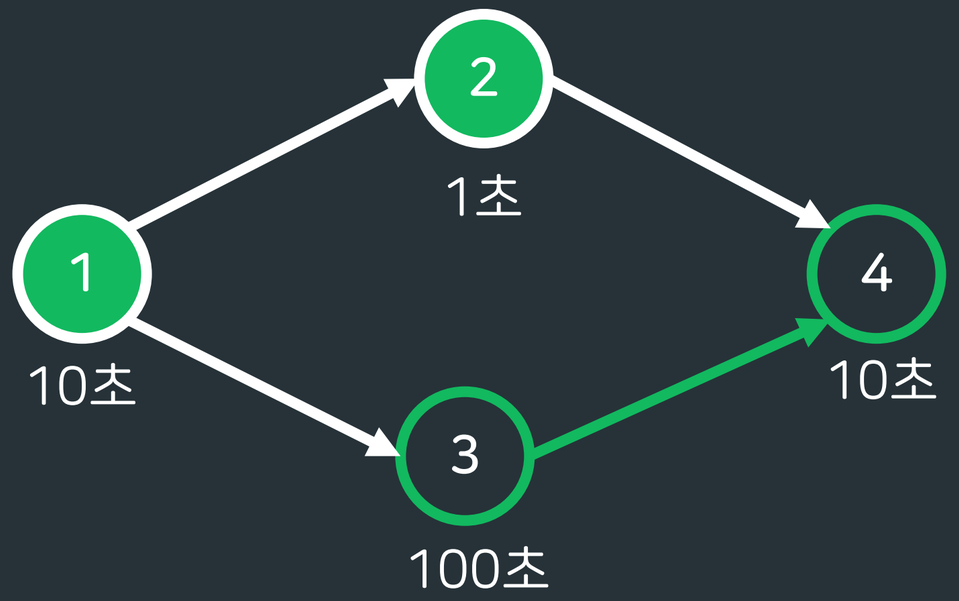
dp	0	0	0	0

ACM Craft



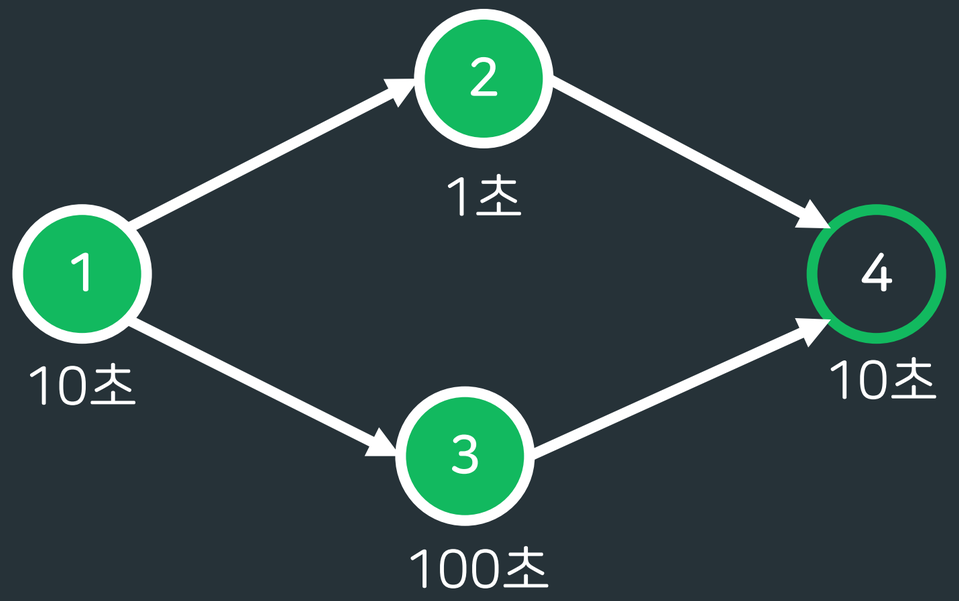
dp	10	0	0	0

ACM Craft



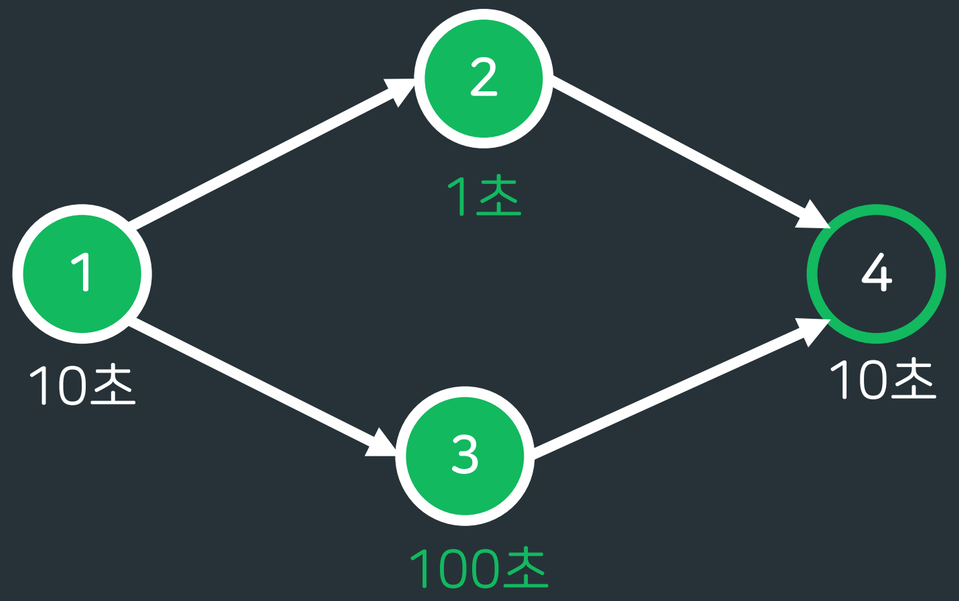
dp	10	11	0	0

ACM Craft



dp	10	11	110	0

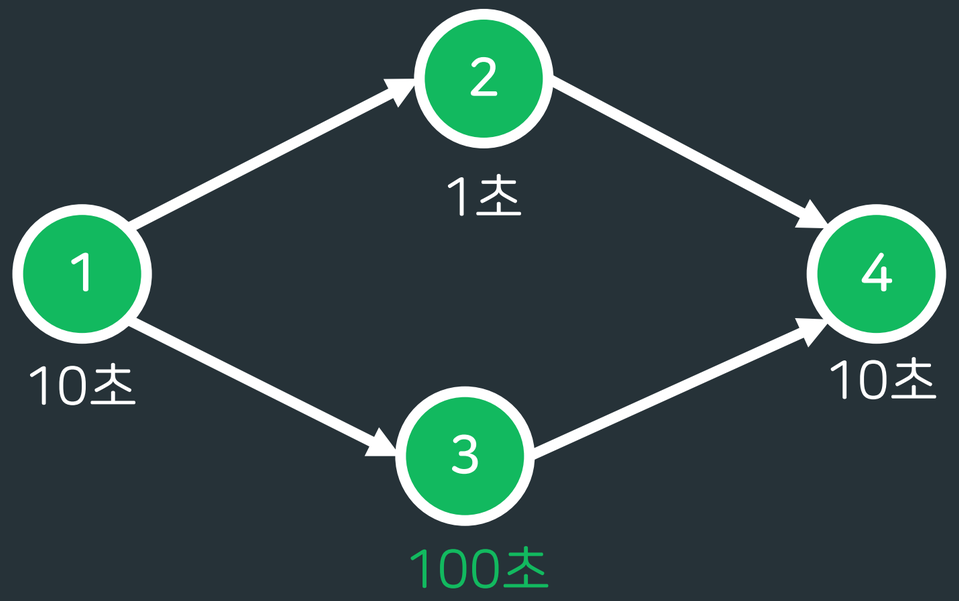
ACM Craft



dp	10	11	110	0

2번과 3번은 동시에 지을 수 있다!

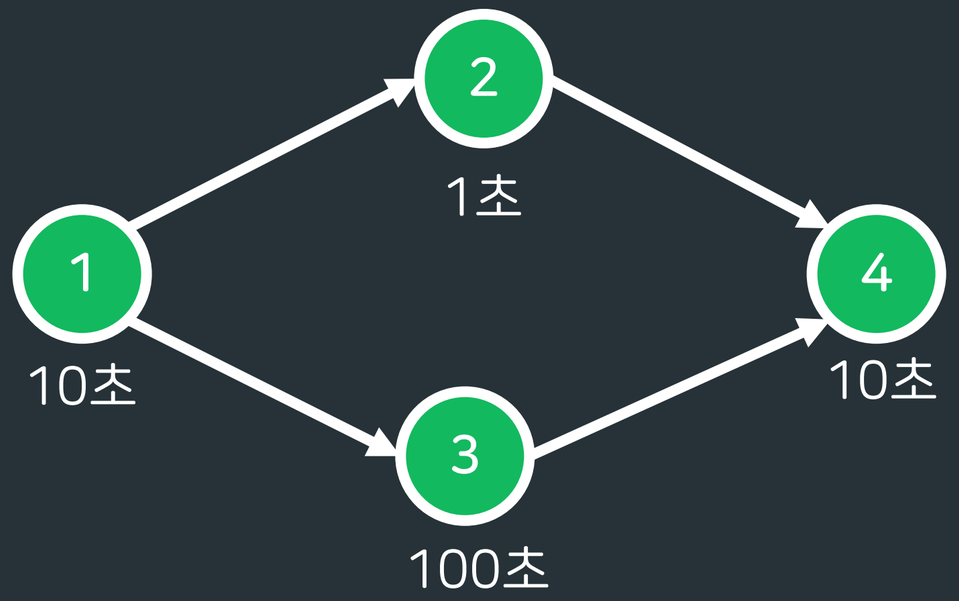
ACM Craft



dp	10	11	110	120

2번과 3번은 동시에 지을 수 있다!
-> 건설 시간이 더 큰 걸 따라감

ACM Craft



dp	10	11	110	120

따라서 현재 지으려는 건물의 총 시간은, 이어진 전 정점 중
가장 시간이 큰 걸 선택해서 현재 건설 시간과 더하면 된다!

마무리

#알튜비튜

정리

- 일상 속에서 일의 순서를 정해야할 때 사용하는 위상 정렬
- 그래프의 선후 관계가 주어졌을 때, 모든 정점을 차례로 나열함
- 선후 관계가 존재하는 정점들만 지키면 되기 때문에, 여러 가지 결과가 나올 수 있음
- 사이클이 존재하면 위상 정렬을 할 수 없음
- 진입차수를 저장한 배열과, 진입차수가 0인 정점을 관리할 큐를 사용하여 구현
- DFS를 활용해서도 구현 가능

이것도 알아보세요

- 오늘은 배열과 큐를 활용한 위상정렬 구현만 다루었습니다. DFS를 활용한 풀이도 직접 디버깅해볼까요!

과제

필수

- 2021 KAKAO BLIND RECRUITMENT: 카드 짝 맞추기 – Level 3
- 3085번 : 사탕 게임 – Silver 3

3문제 이상 선택

- 2623번 : 음악프로그램 – Gold 2
- 2637번 : 장난감 조립 – Gold 2
- 23059번 : 리그 오브 레게노 – Gold 2
- 14907번 : 프로젝트 스케줄링 – Gold 2
- 2020 카카오 인턴십: 경주로 건설 – Level 3
- 2638번 : 치즈 – Gold 4

한 학기동안 정말 수고 많으셨습니다! 😊👍
잘 따라와주셔서 감사합니다!!!!!