Chapter. 02

알고리즘

위상 정렬
Topological Sort

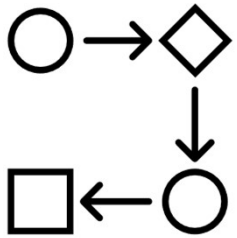
FAST CAMPUS
ONLINE
알고리즘 공채 대비반I

강사. 류호석

Life Changing Education

Chapter. 02

알고리즘
위상 정렬(Topological Sort)



FAST CAMPUS
ONLINE
류호석 강사.

I 위상 정렬(Topological Sort)란?

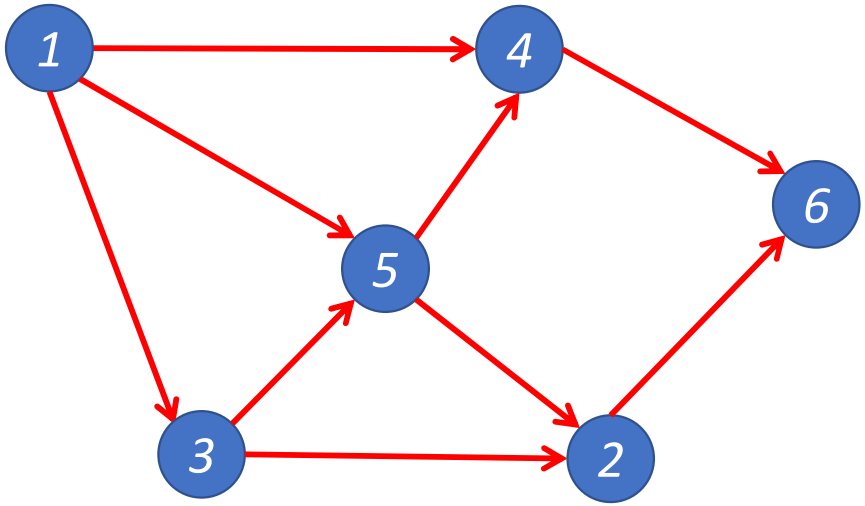
위상 정렬 = “위상” + “정렬”

I 위상 정렬(Topological Sort)란?

위상 정렬 = “위상” + “정렬”

Directed Acyclic Graph (DAG)

- Directed := 간선에 방향성이 있다.
  - ⓐ->ⓑ := ⓐ에서 ⓑ로!
- Acyclic := Cyclic이 없다.
- Graph := 정점(V) + 간선(E)



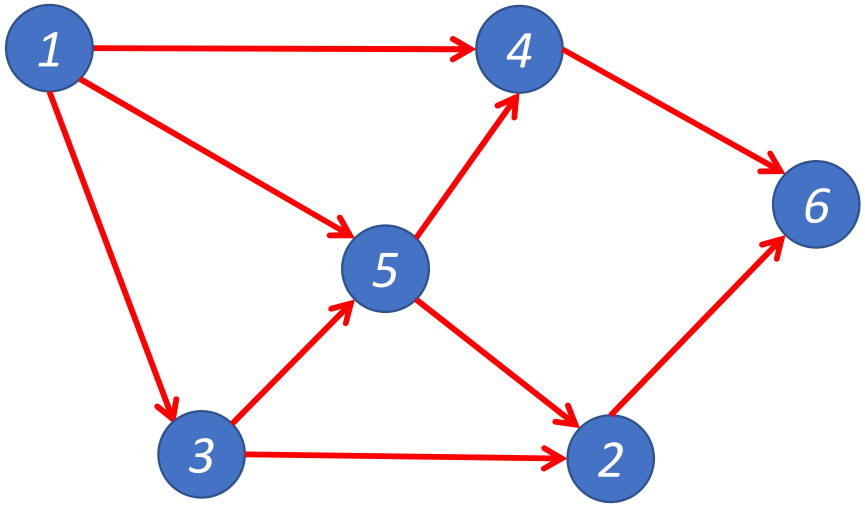
I 위상 정렬(Topological Sort)란?

위상 정렬 = “위상” + “정렬”

차수(Degree)란?

방향성(Directed) Graph에서는?

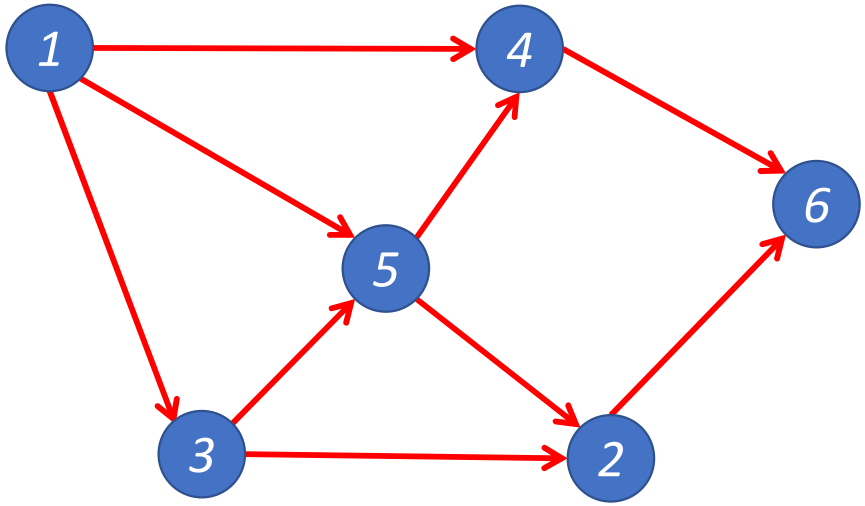
Indegree / Outdegree 로 구별!



I 위상 정렬(Topological Sort)란?

위상 정렬 = “위상” + “정렬”

Directed Acyclic Graph (DAG) → 정점들을 위상(位相)에 맞춰 정렬

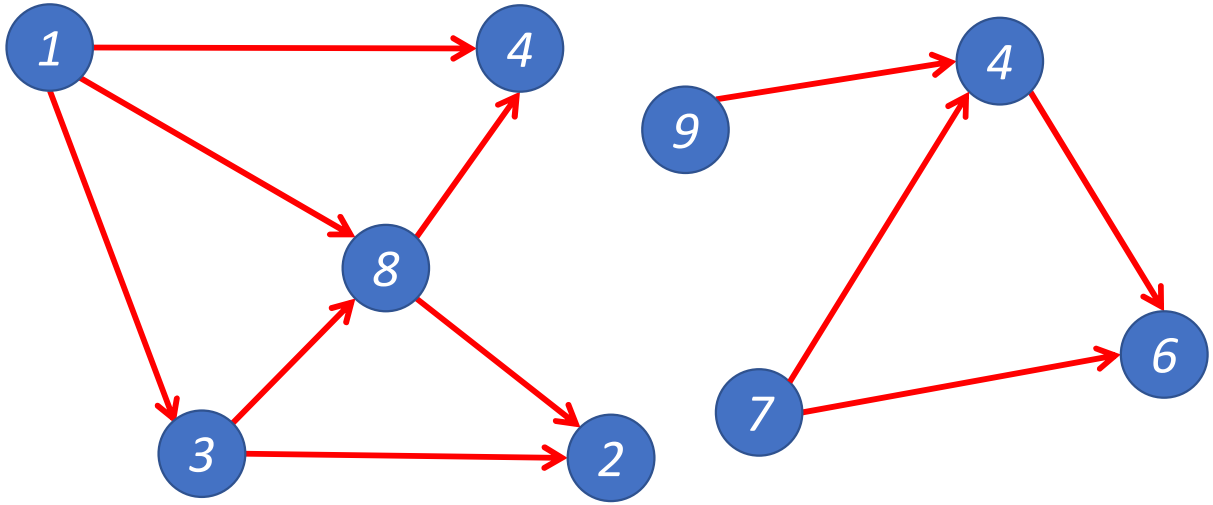


→

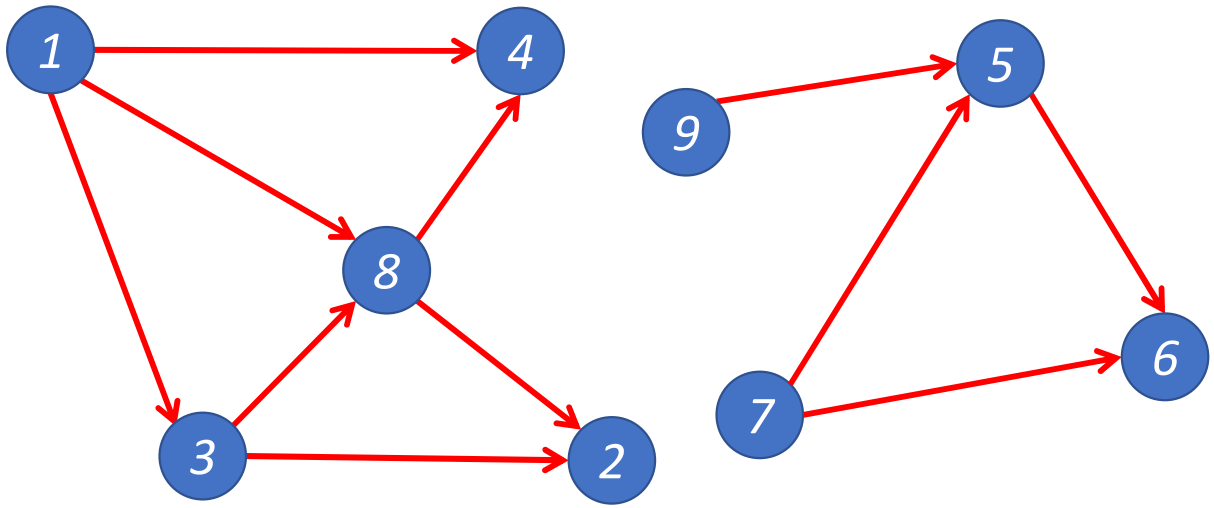
1	3	5	4	2	6

I 위상 정렬; **하나씩 정렬하기!**

아이디어: “제일 먼저 올 수 있는 정점은?”



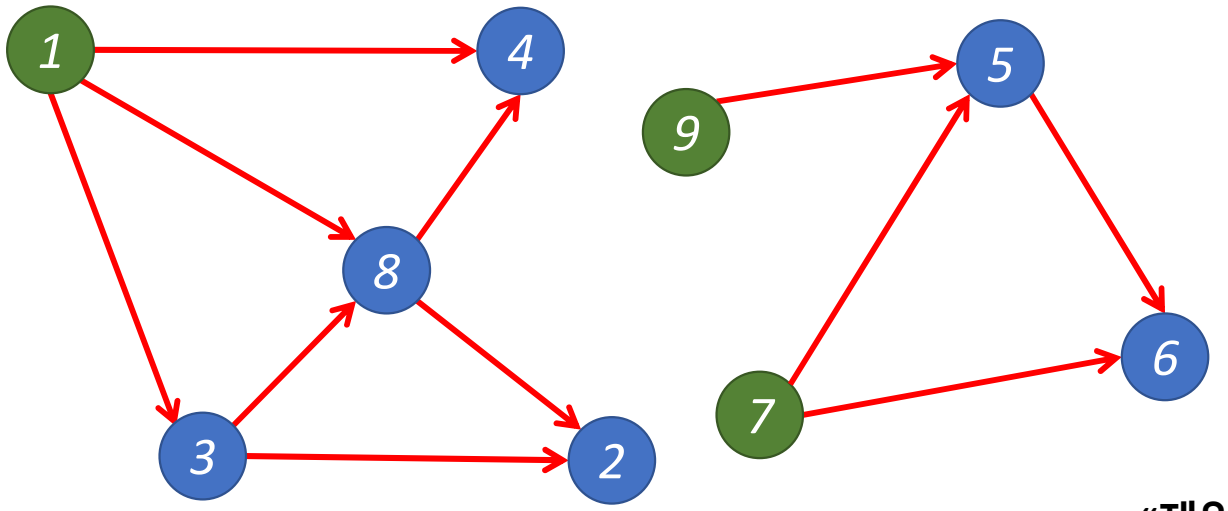
I 위상 정렬; **하나씩 정렬하기!**



Q: **"제일 먼저"** 올 수 있는 정점은?

A: **"들어오는 간선"**이 없는 정점!

I 위상 정렬; **하나씩 정렬하기!**



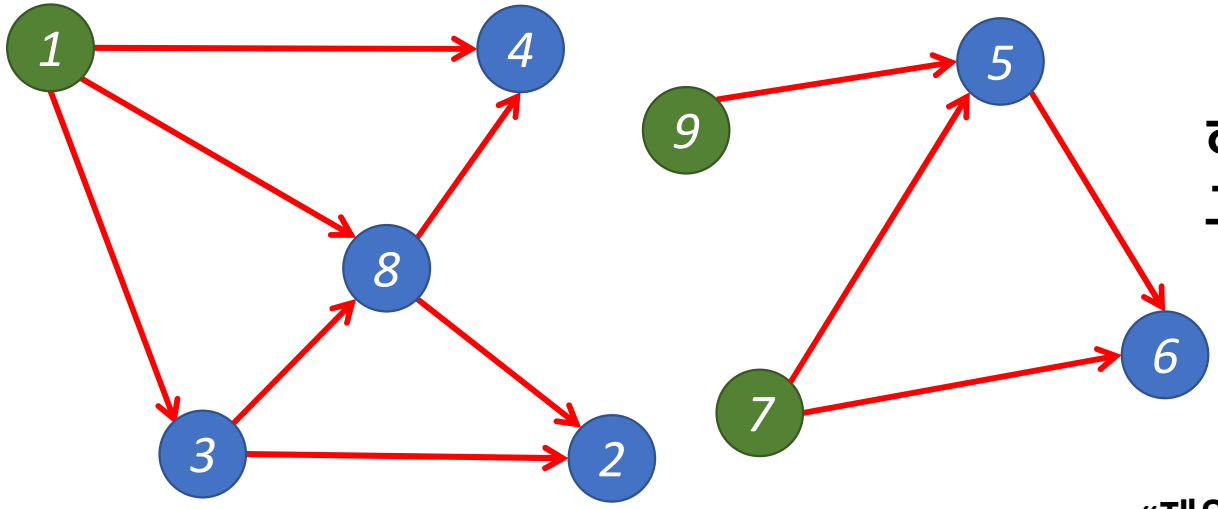
Q: **"제일 먼저"** 올 수 있는 정점은?

A: **"들어오는 간선"**이 없는 정점!

"제일 먼저" 올 수 있는 정점들

9	**1**	**7**			

I 위상 정렬; **하나씩 정렬하기!**



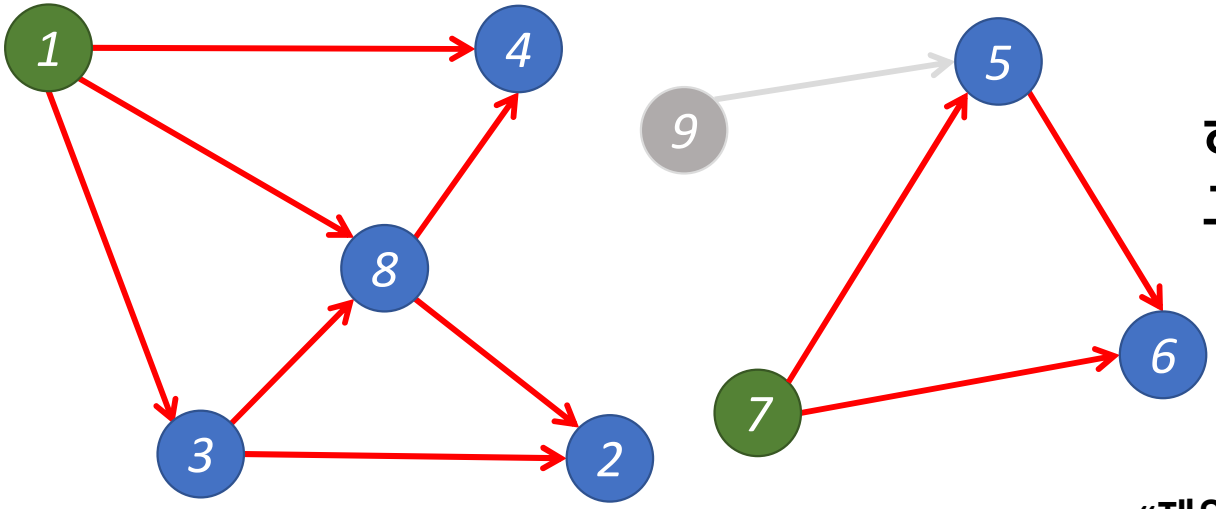
하나씩 정렬시켜 버리면서
그래프에서 삭제하자!

"제일 먼저" 올 수 있는 정점들

9	1	7			

위상 정렬 결과

I 위상 정렬; 하나씩 정렬하기!



하나씩 정렬시켜 버리면서 그래프에서 삭제하자!

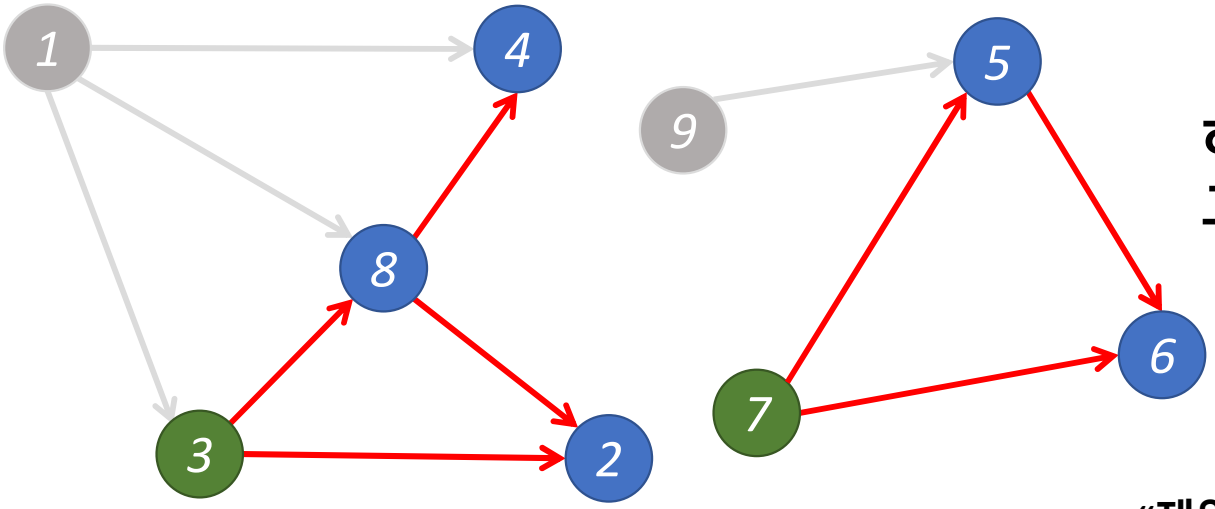
"제일 먼저" 올 수 있는 정점들

1	7				

위상 정렬 결과

9								

I 위상 정렬; 하나씩 정렬하기!



하나씩 정렬시켜 버리면서 그래프에서 삭제하자!

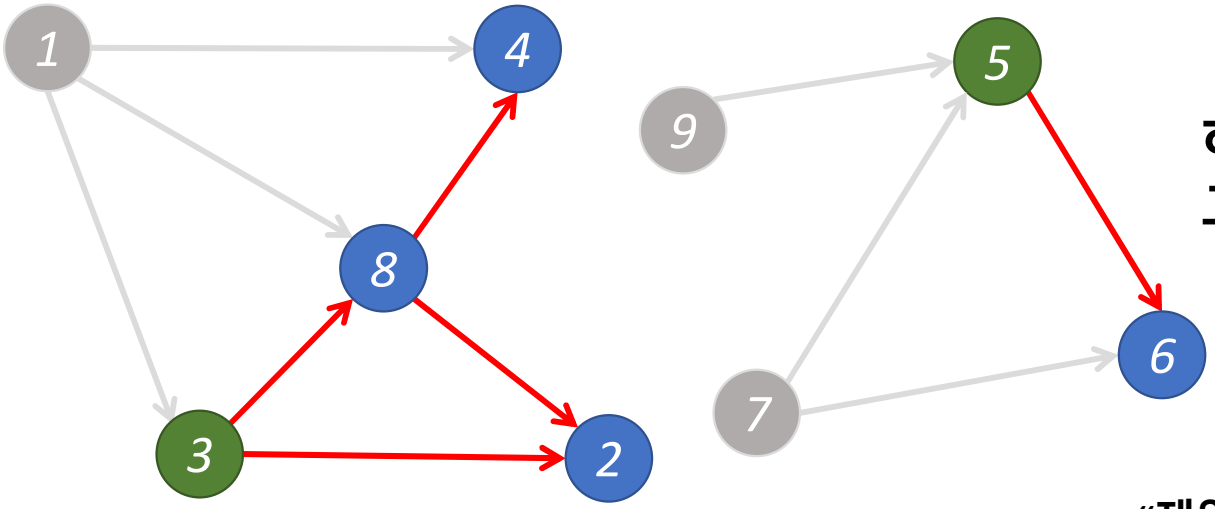
"제일 먼저" 올 수 있는 정점들

7	3				

위상 정렬 결과

9	1							

I 위상 정렬; 하나씩 정렬하기!



하나씩 정렬시켜 버리면서
그래프에서 삭제하자!

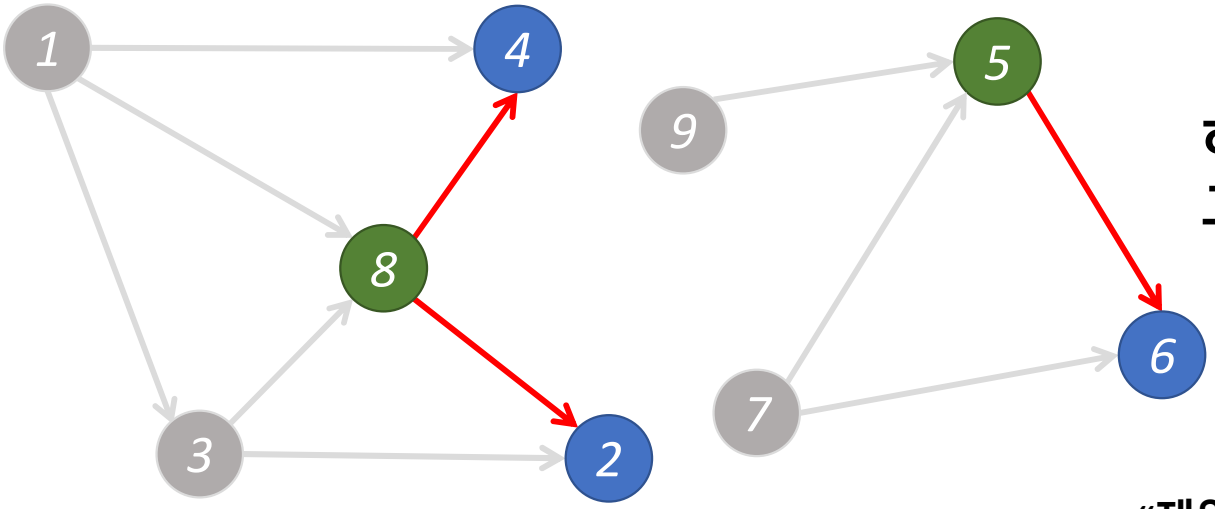
"제일 먼저" 올 수 있는 정점들

3	5				

위상 정렬 결과

9	1	7						

I 위상 정렬; 하나씩 정렬하기!



하나씩 정렬시켜 버리면서
그래프에서 삭제하자!

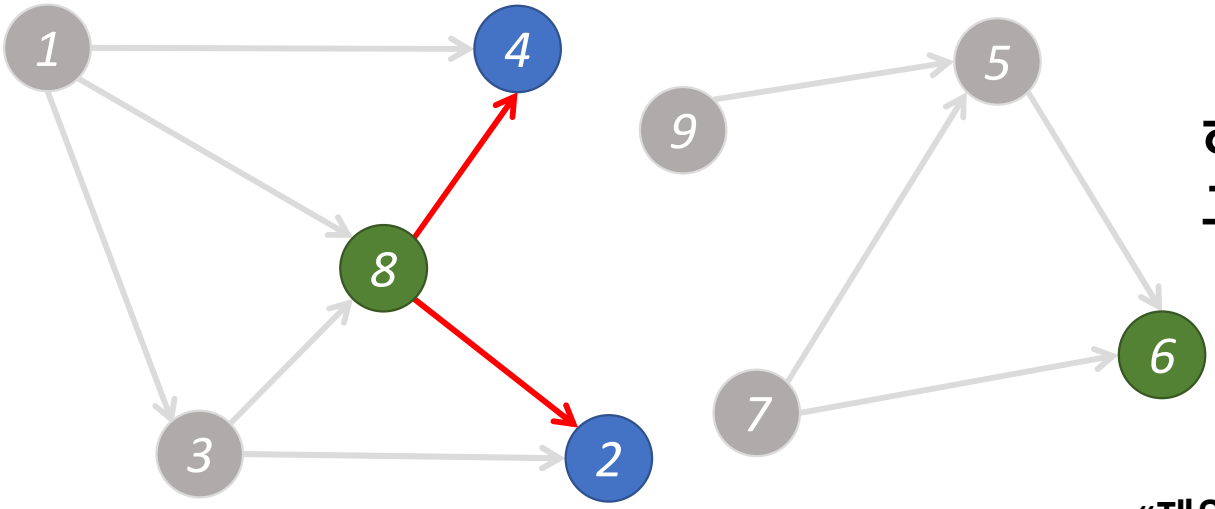
"제일 먼저" 올 수 있는 정점들

5	8				

위상 정렬 결과

9	1	7	3					

I 위상 정렬; 하나씩 정렬하기!



하나씩 정렬시켜 버리면서
그래프에서 삭제하자!

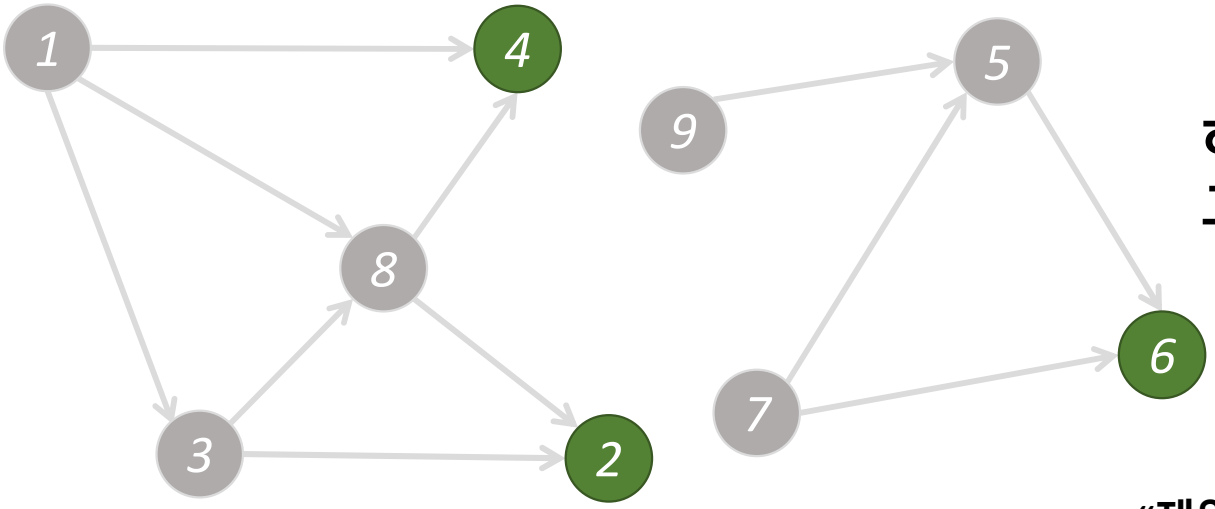
"제일 먼저" 올 수 있는 정점들

8	6				

위상 정렬 결과

9	1	7	3	5				

I 위상 정렬; **하나씩 정렬하기!**



하나씩 정렬시켜 버리면서
그래프에서 삭제하자!

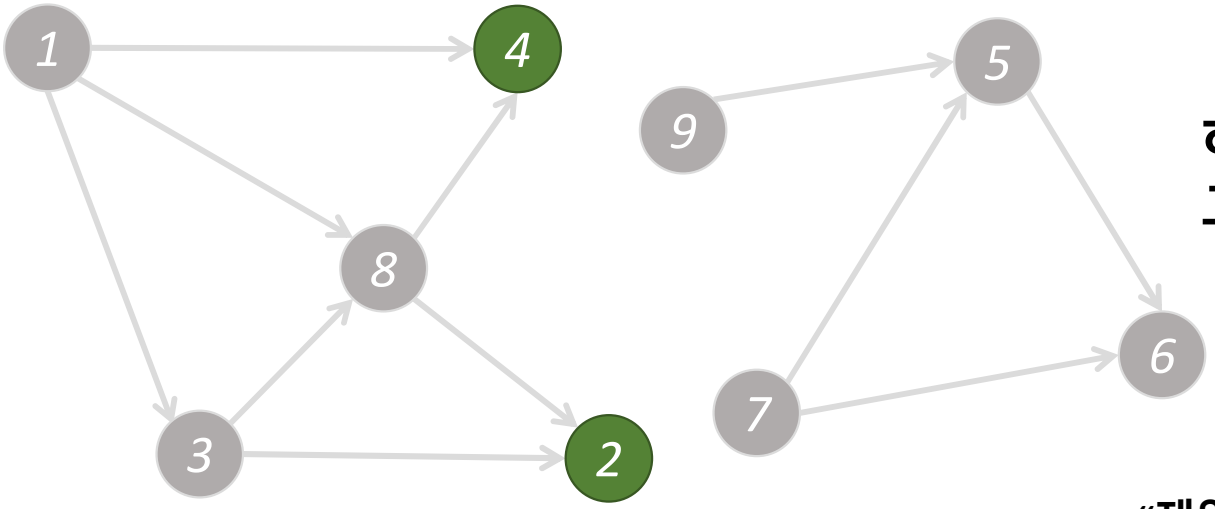
"제일 먼저" 올 수 있는 정점들

6	4	2			

위상 정렬 결과

9	1	7	3	5	8			

I 위상 정렬; 하나씩 정렬하기!



하나씩 정렬시켜 버리면서
그래프에서 삭제하자!

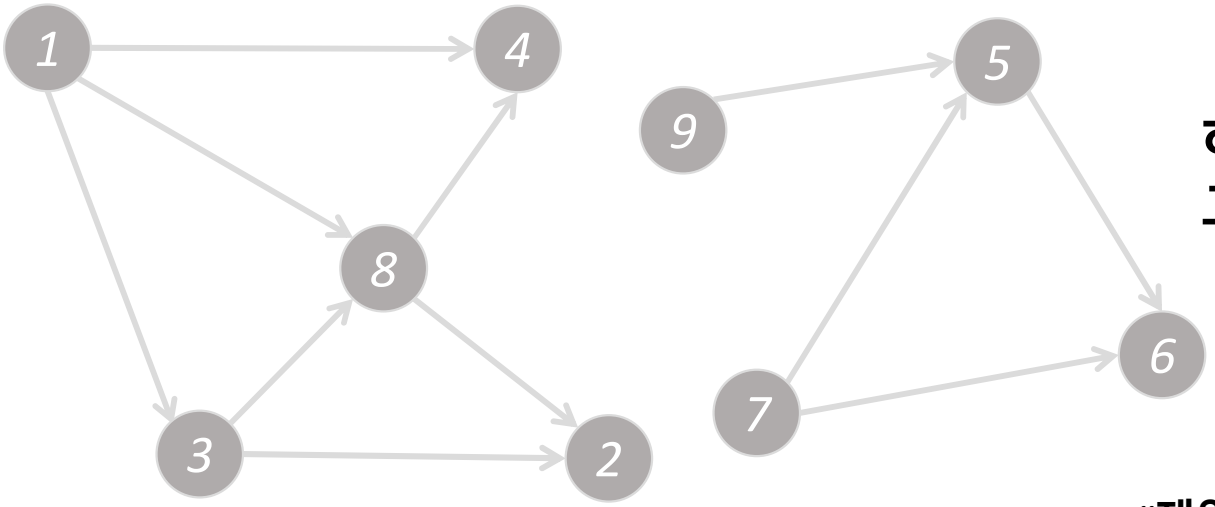
"제일 먼저" 올 수 있는 정점들

4	2				

위상 정렬 결과

9	1	7	3	5	8	6		

I 위상 정렬; **하나씩 정렬하기!**



하나씩 정렬시켜 버리면서
그래프에서 삭제하자!

"제일 먼저" 올 수 있는 정점들

위상 정렬 결과

9	**1**	**7**	**3**	**5**	**8**	**6**	**4**	**2**

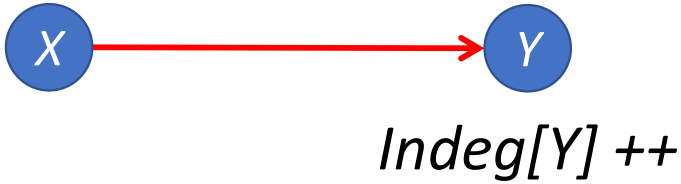
I 위상 정렬; **정리**

<정리>

1. 정점들의 Indegree, *Indeg[1…N]* 계산하기
2. 들어오는 간선이 0개인 (*Indeg[i]* == 0) 정점들을 찾아서 자료구조 ***D*** 에 넣기
3. ***D*** 가 빌 때까지
    1. ***D*** 에서 원소 *x*를 꺼내서 **"정렬"**시키기
    2. Graph 에서 정점 *x* **"제거"**하기
    3. **"새롭게** 정렬 가능한 점"을 찾아서 ***D***에 넣기

I 위상 정렬; **구현 방법**

1. 정점들의 Indegree, *Indeg[1…N]* 계산하기



➔ $O(|E|)$

I 위상 정렬; **구현 방법**

2, 3 ➔ 자료구조 ***D*** 에 필요한 연산은?

- 원소를 추가하기
- 원소를 꺼내기

두 연산을 “빠르게” 해주는 것은? **Queue**, Stack, Linked List, etc.

I 위상 정렬; **구현 방법**

2. 들어오는 간선이 0개인 (*Indeg[i]* == 0) 정점들을 찾아서 자료구조 ***D*** 에 넣기

```
Deque<Integer> queue = new LinkedList<>();
// 제일 앞에 "정렬될 수 있는" 정점 찾기
for (int i = 1; i <= N; i++)
    if (indeg[i] == 0)
        queue.add(i);
```

➔ $O(|V|)$

I 위상 정렬; **구현 방법**

3. ***D*** 가 빌 때까지 원소 *X*를 꺼내서 **"정렬"**시키고 **"제거"**하고

"새롭게 정렬 가능한 점"을 찾아서 ***D***에 넣기

```
while (!queue.isEmpty()) {
    int x = queue.poll();
    sb.append(x).append(' ');
    for (int y : adj[x]) {
        indeg[y]--;
    }
    for (int i = 1; i <= N; i++)
        if (i 가 새롭게 indeg[i]==0 이라면)
            queue.add(i);
}
```

➔ $O(|V|^2)$

I 위상 정렬; **구현 방법**

3. ***D*** 가 빌 때까지 원소 *X*를 꺼내서 **"정렬"**시키고 **"제거"**하고

"새롭게 정렬 가능한 점"을 찾아서 ***D***에 넣기

```
while (!queue.isEmpty()) {
    int x = queue.poll();
    sb.append(x).append(' ');
    for (int y : adj[x]) {
        indeg[y]--;
        if (indeg[y] == 0) queue.add(y);
    }
}
```

➔ $O(|E|)$

"정점 X랑 연결된 점들은?" ➔ 인접 행렬보다 인접 리스트!!

I 위상 정렬; **정리**

<정리>

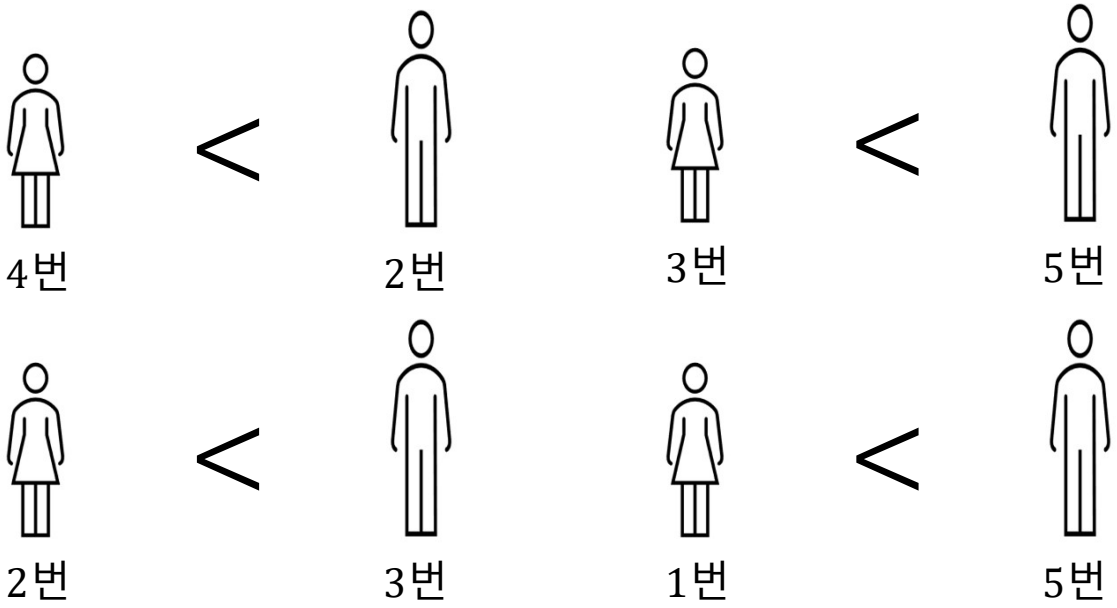
1. 정점들의 Indegree, *Indeg[1…N]* 계산하기
2. 들어오는 간선이 0개인 (*Indeg[i]* == 0) 정점들을 찾아서 자료구조 ***D*** 에 넣기
3. ***D*** 가 빌 때까지
   1. ***D*** 에서 원소 *x*를 꺼내서 **"정렬"**시키기
   2. Graph 에서 정점 *x* **"제거"**하기
   3. **"새롭게** 정렬 가능한 점"을 찾아서 ***D***에 넣기

➔ $O(|V| + |E|)$

BOJ 2252 – 줄 세우기

난이도: 3

$1 \leq$ 학생 수, $N \leq 32{,}000$
$1 \leq$ 키 관계 $\leq 100{,}000$



정렬 결과:

4	2	3	1	5

I 접근 – Graph 만들어 보기

학생들 간에 위상 관계가 주어지고, 이에 맞게 줄을 세운다.

Graph를 정의해보고 위상 정렬을 통해 문제를 해결해보자!

정점(V) := i번 학생이 곧 i번 정점

간선(E) := x번 학생이 y번 학생보다 먼저 서야 한다 ➔ 

I 시간, 공간 복잡도 계산하기

Graph를 만들고 나면 위상 정렬(Topological Sort)

➔ 인접 리스트를 쓴다면 O(V + E)

I 구현

```
static void input() {
    // Adjacent List 생성 및 indegree 계산하기
    /* TODO */
}

static void pro() {
    Deque<Integer> queue = new LinkedList<>();
    // 제일 앞에 "정렬될 수 있는" 정점 찾기
    /* TODO */

    // 정렬될 수 있는 정점이 있다면?
    // 1. 정렬 결과에 추가하기
    // 2. 정점과 연결된 간선 제거하기
    // 3. 새롭에 "정렬 될 수 있는" 정점
    /* TODO */
}
```

I 연습 문제

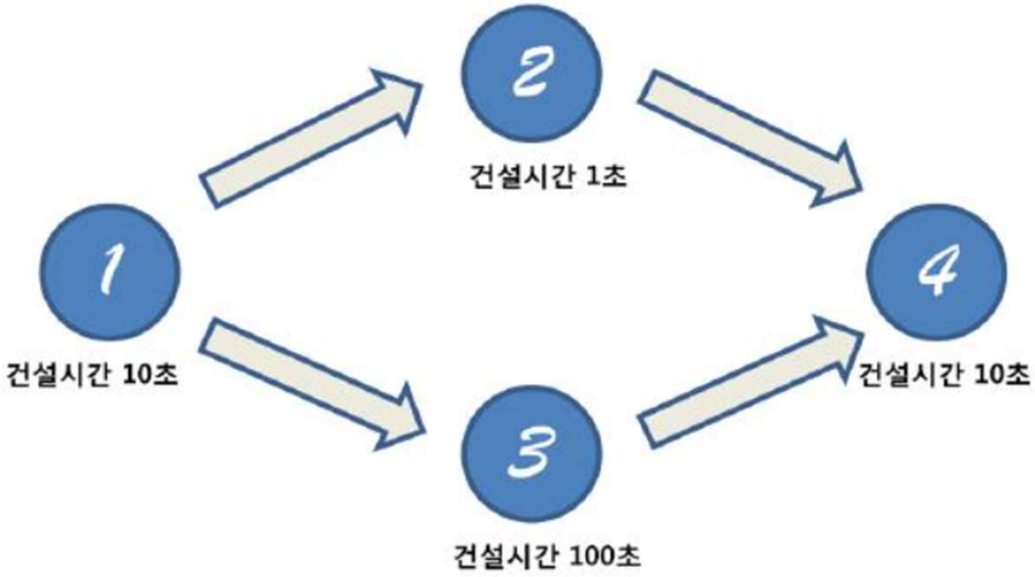
- BOJ 2623 – 음악 프로그램
- BOJ 9470 – Strahler 순서
- BOJ 14676 – 영우는 사기꾼?

이외의 추천 문제가 추가되면 Github 자료에 코드 업로드

I BOJ 1005 – ACM Craft

난이도: 3

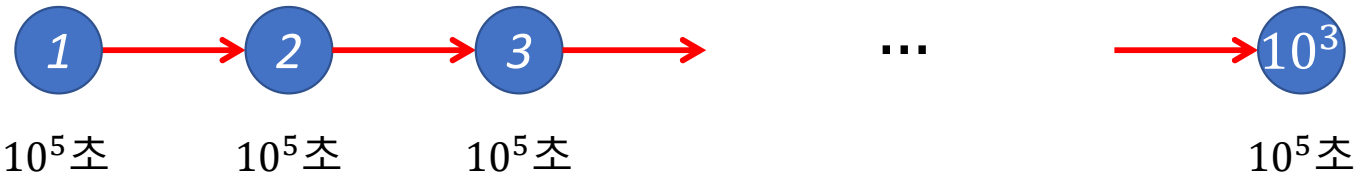
$1 \leq$ 건물의 개수, $N \leq 1{,}000$
$1 \leq$ 건설 규칙 $\leq 100{,}000$
$1 \leq$ 건물 건설 시간 $\leq 100{,}000$



정답: 120

I 접근 – 정답의 최대치

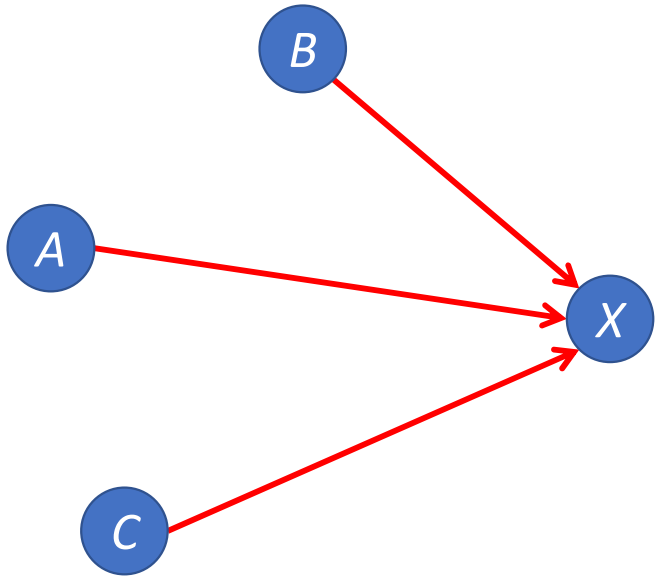
가장 오랜 시간이 걸리려면?



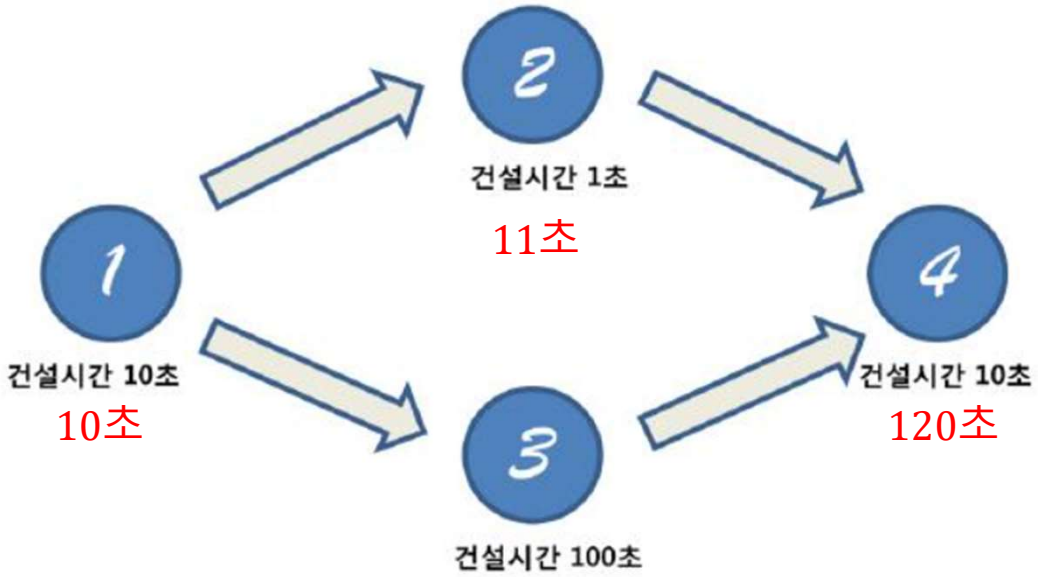
$$10^3 \times 10^5 = 10^8$$

Integer 범위로 충분하다!

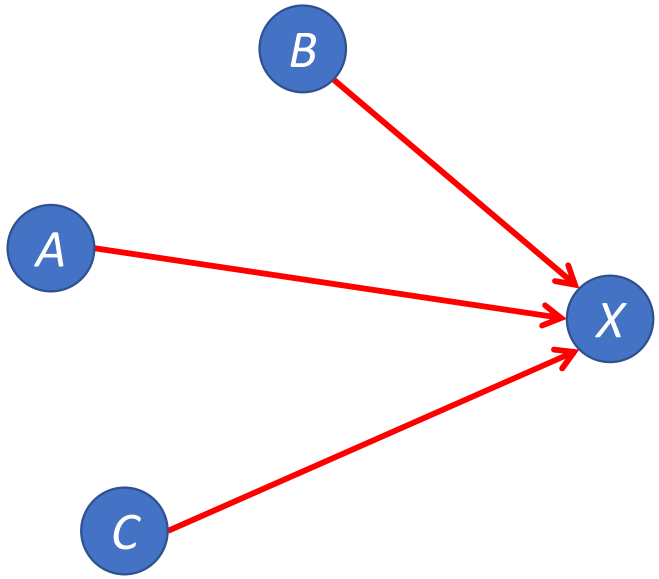
I 접근 – 각 건물을 완성하는 시간은?



$$T_{done}[X] = \max\left(T_{done}[X\text{의 선행 작업}]\right) + T[X]$$



I 접근 – 각 건물을 완성하는 시간은?



$$T_{done}[X] = \max\left(T_{done}[X\text{의 선행 작업}]\right) + T[X]$$

전제 조건

$T_{done}[X]$ 를 계산하기 위해서는

X의 선행 작업들에 대해서 T_{done}이 ***먼저*** 계산되어야 한다!

➔ T_{done} 배열을 위상 정렬 순서로 계산하면 된다!

I 시간, 공간 복잡도 계산하기

위상 정렬(Topological Sort) 이므로 $O(|V| + |E|)$

I 구현

```
static void input() {
    // Testcase 가 존재하는 문제이므로 "배열 초기화"에 유의하자
    /* TODO */
}

static void pro() {
    Deque<Integer> queue = new LinkedList<>();
    // 제일 앞에 "정렬될 수 있는" 정점 찾기
    /* TODO */

    // 위상 정렬 순서대로 T_done 계산을 함께 해주기
    /* TODO */
}
```

I 연습 문제

- BOJ 1516 – 게임 개발
- BOJ 2056 – 작업
- BOJ 2637 – 장난감 조립

이외의 추천 문제가 추가되면 Github 자료에 코드 업로드