Algorithm

[Algorithm] 36강 : 위상정렬 알고리즘의 정의와 구현

2020. 11. 26. 19:08 수정 삭제 공개

위상정렬

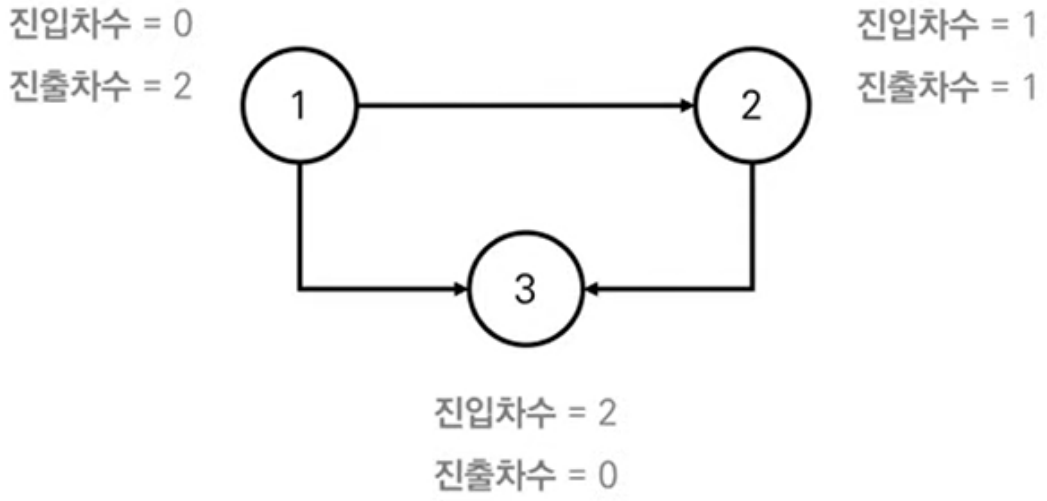
1.1 위상정렬 이란?

- 사이클이 없는 방향 그래프의 모든 노드를 방향성에 거스르지 않도록 순서대로 나열하는 것을 의미

- **예시)** 선수과목을 고려한 학습 순서 설정

1.2 진입 차수 / 진출 차수

- 진입차수(Indegree) : 특정한 노드로 들어오는 간선의 개수
- 진출차수(Outdegree) : 특정한 노드에서 나가는 간선의 개수



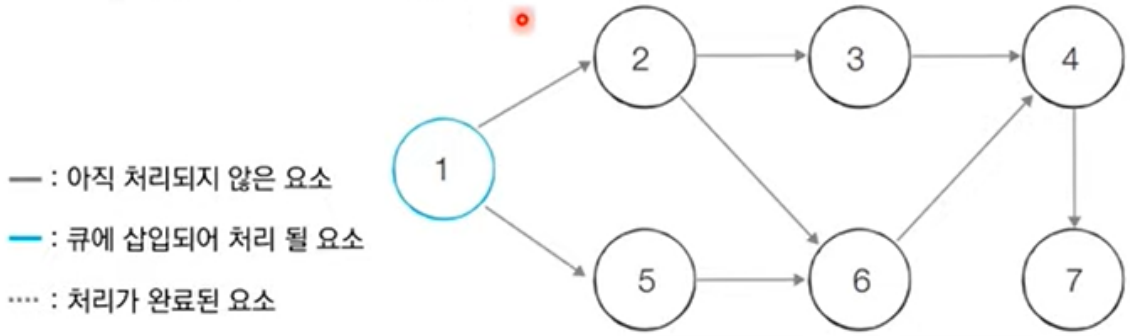
1.3 위상 정렬 알고리즘 동작과정

- 큐를 이용하는 위상 정렬 알고리즘의 동작 과정은 다음과 같다

1. 진입차수가 0인 모든 노드를 큐에 넣는다.
2. 큐가 빌 때까지 다음의 과정을 반복한다
    1. 큐에서 원소를 꺼내 해당노드에서 나가는 간선을 그래프에서 제거한다
    2. 새롭게 진입차수가 0이 된 노드를 큐에 넣는다.

결과적으로 각 노드가 큐에 들어온 순서가 위상 정렬을 수행한 결과와 같다.

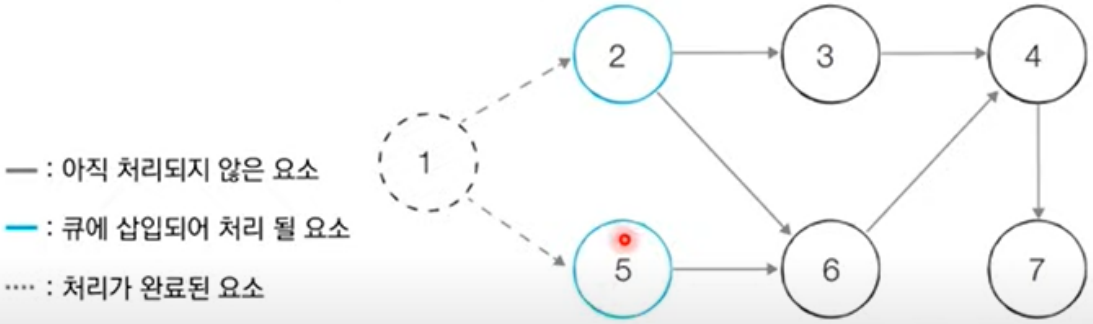
- **[초기 단계]** 초기 단계에서는 **진입차수가 0인 모든 노드**를 큐에 넣습니다.
  - 처음에 **노드 1**이 큐에 삽입됩니다.



노드	1	2	3	4	5	6	7
진입차수	0	1	1	2	1	2	1

큐	노드 1

- **[Step 1]** 큐에서 **노드 1**을 꺼낸 뒤에 **노드 1**에서 나가는 간선을 제거합니다.
  - 새롭게 **진입차수가 0이 된 노드**들을 큐에 삽입합니다.

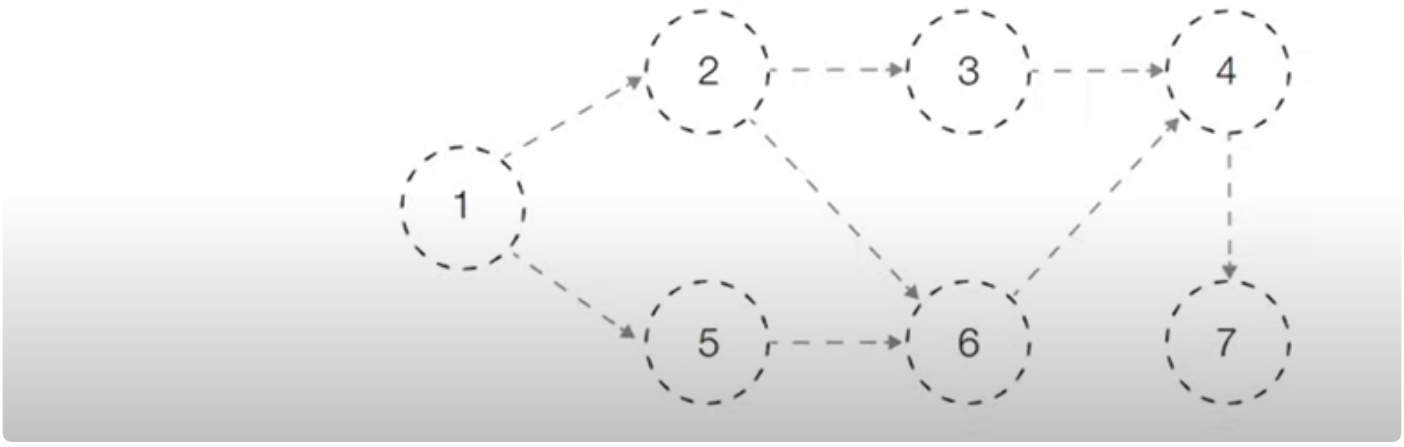


노드	1	2	3	4	5	6	7
진입차수	0	0	1	2	0	2	1

큐	노드 2, 노드 5

이 과정을 계속해서 반복한다. 또한, 같이 0 이되는 노드가 있을경우 올림 차수로 진행된다.

- **[위상 정렬 결과]**
  - 큐에 삽입된 전체 노드 순서: 1 → 2 → 5 → 3 → 6 → 4 → 7



이렇게 반복한 결과는 위와같이 되고 큐에 삽입된 전체 노드의 순서는 위와 같이 된다.

1.4 위상 정렬의 특징

- 위상 정렬은 DAG에 대해서만 수행할 수 있다.
  - **DAG(Direct Acyclic Graph)**: 순환하지 않는 방향 그래프
- 위상정렬에서는 **여러가지 답이 존재**할 수 있다.
  - 한 단계에서 큐에 새롭게 들어가는 원소가 2개 이상인 경우가 있다면 여러 가지 답이 존재한다
- **모든 원소를 방문하기 전에 큐가 빈다면 사이클이 존재**한다고 판단할 수 있다.
  - 사이클에 포함된 원소 주에에서 어떠한 원소도 큐에 들어가지 못한다
- 스택을 활용한 DFS를 이용해 위상 정렬을 수행할 수도 있다.

1.5 위상 정렬 알고리즘 구현

```
from collections import deque

# 노드의 개수와 간선의 개수를 입력받기
v, e = map(int, input().split())

# 모든 노드에 대한 집입차수는 0으로 초기화
indegree = [0] * (v + 1)

# 각 노드에 연결된 간선 정보를 담기위한 연결 리스트 초기화
graph = [[] for i in range(v + 1)]

# 방향 그래프의 모든 간선 정보를 입력 받기
for _ in range(e):
    a, b = map(int, input().split())
    graph[a].append(b)  # 정점 A에서 B로 이동 가능

    # 진입 차수를 1증가
    indegree[b] += 1


# 위상정렬함수
def topology_sort():
    result = []  # 알고리즘 수행 결과를 담을 리스트
    q = deque()  # 큐 기능을 위한 deque 라이브러리 사용
    # 처음 시작할 때는 진입차수가 0인 노드를 큐에 삽입

    for i in range(1, v + 1):
        if indegree[i] == 0:
            q.append(i)
    # 큐가 빌 때까지 반복
    while q:
        # 큐에서 원소 꺼내기
        now = q.popleft()
        result.append(now)

        # 해당 원소와 연결된 노드들의 진입차수에서 1 빼기
        for i in graph[now]:
            indegree[i] -= 1
            if indegree[i] == 0:
                q.append(i)
    for i in result:
        print(i, end=' ')
topology_sort()
```

1.6. 위상 정렬 알고리즘 성능분석

- 위상 정렬을 위해 차례대로 모든 노드를 확인하며 각 노드에서 나가는 간선을 차례대로 제거해야 한다.
    - 위상 정렬 알고리즘의 시간 복잡도는 O(V+E)이다.

이 자료는 동빈 나 님의 **이코 테** 유튜브 영상을 보고 정리한 자료입니다.
참고 : www.youtube.com/watch?v=m-9pAwq1o3w&list=PLRx0vPvlEmdAghTr5mXQxGpHjWqSz0dgC

'Algorithm' 카테고리의 다른 글□

[Algorithm] 36강 : 위상정렬 알고리즘의 정의와 구현□

[Algorithm] 35강 : 크루스칼 알고리즘의 정의와 구현□

[Algorithm] 34강 : 서로소 집합을 활용한 사이클 판별□

[Algorithm] 33강 : 서로소 집합 자료구조의 정의와 구현□

[Algorithm] 32강 : 최단 경로 알고리즘 기초 문제 풀이□

[Algorithm] 31강 : 플로이드 워셜 알고리즘의 정의와 구현□

python 위상 정렬 알고리즘 python 위상정렬 위상정렬 위상정렬 알고리즘 파이썬 위상정렬

나아무늘보

혼자 끄적끄적하는 블로그 입니다.