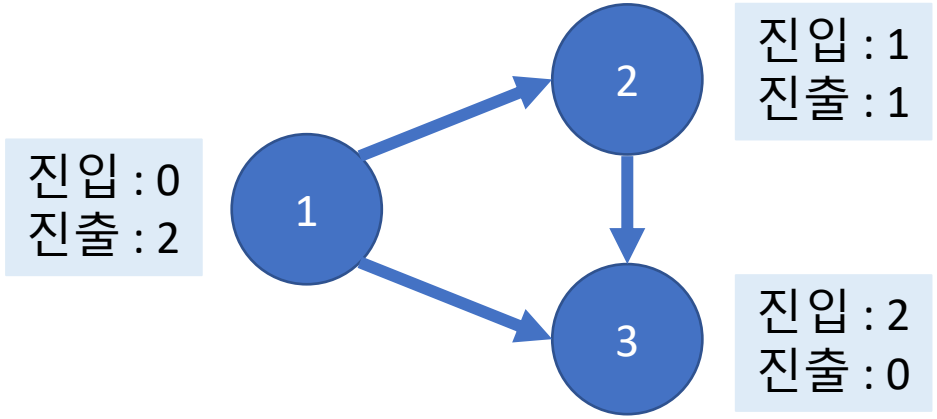
2022 AlgoLive 14th Study

Topology Sort
위상정렬

위상 정렬(Topology Sort)

- 순서가 정해져 있는 작업을 수행할 때 사용하는 알고리즘
- **사이클이 없는 방향 그래프**(Directed Acyclic Graph)
  - 사이클이면? '시작점'이 없다! -> 순서가 없다~

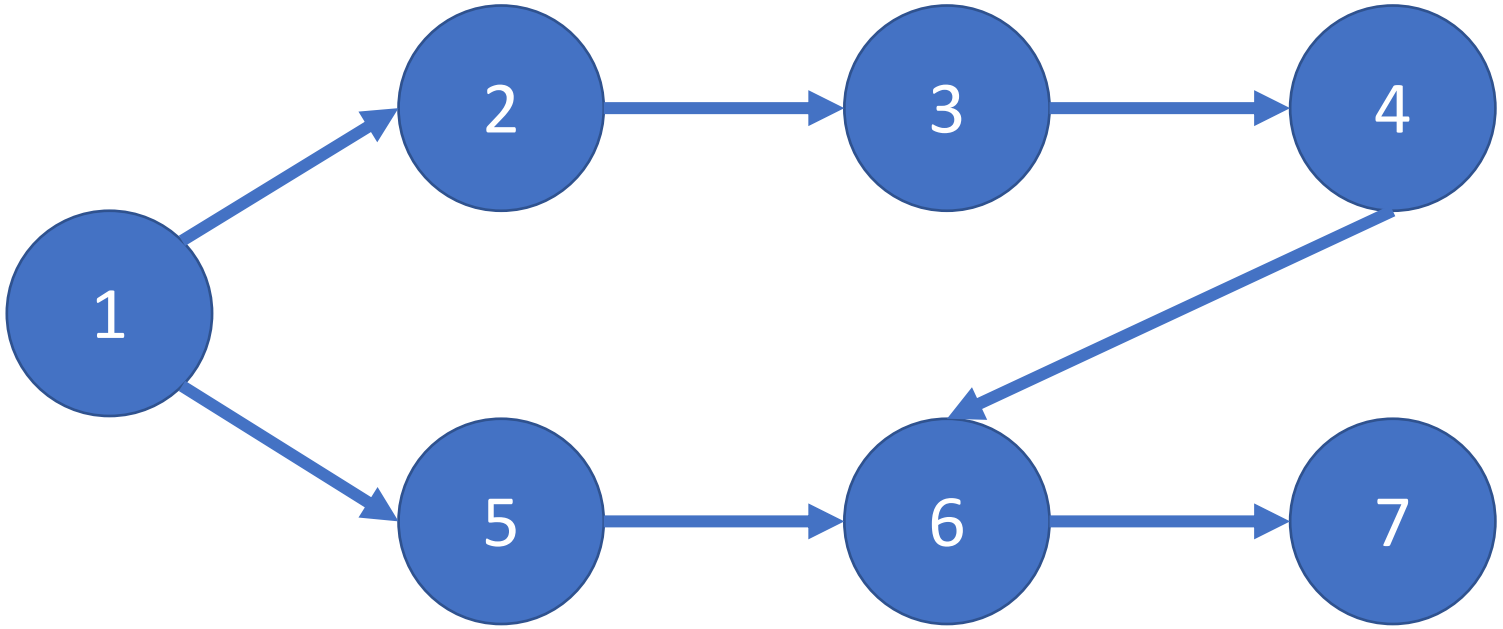
- 진입차수(Indegree) : 특정한 노드로 들어오는 간선 개수
- 진출자수(Outdegree) : 특정한 노드로 나가는 간선 개수



큐(Queue)를 이용한 알고리즘 동작 과정

1. 진입차수가 **0**인 모든 노드를 큐에 삽입
2. 큐에서 원소를 꺼내 **연결된** 모든 간선을 제거
3. 간선 제거 이후 진입차수가 **0**이 된 정점을 큐에 삽입
4. 큐가 빌 때까지 2, 3번 과정을 반복

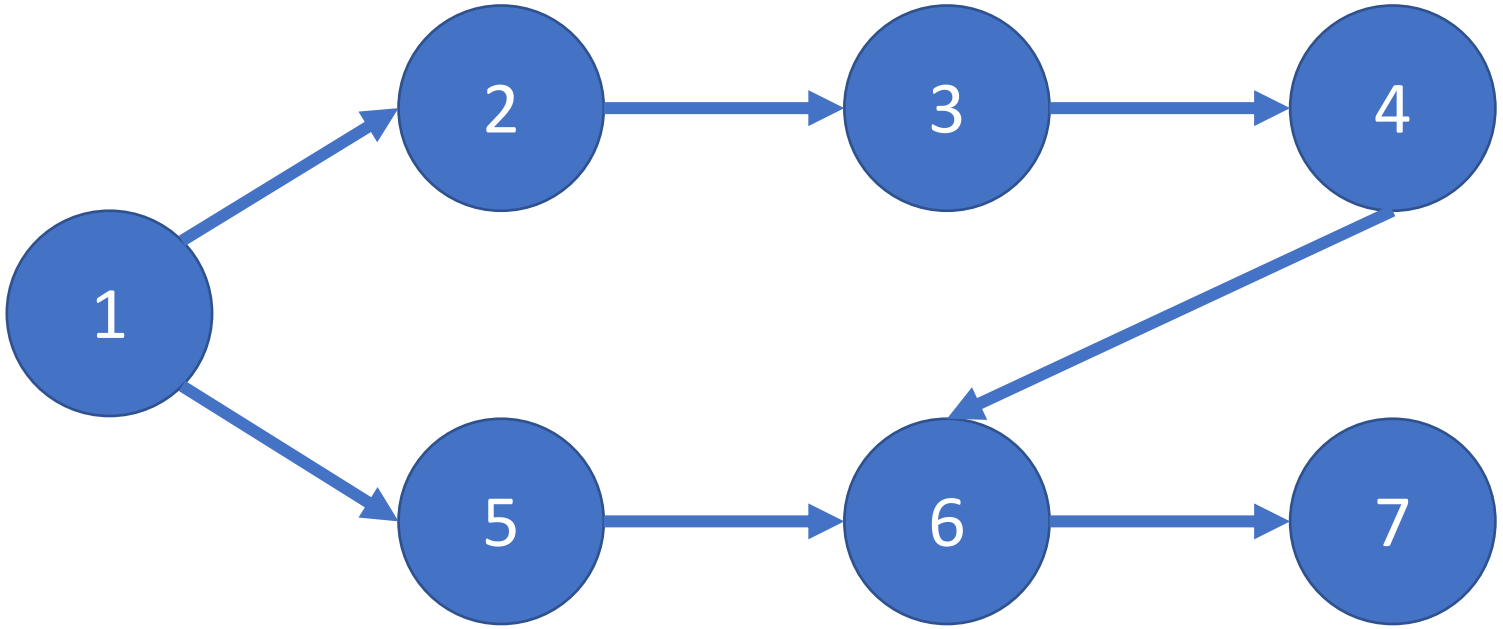
주의) 모든 원소를 방문하기 전에 큐가 비면, 사이클 존재!



Queue

Result[]

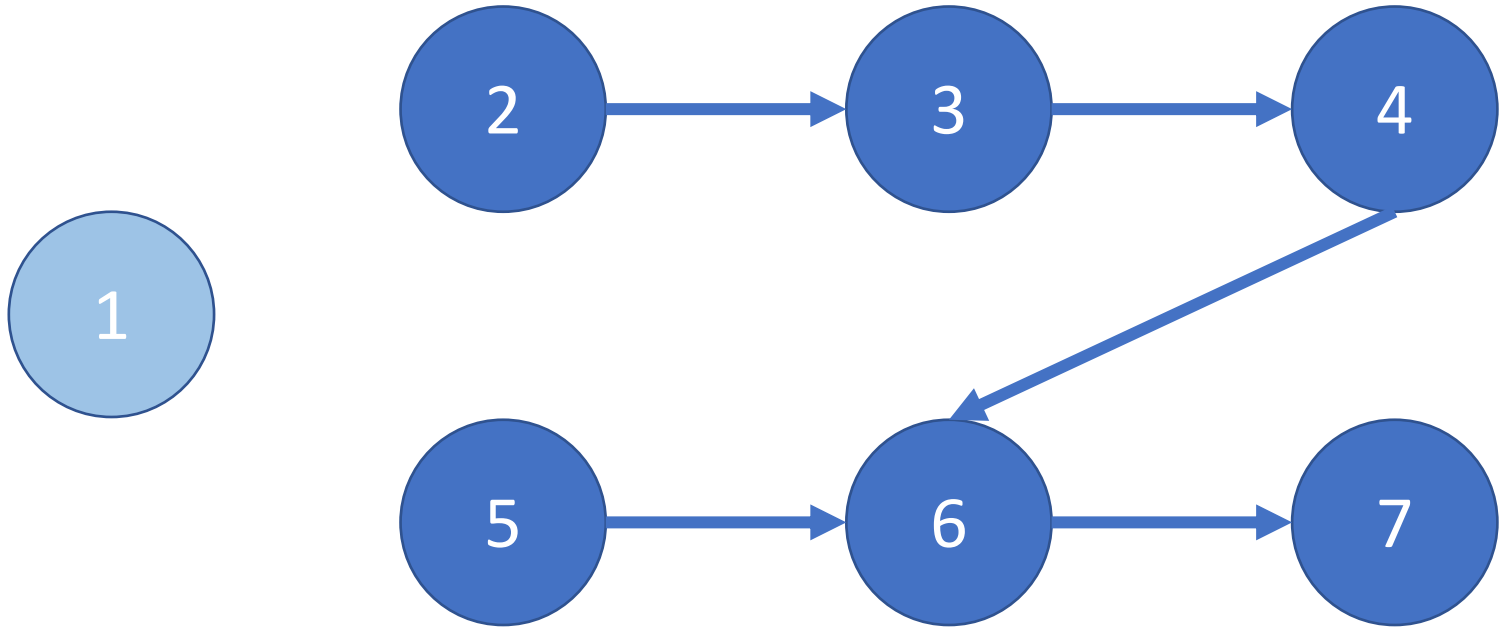
노드	1	2	3	4	5	6	7
진입차수	0	1	1	1	1	2	1



Queue 1

Result[]

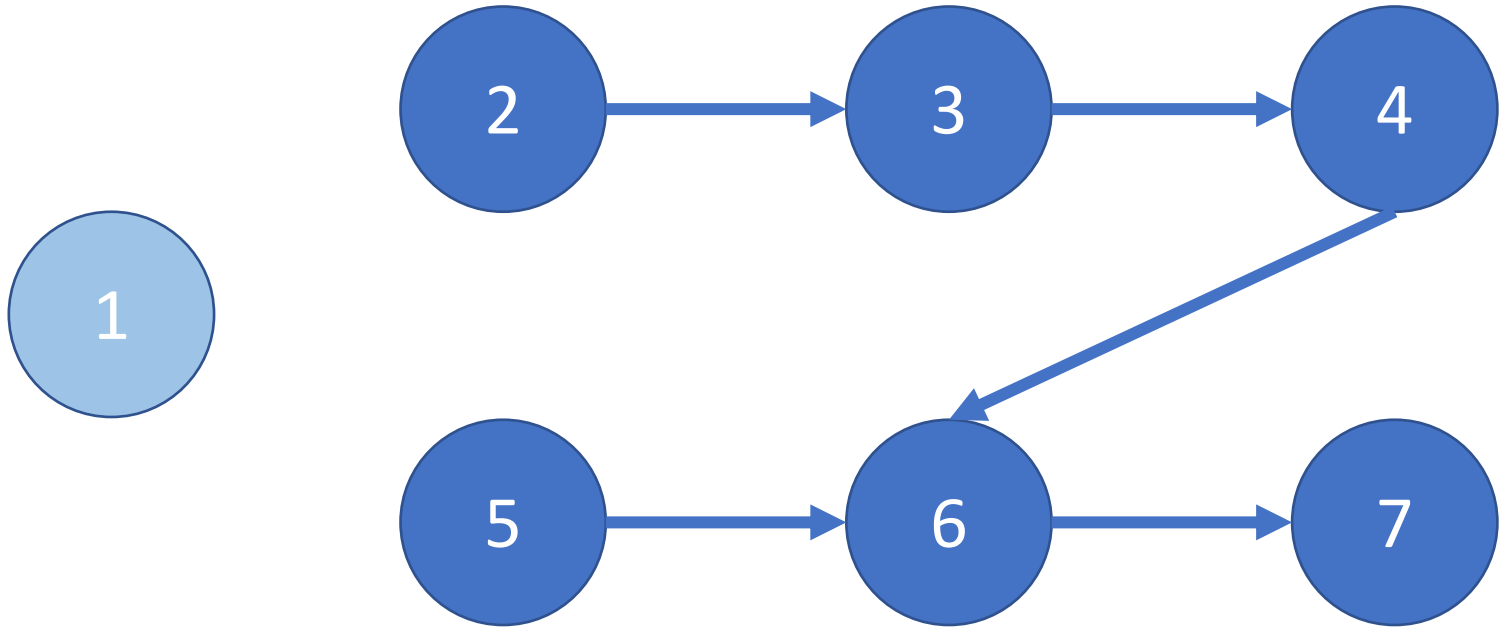
노드	1	2	3	4	5	6	7
진입차수	0	1	1	1	1	2	1



Queue

Result[] 1

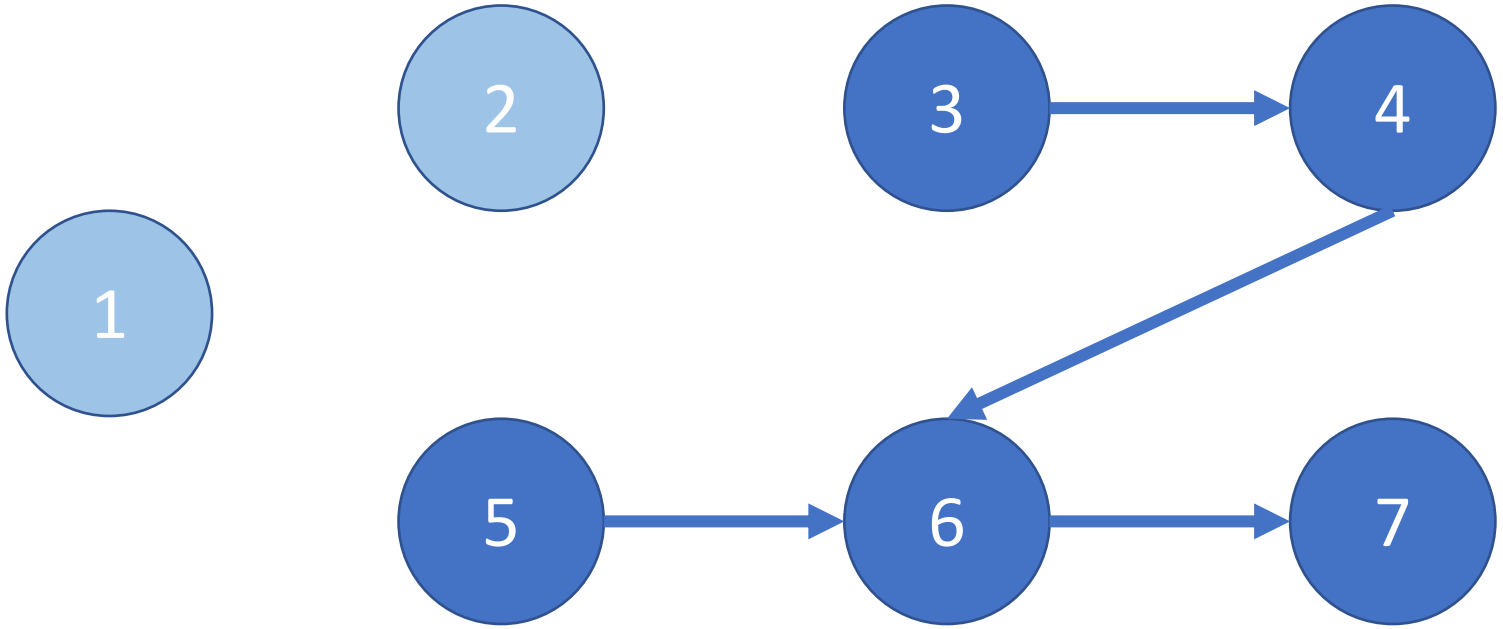
노드	1	2	3	4	5	6	7
진입차수	0	0	1	1	0	2	1



Queue 2 5

Result[] 1

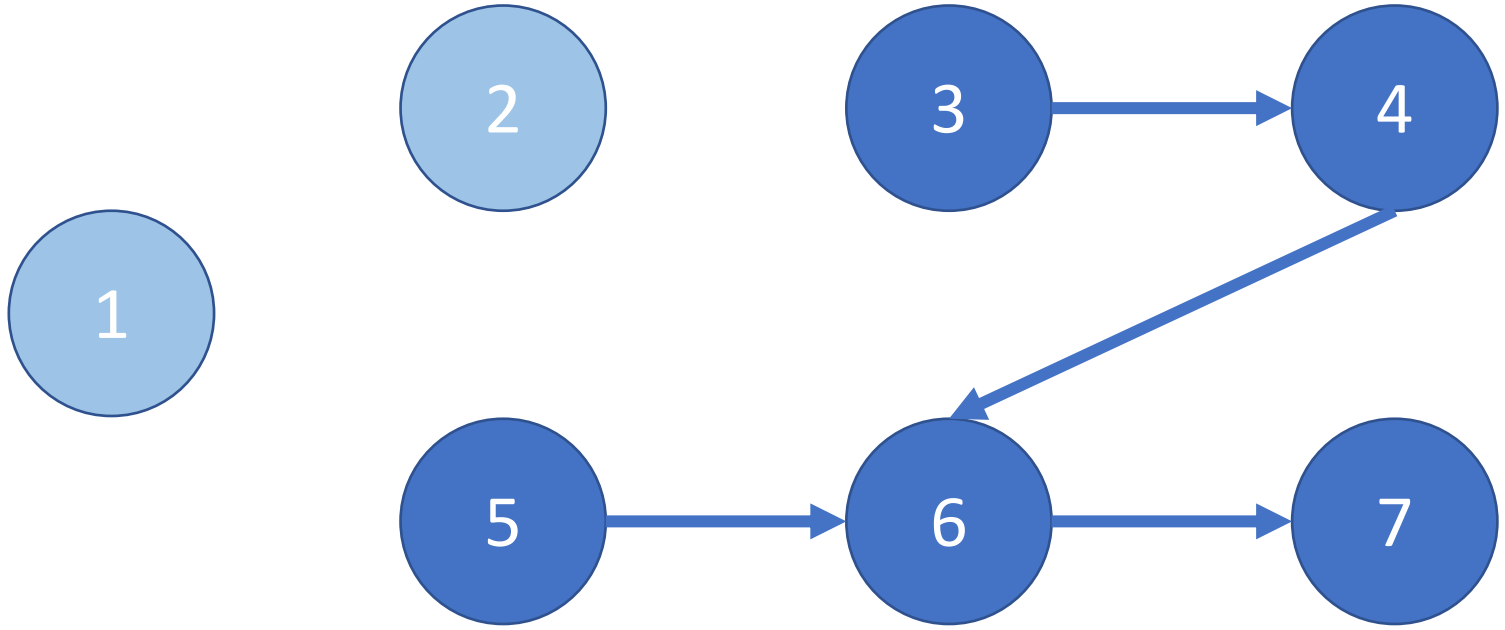
노드	1	2	3	4	5	6	7
진입차수	0	0	1	1	0	2	1



Queue 5

Result[] 1 2

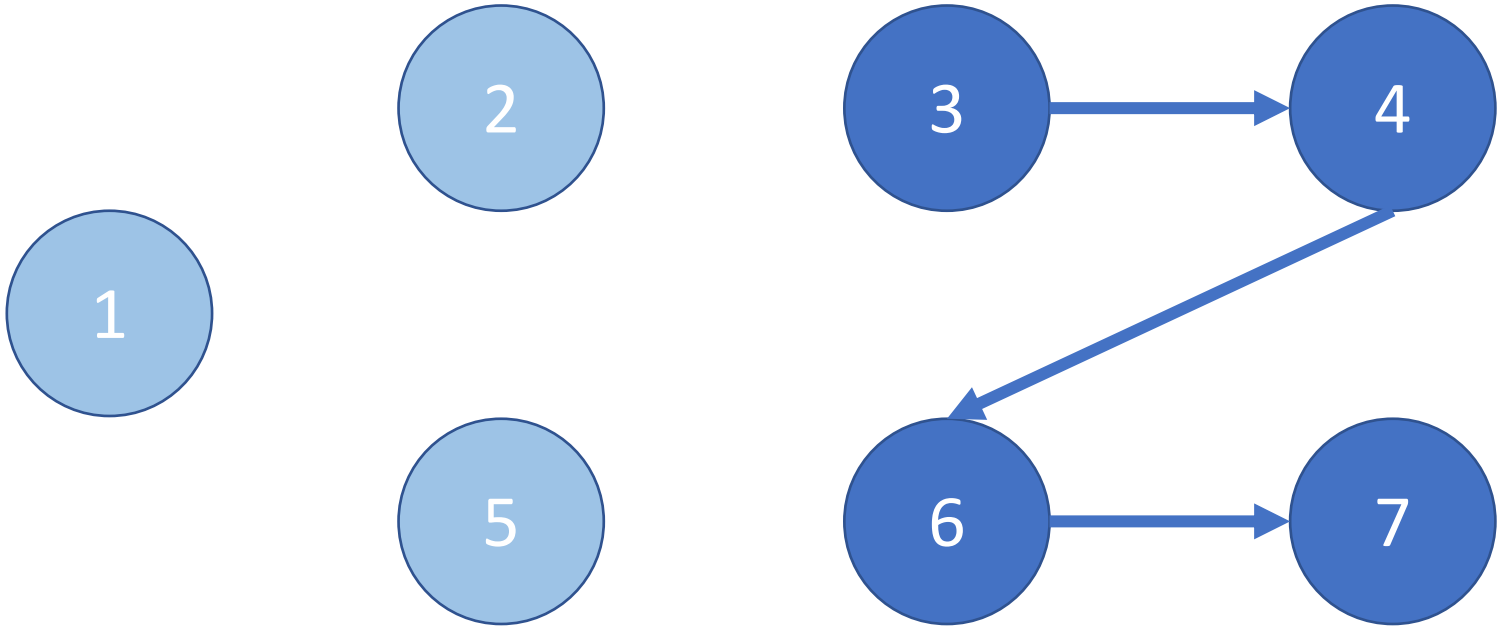
노드	1	2	3	4	5	6	7
진입차수	0	0	0	1	0	2	1



Queue 5 3

Result[] 1 2

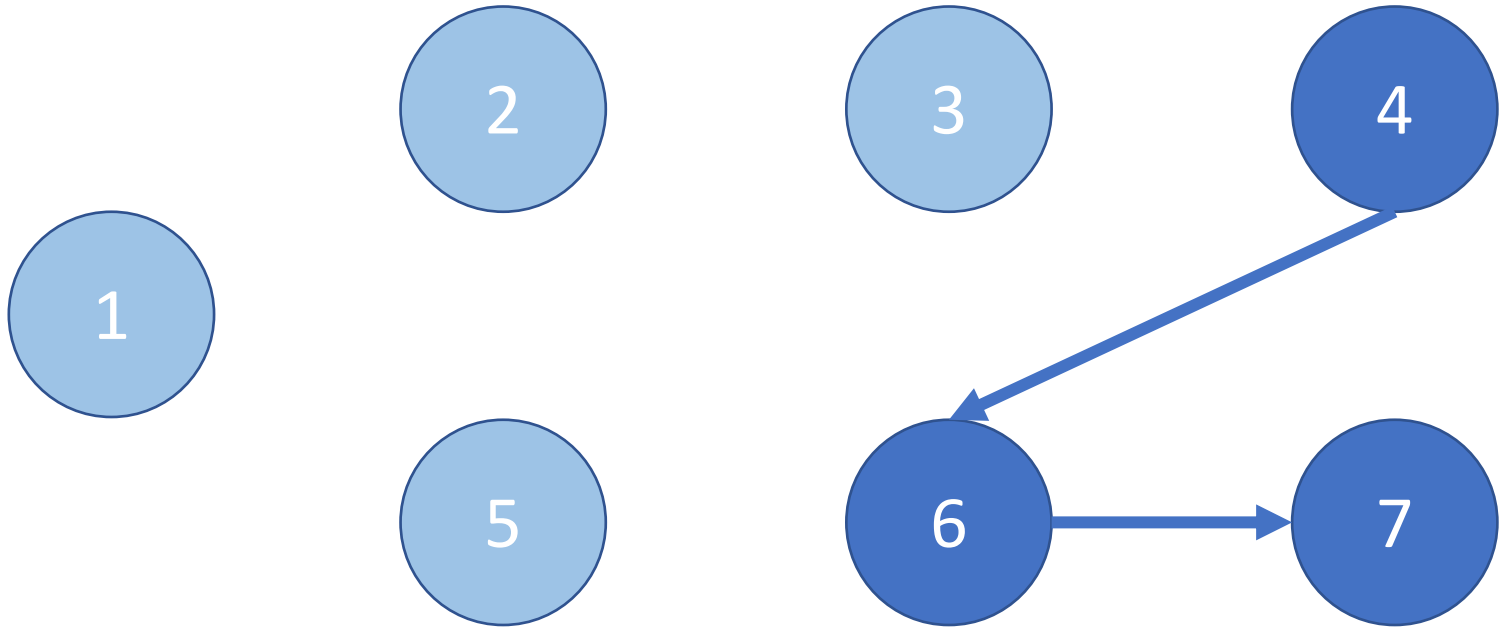
노드	1	2	3	4	5	6	7
진입차수	0	0	0	1	0	2	1



Queue 3

Result[] 1 2 5

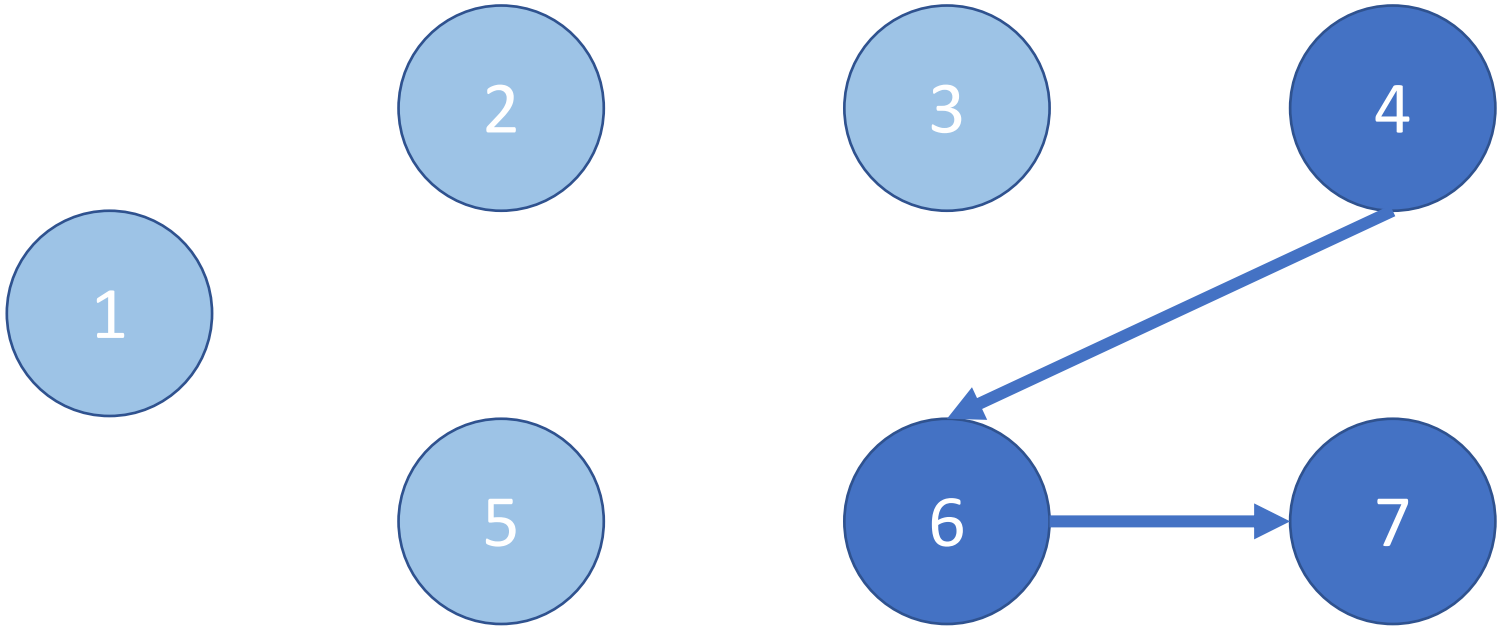
노드	1	2	3	4	5	6	7
진입차수	0	0	0	1	0	1	1



Queue

Result[] 1 2 5 3

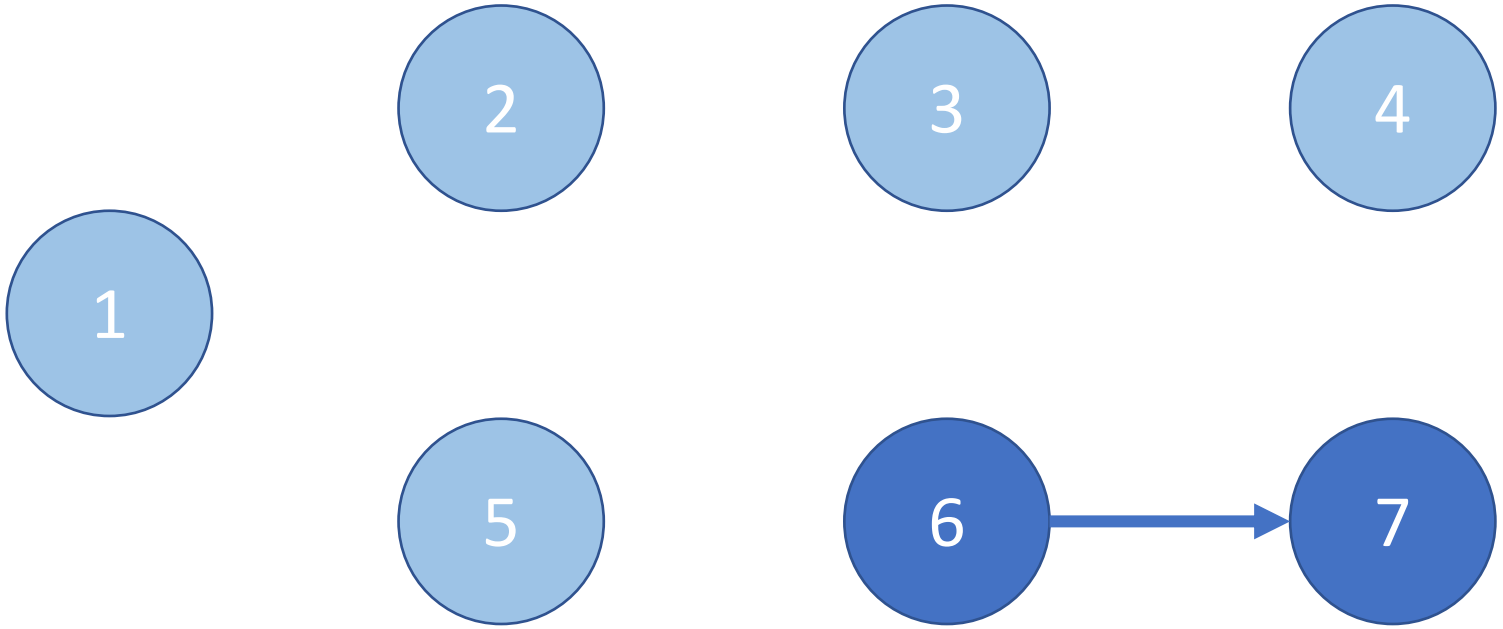
노드	1	2	3	4	5	6	7
진입차수	0	0	0	0	0	1	1



Queue 4

Result[] 1 2 5 3

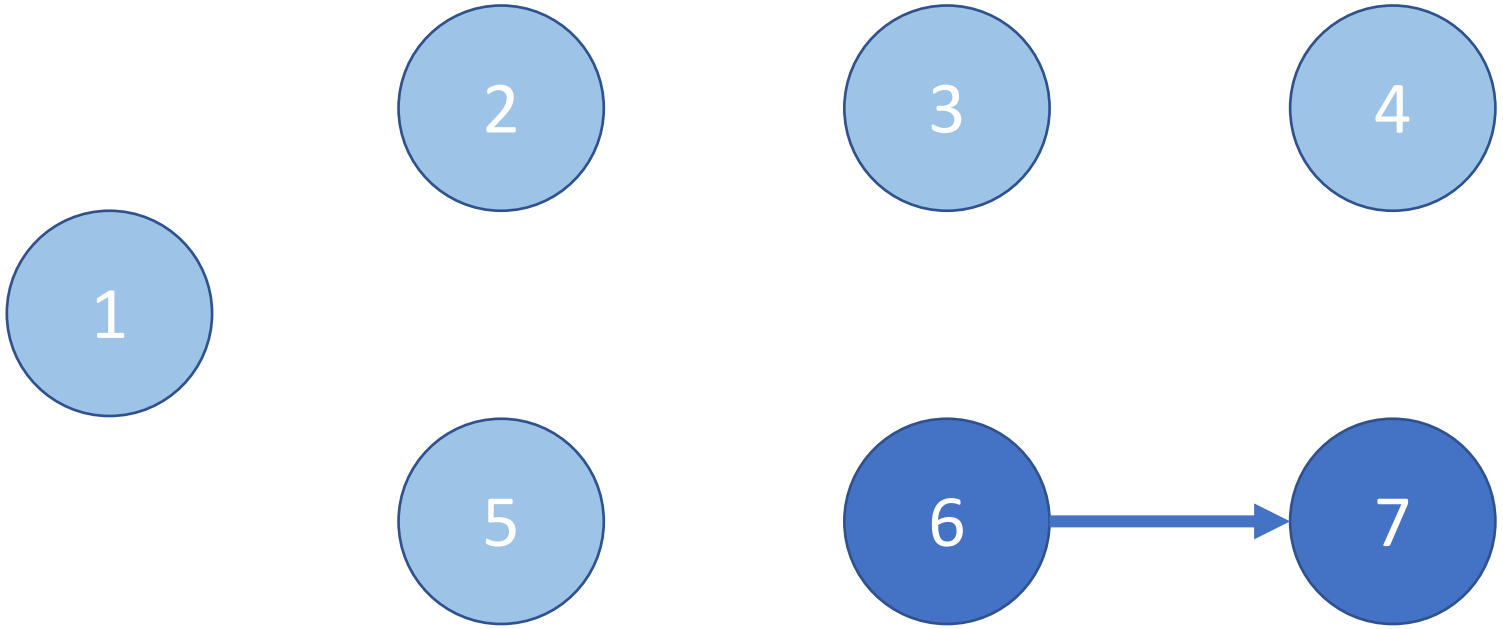
노드	1	2	3	4	5	6	7
진입차수	0	0	0	0	0	1	1



Queue

Result[] 1 2 5 3 4

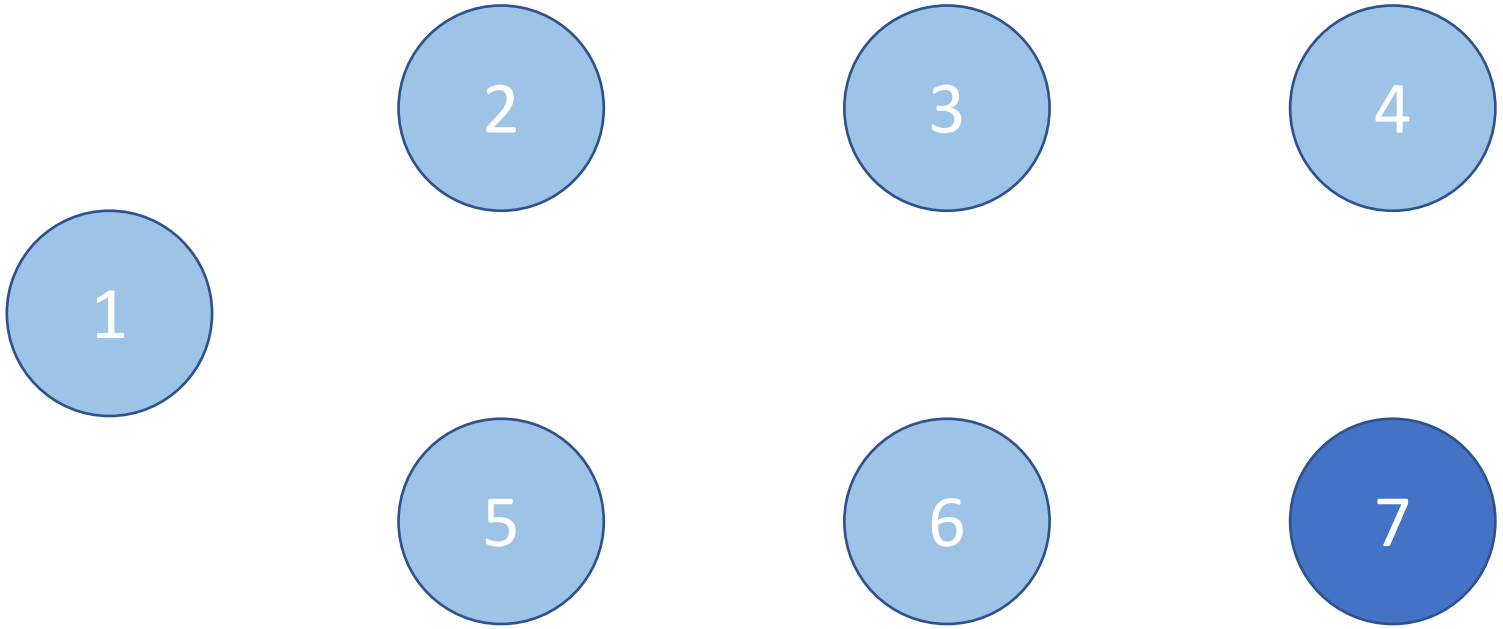
노드	1	2	3	4	5	6	7
진입차수	0	0	0	0	0	0	1



Queue 6

Result[] 1 2 5 3 4

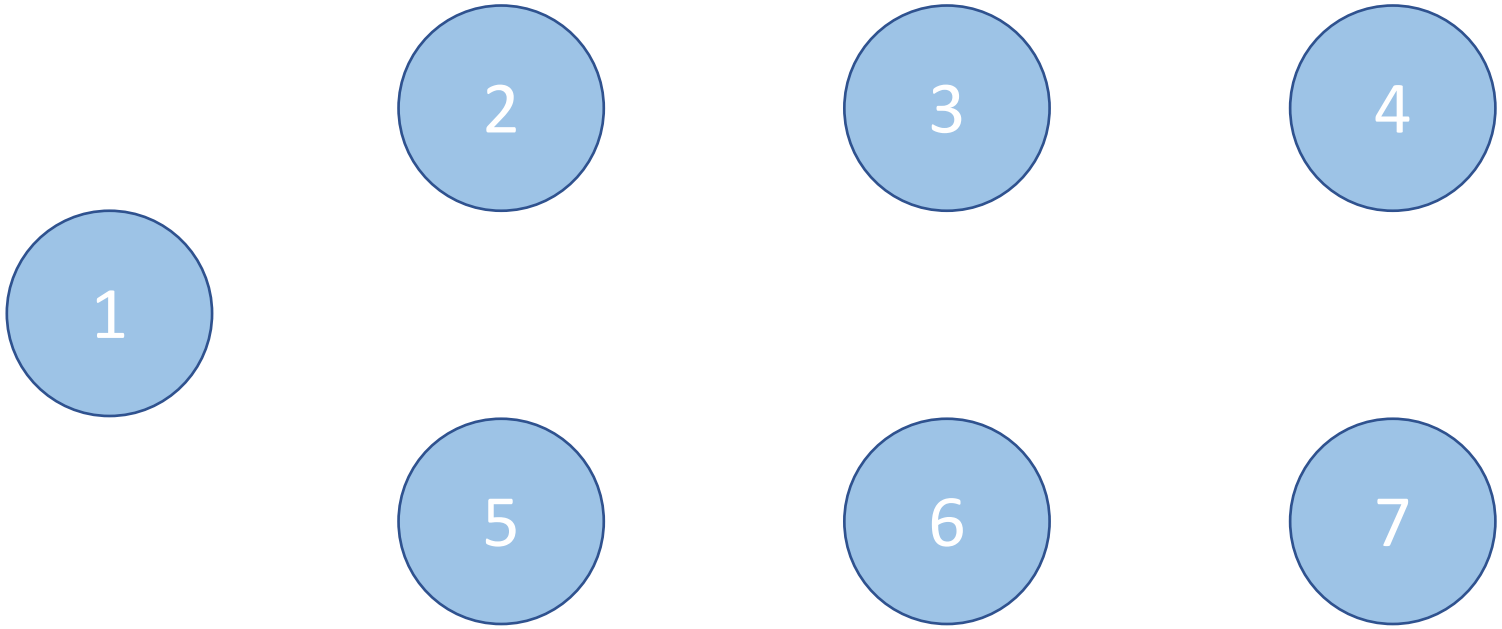
노드	1	2	3	4	5	6	7
진입차수	0	0	0	0	0	0	1



Queue

Result[] 1 2 5 3 4 6

노드	1	2	3	4	5	6	7
진입차수	0	0	0	0	0	0	0



Queue

Result[] 1 2 5 3 4 6 7

노드	1	2	3	4	5	6	7
진입차수	0	0	0	0	0	0	0

위상 정렬의 특징

- DAG에서만 수행 가능
- 여러 가지 답이 존재할 수 있음
  - 한 단계에서 큐에 새롭게 들어가는 원소가 2개 이상인 경우
- 모든 원소 방문 전에 큐가 비면 사이클이 존재함!!

1. 진입차수가 **0**인 모든 노드를 큐에 삽입
2. 큐에서 원소를 꺼내 **연결된** 모든 간선을 제거
3. 간선 제거 이후 진입차수가 **0**이 된 정점을 큐에 삽입
4. 큐가 빌 때까지 2, 3번 과정을 반복

- 스택을 활용한 DFS로도 수행 가능

코드 작성 전, 접 근 하 기 !

- 조건) 정점 개수, 간선 개수 주어지고, 연결된 노드들 각각 입력

- Parameter
- main 함수
- topologySort 함수

C++

```cpp
#include<iostream>
#include<vector>
#include<queue>
using namespace std;

int n, m, inDegree[10], result[10]; //n=정점개수, m=간선개수
vector<vector<int>> graph(10);  //vector<int> graph[10];
```

```cpp
void topologySort(){
    queue<int> q;

    for(int i=1; i<=n; i++){    //위상정렬 전, 진입차수 0인 노드를 큐에 삽입
        if(inDegree[i] == 0)
            q.push(i);
    }
    for(int i=1; i<=n; i++){
        if(q.empty()){  //n개의 정점을 모두 돌기 전에 큐가 비면 사이클
            cout << "사이클 발생";
            return;
        }
        int x = q.front();
        q.pop();
        result[i] = x;  //방문 노드 저장
        for(int i=0; i<graph[x].size(); i++){
            int y = graph[x][i];
            if(--inDegree[y] == 0)    //연결된 노드의 진입차수에서 -1한 값이 0이면 큐에 삽입
                q.push(y);
        }
    }
    for(int i=1; i<=n; i++){    //위상정렬 수행 결과 출력
        cout << result[i] << " ";
    }
}
```

```cpp
int main(){
    cin >> n >> m;
    for(int i=0; i<m; i++){
        int a, b;
        cin >> a >> b;
        graph[a].push_back(b);  //정점 a에서 b로 이동
        inDegree[b]++;  //정점 b의 진입차수 증가
    }
    topologySort();

    return 0;
}
```

시간복잡도

- O(V+E) (=정점의 개수 + 간선의 개수)

#2252

줄 세우기 성공 스페셜 저지

골드 III

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	31965	18256	12034	55.569%

문제

N명의 학생들을 키 순서대로 줄을 세우려고 한다. 각 학생의 키를 직접 재서 정렬하면 간단하겠지만, 마땅한 방법이 없어서 두 학생의 키를 비교하는 방법을 사용하기로 하였다. 그나마도 모든 학생들을 다 비교해 본 것이 아니고, 일부 학생들의 키만을 비교해 보았다.

일부 학생들의 키를 비교한 결과가 주어졌을 때, 줄을 세우는 프로그램을 작성하시오.

```cpp
#include <iostream>
#include <vector>
#include <queue>
using namespace std;

vector<int> graph[32001];
int inDegree[32001];
int N, M;

void TopologicalSort(void){
    queue<int> q;

    for(int i = 1; i <= N; i++)
        if(!inDegree[i])
            q.push(i);

    while(!q.empty()){
        int cur = q.front();
        q.pop();
        cout << cur << ' ';
        for(int i = 0; i < graph[cur].size(); i++){
            inDegree[graph[cur][i]]--;
            if(!inDegree[graph[cur][i]])
                q.push(graph[cur][i]);
        }
    }
}

int main(int argc, const char * argv[]) {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);cout.tie(NULL);
    cin >> N >> M;
    for(int i = 0; i < M; i++){
        int a, b;
        cin >> a >> b;
        graph[a].push_back(b);
        inDegree[b]++;
    }

    TopologicalSort();

    return 0;
}
```

```python
import sys
from collections import deque

n, m = map(int, sys.stdin.readline().split())
inDegree = [0] * (n+1)
inDegree[0] = -1
direct = [[] for _ in range(n+1)]
queue = deque()

for _ in range(m):
    a, b = map(int, sys.stdin.readline().split())
    inDegree[b] += 1
    direct[a].append(b)

for i in range(1, n+1):
    if inDegree[i] == 0:
        queue.append(i)

while queue:
    q = queue.popleft()
    print(q, end=' ')
    for d in direct[q]:
        inDegree[d] -= 1
        if inDegree[d] == 0:
            queue.append(d)
```

```java
import java.util.*;
import java.io.*;

// https://www.acmicpc.net/problem/2252

class Main {
    static int n;   // 노드 갯수
    static int m;   // 간선 갯수

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());

        n = Integer.parseInt(st.nextToken());
        m = Integer.parseInt(st.nextToken());
        List<List<Integer>> list = new ArrayList<List<Integer>>();
        int[] indegree = new int[n+1];

        for(int i=0; i<n+1; i++)
            list.add(new ArrayList<Integer>());

        for(int i=0; i<m; i++) {
            st = new StringTokenizer(br.readLine());

            // v1 -> v2
            int v1 = Integer.parseInt(st.nextToken());
            int v2 = Integer.parseInt(st.nextToken());

            list.get(v1).add(v2);
            indegree[v2]++;
        }

        // Solve
        topologicalSort(indegree, list);
    }

    static void topologicalSort(int[] indegree, List<List<Integer>> list) {
        Queue<Integer> q = new LinkedList<Integer>();
        Queue<Integer> result = new LinkedList<Integer>();

        // indegree가 0 인 노드 Queue 에 넣기
        for(int i=1; i<=n; i++) {
            if(indegree[i] == 0)
                q.offer(i);
        }

        // q 에서 노드를 하나씩 빼며 연결된 노드의 indegree 감소
        // indegree 가 0 이 된 노드들 Queue 에 넣기
        while(!q.isEmpty()) {
            int node = q.poll();
            result.offer(node);

            for(Integer linked : list.get(node)) {
                indegree[linked]--;

                if(indegree[linked] == 0)
                    q.offer(linked);
            }
        }

        // 결과 출력
        while(!result.isEmpty()) {
            System.out.print(result.poll() + " ");
        }
    }
}
```

#14567

선수과목 (Prerequisite) 성공

☆

골드 V

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
5 초	256 MB	2567	1710	1289	65.431%

문제

올해 Z대학 컴퓨터공학부에 새로 입학한 민욱이는 학부에 개설된 모든 전공과목을 듣고 졸업하려는 원대한 목표를 세웠다. 어떤 과목들은 선수과목이 있어 해당되는 모든 과목을 먼저 이수해야만 해당 과목을 이수할 수 있게 되어 있다. 공학인증을 포기할 수 없는 불쌍한 민욱이는 선수과목 조건을 반드시 지켜야만 한다. 민욱이는 선수과목 조건을 지킬 경우 각각의 전공과목을 언제 이수할 수 있는지 궁금해졌다. 계산을 편리하게 하기 위해 아래와 같이 조건을 간소화하여 계산하기로 하였다.

1. 한 학기에 들을 수 있는 과목 수에는 제한이 없다.
2. 모든 과목은 매 학기 항상 개설된다.

모든 과목에 대해 각 과목을 이수하려면 최소 몇 학기가 걸리는지 계산하는 프로그램을 작성하여라.

결과값은 1단위로 변화

```cpp
#include <iostream>
#include <vector>
#include <queue>
using namespace std;

const int MAX = 1001;
vector<int> adj[MAX];
int inDeg[MAX];
queue<int> q;
int result[MAX];

int main() {
    int N, M;
    cin >> N >> M;

    int a, b;
    for (int i = 0; i < M; i++) {
        cin >> a >> b;
        adj[a].push_back(b);
        inDeg[b]++;
    }

    for (int i = 1; i <= N; i++) {
        if (inDeg[i] == 0) {
            q.push(i);
        }
        result[i] = 1; //1학기부터 시작(초기화)
    }

    while (!q.empty()) {
        int cur = q.front();
        q.pop();

        for (int i = 0; i < adj[cur].size(); i++) {
            int next = adj[cur][i];
            inDeg[next]--;
            if (inDeg[next] == 0) {
                q.push(next);
                result[next] = max(result[next], result[cur] + 1); //★
            }
        }
    }

    for (int i = 1; i <=N; i++) {
        cout << result[i] << " ";
    }

    return 0;
}
```

Colored by Colo

```python
import sys
from collections import deque

def topology_sort():
    q = deque()
    # 1. 진입차수가 0인 모든 노드를 큐에 넣는다.
    for i in range(n):
        if indegree[i] == 0:  # 처음은 진입차수가 0인 것부터. 진입차수 0이 시작하는 노드이기 때문
            q.append(i)
    # 2. 큐가 빌 때까지 아래 과정 반복한다.
    while q:
        # 2-1. 큐에서 원소를 꺼내 해당 노드에서 나가는 간선을 그래프에서 제거한다.
        now = q.popleft()
        for i in graph[now]:
            indegree[i] -= 1
            if indegree[i] == 0:  # 2-2. 새롭게 진입차수가 0이된 노드를 큐에 넣기 & 결과 넣기
                result[i] = result[now] + 1  # [추가된 부분] 현재 노드 결과에 + 1한(다음 학기) 학기로 넣기
                q.append(i)

    print(*result)

n, m = map(int, sys.stdin.readline().split())  # 과목수, 선수 조건 수
indegree = [0] * n
result = [1] * n
graph = [[] for i in range(n)]

for _ in range(m):
    a, b = map(int, sys.stdin.readline().split())
    graph[a - 1].append(b - 1)  # a > b
    indegree[b - 1] += 1    # 진입 차수

topology_sort()
```

```java
import java.io.*;
import java.util.*;

public class Main {
    static int N;
    static List<List<Integer>> l = new ArrayList<>();
    static int[] parentNum;
    static int[] answer;

    public static void main(String[] args) throws IOException{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());

        N = Integer.parseInt(st.nextToken());
        int M = Integer.parseInt(st.nextToken());

        parentNum = new int[N+1];
        answer = new int[N+1];
        for(int i = 0; i<=N; i++)
            l.add(new ArrayList<>());

        while(M-->0){
            st = new StringTokenizer(br.readLine());

            int A = Integer.parseInt(st.nextToken());
            int B = Integer.parseInt(st.nextToken());

            l.get(A).add(B);
            parentNum[B]++;
        }

        topologicalSort();

        for(int i = 1; i<=N; i++)
            System.out.print(answer[i]+" ");
        System.out.println();
    }
```

```java
    static void topologicalSort(){
        Queue<Integer> q = new LinkedList<>();

        for(int i = 1; i<=N; i++)
            if(parentNum[i] == 0){
                q.offer(i);
                answer[i] = 1;
            }

        while(!q.isEmpty()){
            int num = q.poll();

            for(int i : l.get(num)){
                parentNum[i]--;

                if(parentNum[i] == 0){
                    q.offer(i);
                    answer[i] = answer[num] + 1;
                }
            }
        }
    }
}
```

#2056

작업 성공

골드 V

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	256 MB	10532	4833	3463	42.875%

문제

수행해야 할 작업 N개 (3 ≤ N ≤ 10000)가 있다. 각각의 작업마다 걸리는 시간(1 ≤ 시간 ≤ 100)이 정수로 주어진다.

몇몇 작업들 사이에는 선행 관계라는 게 있어서, 어떤 작업을 수행하기 위해 반드시 먼저 완료되어야 할 작업들이 있다. 이 작업들은 번호가 아주 예쁘게 매겨져 있어서, K번 작업에 대해 선행 관계에 있는(즉, K번 작업을 시작하기 전에 반드시 먼저 완료되어야 하는) 작업들의 번호는 모두 1 이상 (K-1) 이하이다. 작업들 중에는, 그것에 대해 선행 관계에 있는 작업이 하나도 없는 작업이 반드시 하나 이상 존재한다. (1번 작업이 항상 그러하다)

모든 작업을 완료하기 위해 필요한 최소 시간을 구하여라. 물론, 서로 선행 관계가 없는 작업들은 동시에 수행 가능하다.

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int dp[10001] = { 0, };
int n;
int time = 0, cnt = 0, value = 0;
int result = 0;

int main(void) {

	iostream::sync_with_stdio(false);
	cin.tie(0);

	cin >> n;
	for (int i = 1; i <= n; i++) {
		cin >> time >> cnt;
		if (cnt == 0) {
			dp[i] = time;
			continue;
		}
		int maxed = 0;
		for (int j = 0; j < cnt; j++) {
			cin >> value;
			maxed = max(maxed, dp[value]);
		}
		dp[i] = time + maxed;
	}
	/*
	for (int i = 1; i <= n; i++)
		cout << dp[i] << endl;
	*/


	for (int i = 1; i <= n; i++)
		result = max(result, dp[i]);

	cout << result << endl;


	return 0;
}
```

점화식!
DP 로도 가능

```python
import sys
input = sys.stdin.readline

n = int(input())
times = [0] * (n+1)
graph = {}
for i in range(1, n+1):
    lst = list(map(int, input().split()))
    times[i] = lst[0]
    if lst[1] == 0:
        continue
    for j in lst[2:]:
        if i in graph:
            graph[i].append(j)
        else:
            graph[i] = [j]

for i in range(1, n+1):
    if i in graph:
        time = 0
        for j in graph[i]:
            time = max(time, times[j])
        times[i] += time

print(max(times))
```

```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.StringTokenizer;

public class Main {

	public static void main(String[] args) throws IOException {
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));
		StringTokenizer st;

		int N = Integer.parseInt(br.readLine());
		int[] dp = new int[N + 1]; // 각각의 작업을 수행하는 데 걸리는 시간

		int ans = 0;
		for (int i = 1; i <= N; i++) {
			st = new StringTokenizer(br.readLine());
			int time = Integer.parseInt(st.nextToken());
			int num = Integer.parseInt(st.nextToken());

			dp[i] = time;
			for (int j = 0; j < num; j++) {
				int temp = Integer.parseInt(st.nextToken());

				// 가장 긴 수행 시간으로 설정해야 함.
				dp[i] = Math.max(dp[i], dp[temp] + time);
			}

			ans = Math.max(ans, dp[i]);
		}

		bw.write(ans + "\n");
		bw.flush();
		bw.close();
		br.close();
	}

}
```

끗-!
수고하셨습니다😆