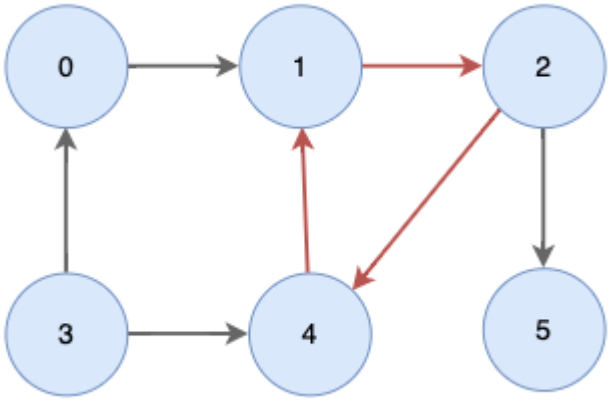
2기 / 3주차_위상 정렬

🌟 위상 정렬

위상 정렬(Topological sort)이란 비순환 방향 그래프(DAG)에서 정점을 선형으로 정렬하는 것이다.

💡 비순환 방향 그래프(DAG): 사이클이 없는 방향 그래프

DAG가 아닌 경우 위상정렬이 불가능한 이유는 다음과 같다.
사이클이 있으면서 두 정점이 사이클에 속한 정점일 경우 어떤 점을 먼저 방문해야 할 지 판단할 수 없다.



예를 들어 다음과 같은 사이클의 경우 1이 4보다 먼저일 수도 있고 4가 1보다 먼저일 수도 있기 때문이다.

일반적으로 진입 차수를 이용한 BFS와 DFS로 구현한다.

1. 백준 14567_선수과목

💡 접근법

위상 정렬을 쓰지 않고도 풀 수 있을 것 같았다.

조건이 first - second라면, 입력 받을 때부터 array[second] = array[first] + 1으로 저장하면 쉽게 풀릴 거라고 생각했다.

~~하지만 인생은 녹록지 않지 ,,~~

🔑 풀이

위상 정렬을 공부하고 풀어보니 정말 정석 그 자체인 문제다.

우선 선수 과목과 다음 과목을 연결리스트로 입력 받고, 다음 과목의 진입차수를 1씩 증가시킨다.

예제 입력 2 복사

```
6 4
1 2
1 3
2 5
4 5
```

위 예제로 생각했을 때 input() 메소드가 끝난 후 degree 배열(진입차수)의 값은 {0, 1, 1, 0, 2, 0}이 될 것이다.
즉, index 2와 3은 선수과목이 1개씩 존재한다는 뜻이다.
여기서 주의해야 할 점은 선수과목의 개수 ≠ 해당 과목을 들을 수 있는 학기라는 것이다.

메인 로직은 다음과 같다.

1. 진입 차수가 0인 과목들을 큐에 담고, 결과 배열에 1을 넣는다.
   (이 때, 결과 배열이란 해당 과목을 들을 수 있는 학기를 의미한다.)
2. 큐가 비어있지 않을 때까지 loop를 실행한다.
   a. 큐에서 값을 꺼낸다.
   b. 꺼낸 값과 연결된 정점의 진입 차수에 -1을 한다. (연결된 간선을 끊는다는 뜻)
   c. 진입 차수가 0이 된 정점은 큐에 담고, 결과 배열 값을 넣어준다.

💻 코드

688 ms, 128696 KB

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

public class BOJ_14567 {
    private static int cntSubject;
```

```
    private static List<List<Integer>> graph;
    private static int[] degree;
    private static int[] result;

    public static void main(String[] args) throws IOException {
        input();
        topologicalSort();
        output();
    }

    private static void topologicalSort() {
        result = new int[cntSubject + 1];
        Queue<Integer> queue = new LinkedList<>();

        for (int i = 1; i <= cntSubject; i++) {
            if (degree[i] == 0) {
                queue.offer(i);
                result[i] = 1;
            }
        }

        while (!queue.isEmpty()) {
            int current = queue.poll();

            for (int i : graph.get(current)) {
                degree[i]--;
                if (degree[i] == 0) {
                    queue.offer(i);
                    result[i] = result[current] + 1;
                }
            }
        }
    }

    private static void output() throws IOException {
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));
        for (int i = 1; i <= cntSubject; i++) {
            bw.write(result[i] + " ");
        }
        bw.flush();
        bw.close();
    }

    private static void input() throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());

        cntSubject = Integer.parseInt(st.nextToken());
        int cntEdge = Integer.parseInt(st.nextToken());

        degree = new int[cntSubject + 1];
        graph = new ArrayList<>();
        for (int i = 0; i <= cntSubject; i++) {
            graph.add(new ArrayList<>());
        }

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

```
            int first = Integer.parseInt(st.nextToken());
            int second = Integer.parseInt(st.nextToken());

            graph.get(first).add(second);
            degree[second]++;
        }
    }
}
```

2. 백준 5021_왕위 계승

💡 접근법

🔑 풀이

💻 코드

672 ms, 53988 KB

```

```

3. 백준 2637_장난감 조립

💡 접근법

🔑 풀이

💻 코드

1228 ms, 263976 KB

```

```