Chapter 03.
DFS - 깊이우선 탐색

Clip 01 | [10451] 순열 사이클
순열과 배열을 사용한 그래프
진입 차수와 진출 차수

Clip 02 | [9466] 텀 프로젝트
사이클 처리와 최적화

Ch03. DFS – 깊이 우선 탐색

1. [10451] 순열 사이클

BOJ10451: 순열 사이클

문제 요약

- 2<=N<=1000 크기의 순열이 주어진다
- n번째로 오는 숫자는 n번 노드에서 이동할 수 있는 노드의 번호를 뜻한다
- 정점 하나에서 출발해 다시 자기 자신으로 돌아오는 경로에 존재하는 모든 노드들을 묶어서 순열 사이클이라고 한다
- 순열로 주어진 그래프에서 순열 사이클의 개수를 구하는 문제

BOJ10451: 순열 사이클

예제

입력 데이터
2 8 3 2 7 8 1 4 5 6 10 2 1 3 4 5 6 7 9 10 8

출력 데이터
3 7

BOJ10451: 순열 사이클

예제

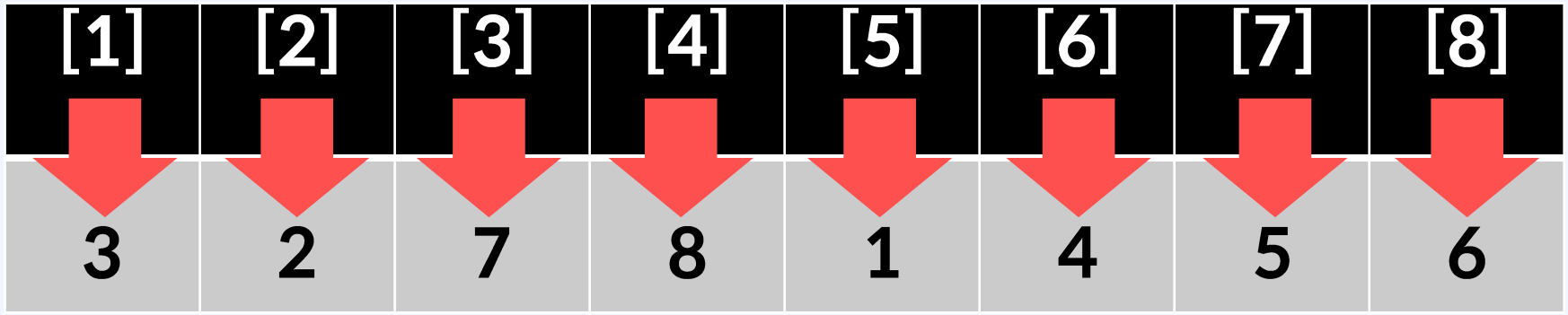
입력 데이터
8 3 2 7 8 1 4 5 6

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
3	2	7	8	1	4	5	6

정점 하나에서 출발해 다시 자기 자신으로 돌아오는 경로에 존재하는
모든 노드들을 묶어서 순열 사이클이라고 한다

BOJ10451: 순열 사이클

예제

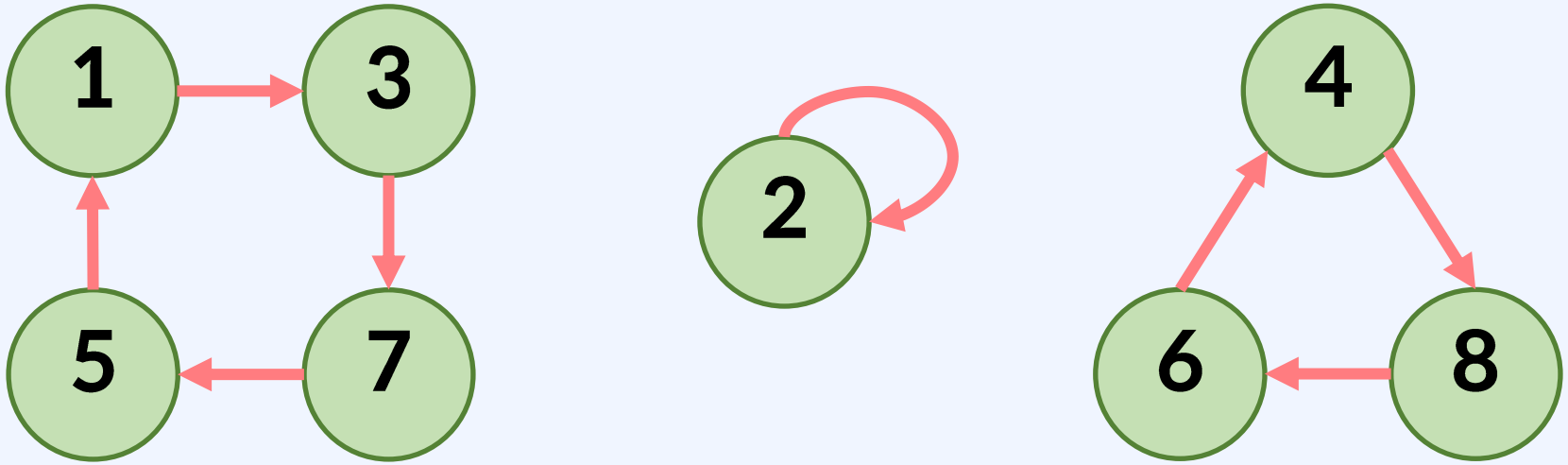


정점 하나에서 출발해 다시 자기 자신으로 돌아오는 경로에 존재하는
모든 노드들을 묶어서 순열 사이클이라고 한다

BOJ10451: 순열 사이클

예제

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
3	2	7	8	1	4	5	6



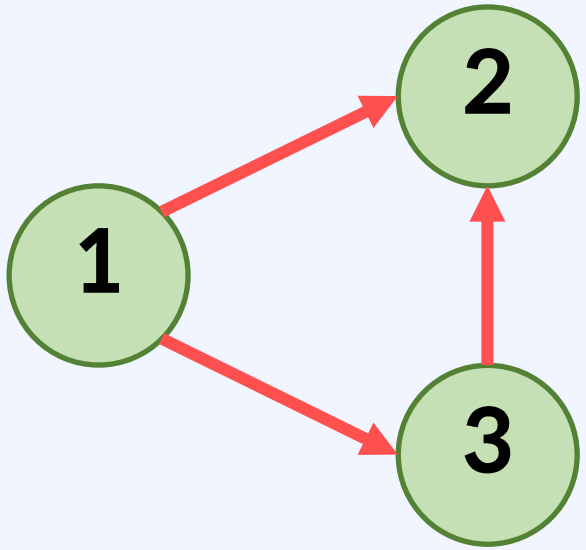
BOJ10451: 순열 사이클

문제 분석

- 입력이 순열로 주어진다
  - 1 ~ N 까지 수가 1개씩 등장한다
- index와 value가 1:1 매핑 된다
- 두 조건을 조합하면?
  - (진입차수: 1) (진출차수: 1) 이 항상 유지된다
  - 자기 자신에게 돌아오는 경로는 무조건 생긴다
  - 왜?

BOJ10451: 순열 사이클

문제 분석



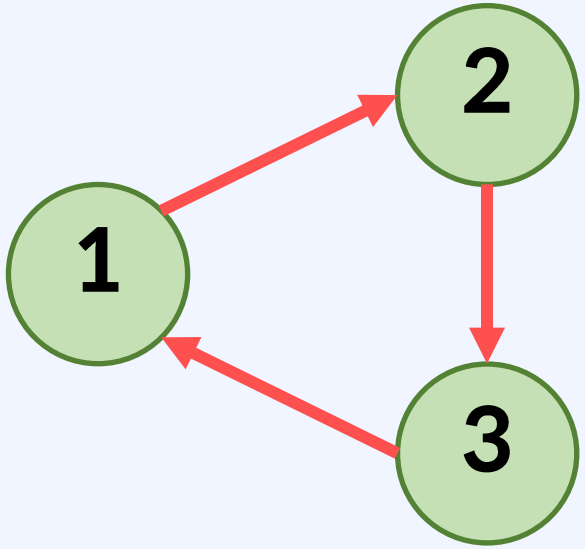
[1]	[2]	[3]
2, 3	X	2

진출 차수가 0인 케이스
- 배열 한칸에 두개의 값이 들어가야 한다 (입력으로 들어올 수 없다)

진입 차수가 0인 케이스
- 특정 배열 칸에는 값이 들어가지 않는다 (입력으로 들어올 수 없다)

BOJ10451: 순열 사이클

문제 분석



[1]	[2]	[3]
2	3	1

(진입차수: 1) (진출차수: 1)
항상 유지된다

모든 노드들은 사이클
(자기 자신으로 돌아오는 경로)
이 존재한다

BOJ10451: 순열 사이클

문제 분석

모든 노드들은 사이클이 존재한다
→ 코드로 따로 사이클을 찾을 필요가 없다

그렇다면?
문제는 pt3 ch1: 연결요소의 개수 와 거의 유사해진다
- 연결 되어있는 그래프들의 개수를 세면 된다

BOJ10451: 순열 사이클

구현

```
for(int i = 1; i <= n; i++) {
    if(!visited[i]) {
        dfs(i);
        cnt++;
    }
}
```

```
public static void dfs(int node) {
    visited[node] = true;
    if(!visited[nextNode[node]]) {
        dfs(nextNode[node]);
    }
}
```

DFS를 수행하며 연결 노드 체크, Main에서 수행한 탐색이 끝나면 연결된 그래프의 개수를 1 증가

Ch03. DFS – 깊이 우선 탐색

2. [9466] 텀 프로젝트

BOJ9466: 텀 프로젝트

문제 요약

- 학생은 프로젝트를 같이할 다른 학생을 1명 고를 수 있음
  - 자기 자신을 고르는 것도 가능
- 사이클이 만들어지는 그래프는 같은 팀이 될 수 있음
- 팀에 속하지 않는 학생들의 수를 계산

BOJ9466: 텀 프로젝트

예제

입력 데이터
2 7 3 1 3 7 3 4 6 8 1 2 3 4 5 6 7 8

출력 데이터
3 0

BOJ9466: 텀 프로젝트

예제

입력 데이터
7 3 1 3 7 3 4 6

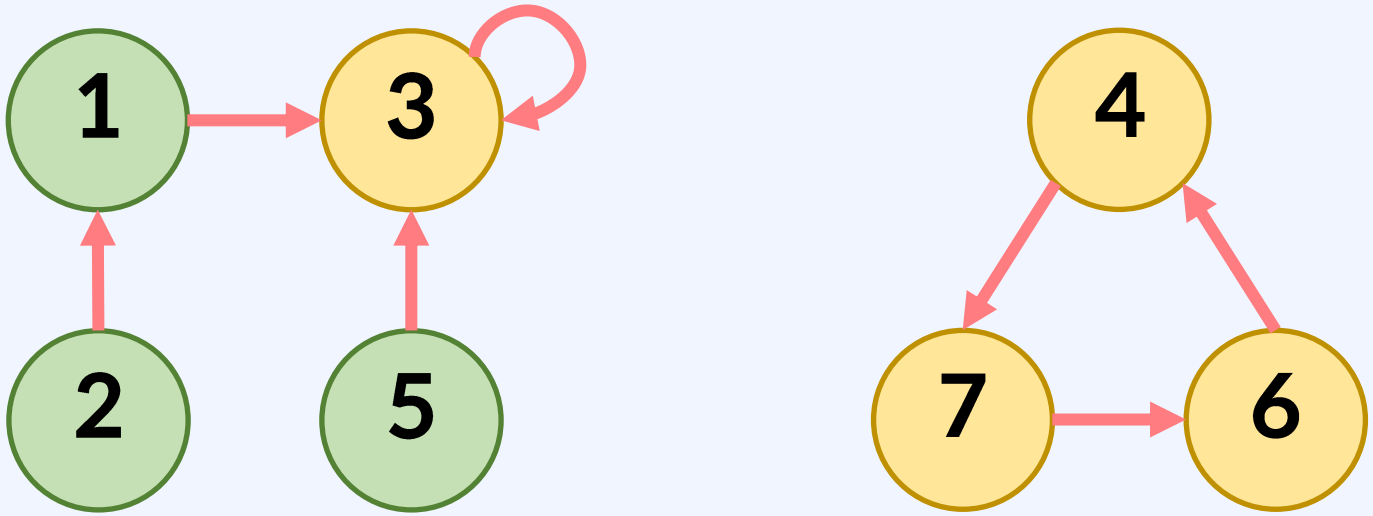
[1]	[2]	[3]	[4]	[5]	[6]	[7]
3	1	3	7	3	4	6

배열의 {인덱스} → {값} 으로 연결정보가 주어지는 그래프

BOJ9466: 텀 프로젝트

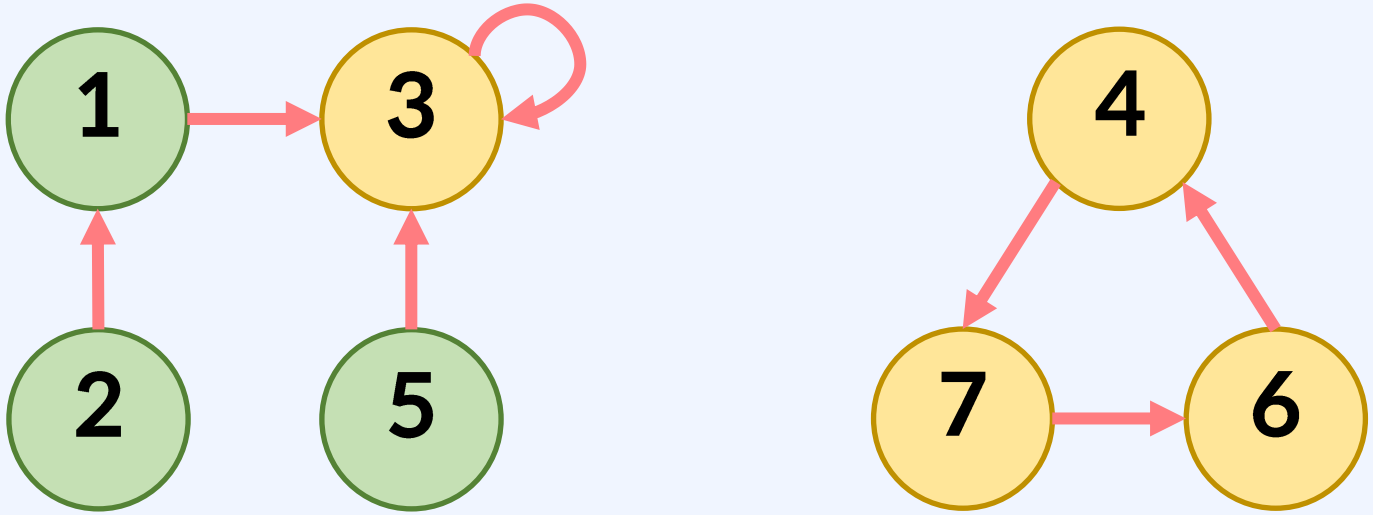
예제

[1]	[2]	[3]	[4]	[5]	[6]	[7]
3	1	3	7	3	4	6



BOJ9466: 텀 프로젝트

예제



사이클에 포함되는 노드: 노란색
사이클에 포함되지 않는 노드: 녹색

BOJ9466: 텀 프로젝트

문제 분석

- BOJ10451 순열 사이클과 비슷한 문제
  - 단, 진입 차수가 1이 아닐 수 있음
    - 여러 명의 선택을 받은 사람: 1 초과
    - 선택을 받지 못한사람: 0
  - 진출 차수는 항상 1을 유지함

- 사이클에 포함되지 않는 노드의 수를 세는 문제
  - {전체 노드의 수} – {사이클에 포함되는 노드 수} 로 접근

BOJ9466: 텀 프로젝트

문제 분석

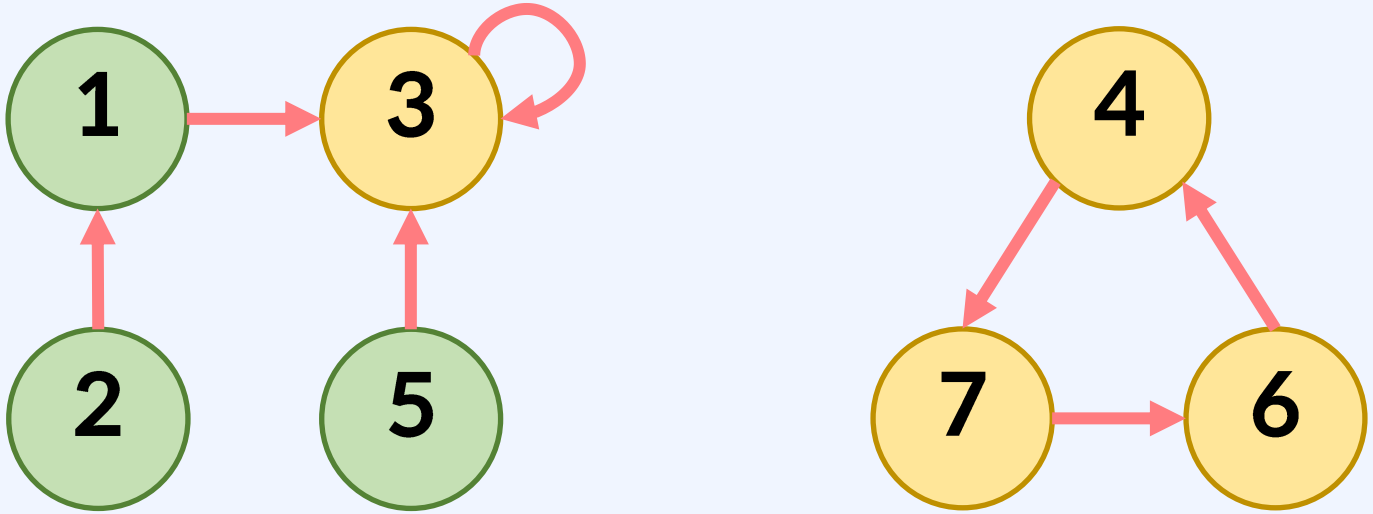
- 노드의 수가 2 <= N <= 100,000
  - 모든 노드에서 DFS를 수행하면 O(N^2) 로 시간초과

- 한번의 DFS로 구할 수 있어야 함

BOJ9466: 텀 프로젝트

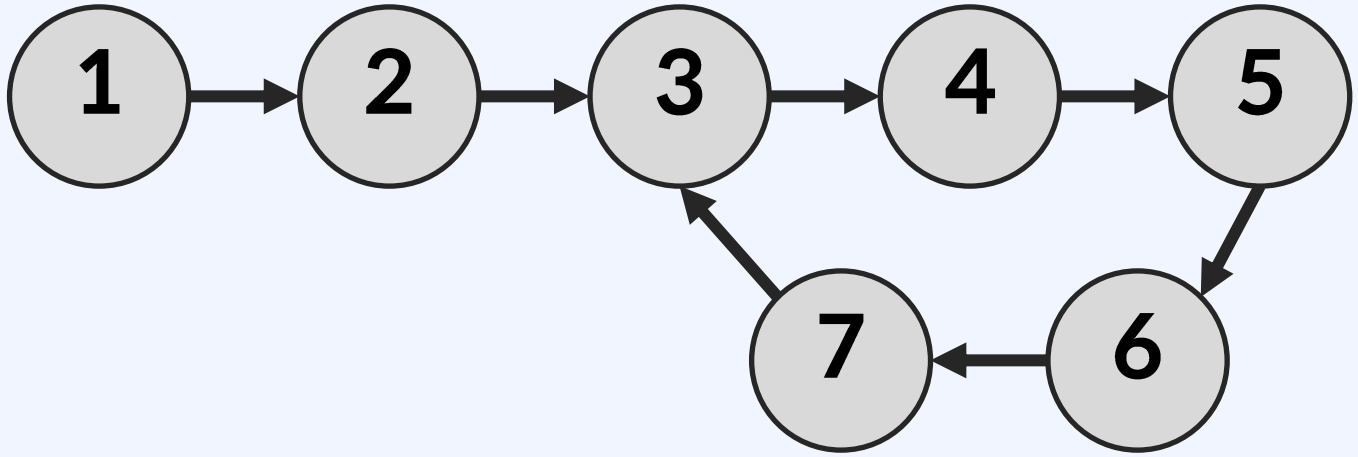
문제 분석

- 노드의 특징
  - 사이클에 포함되는 노드
  - 사이클에 포함되지 않는 노드
    - 사이클로 향하는 경로에 있음



BOJ9466: 텀 프로젝트

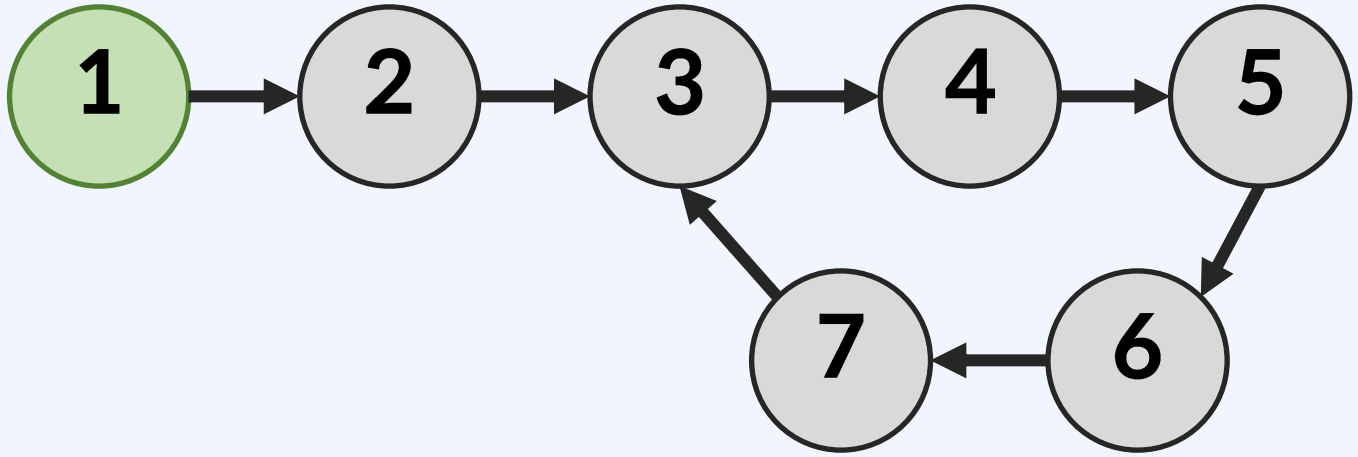
문제 분석



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	0	0	0	0	0	0	0

BOJ9466: 텀 프로젝트

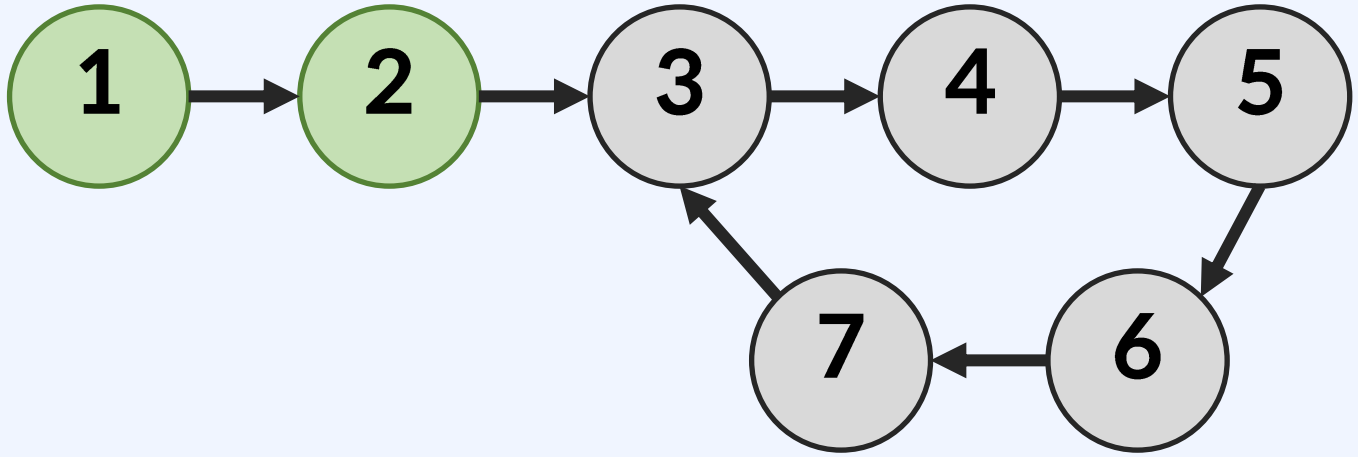
문제 분석



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	1	0	0	0	0	0	0

BOJ9466: 텀 프로젝트

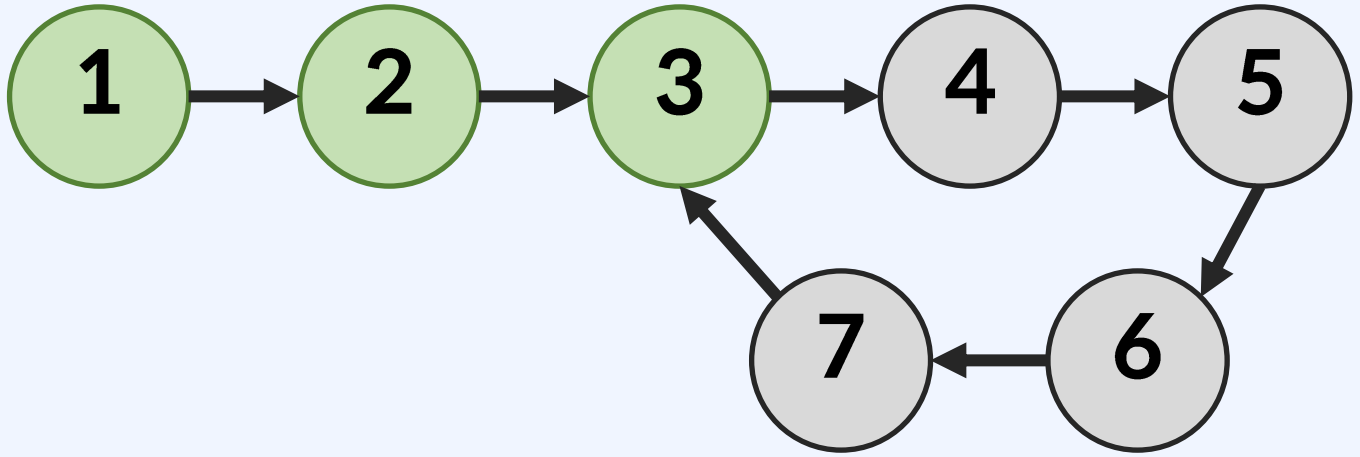
문제 분석



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	1	2	0	0	0	0	0

BOJ9466: 텀 프로젝트

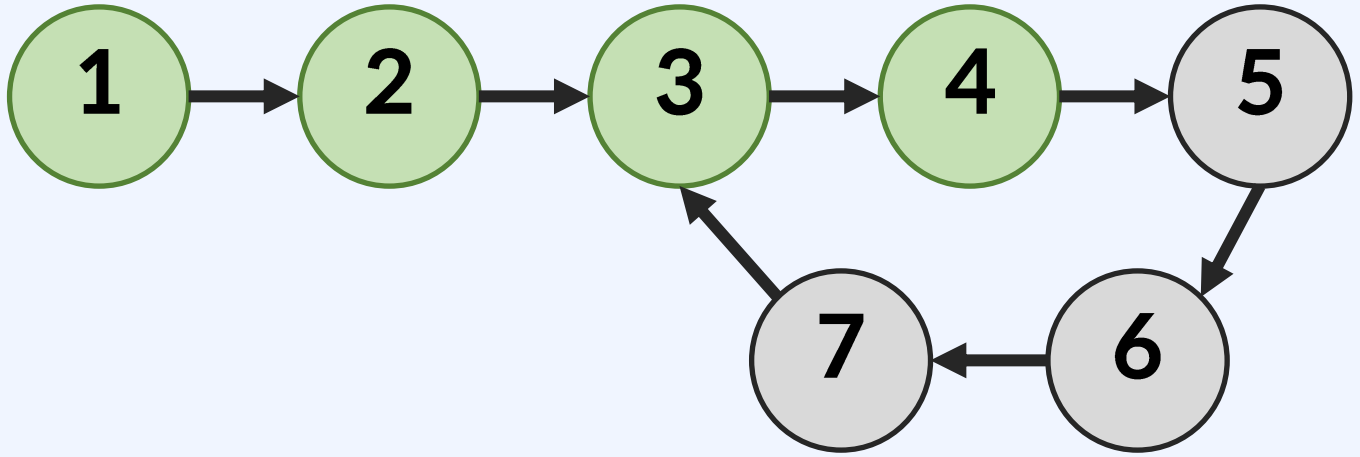
문제 분석



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	1	2	3	0	0	0	0

BOJ9466: 텀 프로젝트

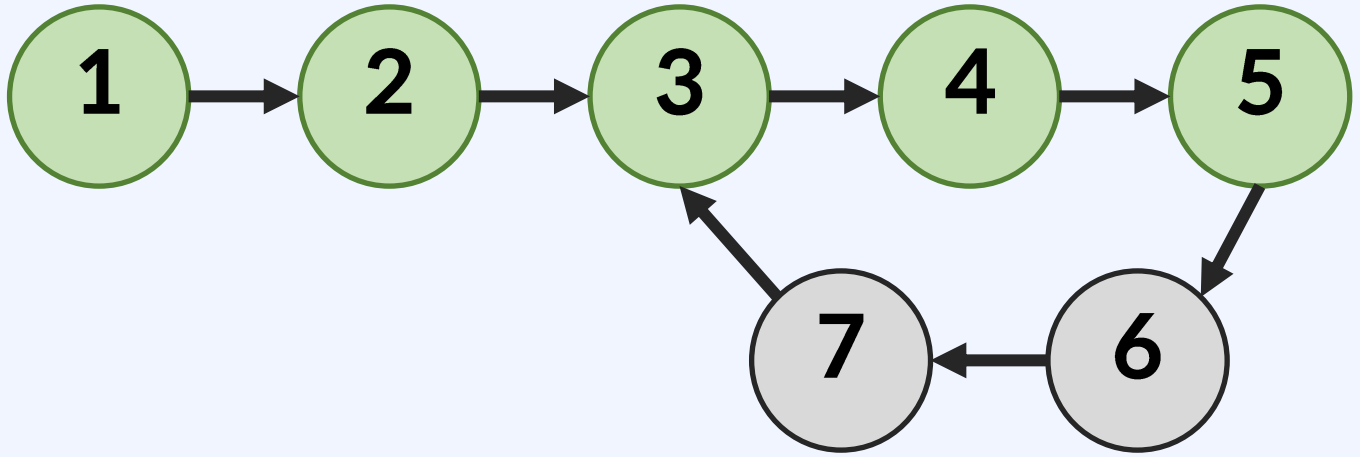
문제 분석



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	1	2	3	4	0	0	0

BOJ9466: 텀 프로젝트

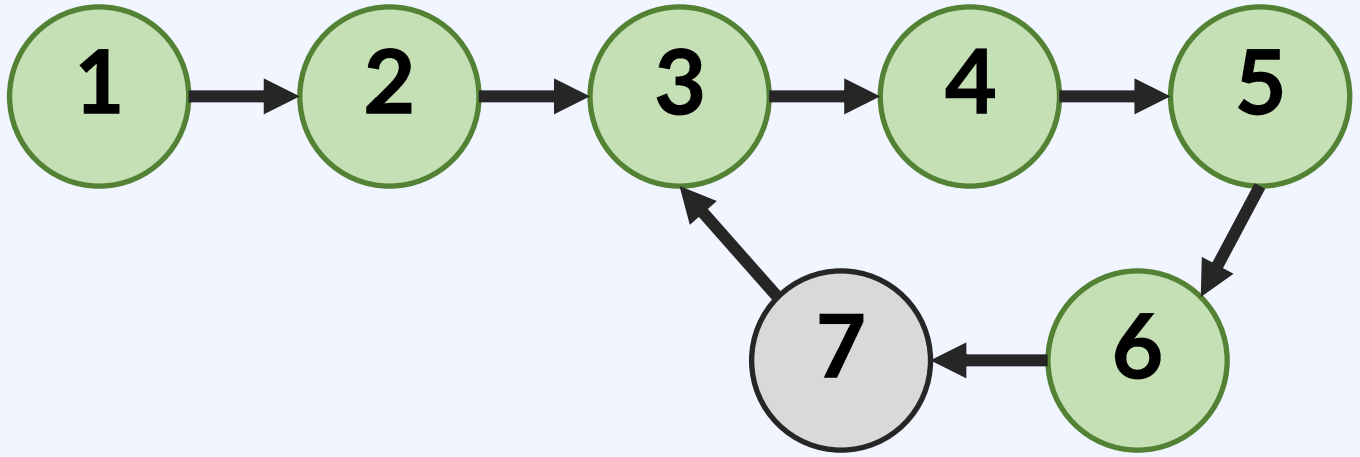
문제 분석



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	1	2	3	4	5	0	0

BOJ9466: 텀 프로젝트

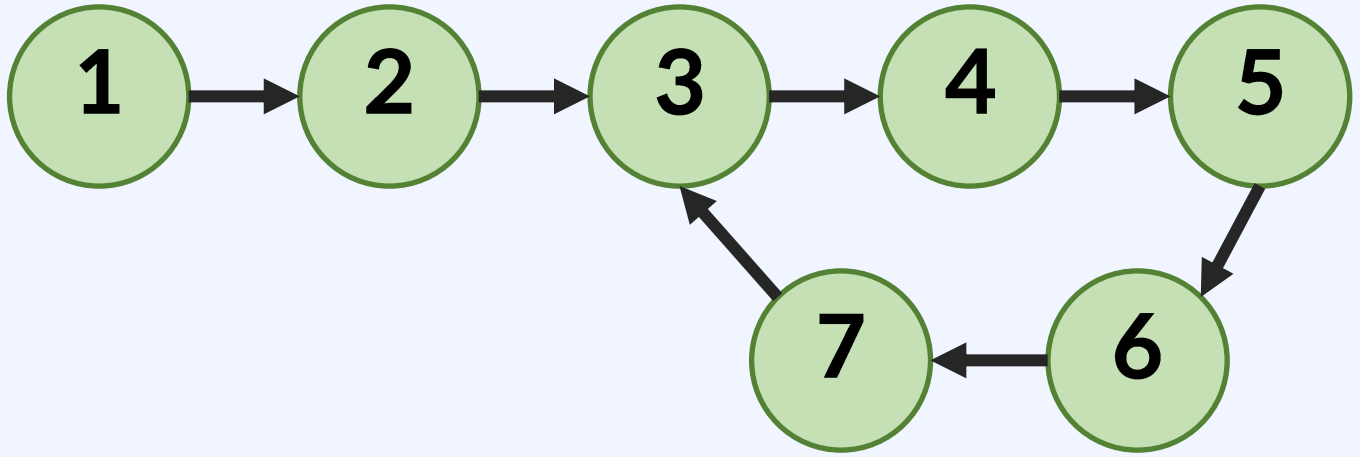
문제 분석



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	1	2	3	4	5	6	0

BOJ9466: 텀 프로젝트

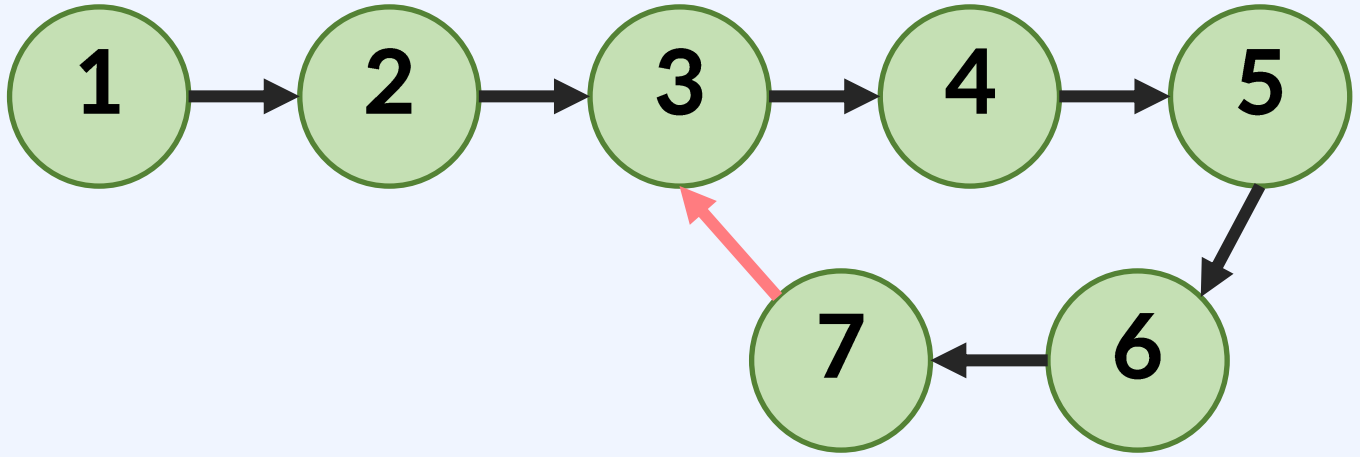
문제 분석



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	1	2	3	4	5	6	7

BOJ9466: 텀 프로젝트

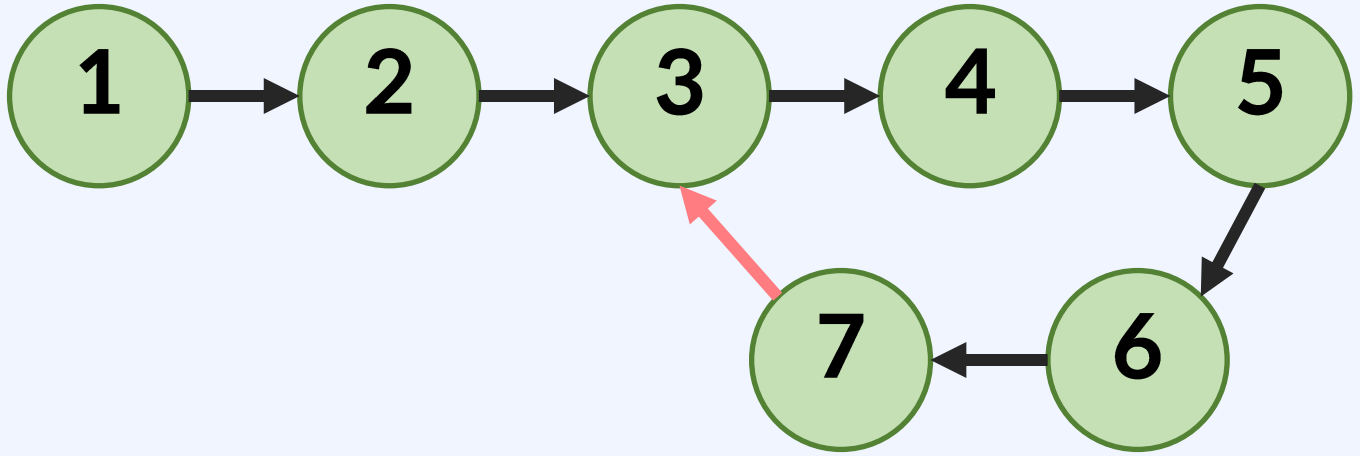
문제 분석



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	1	2	3	4	5	6	7

BOJ9466: 텀 프로젝트

문제 분석

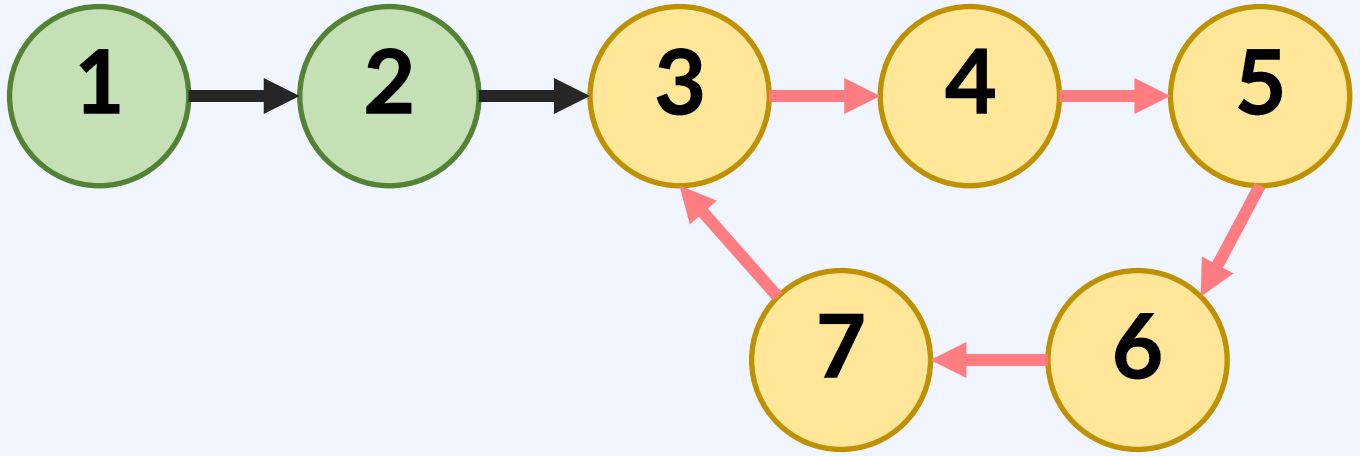


	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	1	2	3	4	5	6	7

다음 탐색할 노드의 Depth[next] 에 값이 들어 있다면? (3)

BOJ9466: 텀 프로젝트

문제 분석

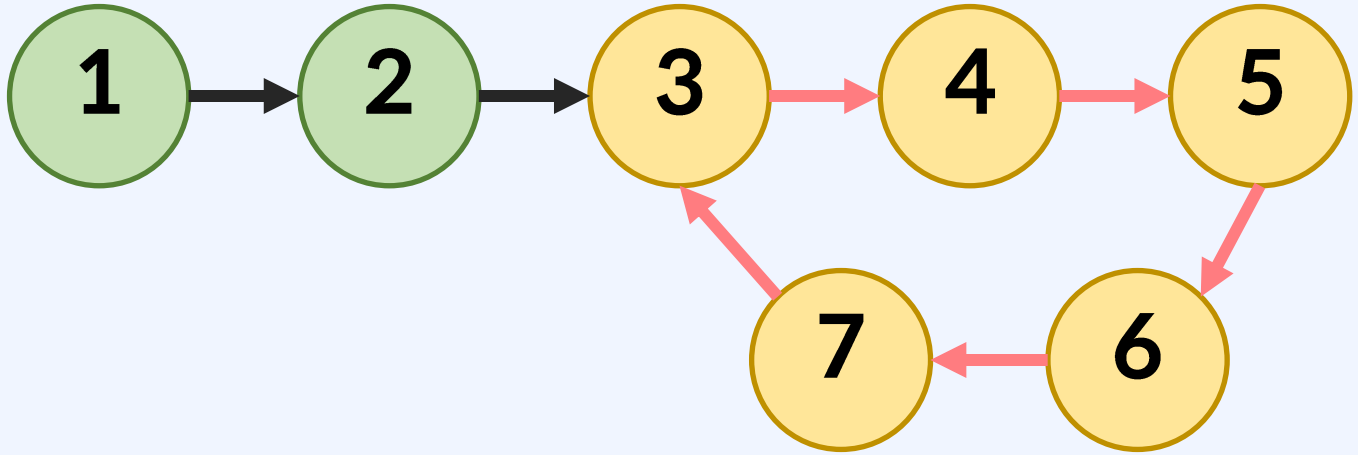


	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	1	2	3	4	5	6	7

Depth[next] ~ Depth[now] 는 모두 사이클 노드 (3~7)

BOJ9466: 텀 프로젝트

문제 분석



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Depth[]	1	2	3	4	5	6	7

사이클 노드의 개수
= depth[now] – depth[next] + 1 (7 - 3 + 1)

BOJ9466: 텀 프로젝트

구현

사이클 노드의 개수
= depth[now] – depth[next] + 1 (7 - 3 + 1)

```
public static int dfs(int nodeNum) {
    int next = nextNode[nodeNum];
    int cycleCnt = 0;
  // 첫 방문
     if (depth[next] == 0) {
       depth[next] = depth[nodeNum] + 1;
       cycleCnt = dfs(next);
    }
```

```
// 재방문 (사이클)
    else {
        cycleCnt = depth[nodeNum] - depth[next] + 1;
    }
// 다음 탐색을 위해 재귀 안에서 초기화
    depth[nodeNum] = 100001;
// 사이클이 아니면(음수) 0을 리턴
    return cycleCnt < 0 ? 0 : cycleCnt;
}
```