JavaScript DFS 알고리즘
DFS 문제 풀이

DFS 문제 풀이 | 코딩 테스트에서 자주 등장하는 DFS 알고리즘 이해하기

강사 나동빈

JavaScript
DFS 알고리즘

DFS 문제 풀이

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

문제 제목: 텀 프로젝트

문제 난이도: ★★★☆☆

문제 유형: 깊이 우선 탐색, 방향 그래프 내 사이클 판별

추천 풀이 시간: 60분

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 모든 학생들은 프로젝트를 함께 하고 싶은 학생을 **한 명씩 선택**한다.
- 자기 자신을 선택하는 것도 가능하다.
- 결과적으로 어느 프로젝트 팀에도 속하지 못한 학생들의 수를 계산해야 한다.

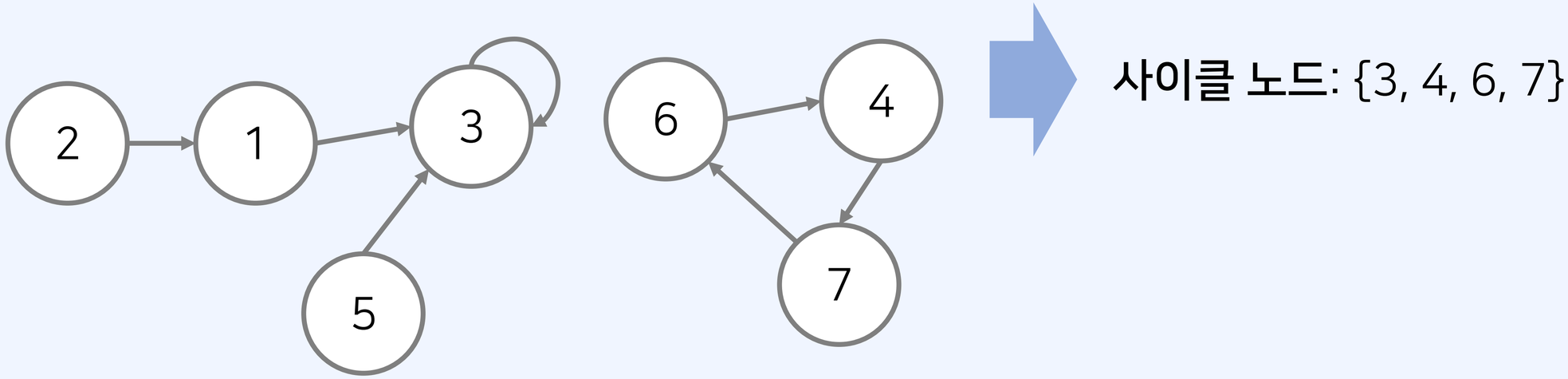
텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

[핵심 아이디어]

- 모든 학생들은 자신이 원하는 학생과 같은 팀에 소속되고자 한다.

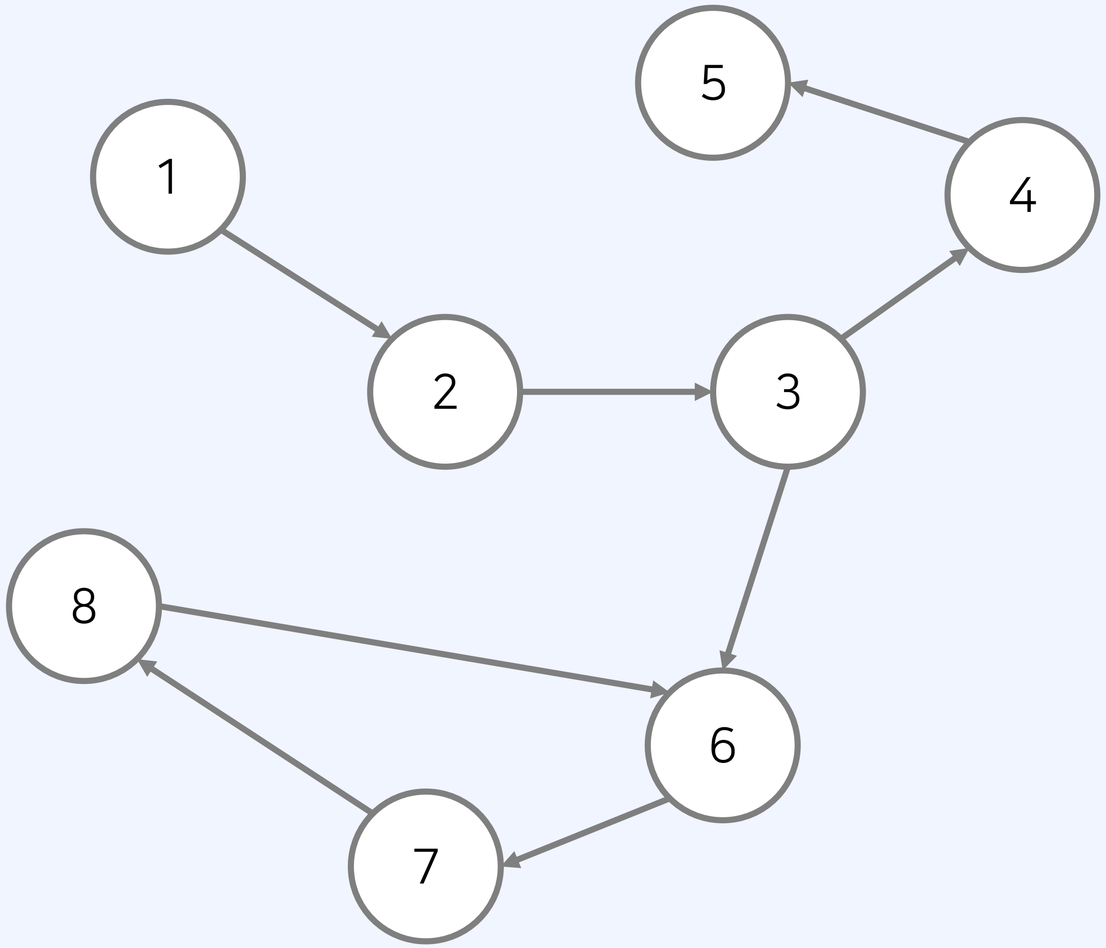
  각 학생들의 선택을 **방향 간선으로 표현하여 그래프를 구성**할 수 있다.
- 한 팀에 포함된 임의의 학생 A와 B가 있을 때, A에서 B로 도달할 수 있어야 한다.
- 즉, 본 문제는 **사이클(cycle)을 구성하는 부분 그래프에 포함된 노드의 개수를 세는 문제**다.

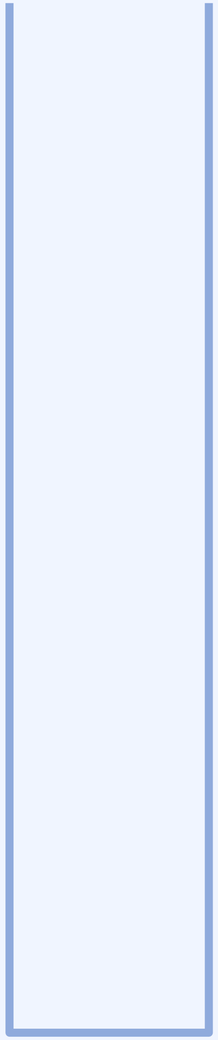


텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: {}
- 방문한 노드: {}

: 방문한 노드

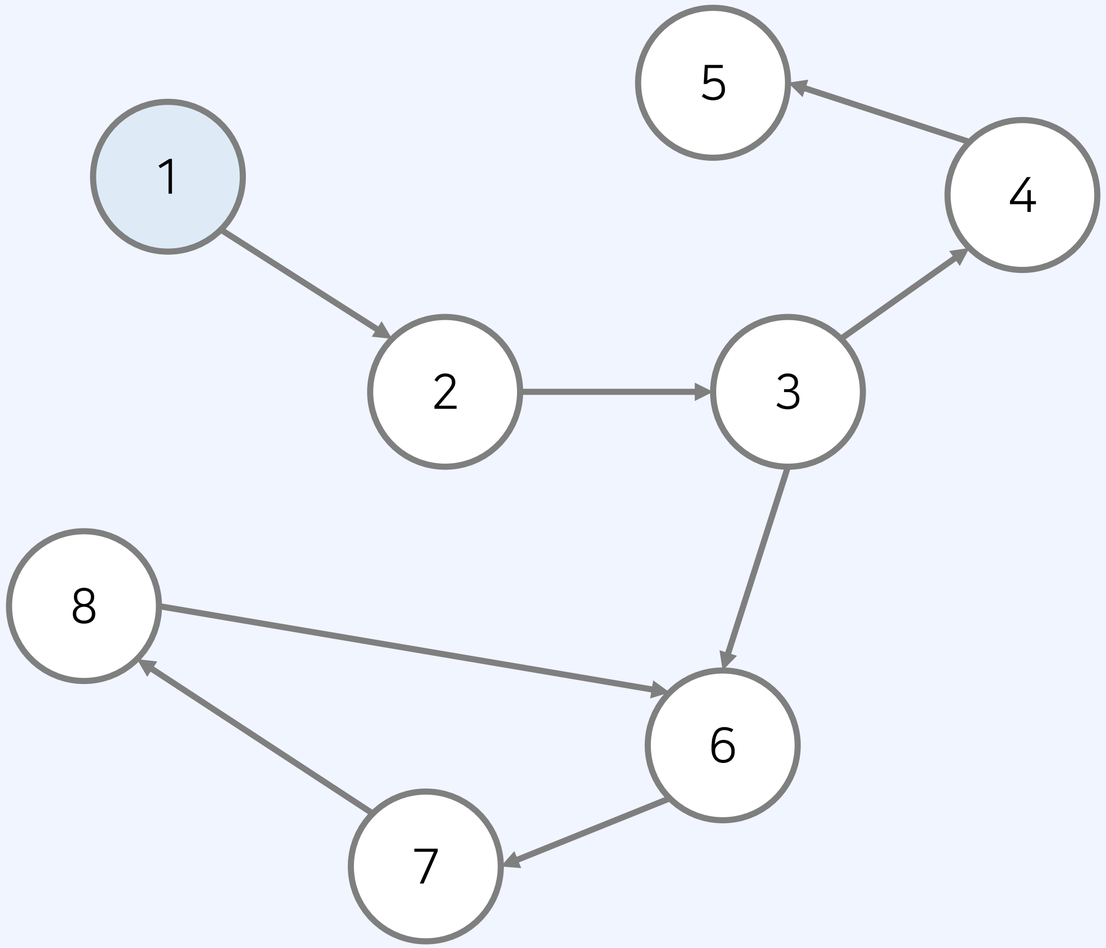
: 처리가 완료된 노드 (스택에서 추출된)

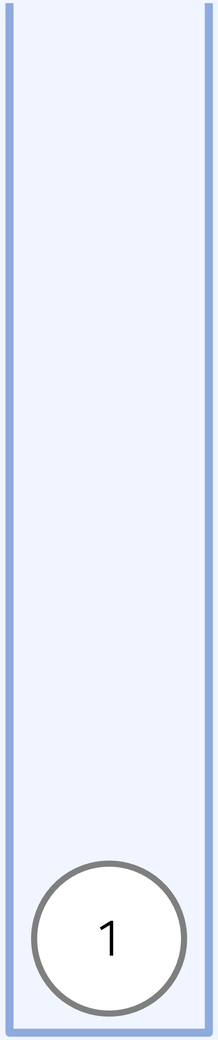
[해설] 전체 그래프를 확인한다.

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: {}
- 방문한 노드: {1}

: 방문한 노드

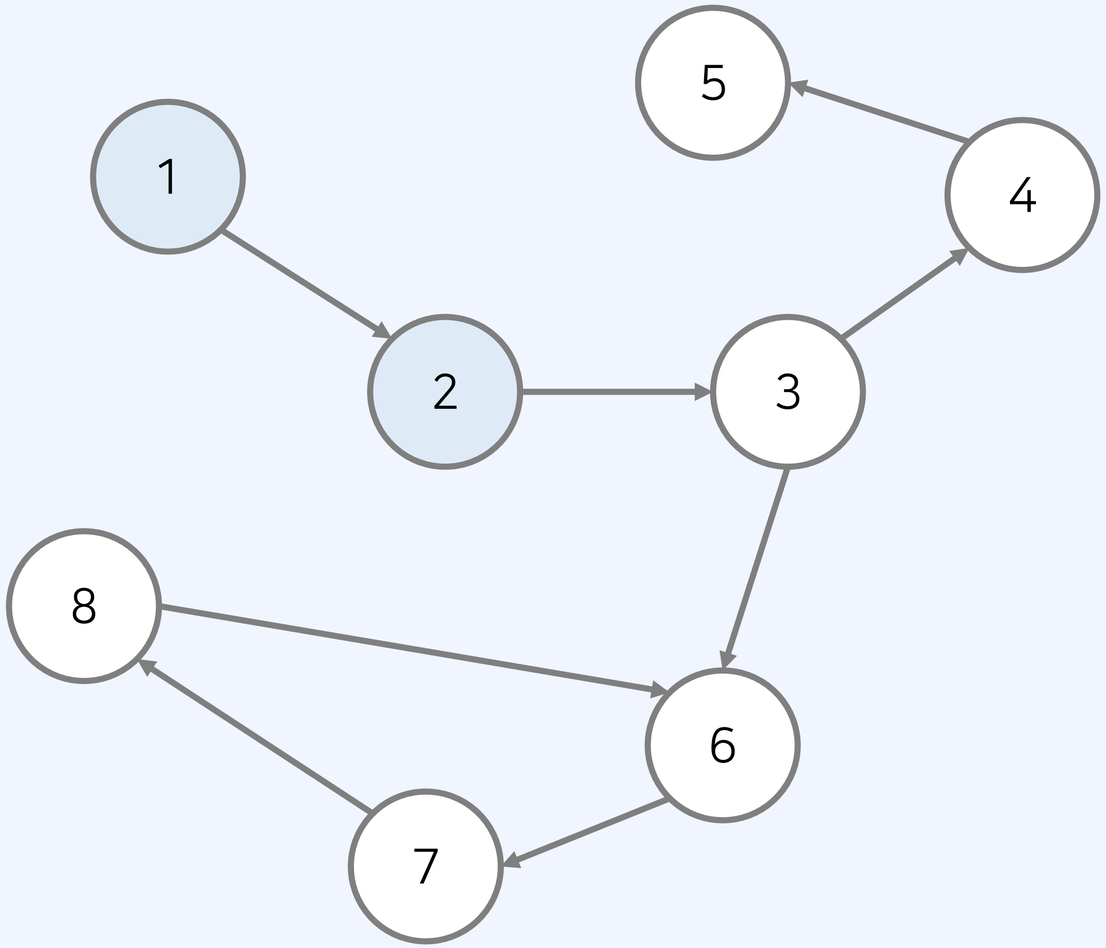
: 처리가 완료된 노드 (스택에서 추출된)

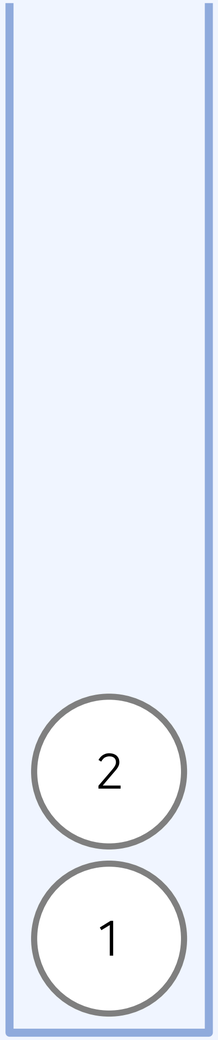
[해설] 노드 1을 방문해 스택에 삽입한다.

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: {}
- 방문한 노드: $\{1, 2\}$

: 방문한 노드

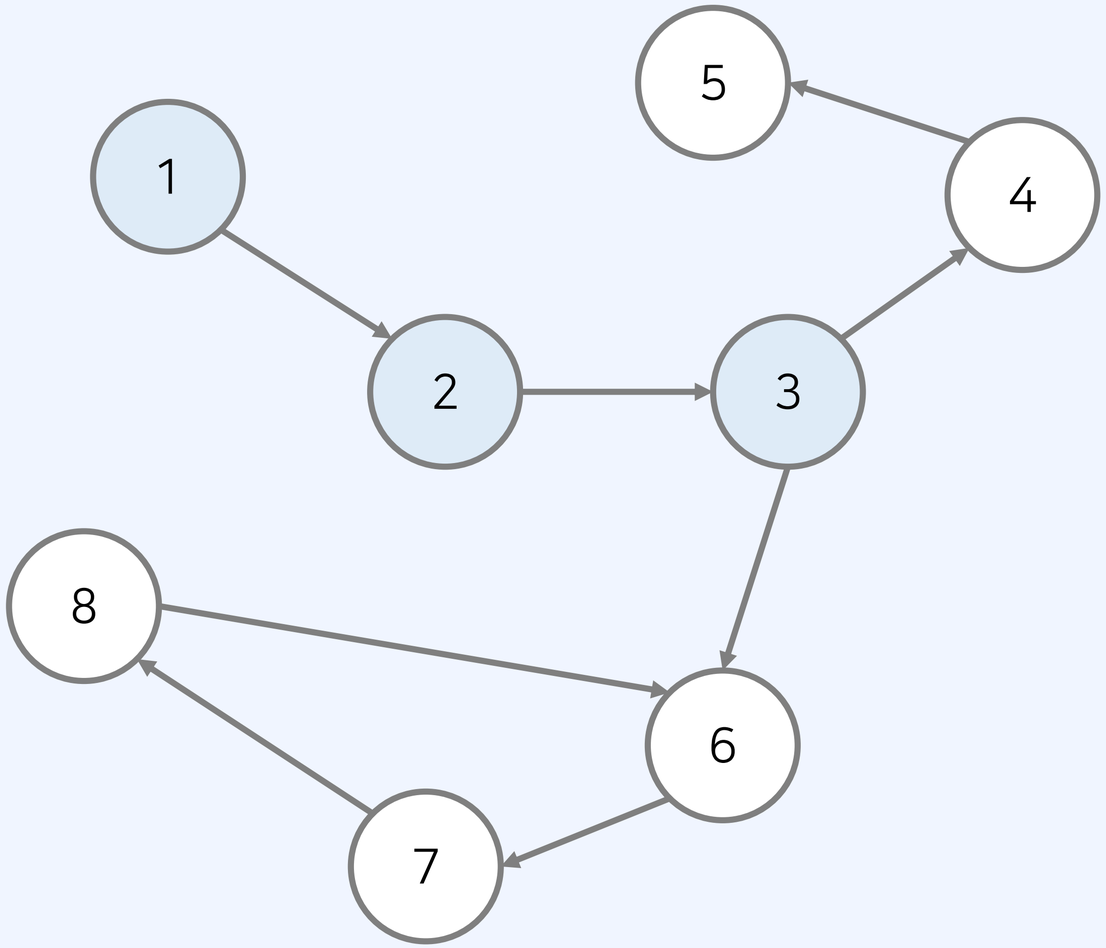
: 처리가 완료된 노드 (스택에서 추출된)

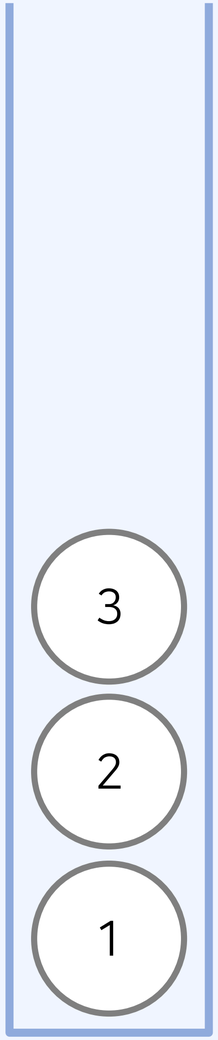
[해설] 스택의 최상단 노드와 인접한 **노드 2**를 방문해 스택에 삽입한다.

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: {}
- 방문한 노드: $\{1, 2, 3\}$

: 방문한 노드

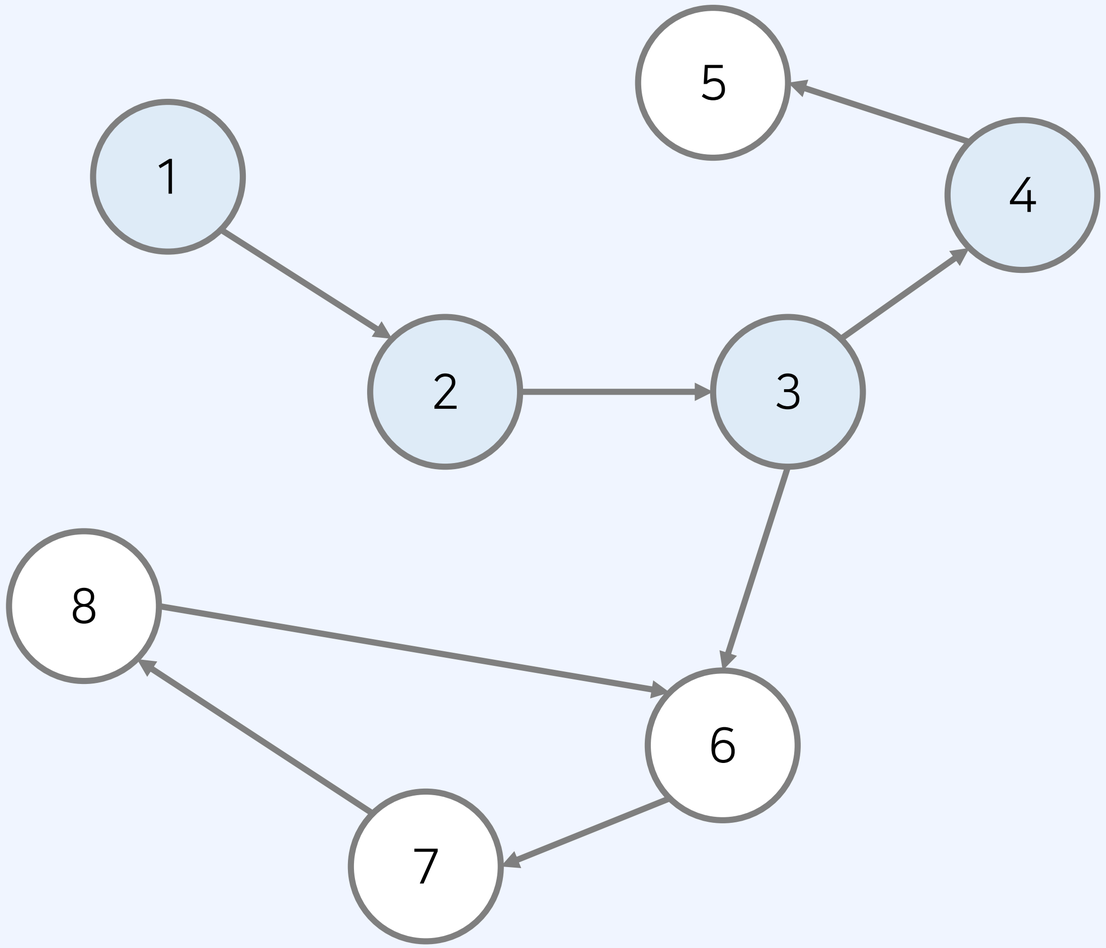
: 처리가 완료된 노드 (스택에서 추출된)

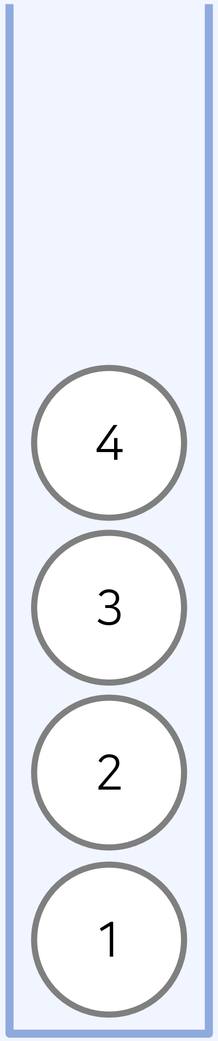
[해설] 스택의 최상단 노드와 인접한 **노드 3**을 방문해 스택에 삽입한다.

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: {}
- 방문한 노드: $\{1, 2, 3, 4\}$

: 방문한 노드

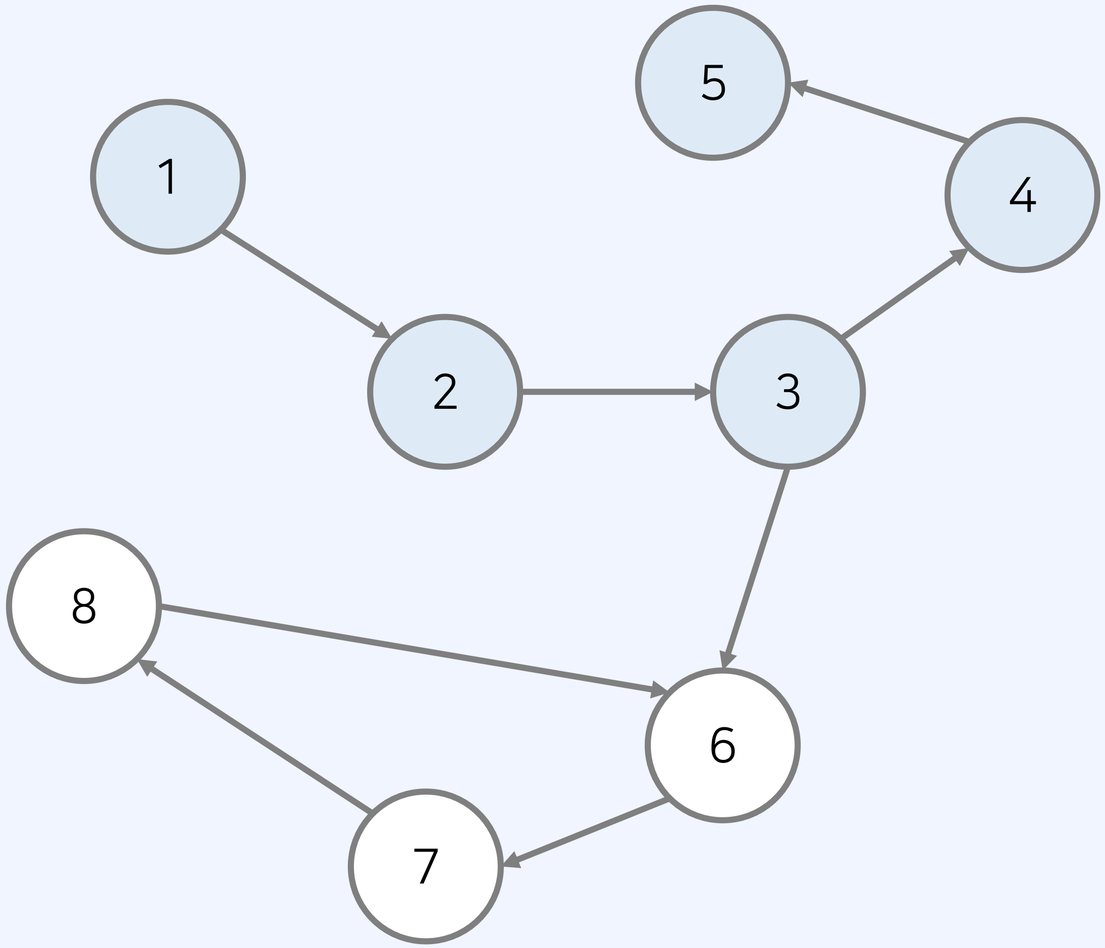
: 처리가 완료된 노드 (스택에서 추출된)

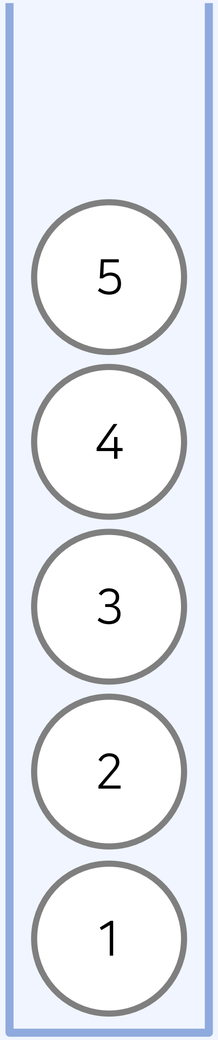
[해설] 스택의 최상단 노드와 인접한 **노드 4**를 방문해 스택에 삽입한다.

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: $\{\}$
- 방문한 노드: $\{1, 2, 3, 4, 5\}$

: 방문한 노드

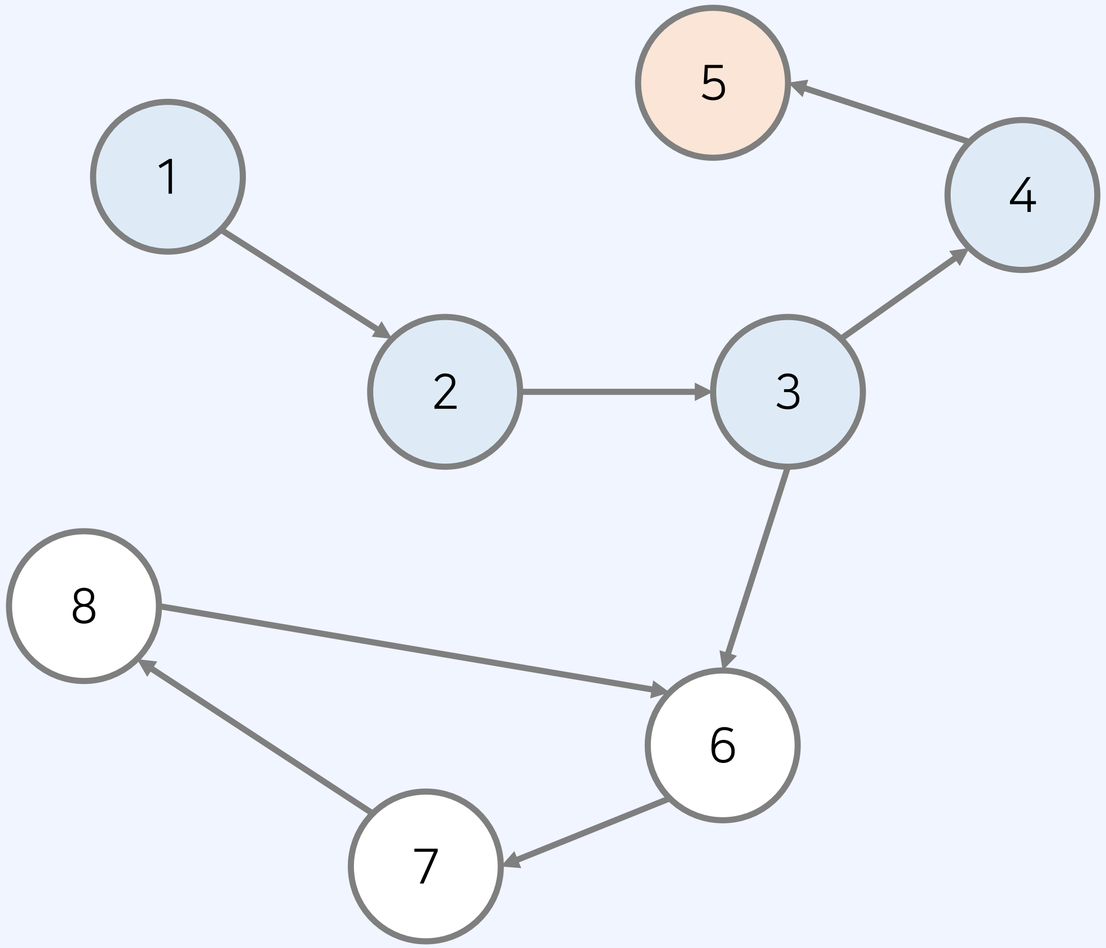
: 처리가 완료된 노드 (스택에서 추출된)

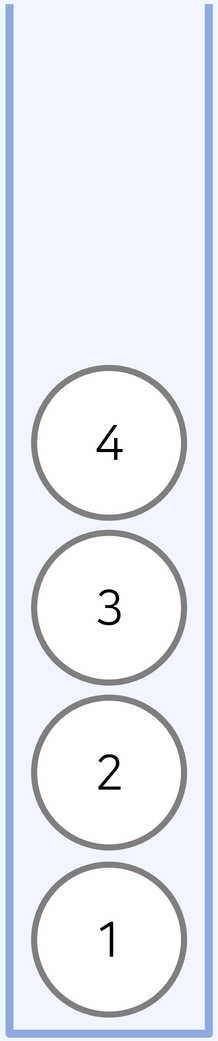
[해설] 스택의 최상단 노드와 인접한 **노드 5**를 방문해 스택에 삽입한다.

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: $\{5\}$
- 방문한 노드: $\{1, 2, 3, 4, 5\}$

: 방문한 노드

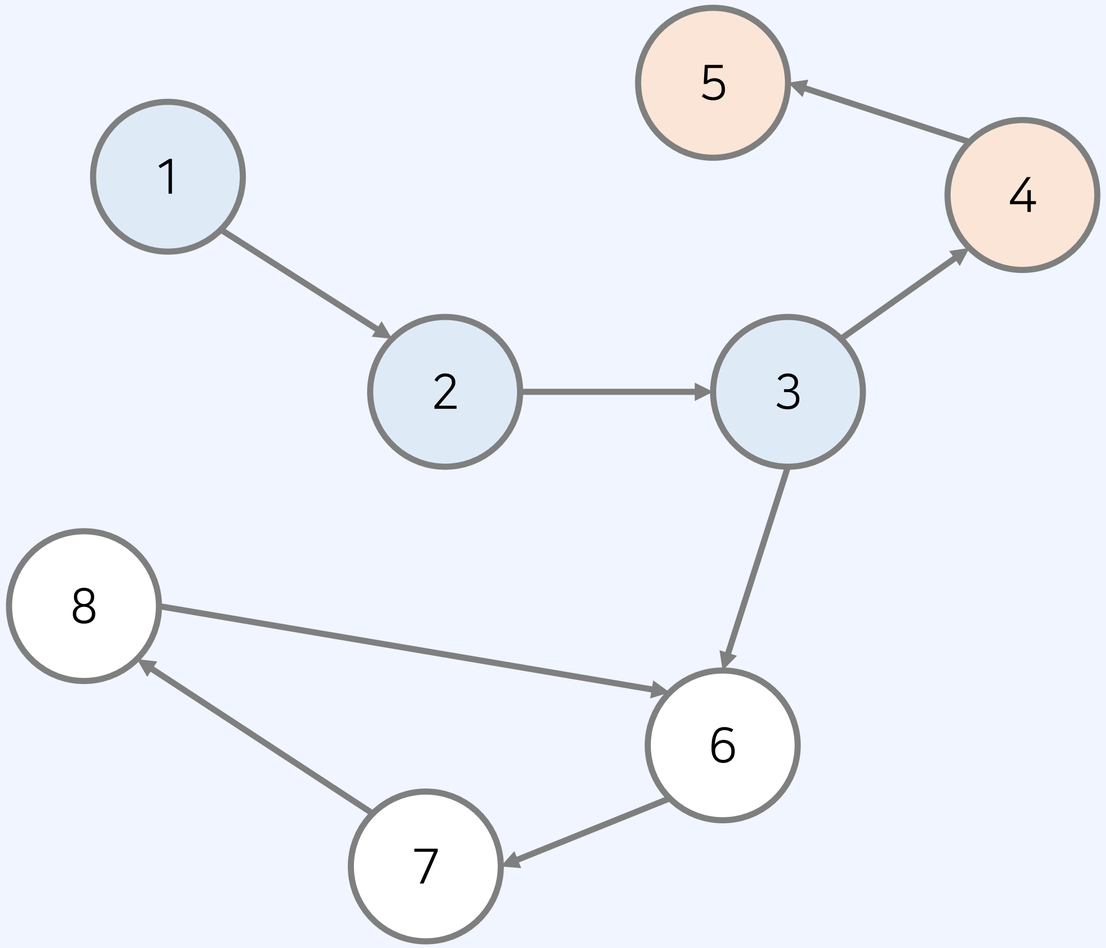
: 처리가 완료된 노드 (스택에서 추출된)

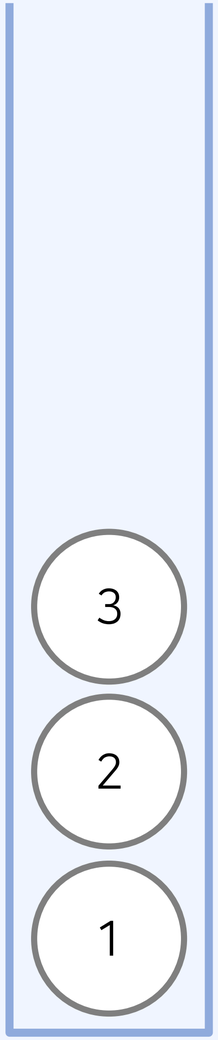
[해설] 스택의 최상단 노드에게
방문하지 않은 인접 노드가 없으므로
최상단 노드를 추출한다.

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: $\{4, 5\}$
- 방문한 노드: $\{1, 2, 3, 4, 5\}$

: 방문한 노드

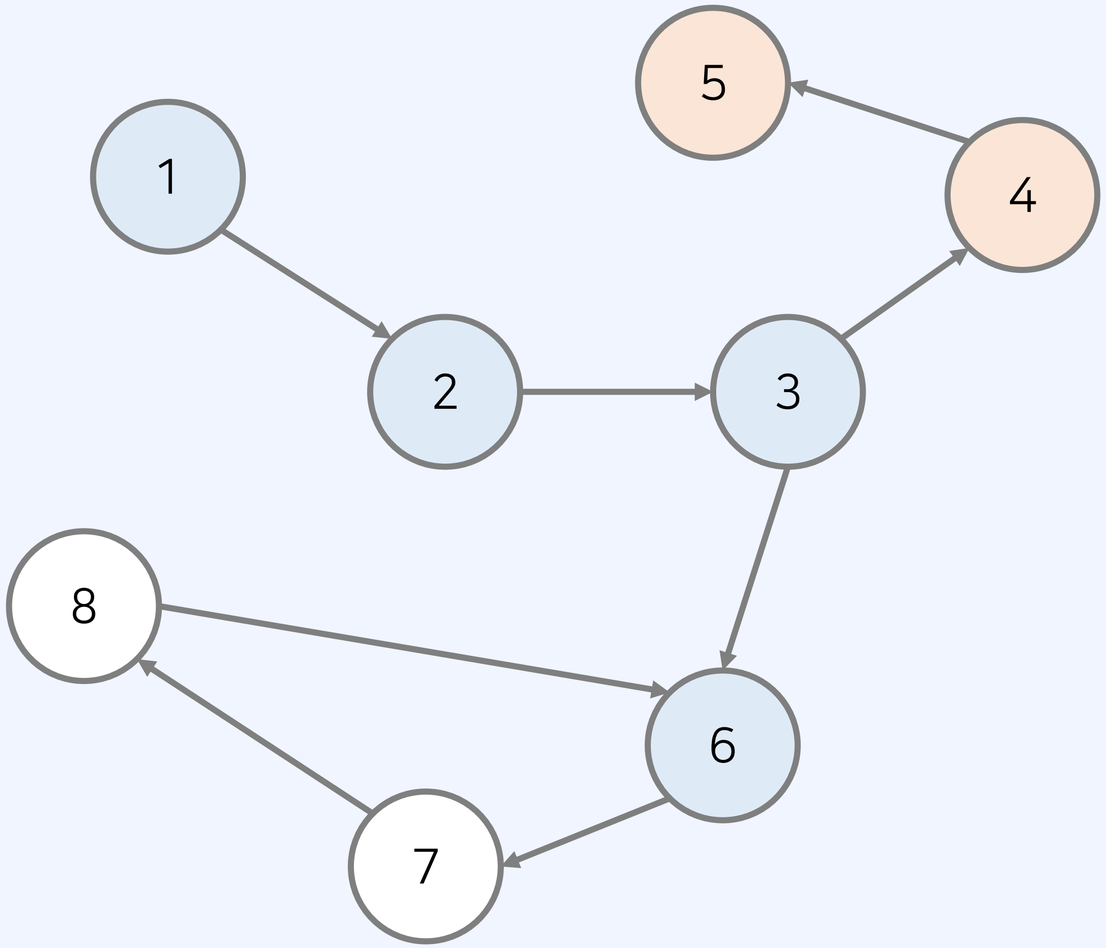
: 처리가 완료된 노드 (스택에서 추출된)

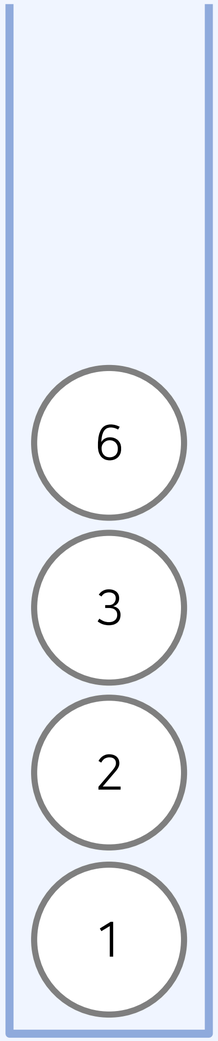
[해설] 스택의 최상단 노드에게
방문하지 않은 인접 노드가 없으므로
최상단 노드를 추출한다.

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: $\{4, 5\}$
- 방문한 노드: $\{1, 2, 3, 4, 5, 6\}$

: 방문한 노드

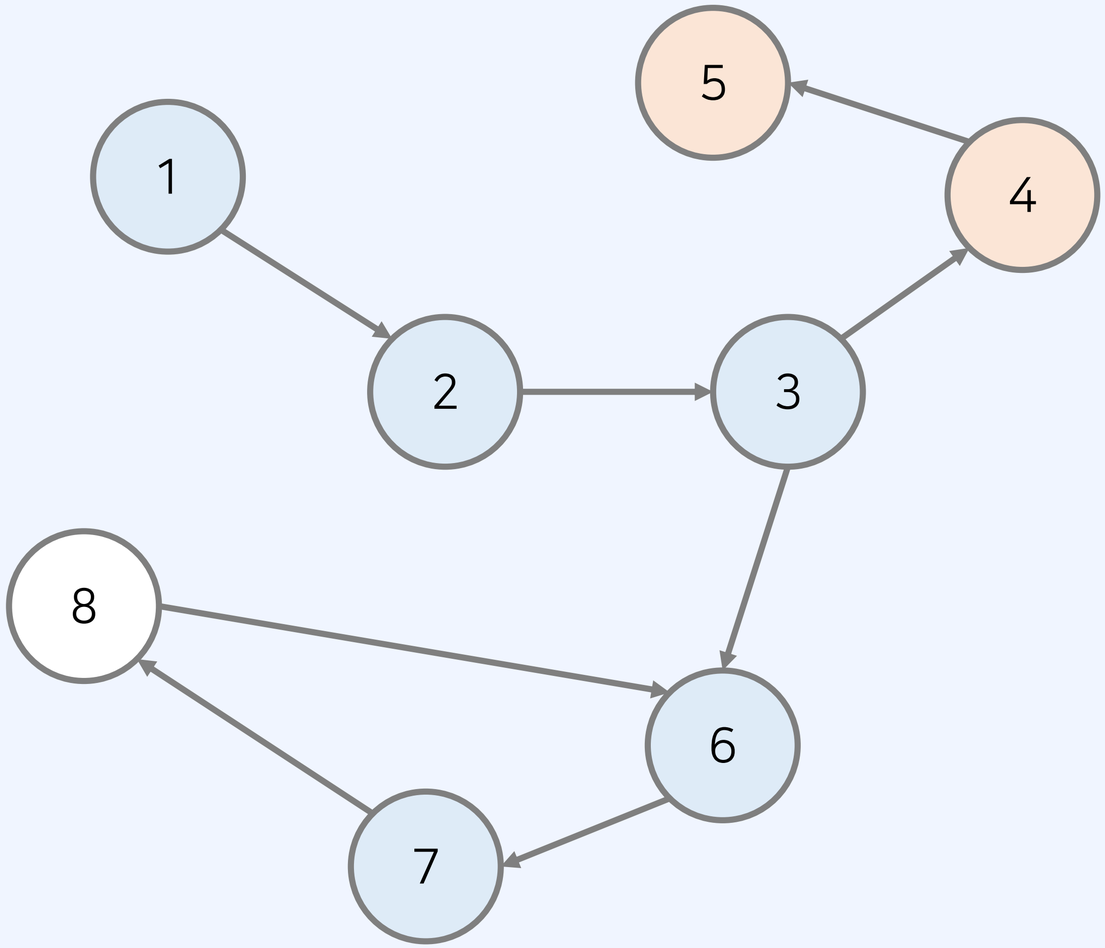
: 처리가 완료된 노드 (스택에서 추출된)

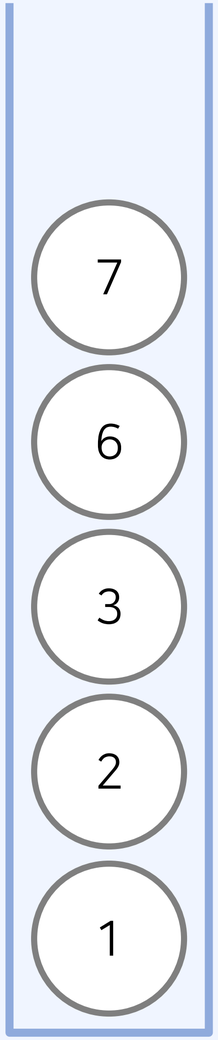
[해설] 스택의 최상단 노드와 인접한 **노드 6**를 방문해 스택에 삽입한다.

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: $\{4, 5\}$
- 방문한 노드: $\{1, 2, 3, 4, 5, 6, 7\}$

: 방문한 노드

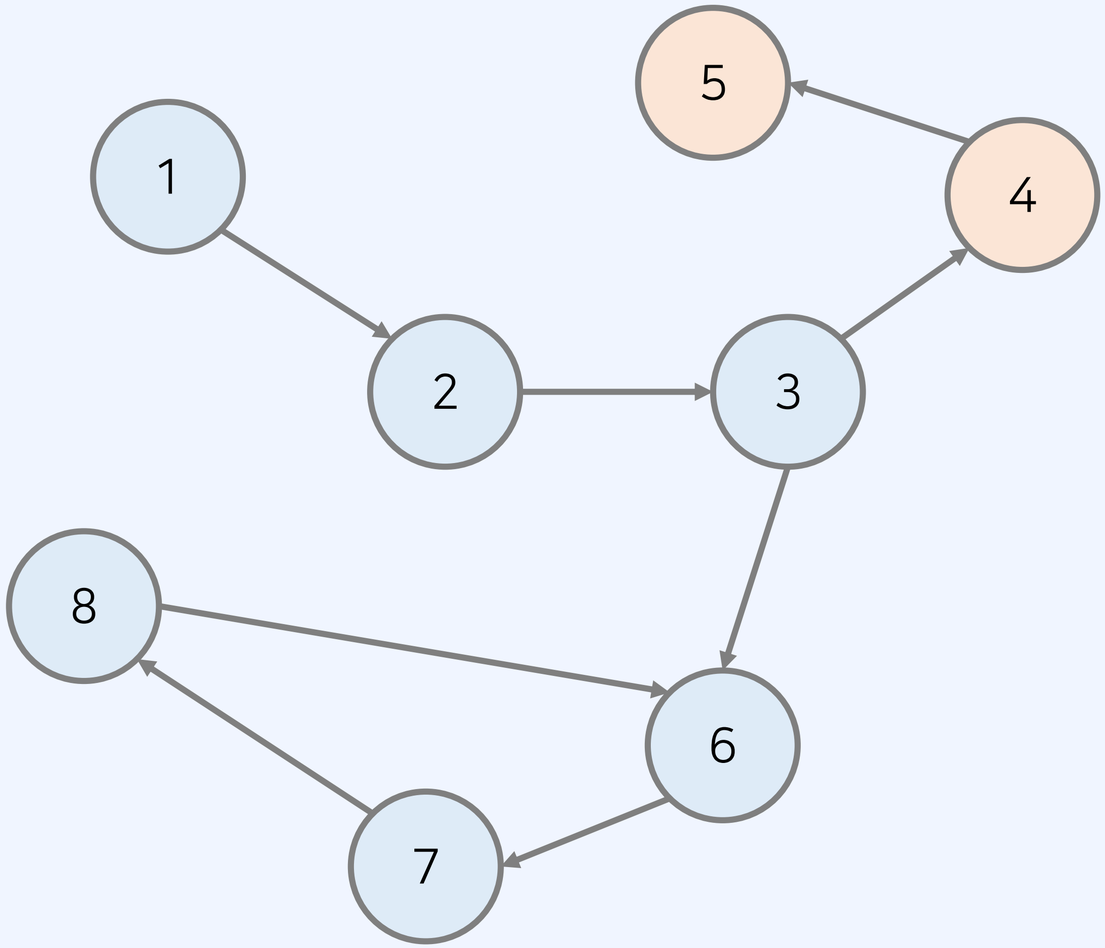
: 처리가 완료된 노드 (스택에서 추출된)

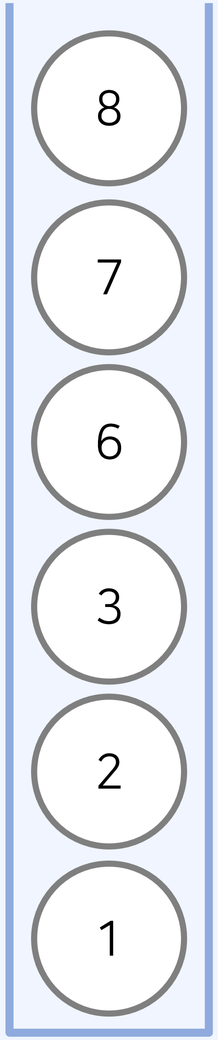
[해설] 스택의 최상단 노드와 인접한 **노드 7**을 방문해 스택에 삽입한다.

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: $\{4, 5\}$
- 방문한 노드: $\{1, 2, 3, 4, 5, 6, 7, 8\}$

: 방문한 노드

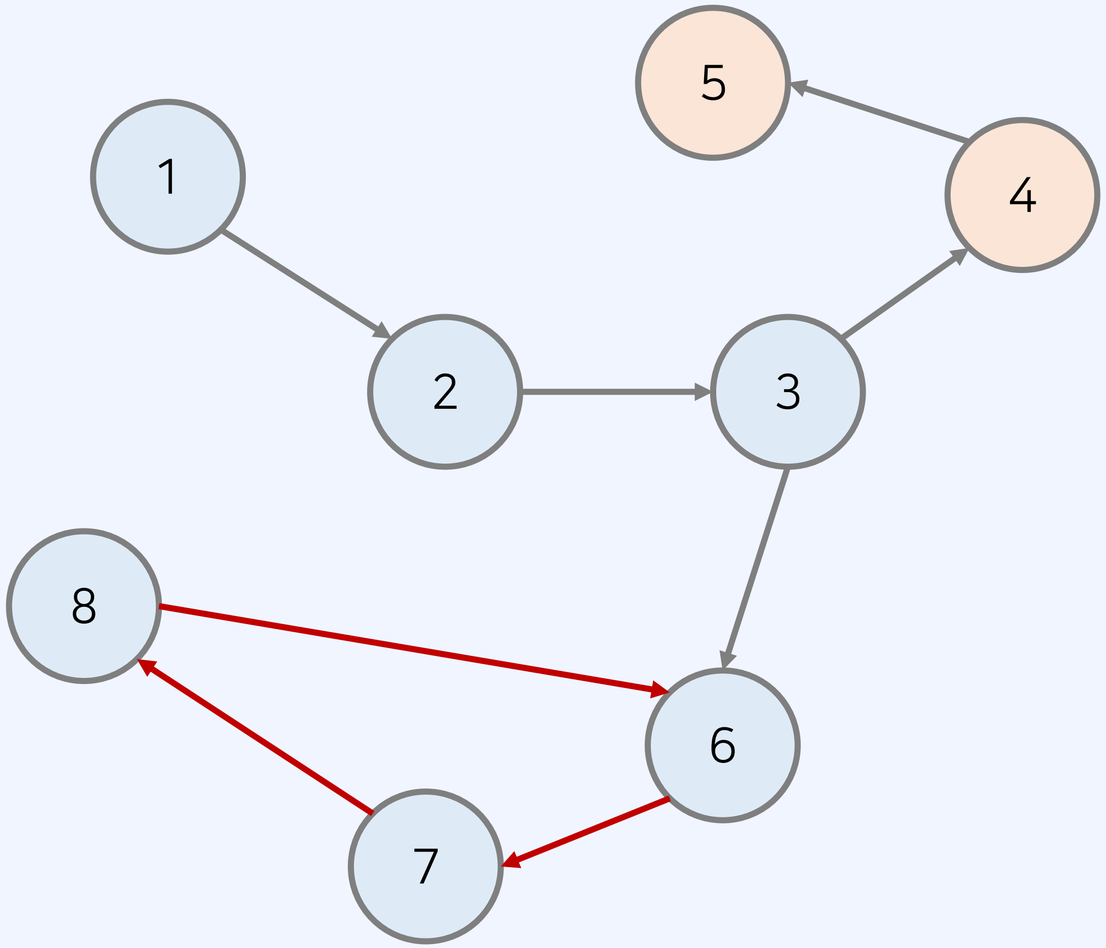
: 처리가 완료된 노드 (스택에서 추출된)

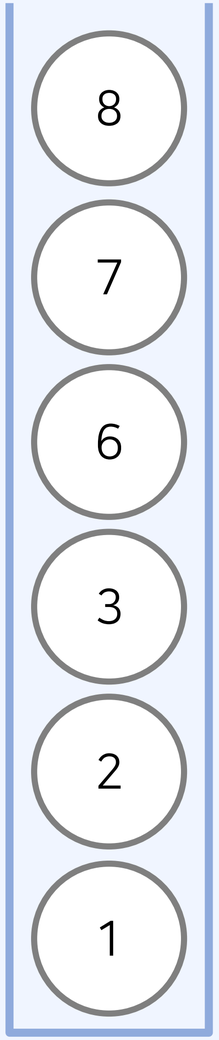
[해설] 스택의 최상단 노드와 인접한 **노드 8**을 방문해 스택에 삽입한다.

텀 프로젝트: https://www.acmicpc.net/problem/9466

혼자 힘으로 풀어보기

- 방향 그래프 내 사이클 판별





- 처리가 완료된 노드: $\{4, 5\}$
- 방문한 노드: $\{1, 2, 3, 4, 5, 6, 7, 8\}$

: 방문한 노드

: 처리가 완료된 노드 (스택에서 추출된)

[해설] 스택의 최상단 노드와 인접한 **노드 6**을 이미 방문한 적이 있으며 **처리가 완료되지 않았다.**

사이클 발생

정답 코드 예시


```
function dfs(x, graph, visited, finished, result) {
  visited[x] = true; // 현재 노드 방문 처리
  let y = graph[x]; // 다음 노드
  if (!visited[y]) { // 다음 노드를 아직 방문하지 않았다면
    dfs(y, graph, visited, finished, result);
  }
  // 다음 노드를 방문한 적 있고, 완료되지 않았다면
  else if (!finished[y]) {
    // 사이클이 발생한 것이므로 사이클에 포함된 노드 저장
    while (y != x) {
      result.push(y);
      y = graph[y];
    }
    result.push(x);
  }
  finished[x] = true; // 현재 노드의 처리가 완료됨
}
```

텀 프로젝트: https://www.acmicpc.net/problem/9466

정답 코드 예시


```
let file = require('fs').readFileSync('/dev/stdin');
let input = file.toString().split('\n');

let testCases = Number(input[0]); // 테스트 케이스의 수
let line = 1;
while (testCases--) {
  let n = Number(input[line]);
  let graph = [0, ...input[line + 1].split(' ').map(Number)];
  let visited = new Array(n + 1).fill(false);
  let finished = new Array(n + 1).fill(false);
  let result = [];

  for (let x = 1; x <= n; x++) { // 각 위치에서 연결 요소 계산 및 사이클 판단
    if (!visited[x]) dfs(x, graph, visited, finished, result);
  }

  line += 2; // 다음 테스트 케이스로 이동
  console.log(n - result.length);
}
```

숫자고르기: https://www.acmicpc.net/problem/2668

혼자 힘으로 풀어보기

문제 제목: 숫자고르기

문제 난이도: ★★★☆☆

문제 유형: 깊이 우선 탐색, 방향 그래프 내 사이클 판별

추천 풀이 시간: 60분

숫자고르기: https://www.acmicpc.net/problem/2668

혼자 힘으로 풀어보기

- 첫째 줄에서 뽑은 정수들이 이루는 집합 A와 뽑힌 정수들의 바로 밑에 있는 정수들이 이루는 집합 B가 **일치**하도록 하는 집합 A의 **최대 크기를 계산**한다.
- 아래 예시에서는 $A = \{1, 3, 5\}$일 때 $B = \{3, 1, 5\}$이며, 이것이 최대 크기다.

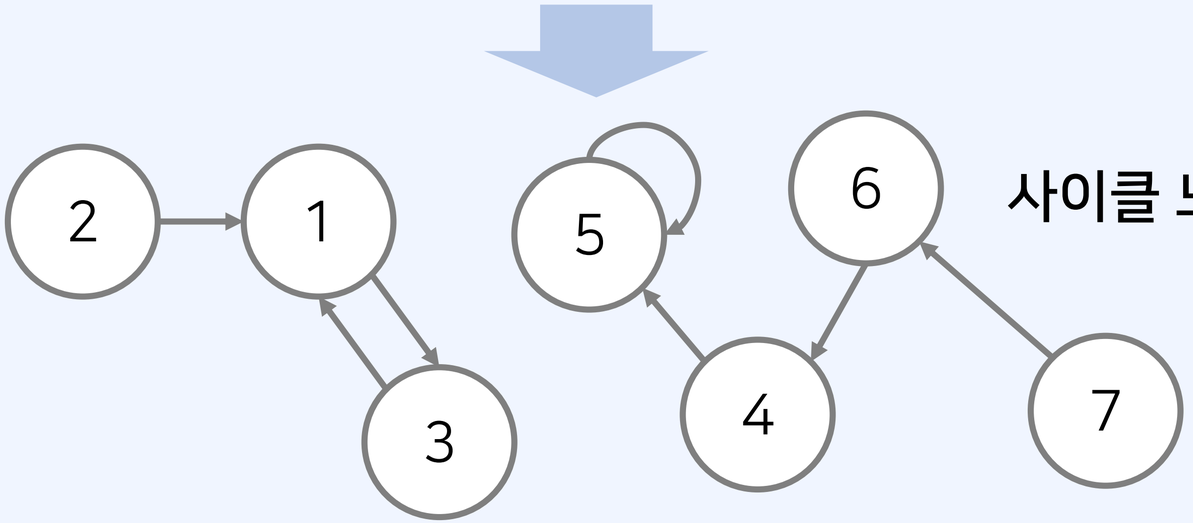
1	2	3	4	5	6	7
3	1	1	5	5	4	6

숫자고르기: https://www.acmicpc.net/problem/2668

혼자 힘으로 풀어보기

- 첫째 줄과 둘째 줄의 관계를 **방향 간선으로 표현하여 그래프를 구성**할 수 있다.
- [핵심] 본 문제는 **사이클(cycle)을 구성하는 부분 그래프에 포함된 노드의 개수를 세는 문제**다.

1	2	3	4	5	6	7
3	1	1	5	5	4	6



사이클 노드: {1, 3, 5}

숫자고르기: https://www.acmicpc.net/problem/2668

정답 코드 예시


```
function dfs(x, graph, visited, finished, result) {
  visited[x] = true; // 현재 노드 방문 처리
  let y = graph[x]; // 다음 노드
  if (!visited[y]) { // 다음 노드를 아직 방문하지 않았다면
    dfs(y, graph, visited, finished, result);
  }
  // 다음 노드를 방문한 적 있고, 완료되지 않았다면
  else if (!finished[y]) {
    // 사이클이 발생한 것이므로 사이클에 포함된 노드 저장
    while (y != x) {
      result.push(y);
      y = graph[y];
    }
    result.push(x);
  }
  finished[x] = true; // 현재 노드의 처리가 완료됨
}
```

숫자고르기: https://www.acmicpc.net/problem/2668

정답 코드 예시


```
let file = require('fs').readFileSync('/dev/stdin');
let input = file.toString().split('\n');

let n = Number(input[0]);
let graph = [0];
for (let i = 1; i <= n; i++) {
  graph.push(Number(input[i]));
}
let visited = new Array(n + 1).fill(false);
let finished = new Array(n + 1).fill(false);
let result = [];

for (let x = 1; x <= n; x++) {
  if (!visited[x]) dfs(x, graph, visited, finished, result);
}

console.log(result.length);
result.sort((a, b) => a - b);
for (let x of result) console.log(x);
```