ANSI 알고리즘 스터디

02 DFS

01 그래프 이론

02 깊이 우선 탐색 DFS

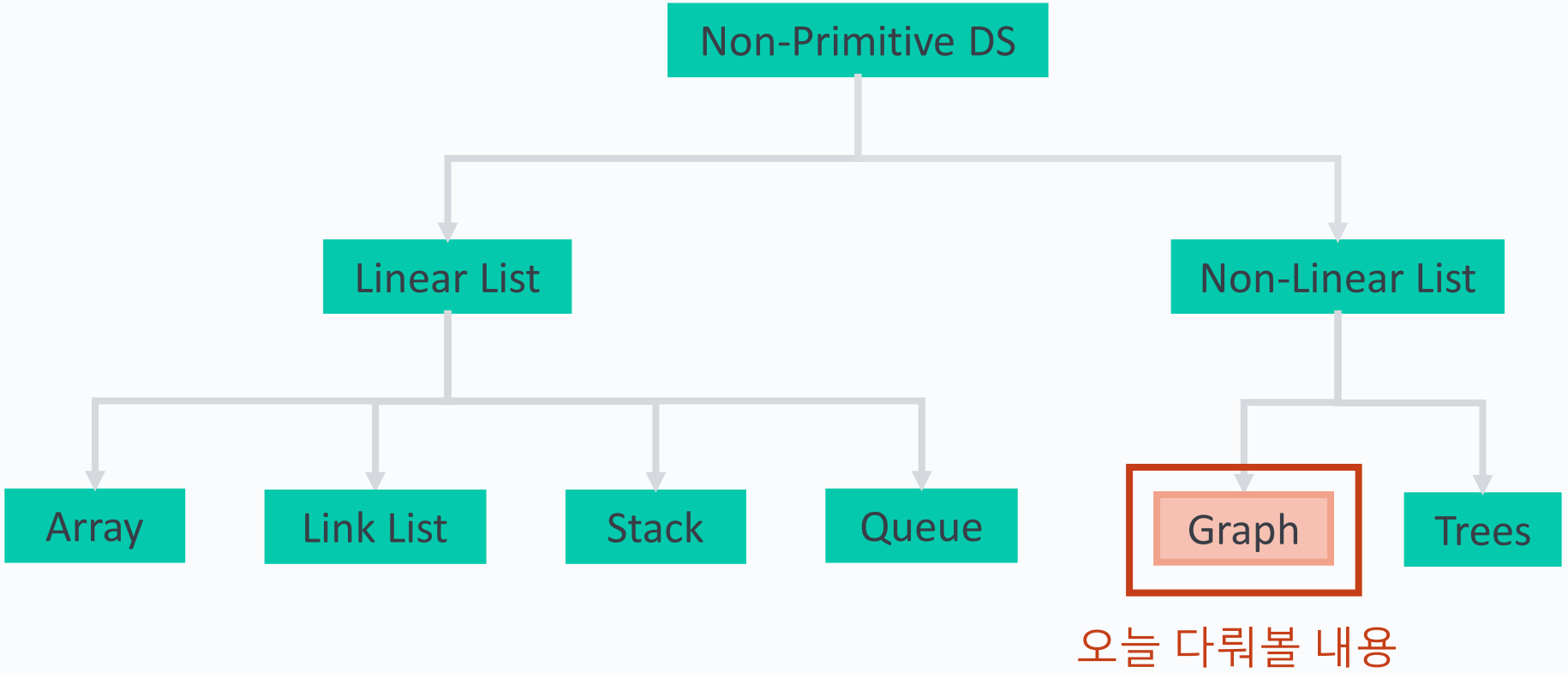
03 문제 풀어보기

ANSI 아주대학교 알고리즘 소학회 Analyze N' Solve It

NAVER D2

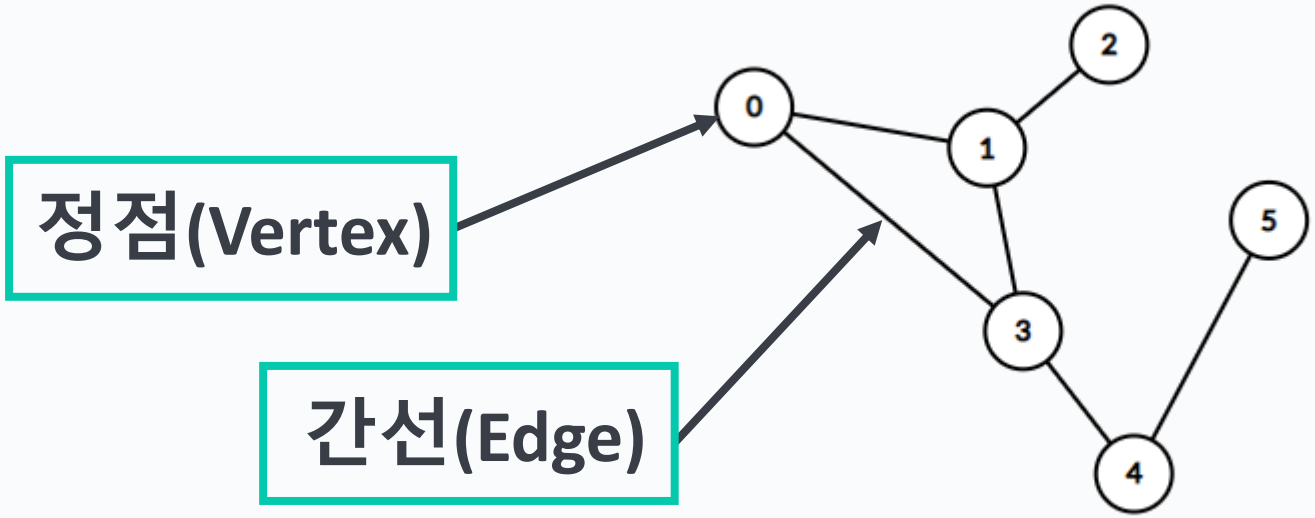
01 그래프 이론

자료 구조의 종류


Non-Primitive DS
Linear List
Non-Linear List
Array
Link List
Stack
Queue
Graph
Trees
오늘 다뤄볼 내용

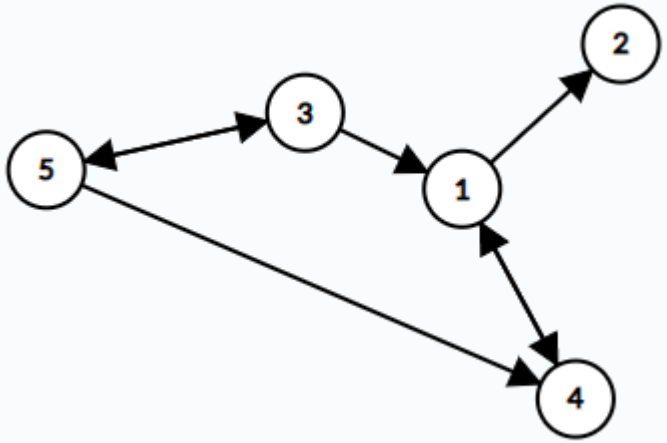
그래프 정의

- **정점(Vertex)**과 정점들을 연결하는 **간선(Edge)**들로 구성된 자료구조

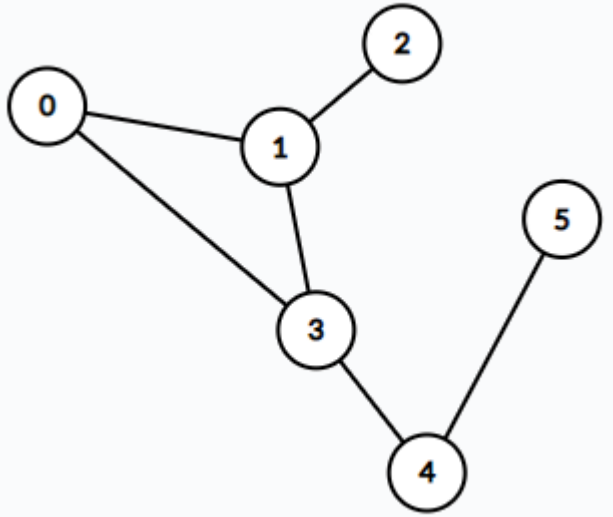


그래프 종류

- 방향 그래프(directed graph)

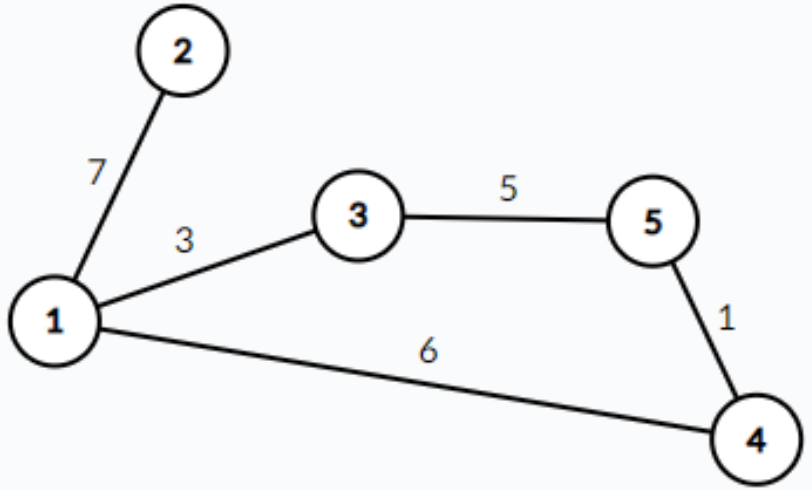


- 무향 그래프(undirected graph)



그래프 종류

- 가중치 그래프(weighted graph)



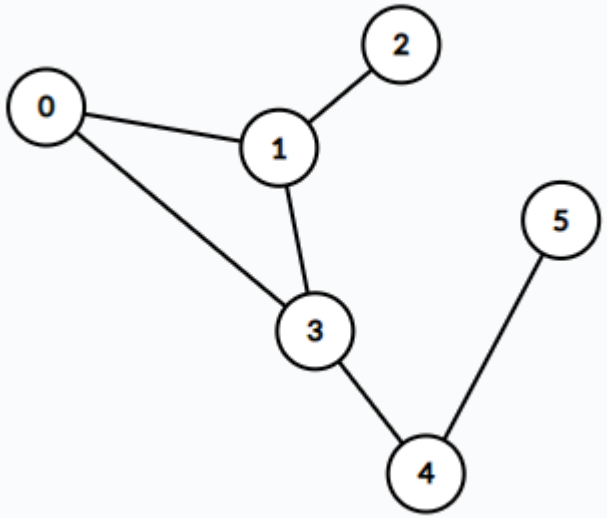
그래프 표현

- 인접 행렬 표현

```
int graph[6][6];

graph[0][1] = 1;
graph[1][0] = 1;
graph[0][3] = 1;
graph[3][0] = 1;
...
```

	0	*1*	*2*	*3*	*4*	*5*
0	0	1	0	1	0	0
1	1	0	1	1	0	0
2	0	1	0	0	0	0
3	1	1	0	0	1	0
4	0	0	0	1	0	1
5	0	0	0	0	1	0



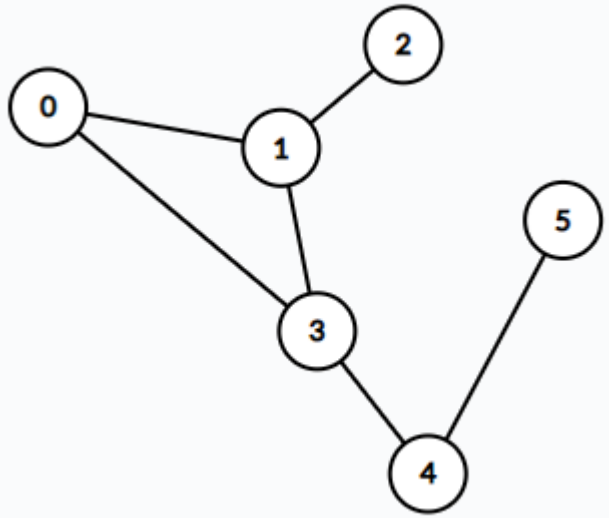
그래프 표현

- 인접 리스트 표현

```
vector<int> graph[6];

graph[0].push_back(1);
graph[1].push_back(0);
graph[0].push_back(3);
graph[3].push_back(0);
...
```

0	1	3	
1	0	2	3
2	1		
3	0	1	4
4	3	5	
5	4		



인접 행렬과 인접 리스트 비교

- **인접 행렬 표현**

두 정점 사이에 간선이 있는지를 한 번의 배열 접근만으로 확인 가능.

실제 간선의 개수와 관계없이 항상 V*V 크기의 공간이 필요.

	0	1	2	3	4	5
0	0	1	0	1	0	0
1	1	0	1	1	0	0
2	0	1	0	0	0	0
3	1	1	0	0	1	0
4	0	0	0	1	0	1
5	0	0	0	0	1	0

- **인접 리스트 표현**

두 정점 사이에 간선이 있는지 확인하려면 해당 행을 처음부터 읽어 가면서 일일이 확인해야 함.

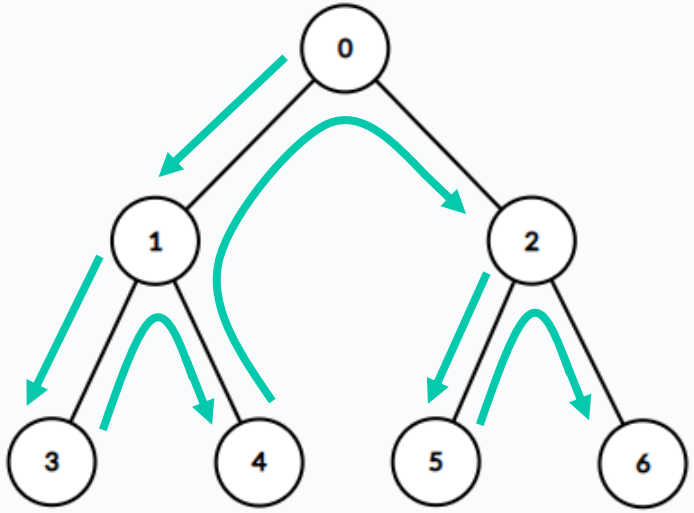
실제 간선 수만큼만 공간 필요.

0	1	3	
1	0	2	3
2	1		
3	0	1	4
4	3	5	
5	4		

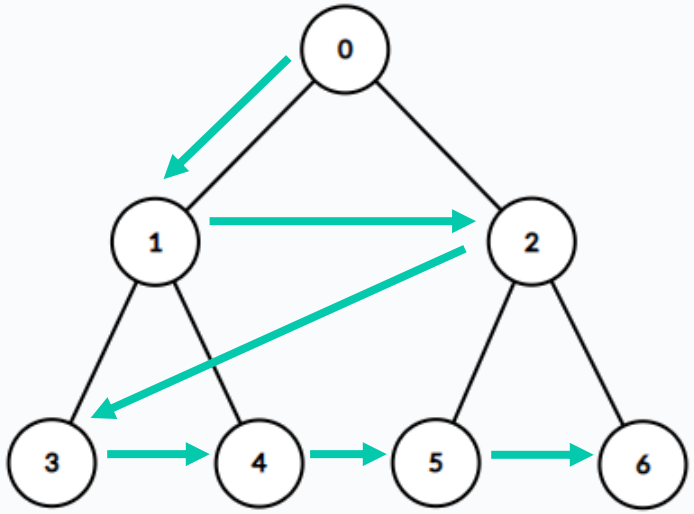
그래프 탐색 알고리즘

그래프의 모든 정점들을 특정한 순서에 따라 방문하는 알고리즘

- **깊이 우선 탐색 DFS** (Depth First Search)



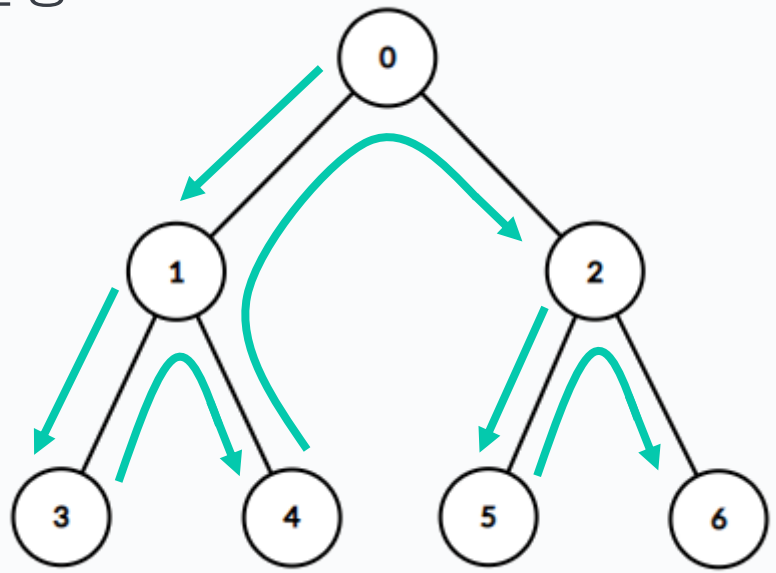
- **너비 우선 탐색 BFS** (Breadth First Search)



02 깊이 우선 탐색 DFS

깊이 우선 탐색 DFS

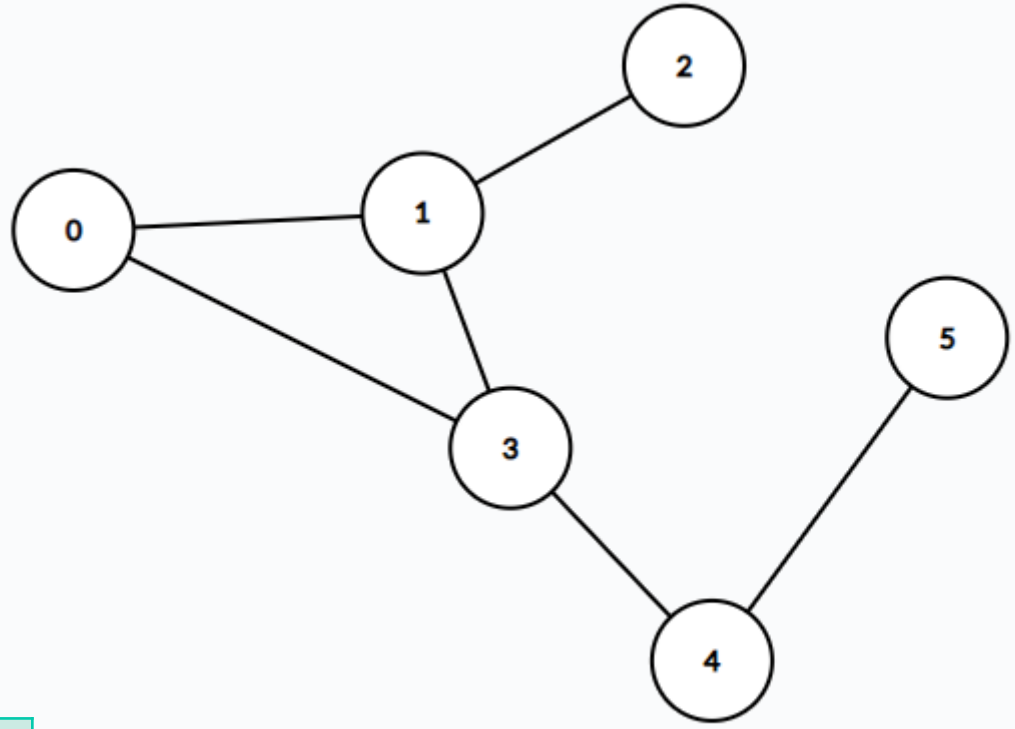
- **현재 정점에서 갈 수 있는 정점들까지 들어가며 탐색**
  - ✓한 방향으로 계속 갈 수 있을 때까지 탐색하다 막히면, 가장 가까운 갈림길로 돌아와서 다른 방향으로 탐색을 진행



깊이 우선 탐색 DFS

vector<int> graph[6];

graph[0]	3	1	
graph[1]	0	2	3
graph[2]	1		
graph[3]	0	4	1
graph[4]	3	5	
graph[5]	4		



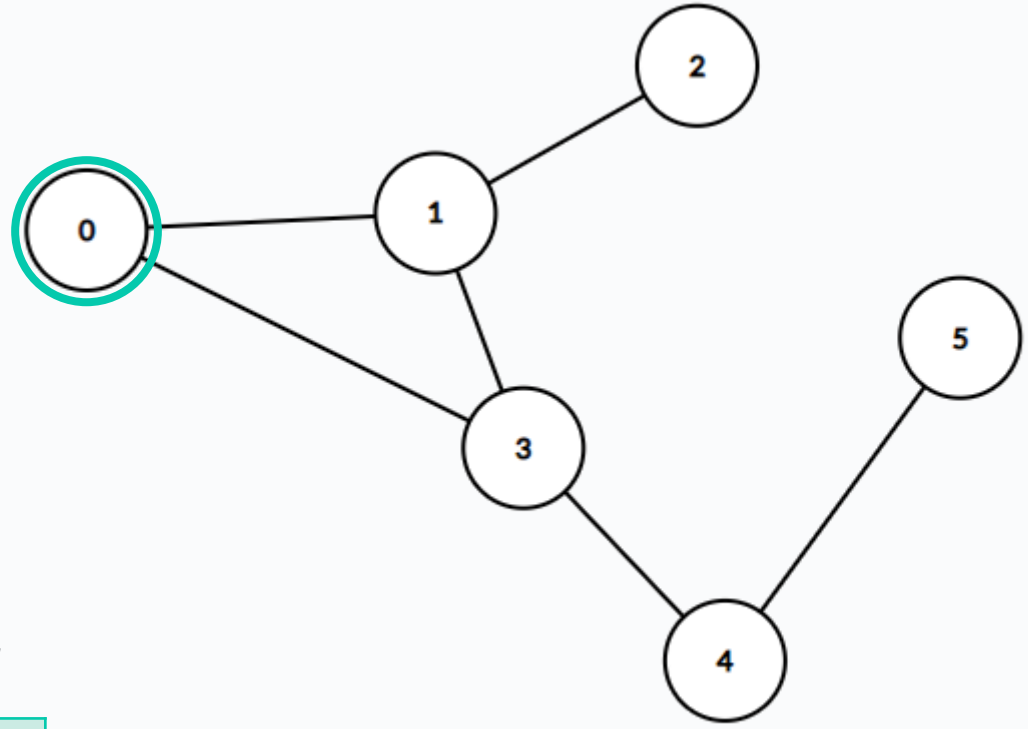
방문 순서

	0	*1*	*2*	*3*	*4*	*5*
int visited[6]	0	0	0	0	0	0

깊이 우선 탐색 DFS

vector<int> graph[6];

graph[0]	3	1	
graph[1]	0	2	3
graph[2]	1		
graph[3]	0	4	1
graph[4]	3	5	
graph[5]	4		



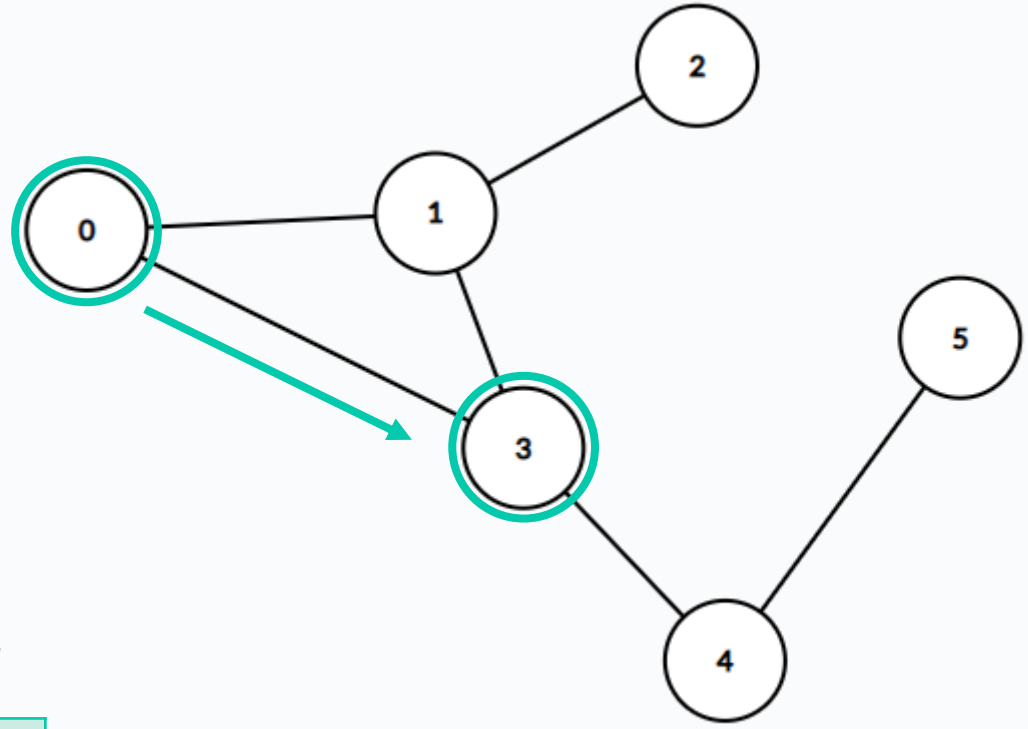
방문 순서 0

	0	*1*	*2*	*3*	*4*	*5*
int visited[6]	1	0	0	0	0	0

깊이 우선 탐색 DFS

vector<int> graph[6];

graph[0]	3	1	
graph[1]	0	2	3
graph[2]	1		
graph[3]	0	4	1
graph[4]	3	5	
graph[5]	4		



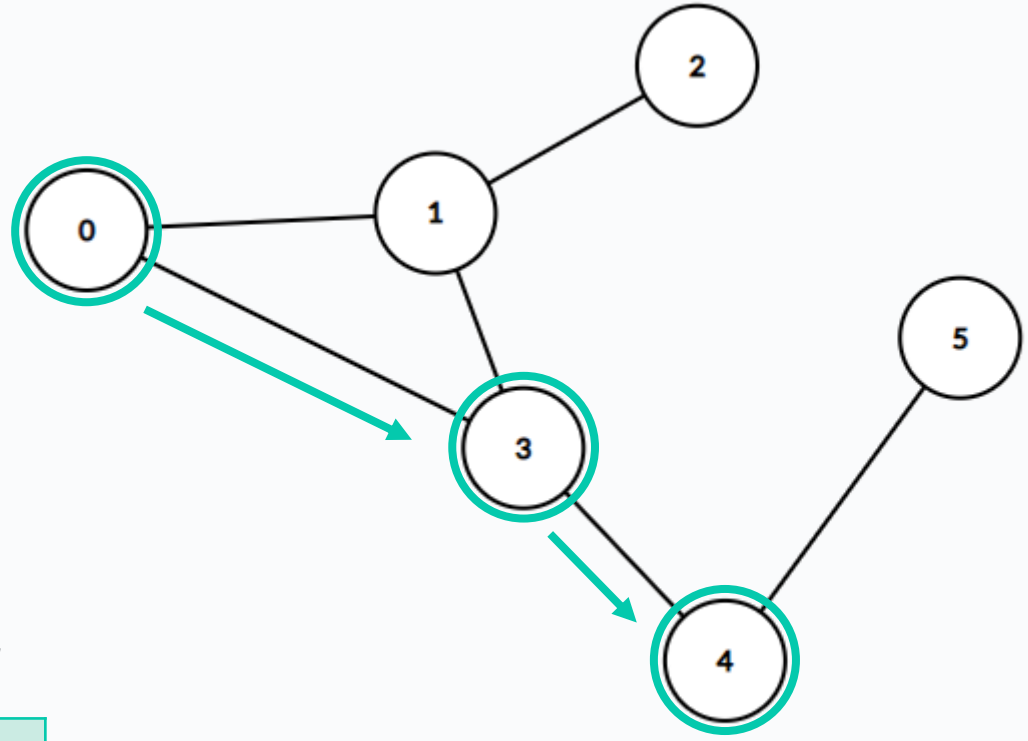
방문 순서 0 -> 3

	0	*1*	*2*	*3*	*4*	*5*
int visited[6]	1	0	0	1	0	0

깊이 우선 탐색 DFS

vector<int> graph[6];

graph[0]	3	1	
graph[1]	0	2	3
graph[2]	1		
graph[3]	0	4	1
graph[4]	3	5	
graph[5]	4		



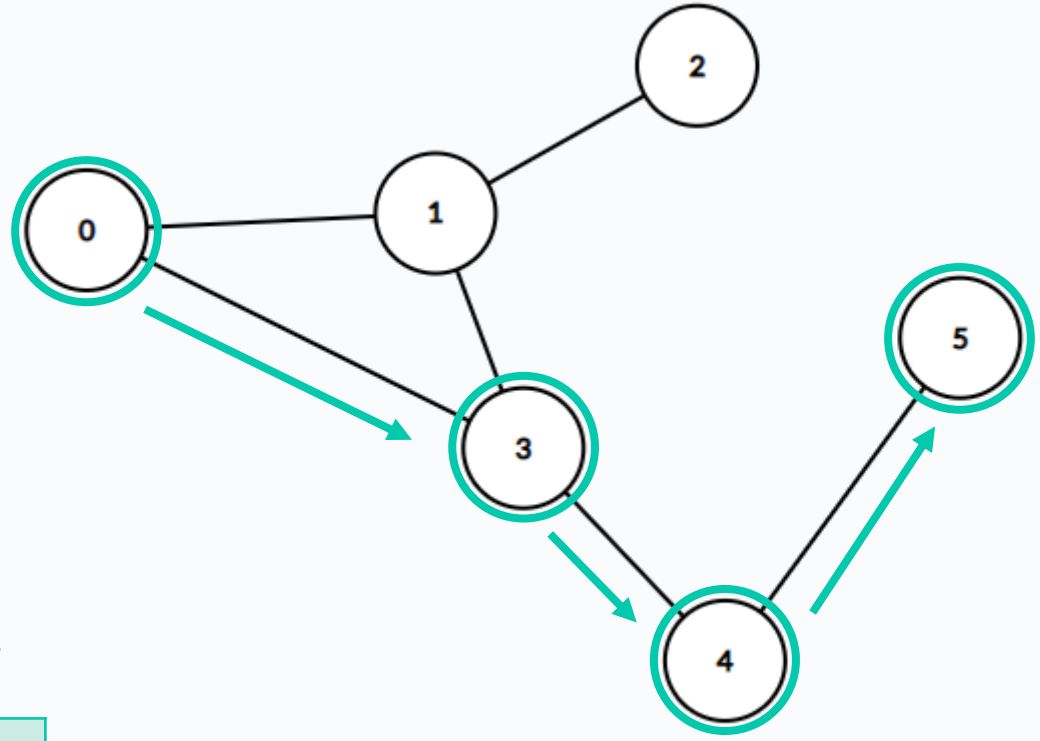
방문 순서 **0 -> 3 -> 4**

	0	*1*	*2*	*3*	*4*	*5*
int visited[6]	1	0	0	1	1	0

깊이 우선 탐색 DFS

vector<int> graph[6];

graph[0]	3	1	
graph[1]	0	2	3
graph[2]	1		
graph[3]	0	4	1
graph[4]	3	5	
graph[5]	4		



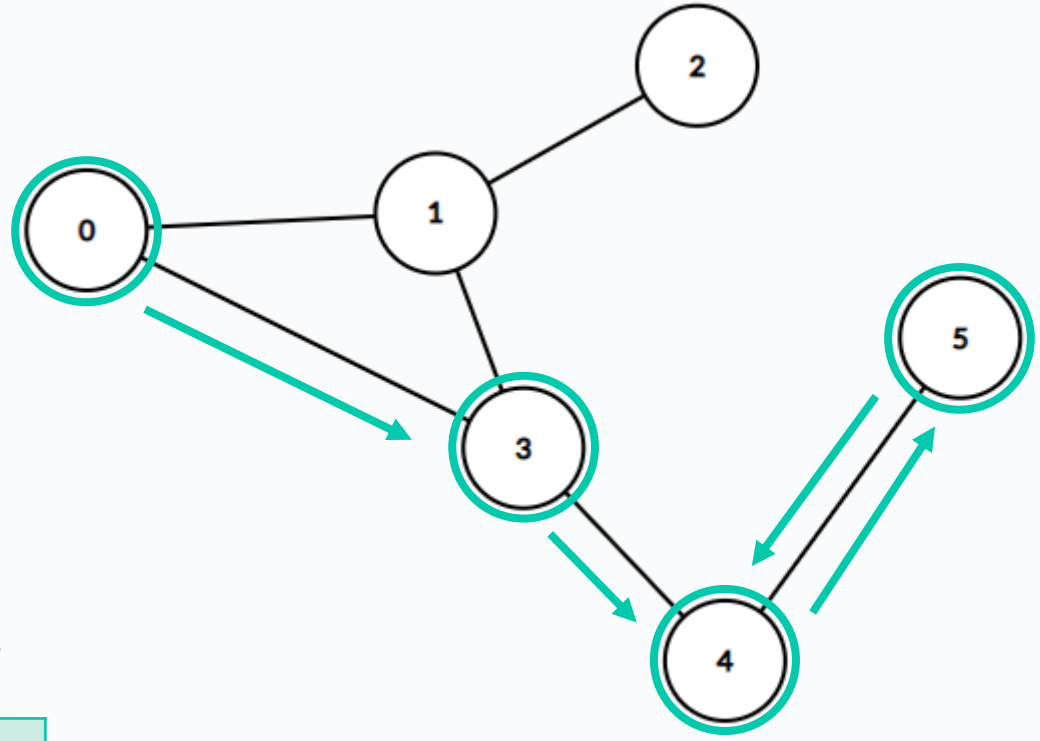
방문 순서 0 -> 3 -> 4 -> 5

	0	*1*	*2*	*3*	*4*	*5*
int visited[6]	1	0	0	1	1	1

깊이 우선 탐색 DFS

vector<int> graph[6];

graph[0]	3	1	
graph[1]	0	2	3
graph[2]	1		
graph[3]	0	4	1
graph[4]	3	5	
graph[5]	4		



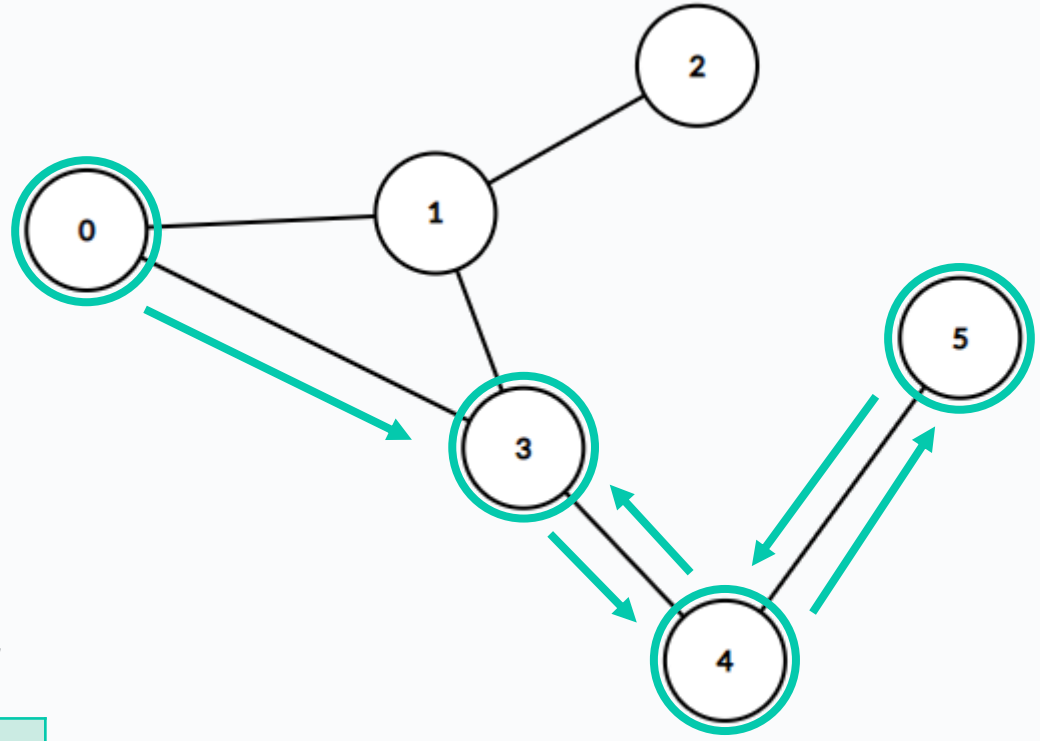
방문 순서 0 -> 3 -> 4 -> 5

	0	*1*	*2*	*3*	*4*	*5*
int visited[6]	1	0	0	1	1	1

깊이 우선 탐색 DFS

vector<int> graph[6];

graph[0]	3	1	
graph[1]	0	2	3
graph[2]	1		
graph[3]	0	4	1
graph[4]	3	5	
graph[5]	4		



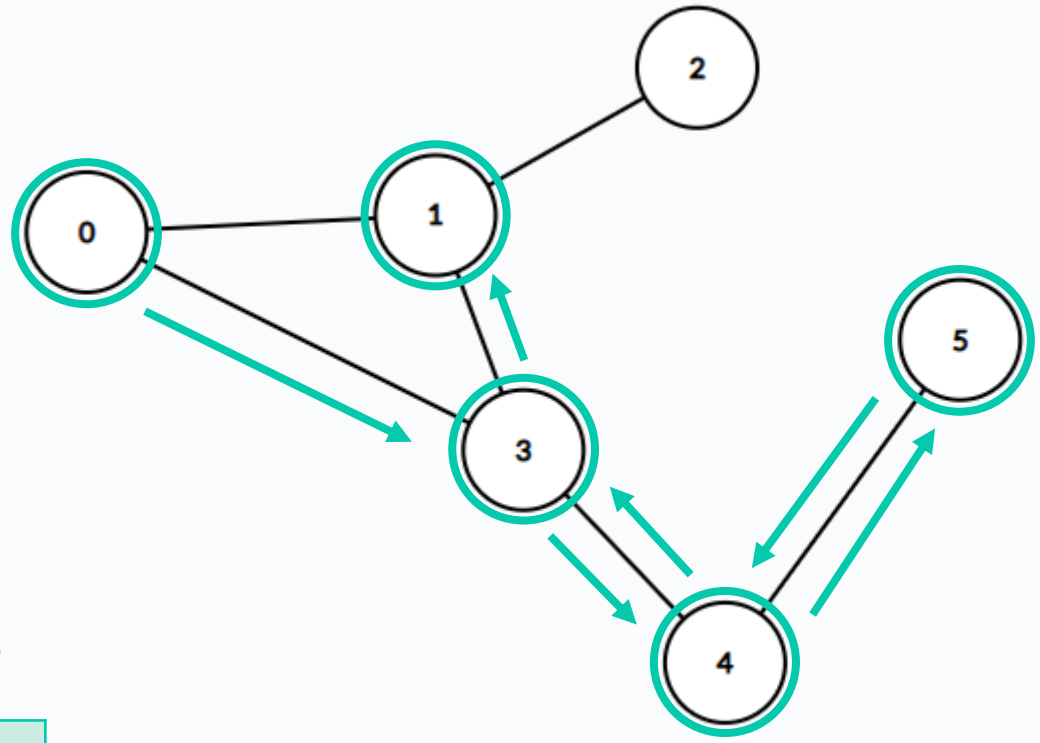
방문 순서 0 -> 3 -> 4 -> 5

	0	*1*	*2*	*3*	*4*	*5*
int visited[6]	1	0	0	1	1	1

깊이 우선 탐색 DFS

vector<int> graph[6];

graph[0]	3	1	
graph[1]	0	2	3
graph[2]	1		
graph[3]	0	4	1
graph[4]	3	5	
graph[5]	4		



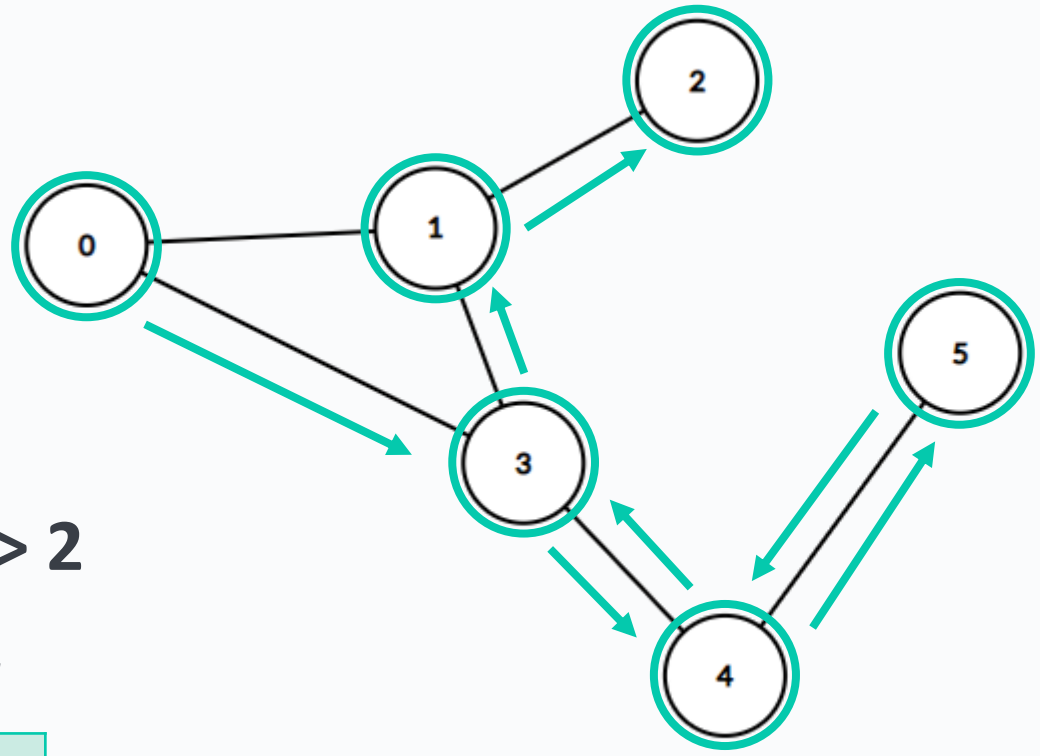
방문 순서 0 -> 3 -> 4 -> 5 -> 1

	0	*1*	*2*	*3*	*4*	*5*
int visited[6]	1	1	0	1	1	1

깊이 우선 탐색 DFS

vector<int> graph[6];

graph[0]	3	1	
graph[1]	0	2	3
graph[2]	1		
graph[3]	0	4	1
graph[4]	3	5	
graph[5]	4		

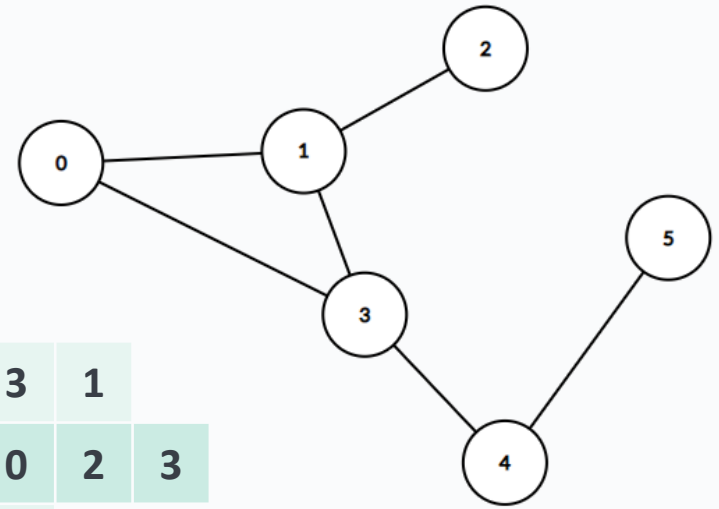


방문 순서 0 -> 3 -> 4 -> 5 -> 1 -> 2

	0	*1*	*2*	*3*	*4*	*5*
int visited[6]	1	1	1	1	1	1

깊이 우선 탐색 DFS

- DFS 구현해보기



graph[0]	3	1	
graph[1]	0	2	3
graph[2]	1		
graph[3]	0	4	1
graph[4]	3	5	
graph[5]	4		

```cpp
vector<int> graph[6];
int visited[6];

void dfs(int cur)
{
    visited[cur] = 1;

    for (int i = 0; i < graph[cur].size(); i++)
    {
        int next = graph[cur][i];
        if (visited[next] == 0)
            dfs(next);
    }
}
```

03 문제 풀어보기

문제 풀어보기

- #2606 바이러스 (https://www.acmicpc.net/problem/2606)
- #11724 연결 요소의 개수 (https://www.acmicpc.net/problem/11724)
- #1012 유기농 배추 (https://www.acmicpc.net/problem/1012)
- #4963 섬의 개수 (https://www.acmicpc.net/problem/4963)
- #2667 단지번호붙이기 (https://www.acmicpc.net/problem/2667)

수고하셨습니다

다음 강의는 BFS입니다

ANSi