

/* 데이터 구조 및 알고리즘 */

신현규 강사, 화/목 20:00

그래프 소개



/* elice */

주차별 커리큘럼

1주차 과정 소개, 배열, 연결리스트, 클래스

2주차 스택, 큐, 해싱

3주차 시간복잡도

4주차 트리, 트리순회, 재귀호출

5주차 힙

6주차 **그래프 소개, DFS**

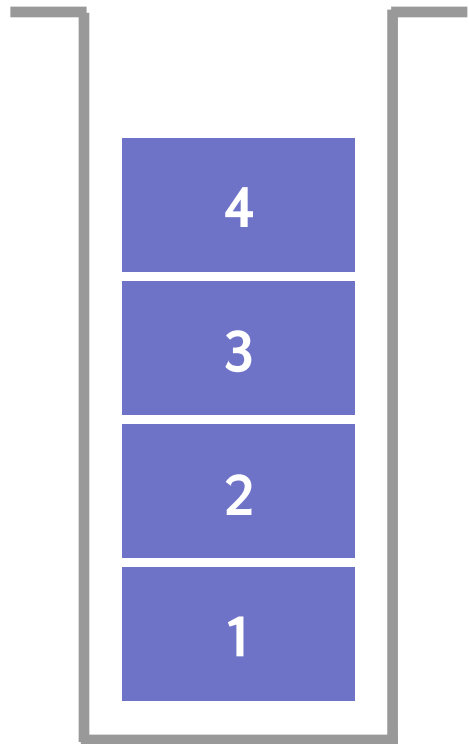
7주차 그래프 심화, BFS

8주차 강의 요약, 알고리즘 과정 소개

컴퓨터를 이용한 문제 해결 과정

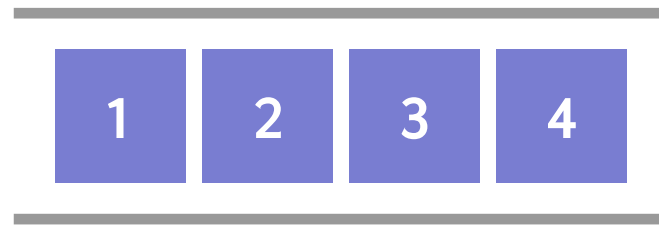
1. 문제를 정확히 이해한다
2. 문제를 해결하는 알고리즘을 개발한다
3. 알고리즘이 문제를 해결한다는 것을 증명한다
4. 알고리즘이 제한시간 내에 동작한다는 것을 보인다
5. 알고리즘을 코드로 작성한다
6. 제출 후 만점을 받고 매우 기뻐한다

대표적인 자료구조



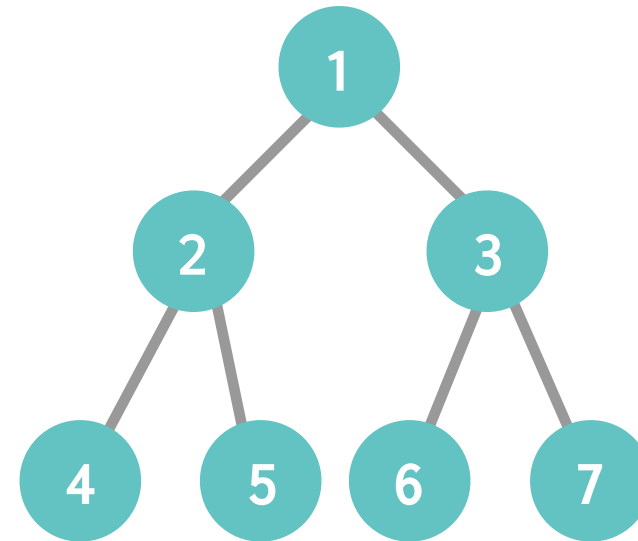
스택 (Stack)

Last In First Out

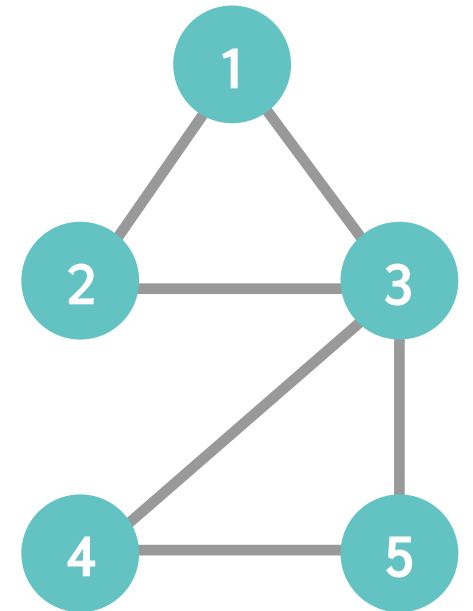


큐 (Queue)

First In First Out



트리 (Tree)



그래프 (Graph)

유명한 자료구조는 이미 구현되어 있다

심지어 유능한 개발자가 구현을 해 놓았다

내가 짠 코드보다 훨씬 성능이 좋다

버그가 있는지 고민할 필요가 없다

이미 많은 사람들이 잘 쓰고있다

따라서, 쓰지 않을 이유가 없다

아니, 반드시 써야한다

아놔 근데 이걸 왜 구현하고 있는가 ?

구현을 해봐야 자료구조를 완전히 이해한다

이론 외에 구현에서 발생하는 이슈가 있다

내부 구현을 알아야 정확한 분석을 할 수 있다

구현을 해봐야 연산의 시간복잡도를 명확히 이해한다

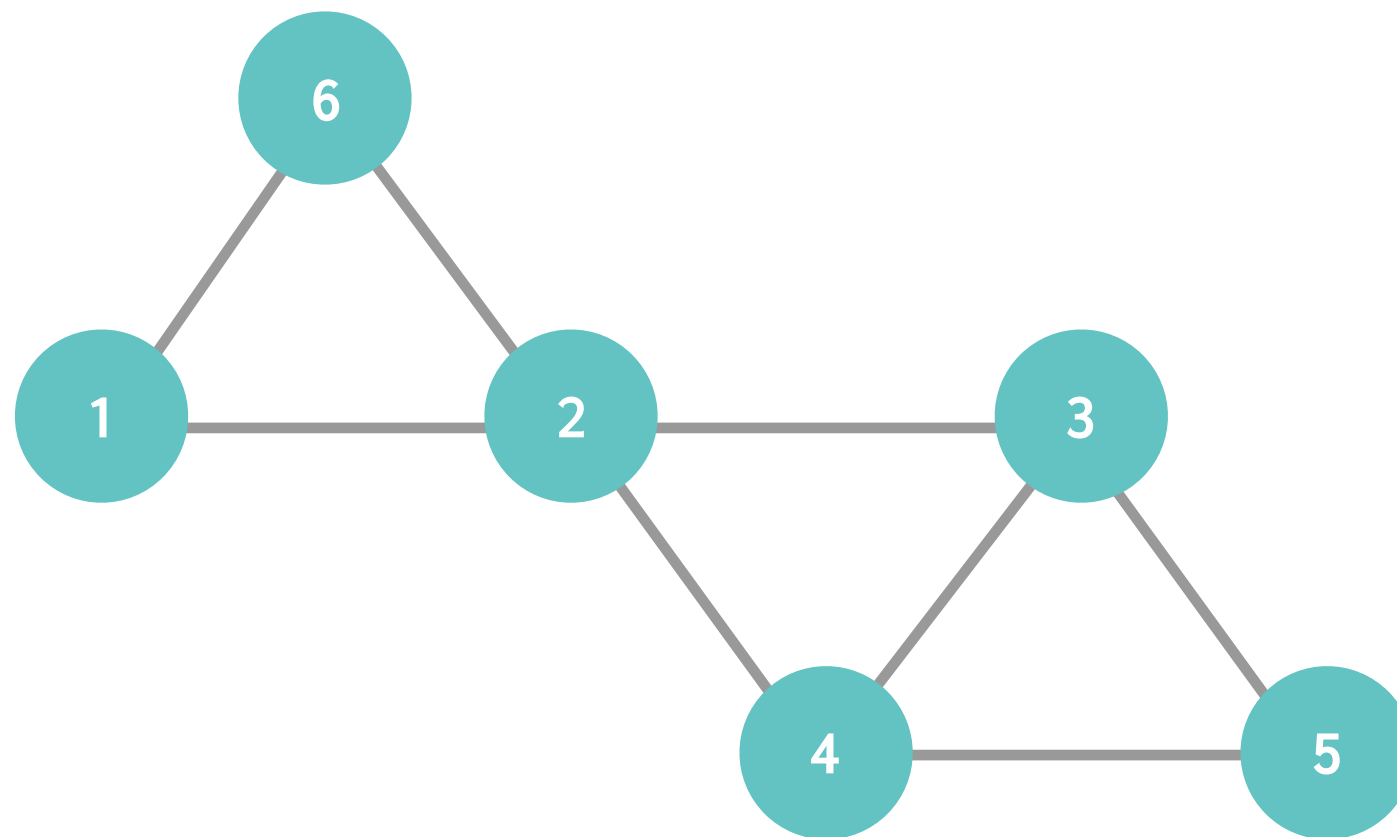


엘리스 자료구조 과정이기 때문이다

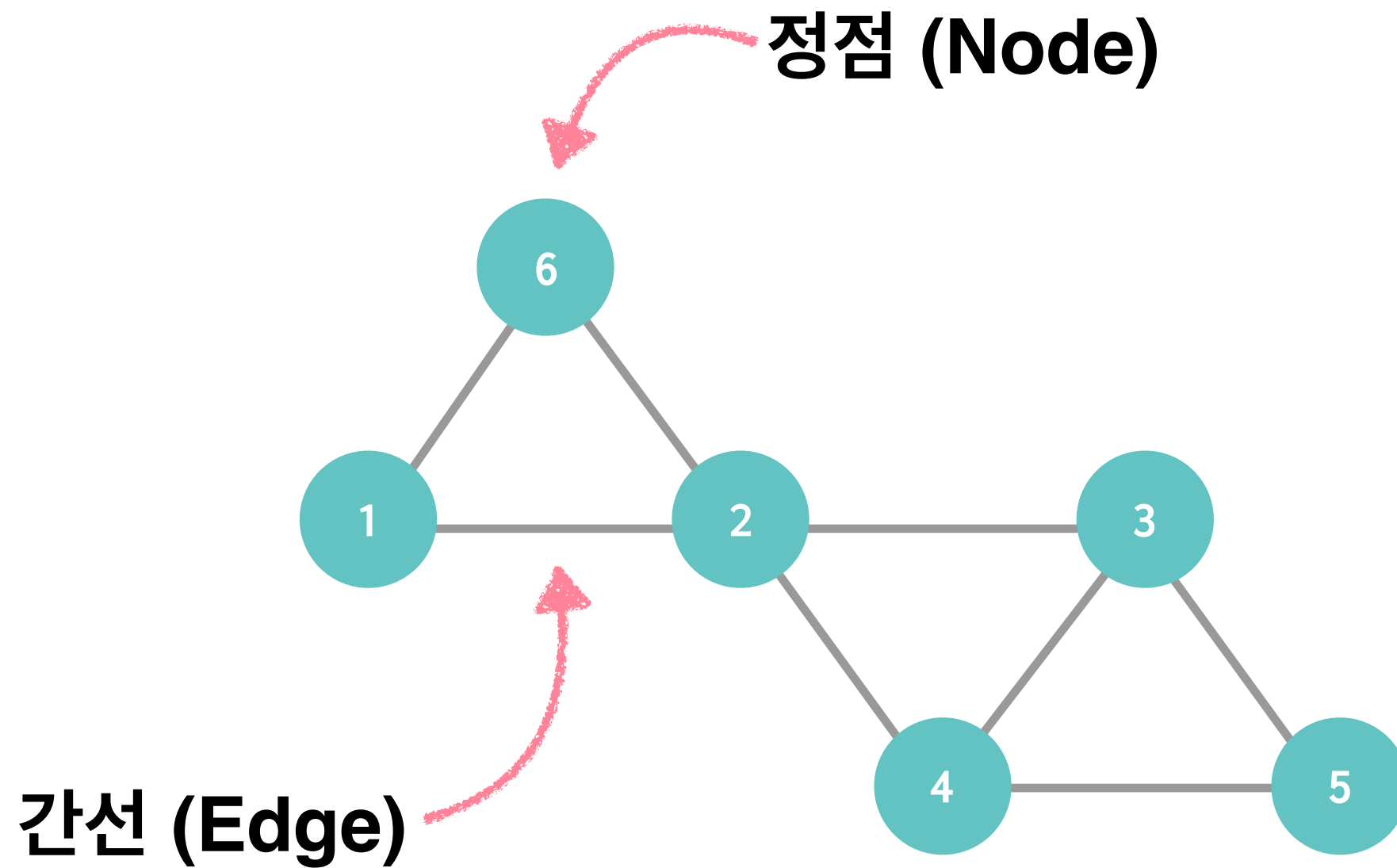
엘리스에서 배웠다 = **내 손으로** 구현까지 해봤다

그래프

정점과 간선으로 이루어진 자료구조



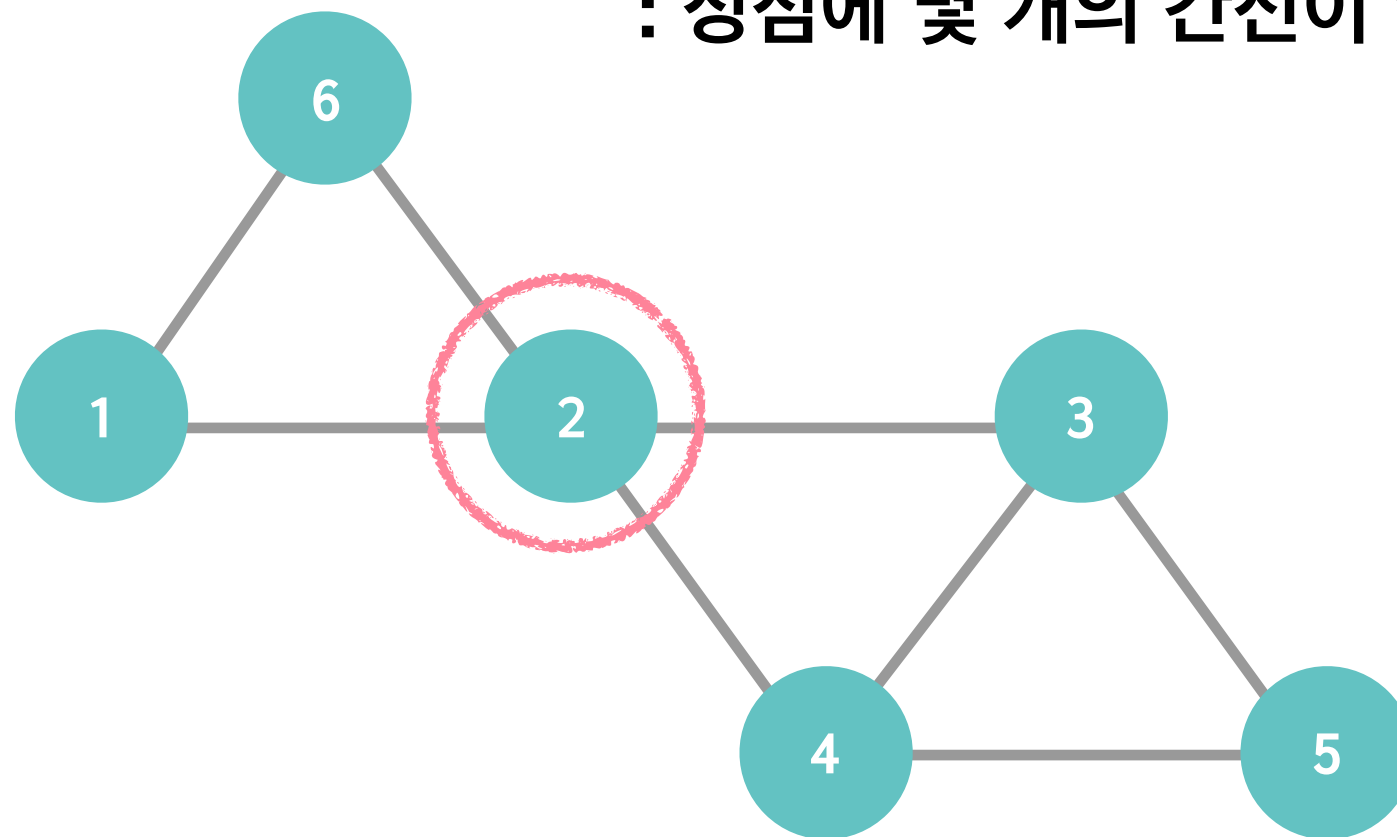
용어



용어

차수 (Degree)

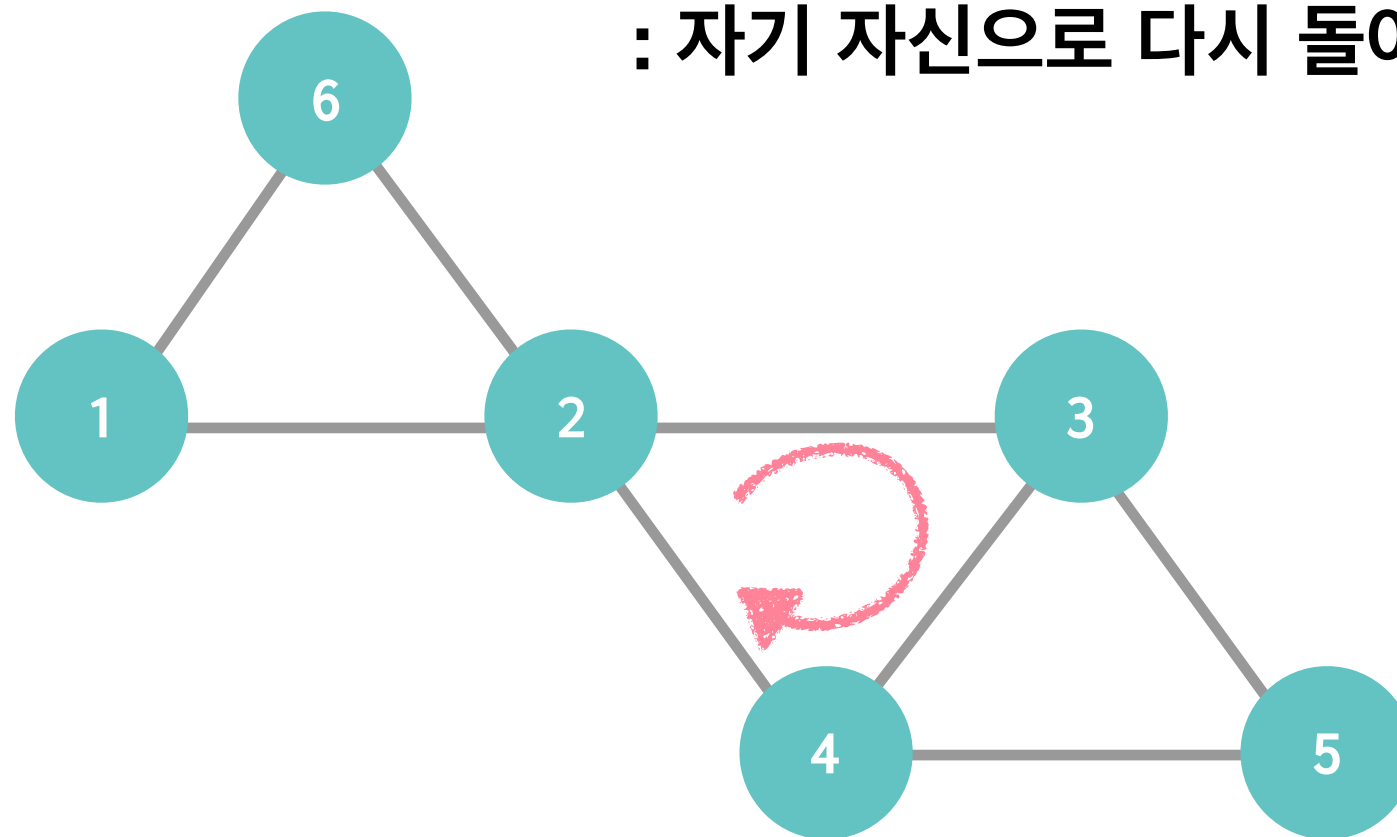
: 정점에 몇 개의 간선이 연결되어 있는가 ?



용어

사이클 (Cycle)

: 자기 자신으로 다시 돌아올 수 있는 경로



그래프는 왜 중요한가 ?

현실 세계의 많은 것들을 그래프로 나타낼 수 있다

즉, 그래프와 관련된 문제가 매우 많다

그래프와 관련된 수학적 정리가 매우 많다

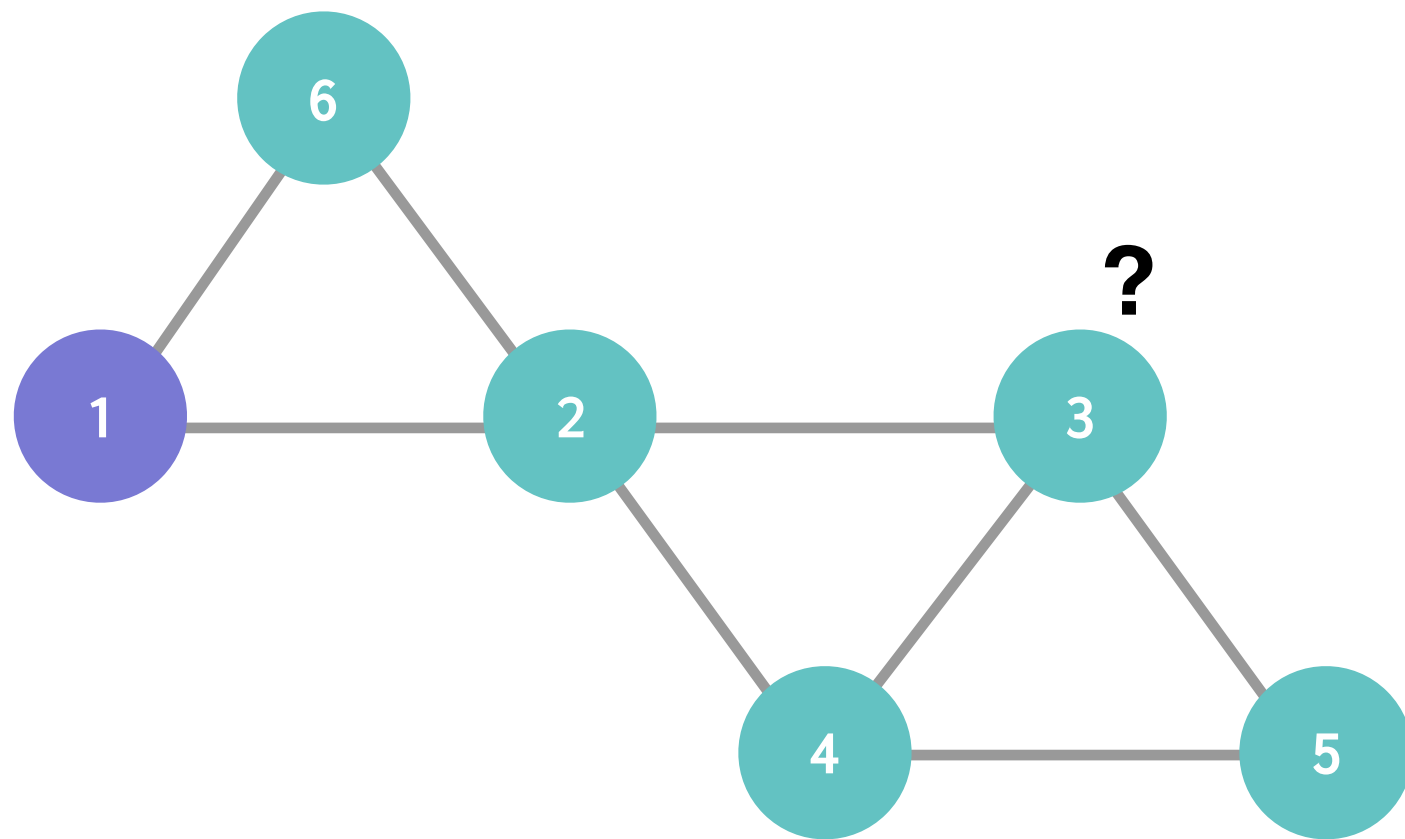
그래프 이론이라는 분야가 따로 있다 (Graph theory)

어렵다(...)

그래프와 관련된 이론도 어렵고 구현도 어렵고...ㅠㅠ

그래프의 구현 : 인접행렬

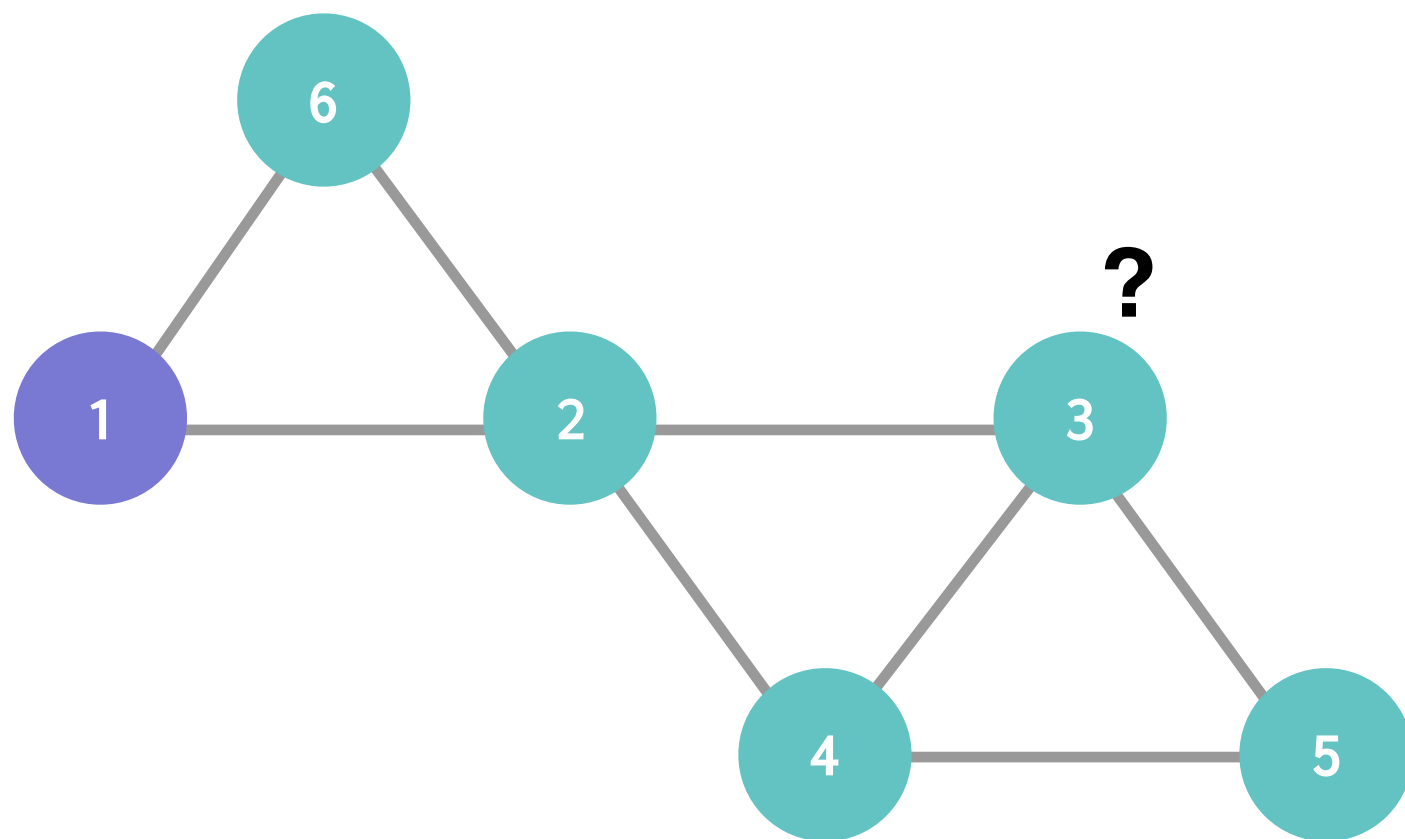
정점의 연결관계를 2차원 리스트에 0, 1으로 표현



	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	1	0	1
3	0	1	0	1	1	0
4	0	1	1	0	1	0
5	0	0	1	1	0	0
6	1	1	0	0	0	0

그래프의 구현 : 인접행렬

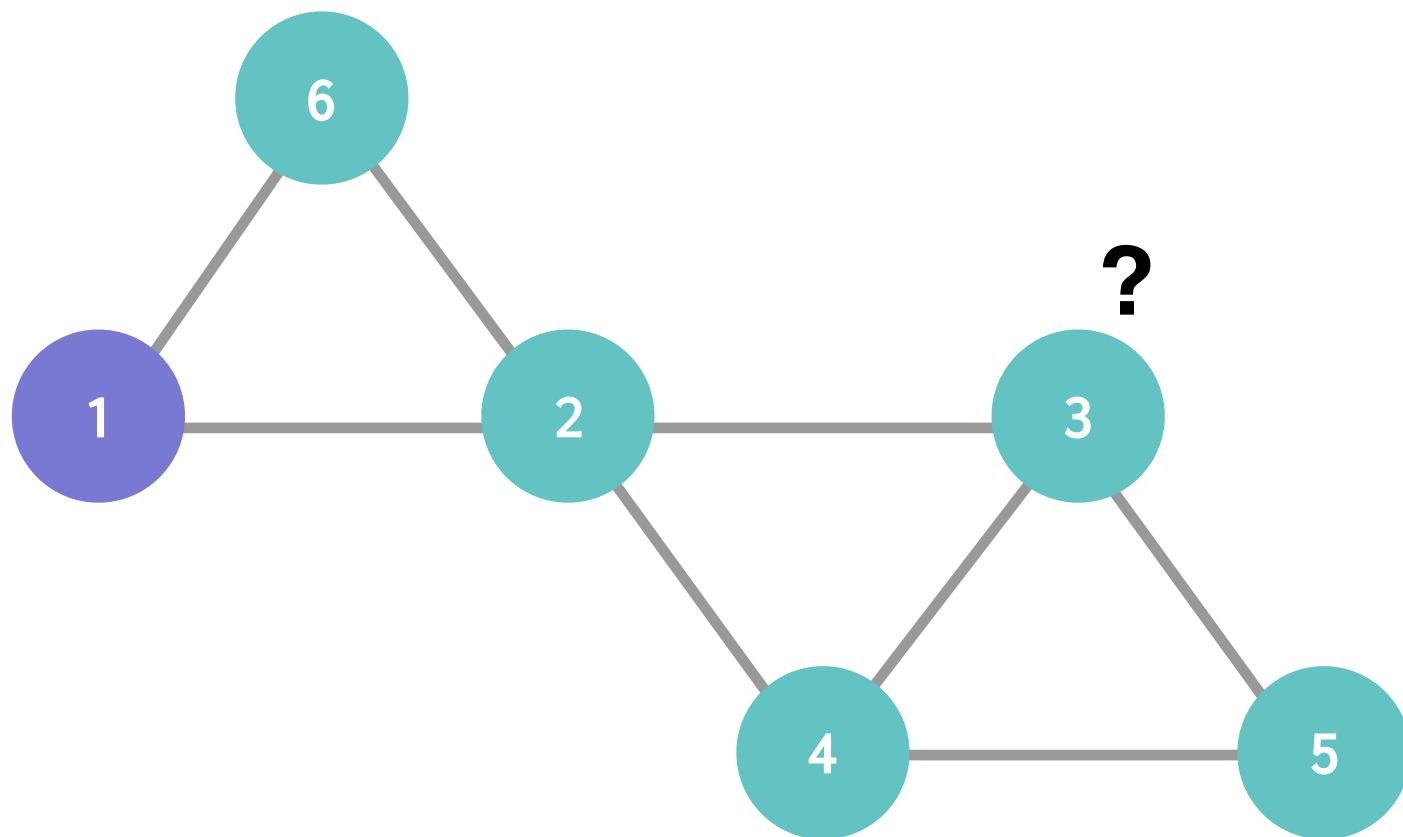
정점의 연결관계를 2차원 리스트에 0, 1으로 표현



	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	1	0	1
3	0	1	0	1	1	0
4	0	1	1	0	1	0
5	0	0	1	1	0	0
6	1	1	0	0	0	0

그래프의 구현 : 인접행렬

장점 : 연결 여부를 $O(1)$ 에 알 수 있다

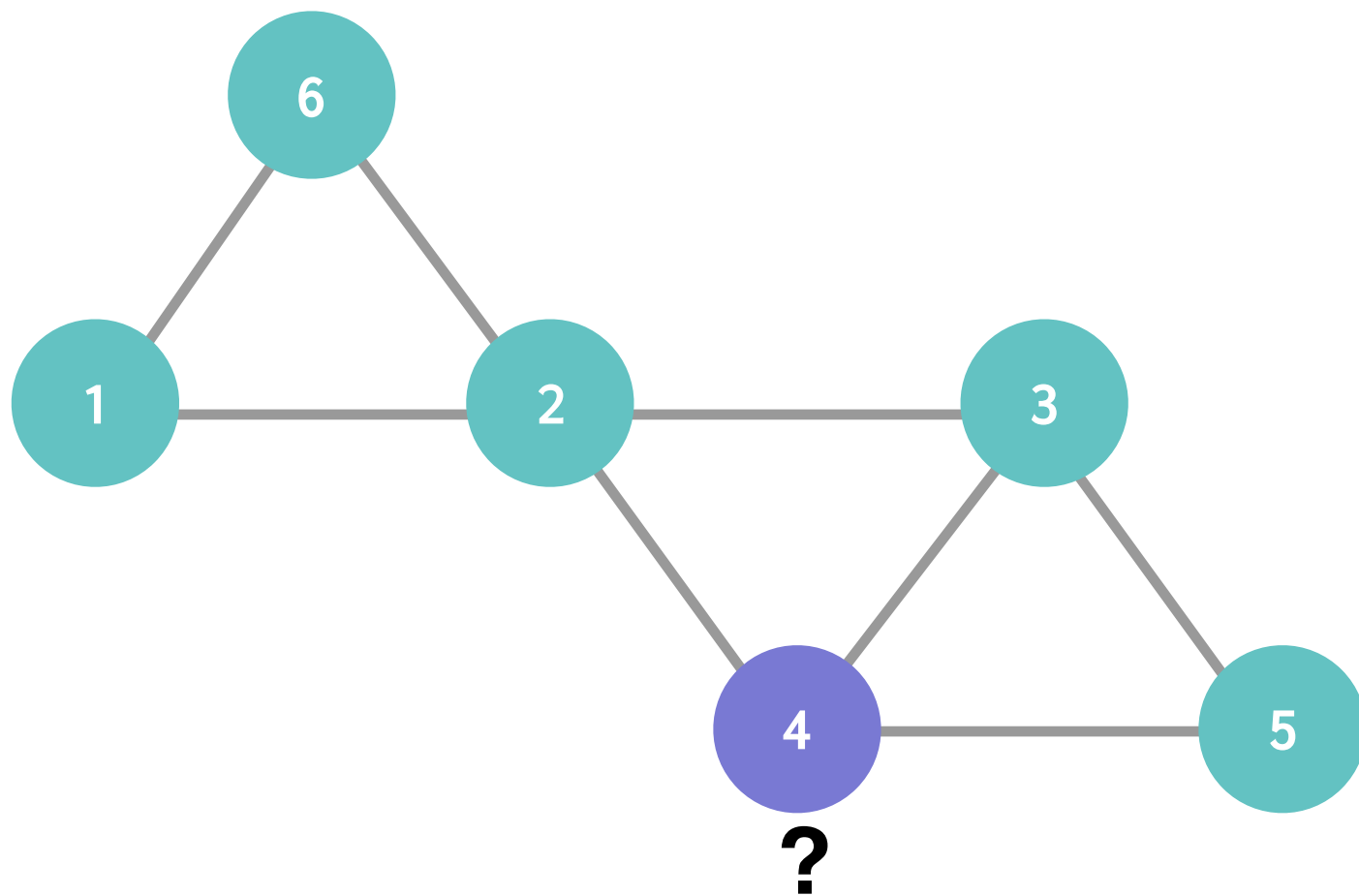


	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	1	0	1
3	0	1	0	1	1	0
4	0	1	1	0	1	0
5	0	0	1	1	0	0
6	1	1	0	0	0	0

그래프의 구현 : 인접행렬

단점 : 인접한 정점을 찾는데 $O(n)$ 이 걸린다

(실제 인접한 정점 수와 관계없이)

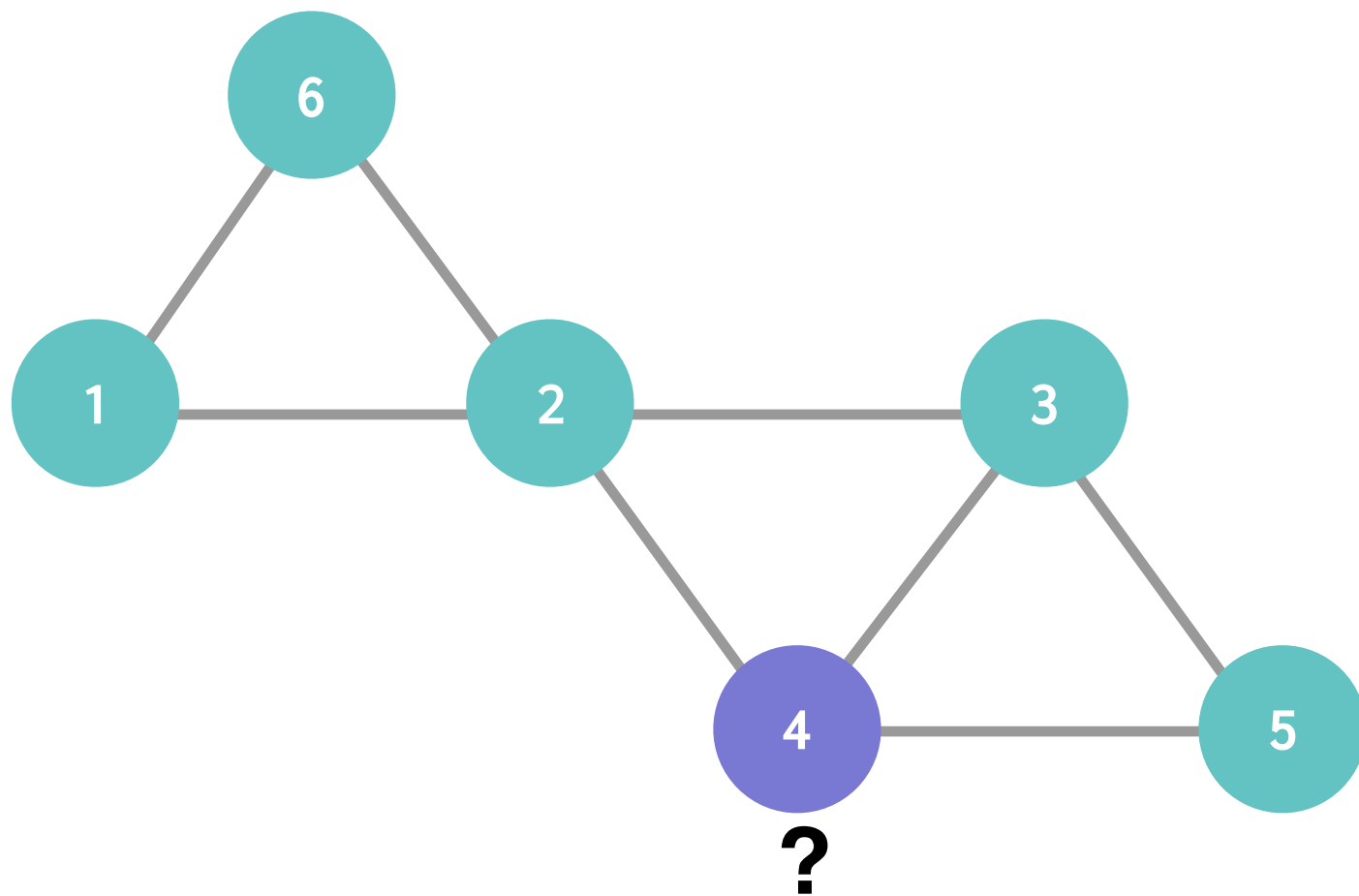


	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	1	0	1
3	0	1	0	1	1	0
4	0	1	1	0	1	0
5	0	0	1	1	0	0
6	1	1	0	0	0	0

그래프의 구현 : 인접행렬

단점 : 무조건 n^2 개의 칸을 써야 한다

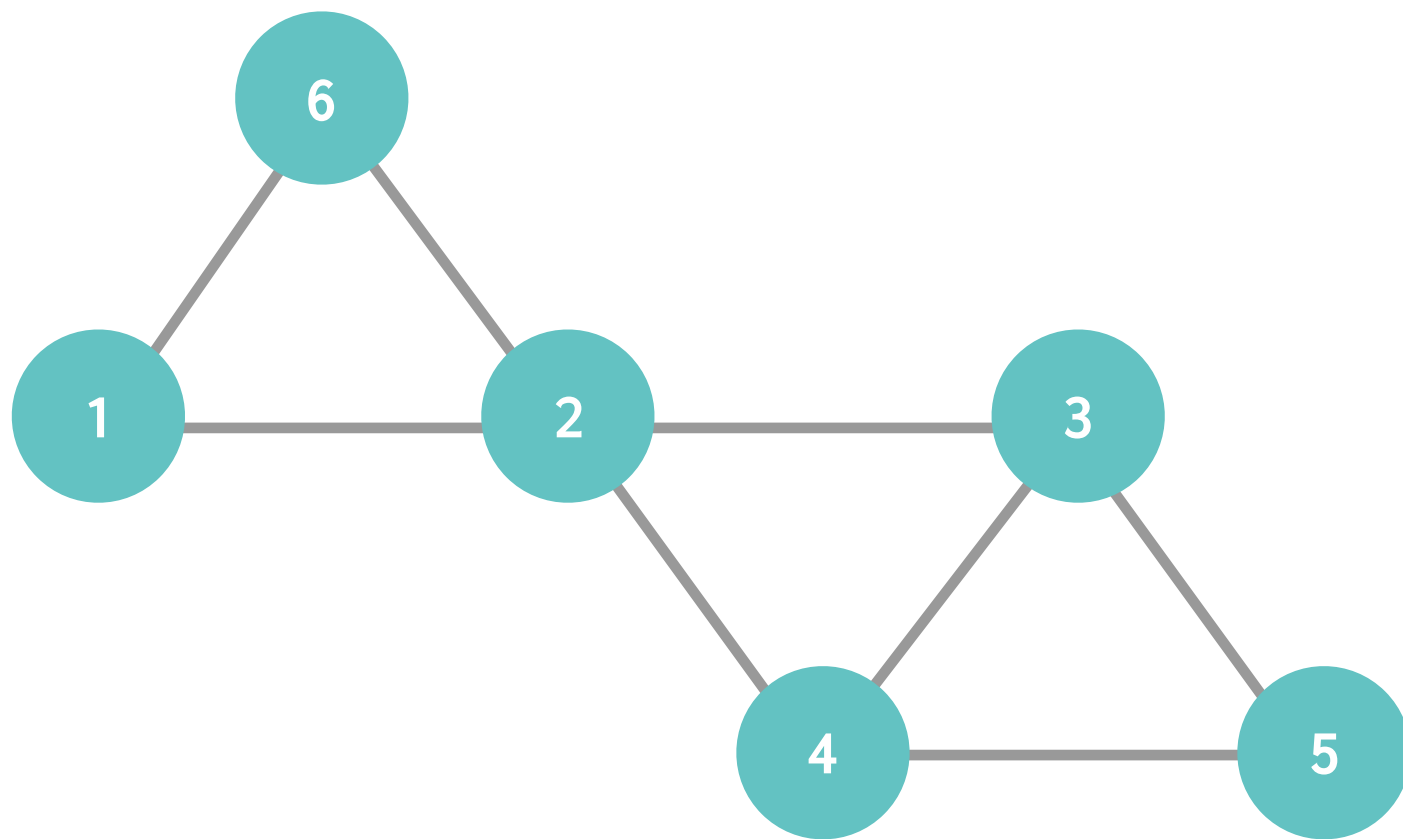
(실제 간선의 개수와 관계없이)



	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	1	0	1
3	0	1	0	1	1	0
4	0	1	1	0	1	0
5	0	0	1	1	0	0
6	1	1	0	0	0	0

그래프의 구현 : 인접 리스트

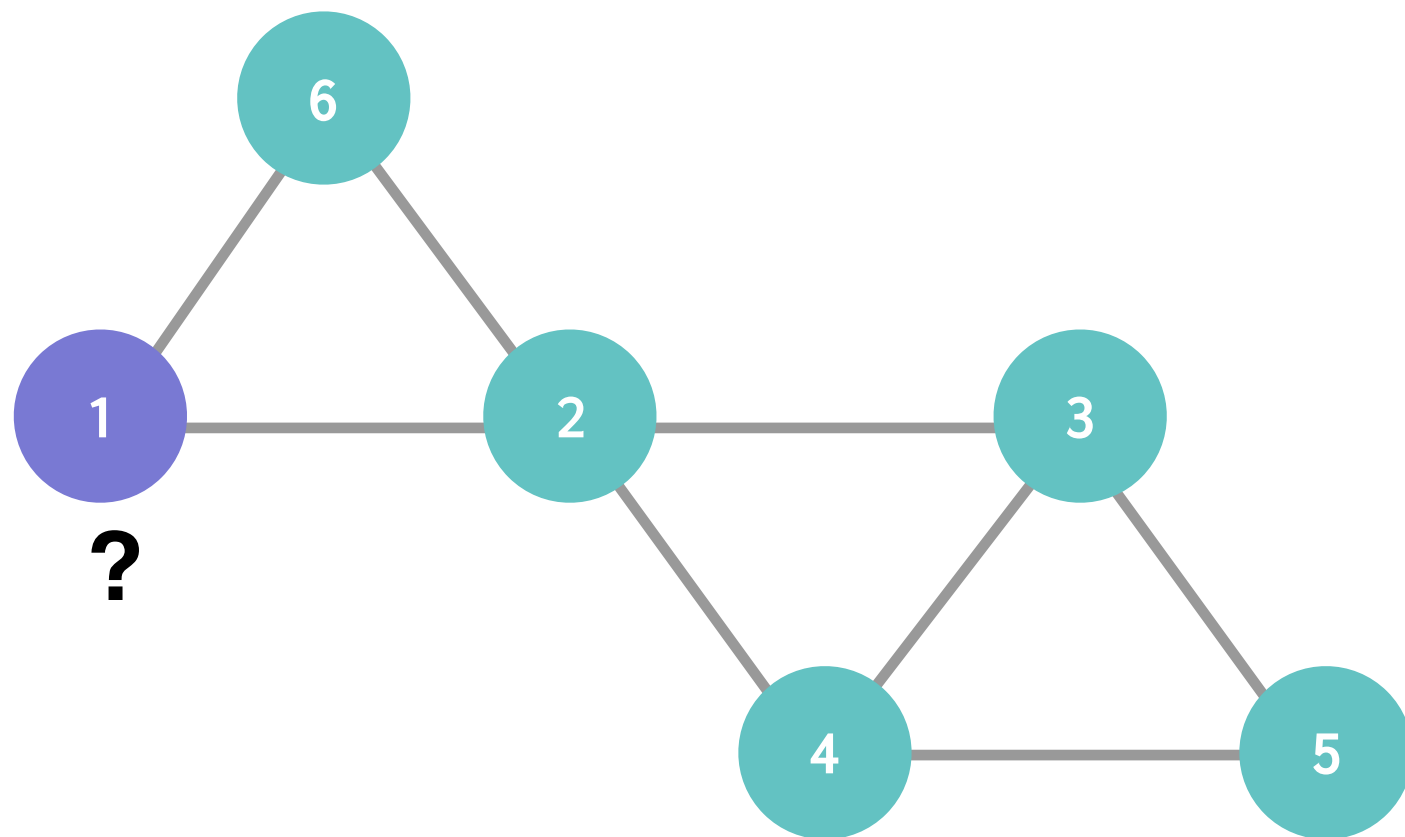
각각의 정점에 대하여 **인접한 정점 번호**를 저장



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

그래프의 구현 : 인접 리스트

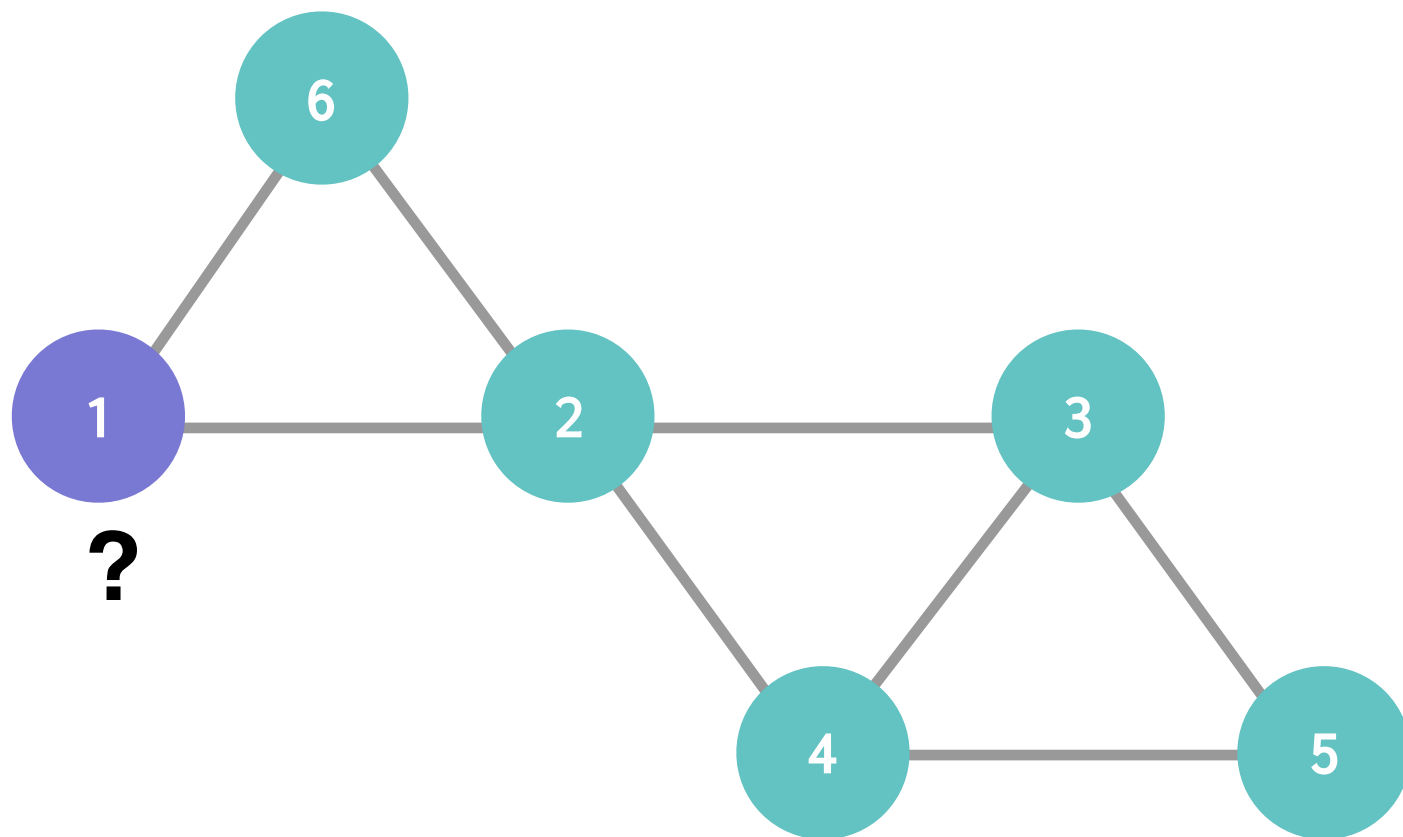
각각의 정점에 대하여 **인접한 정점 번호**를 저장



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

그래프의 구현 : 인접 리스트

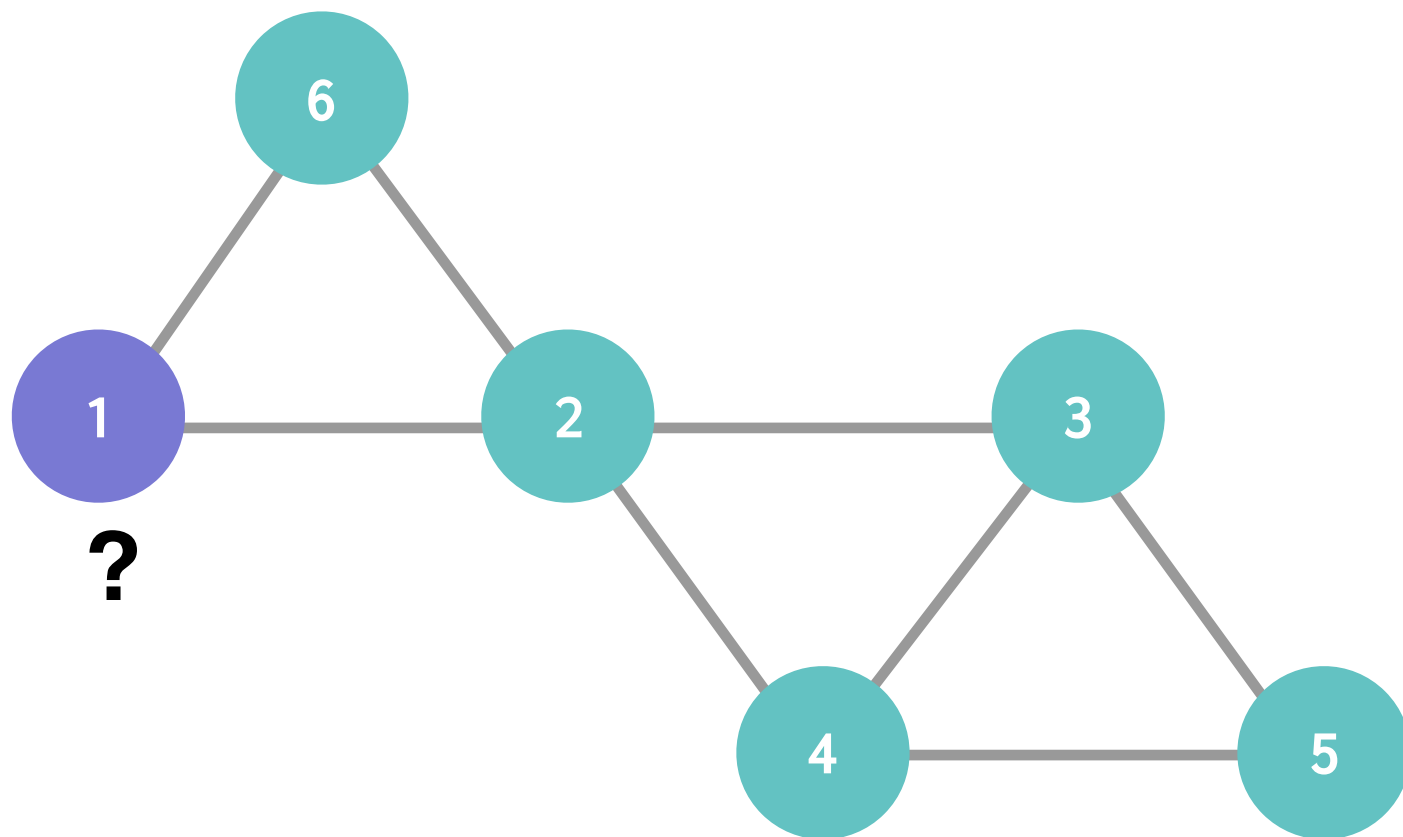
각각의 정점에 대하여 **인접한 정점 번호**를 저장



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

그래프의 구현 : 인접 리스트

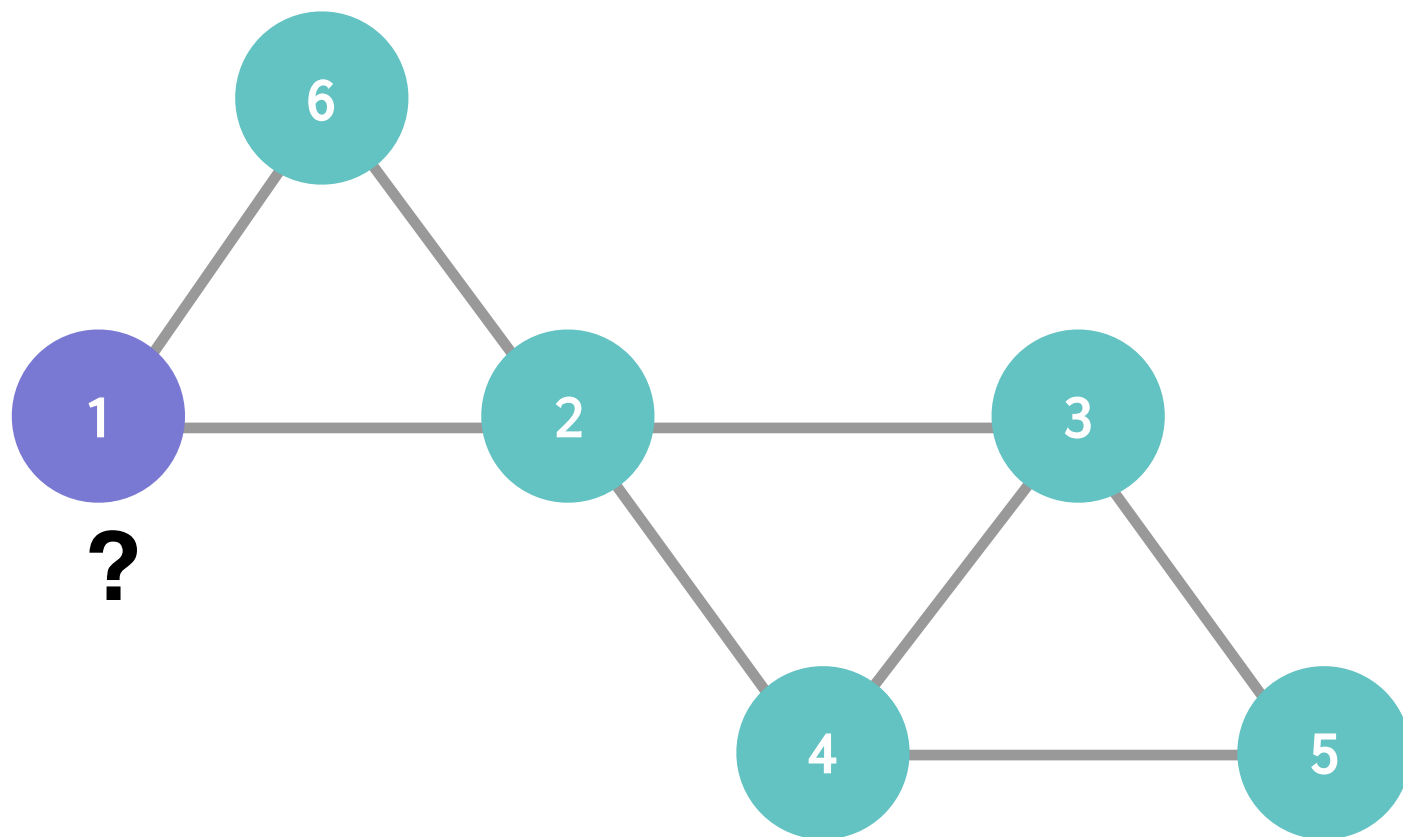
장점 : 인접한 정점을 모두 찾는데 필요한 만큼만 본다



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

그래프의 구현 : 인접 리스트

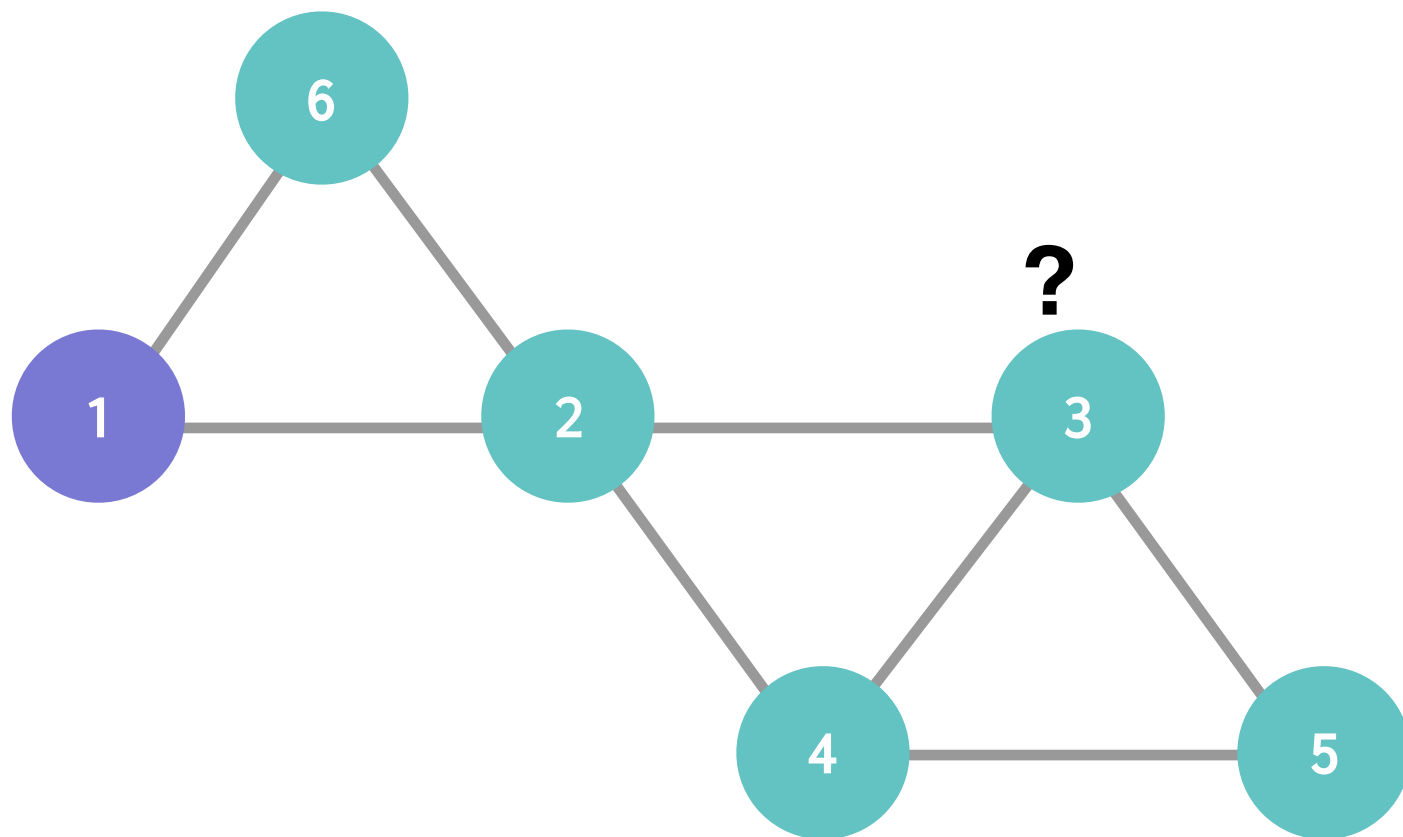
장점 : 필요한 만큼만 공간을 활용한다



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

그래프의 구현 : 인접 리스트

단점 : 인접 여부를 보려면 인접한 정점을 모두 봐야한다

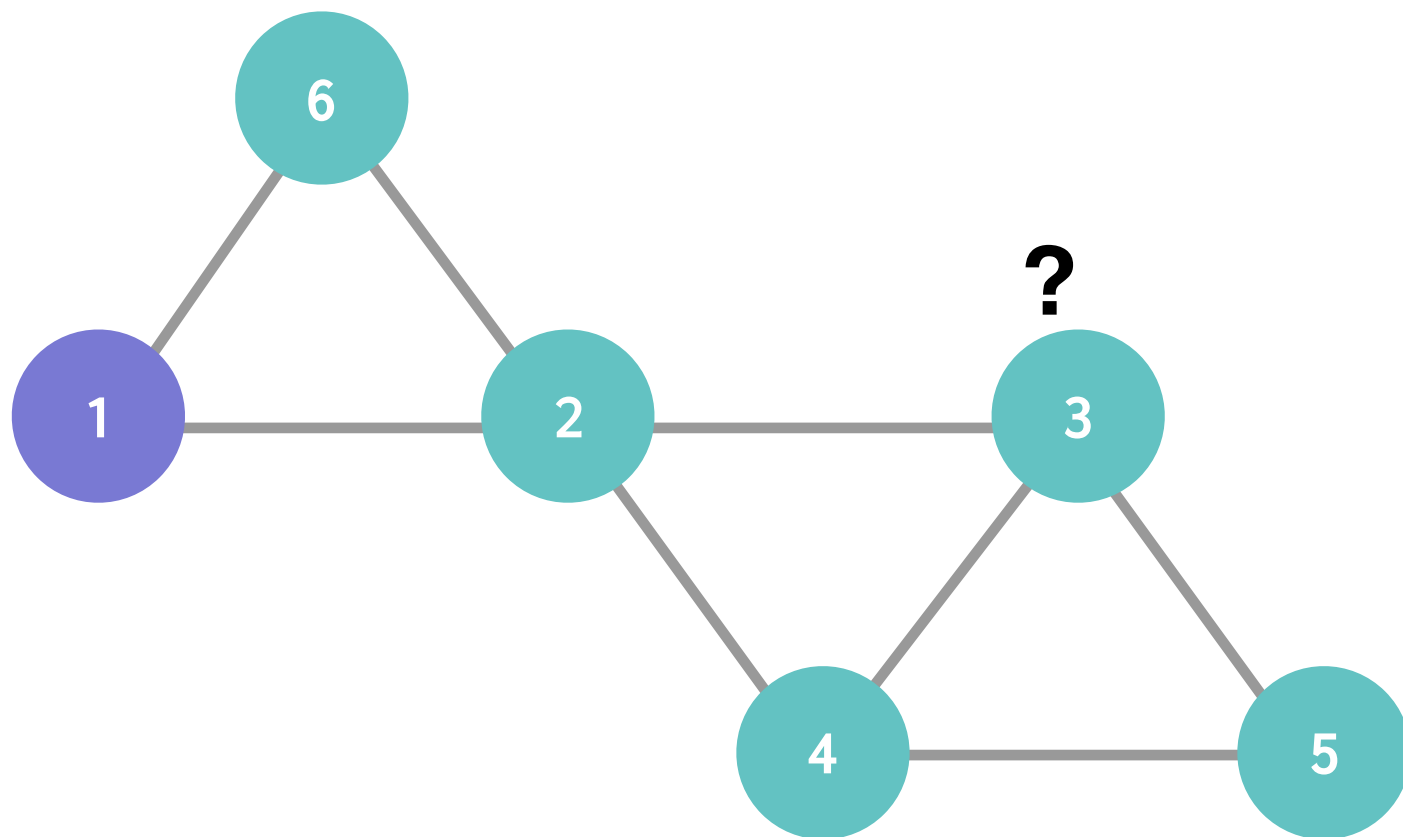


1	2	6		
2	1	3	4	6
3	2	4	5	
4	2	3	5	
5	3	4		
6	1	2		

그래프의 구현 : 인접 리스트

단점 : 인접 여부를 보려면 인접한 정점을 모두 봐야한다

$O(\deg(v))$



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

그래프의 구현 요약

	인접행렬	인접리스트
장점	인접 여부확인 : $O(1)$	인접한 모든 정점 : $O(\deg(v))$ 그래프 저장을 위한 공간 : $O(E)$
단점	인접한 모든 정점 : $O(n)$ 그래프 저장을 위한 공간 : $O(n^2)$	인접 여부확인 : $O(\deg(v))$

그래서 무엇을 써야하나

인접행렬을 써야하나

인접 리스트를 써야하나 ?



그때그때 달라요

목적이 무엇인가 ?

두 정점이 인접하여 있는지를 빠르게 알아야 한다
인접행렬

정점 v 와 인접해 있는 모든 정점을 빠르게 알아야 한다
인접 리스트

[예제 2] 그래프의 구현

그래프를 인접행렬 / 인접리스트로 구현하시오

시스템 입력

```
myGraph = graph(3)

myGraph.addEdge(1, 2)
myGraph.addEdge(1, 3)

print(myGraph.isEdge(1,
2))
print(myGraph.isEdge(2,
3))
print(myGraph.getAdj(1))
```

시스템 출력

```
True
False
[2, 3]
```

[예제 2] 그래프의 구현



```
/* elice */
```

요약

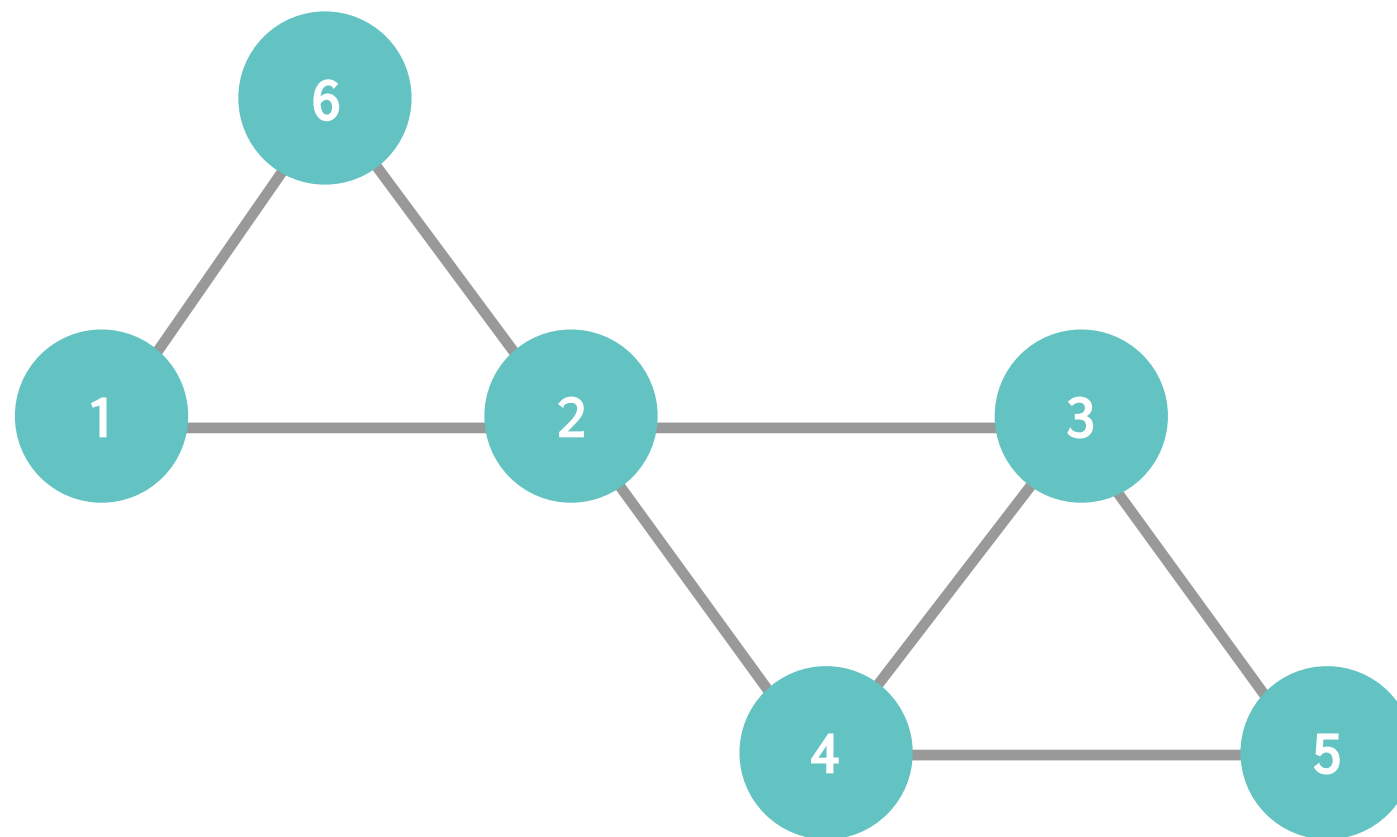
유명한 자료구조는 이미 구현되어 있는걸 찾고, 쓰자
직접 구현해서 좋을 것이 거의 없다

그래프는 중요하다
응용할 수 있는 여지가 많고, 많은 수학적 성질이 있다

그래프는 그 이론과 구현이 어렵다
차별점을 둘 수 있는 기회이다

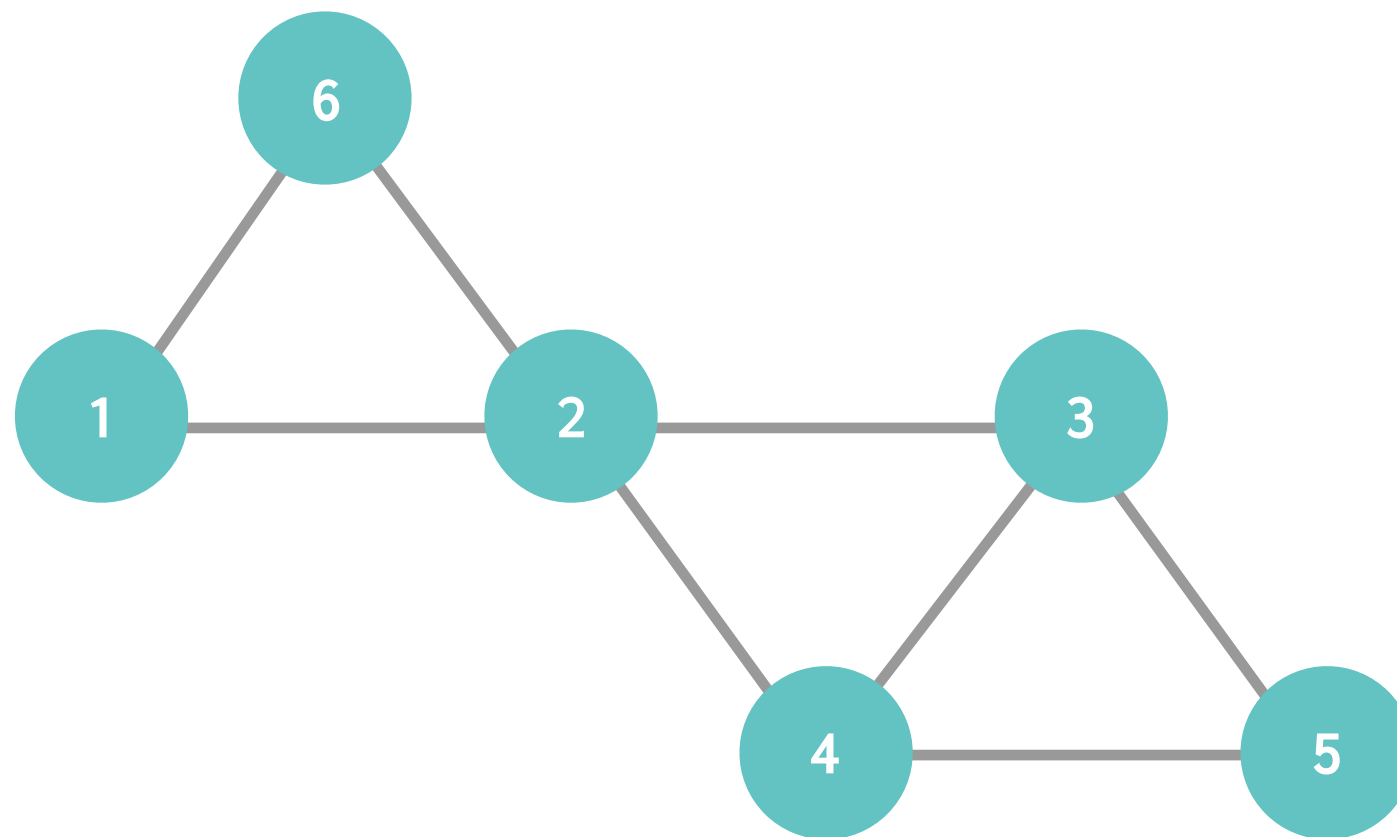
그래프

정점과 간선으로 이루어진 자료구조



그래프 순회

그래프라는 자료구조에 어떠한 값이 저장되어 있는가?



그래프 순회

깊이우선탐색 (Depth First Search)

너비우선탐색 (Breadth First Search)

그래프 순회

깊이우선탐색 (Depth First Search)

스택을 이용하여 그래프를 순회하는 방법

너비우선탐색 (Breadth First Search)

큐를 이용하여 그래프를 순회하는 방법

깊이우선탐색 (DFS)

깊이우선탐색 (Depth First Search)

스택을 이용하여 그래프를 순회하는 방법

깊이우선탐색 (DFS)

깊이우선탐색 (Depth First Search)

스택을 이용하여 그래프를 순회하는 방법

스택 = 선행관계

깊이우선탐색 (DFS)

깊이우선탐색 (Depth First Search)

스택을 이용하여 그래프를 순회하는 방법

스택 = 선행관계

나를 먼저 방문하고 그 다음으로 인접한 노드를 차례로 방문
단, 방문했던 노드는 방문하지 않는다

깊이우선탐색의 과정 (재귀호출)

DFS(v, visited)

현재 정점이 v이고 지금까지 방문한 정점이 visited일 때,
깊이우선탐색 결과를 리스트로 반환하는 함수

깊이우선탐색의 과정 (재귀호출)

DFS(v, visited)

현재 정점이 v이고 지금까지 방문한 정점이 visited일 때,
깊이우선탐색 결과를 리스트로 반환하는 함수

인접한 정점을 모두 방문한 경우

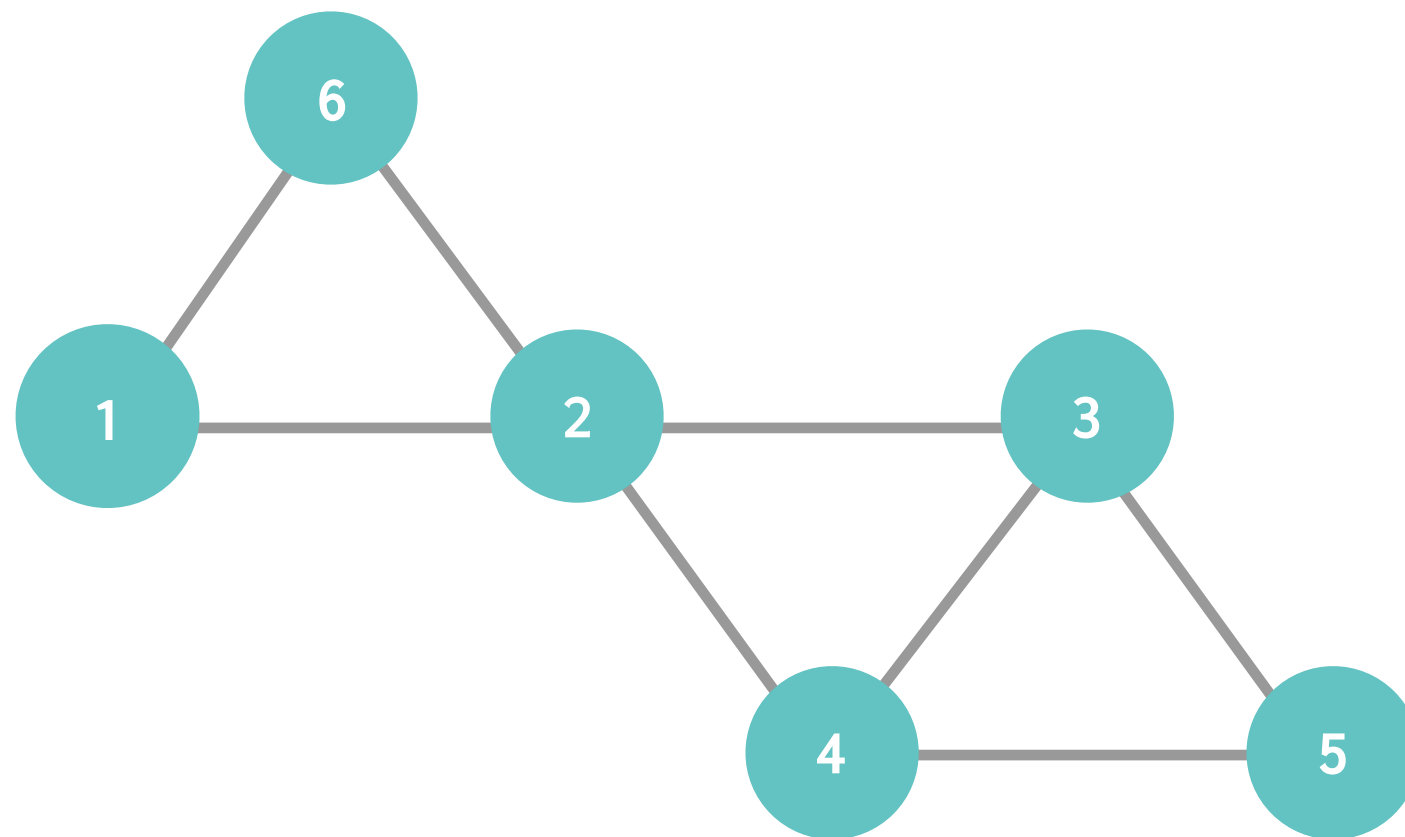
깊이우선탐색의 과정

DFS(v , visited) :

1. v 를 방문했다고 처리한다
2. v 와 인접한 모든 w 에 대하여 다음을 반복
3. 만약 w 를 아직 방문하지 않았다면,
4. DFS(w , visited)
5. 방문순서 반환

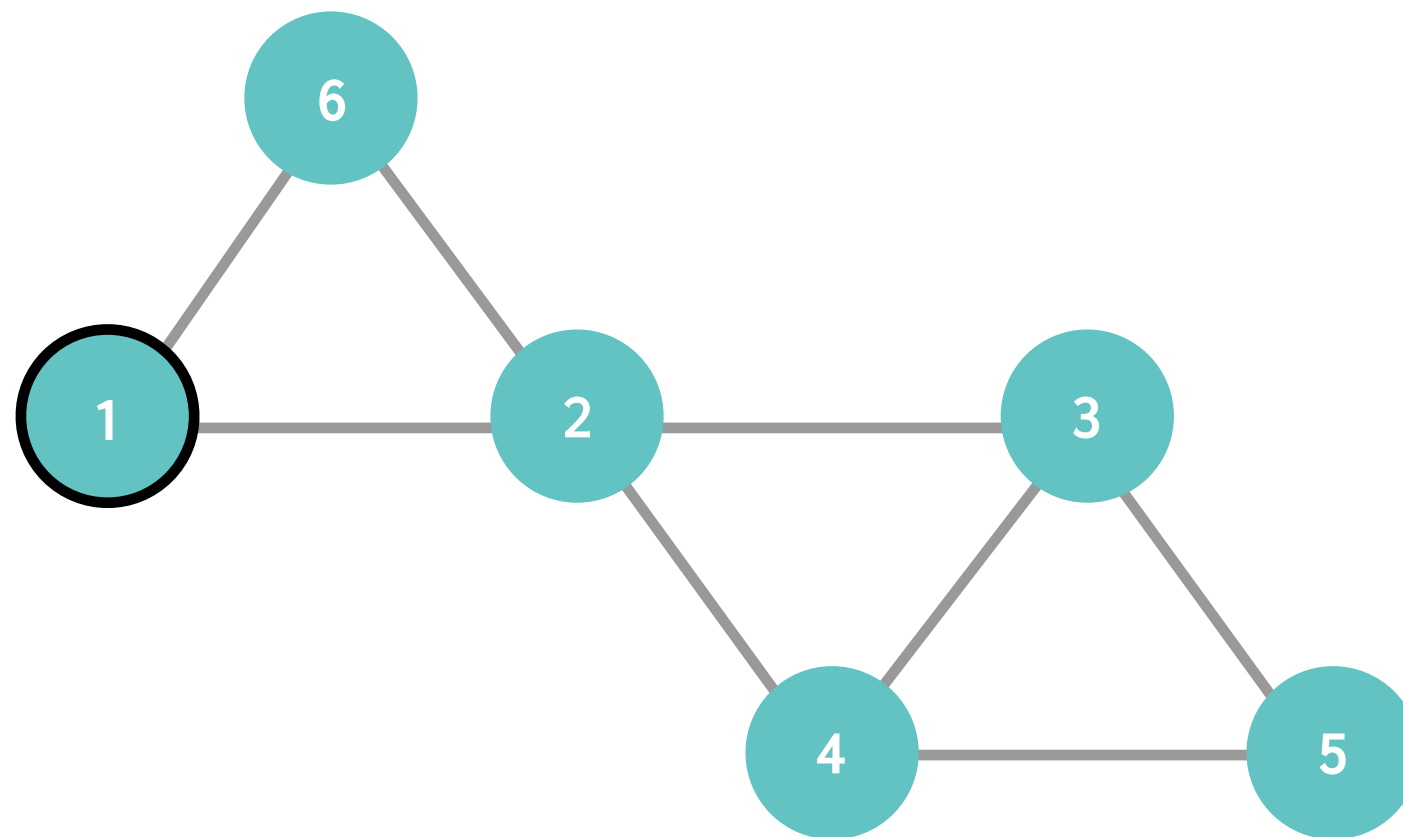
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



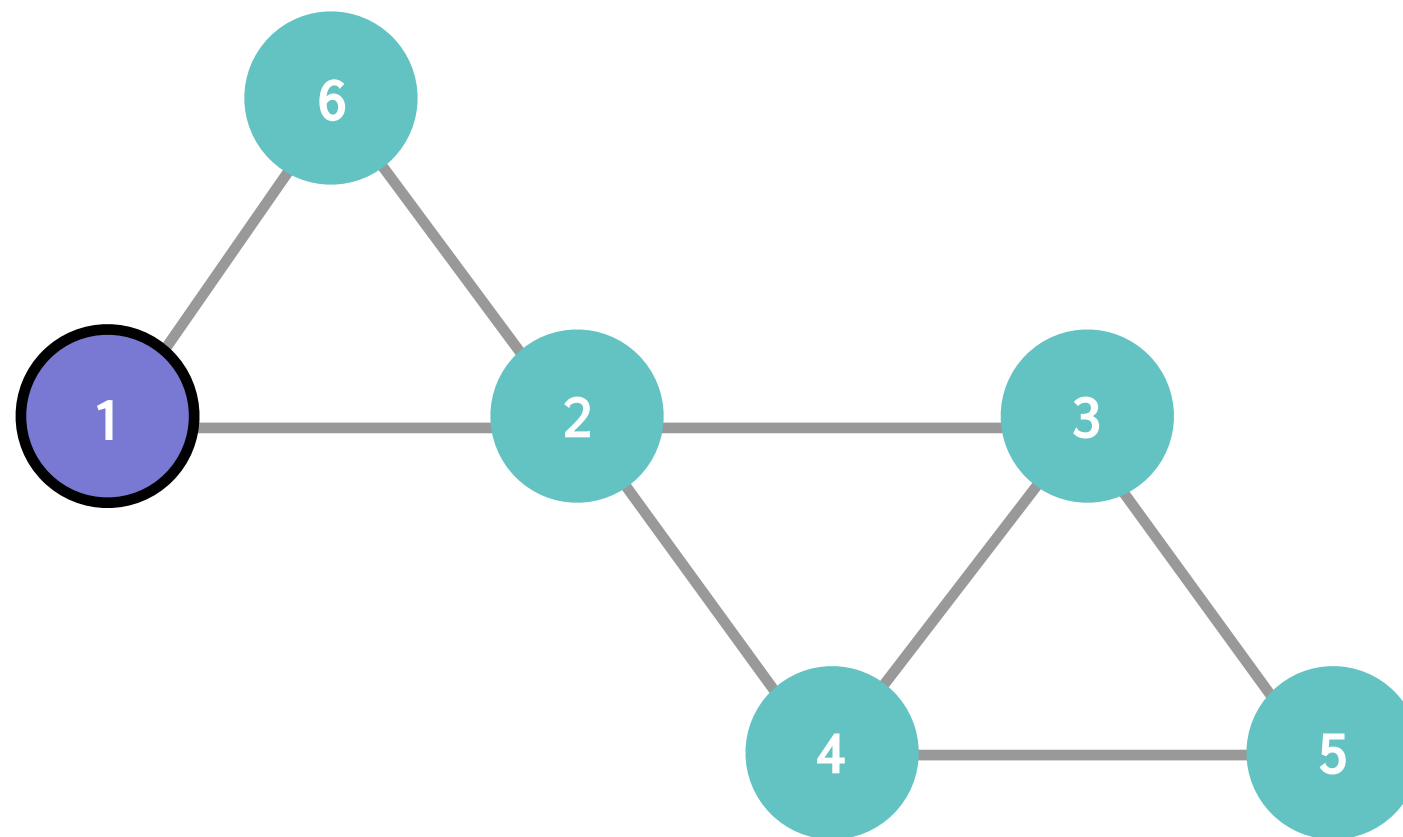
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



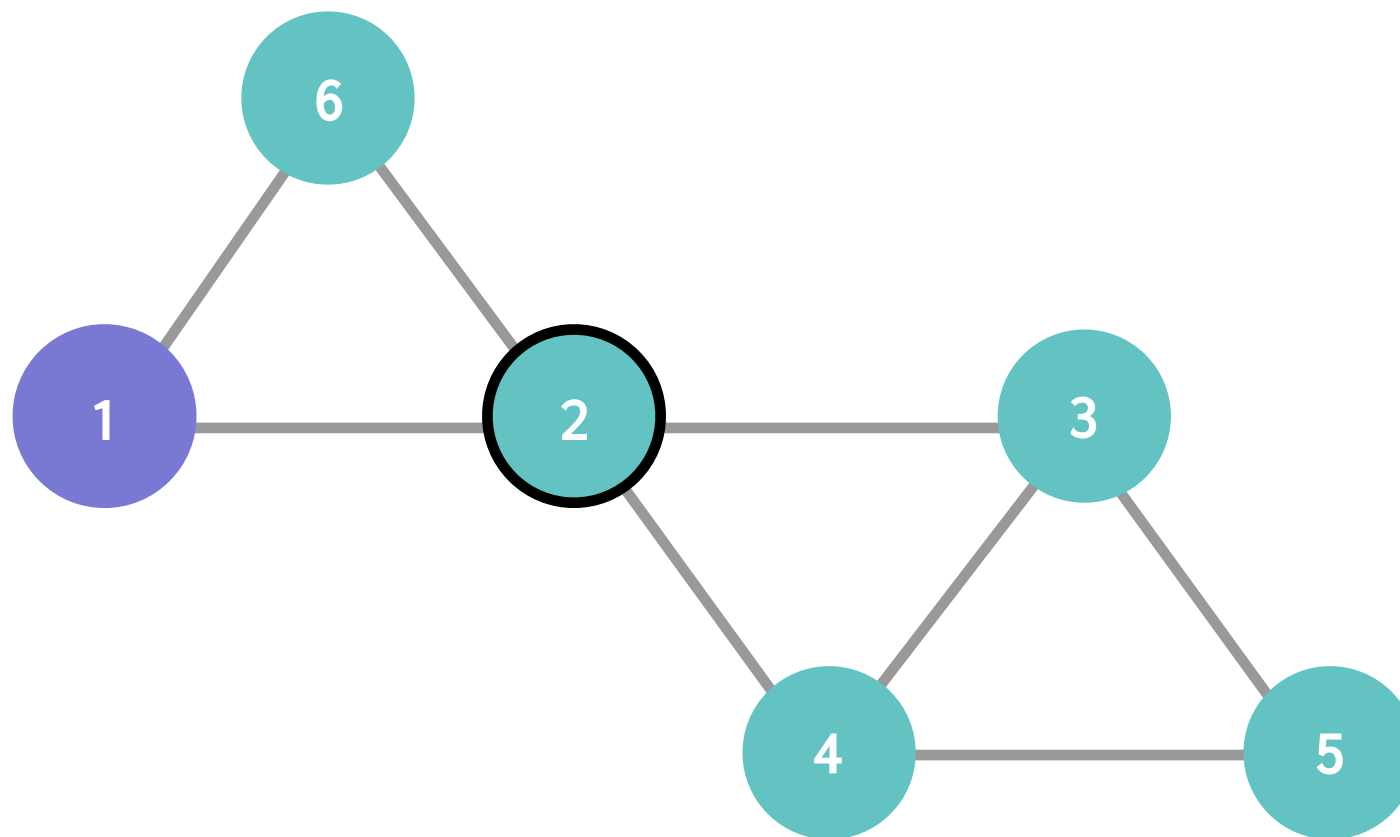
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



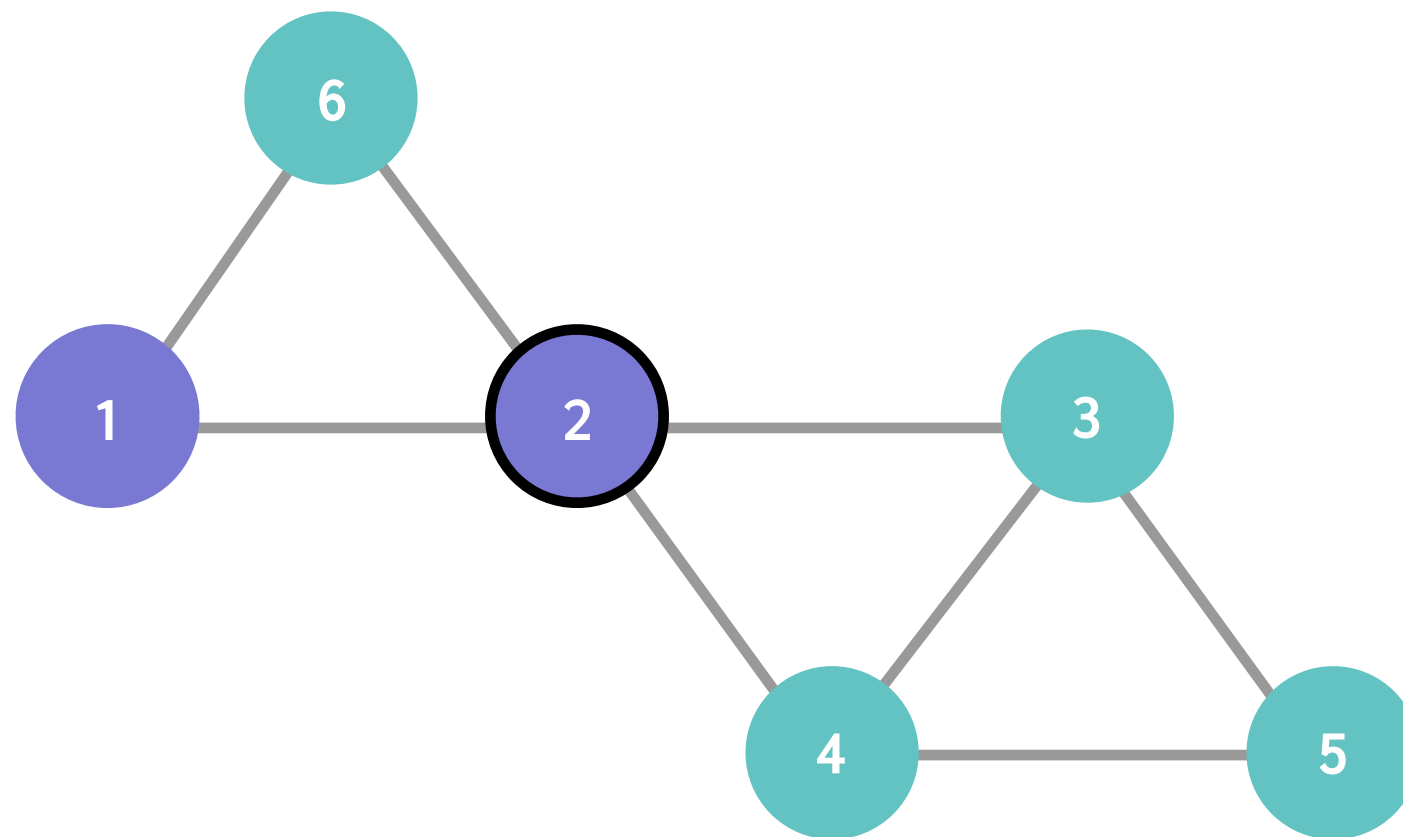
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



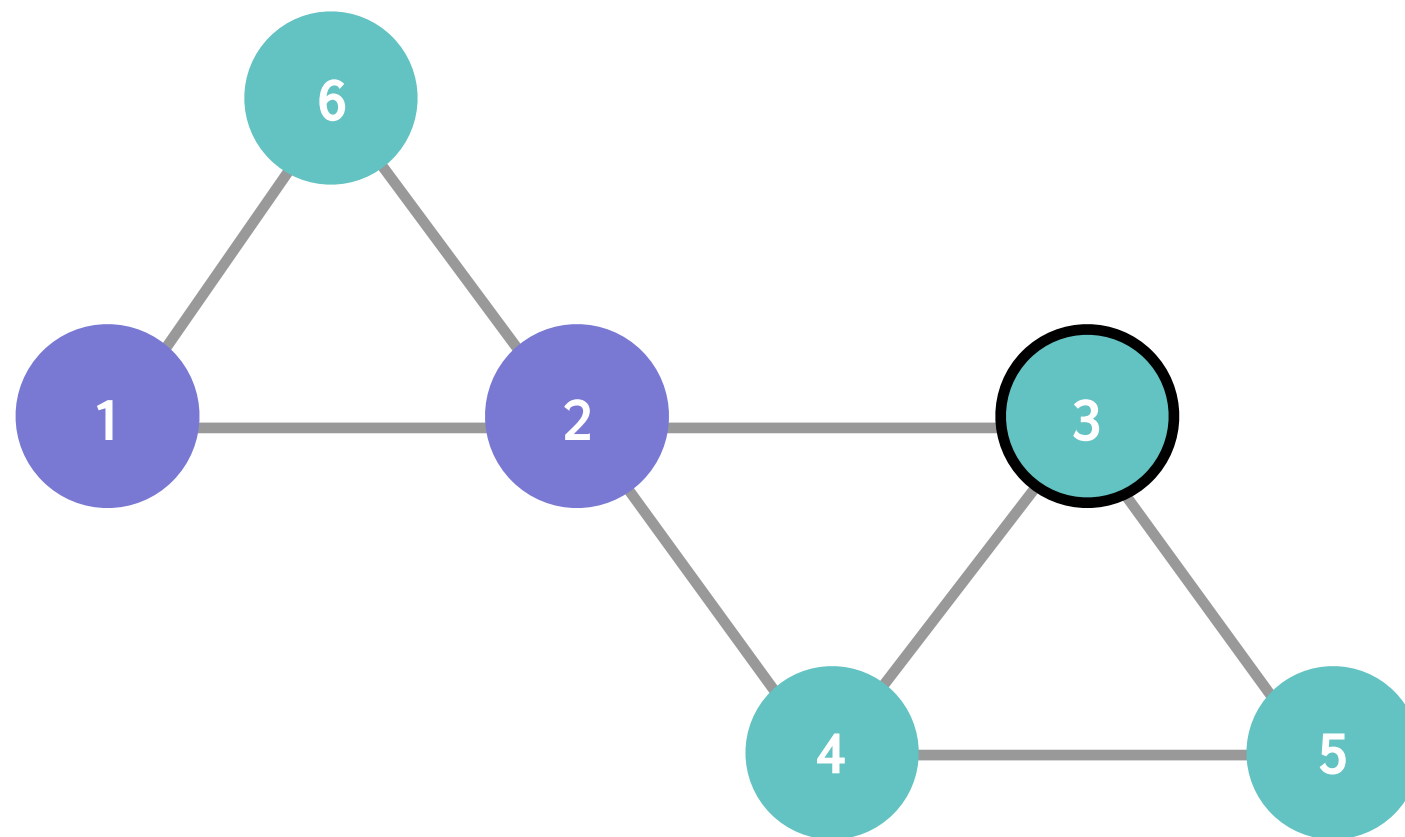
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



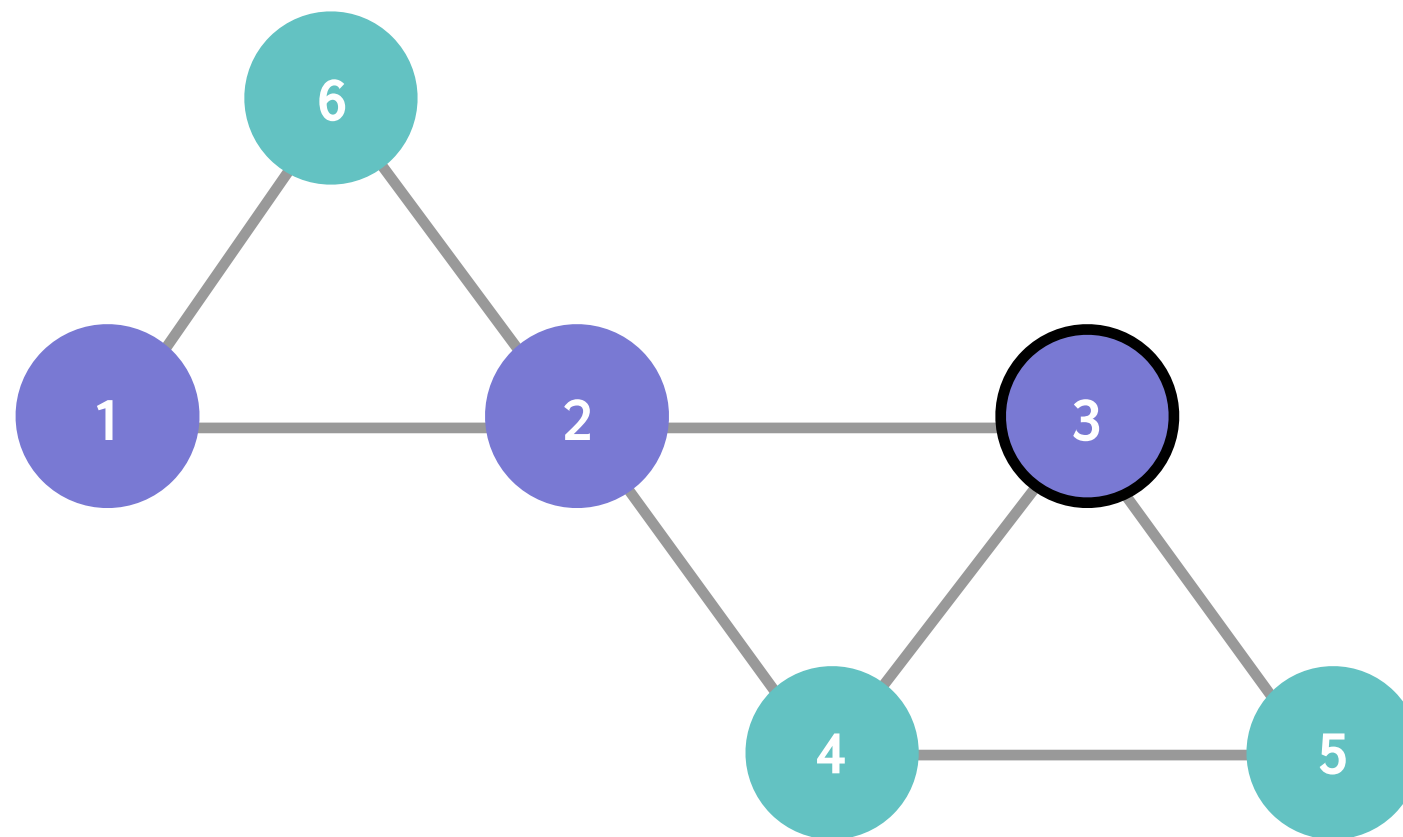
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



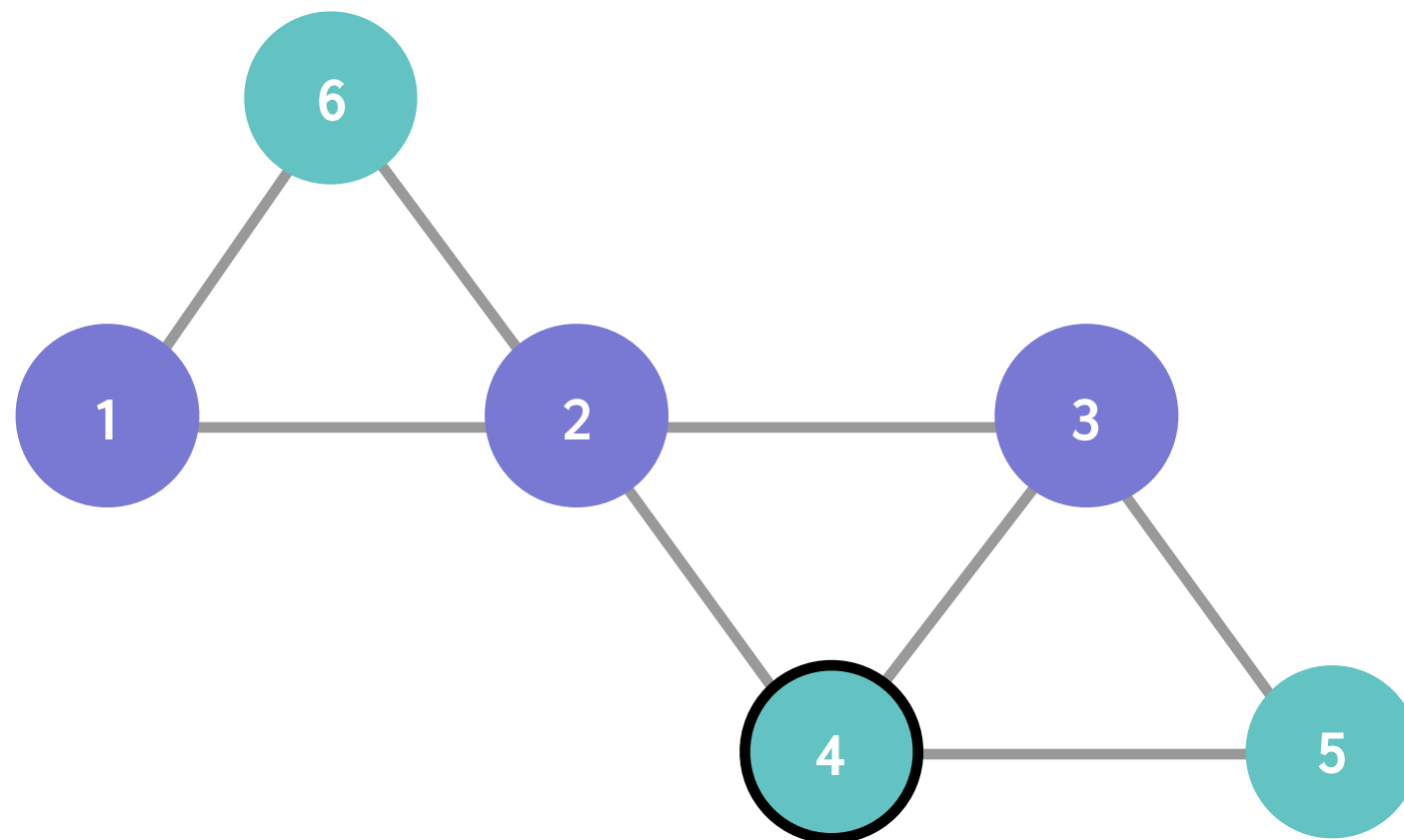
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



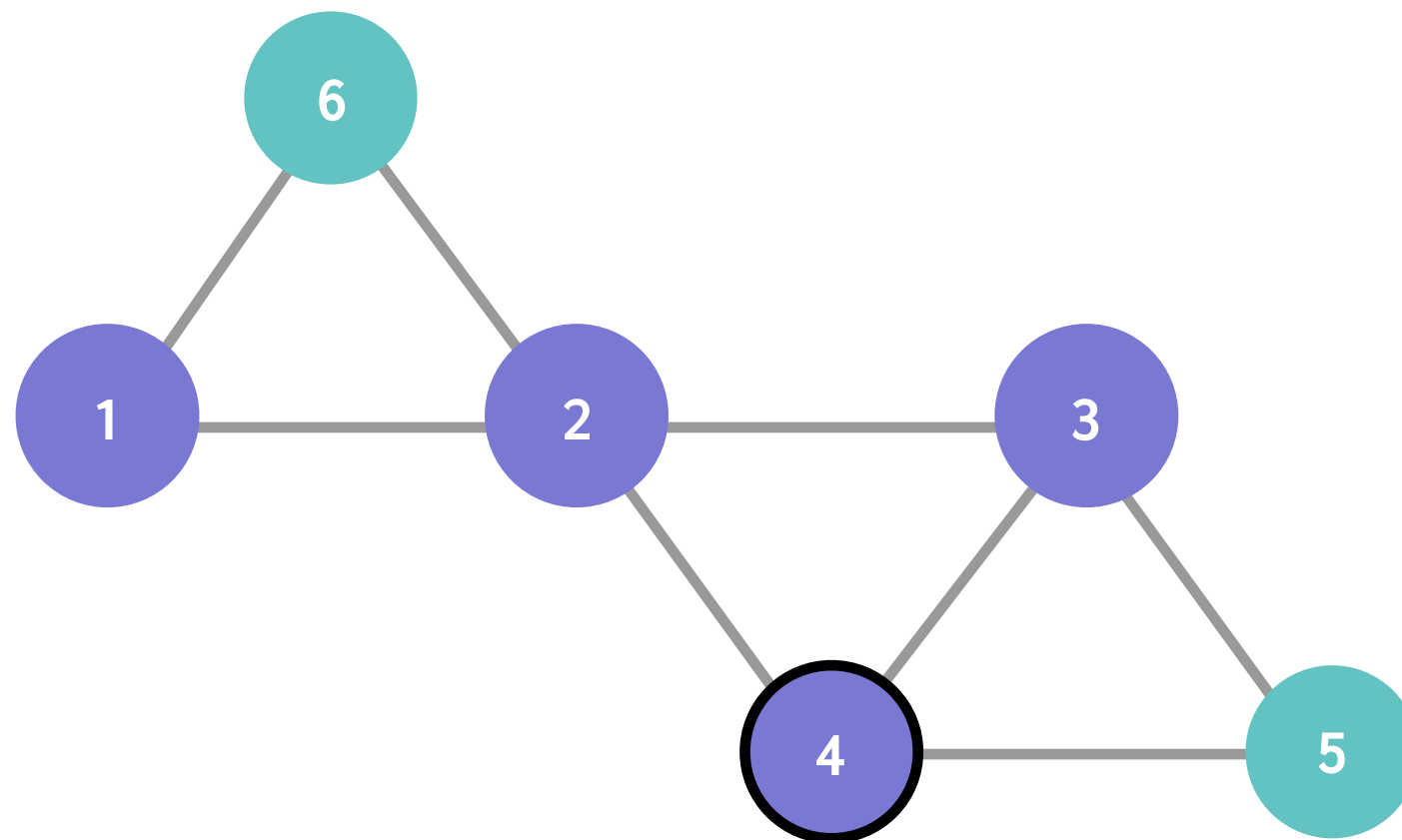
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



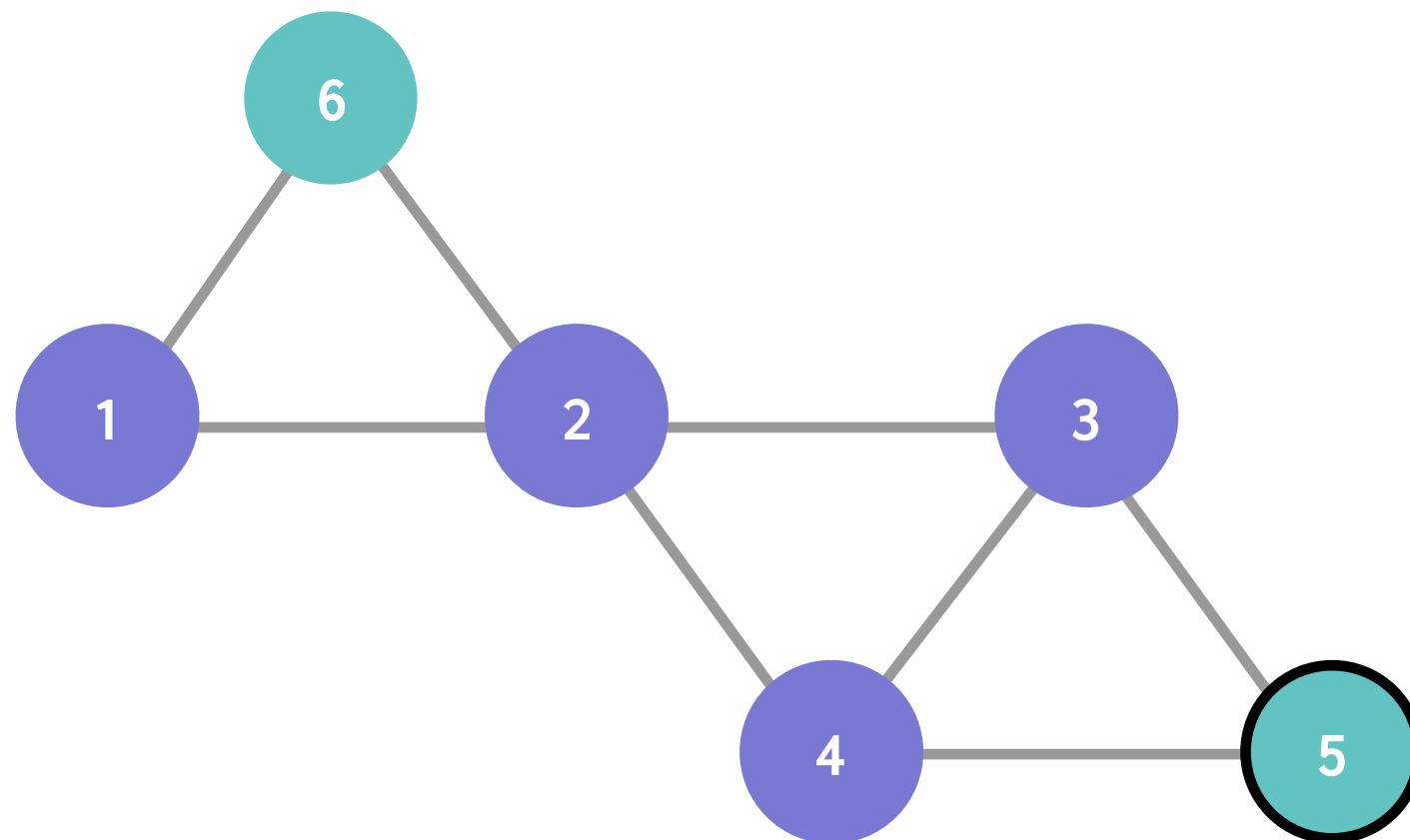
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



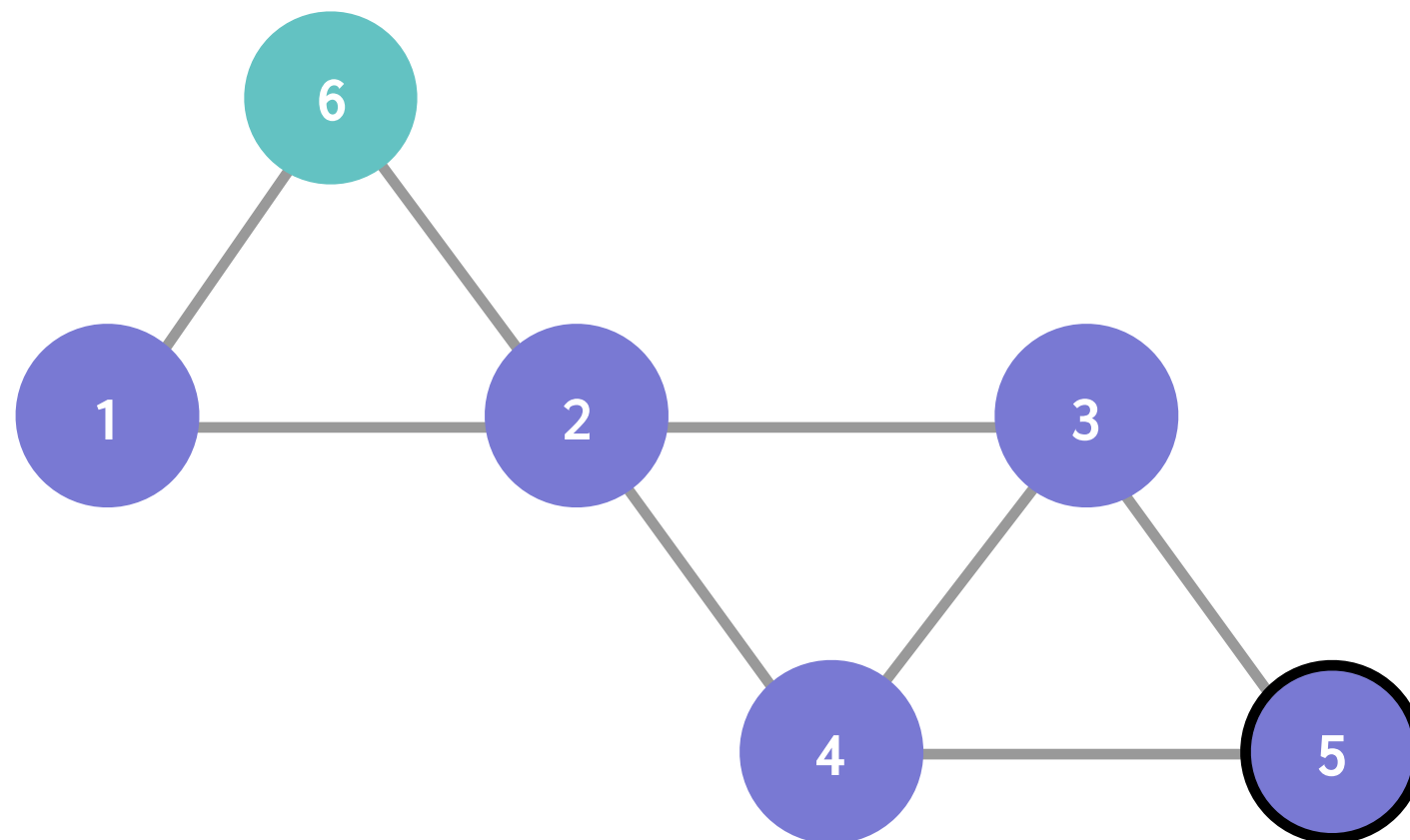
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



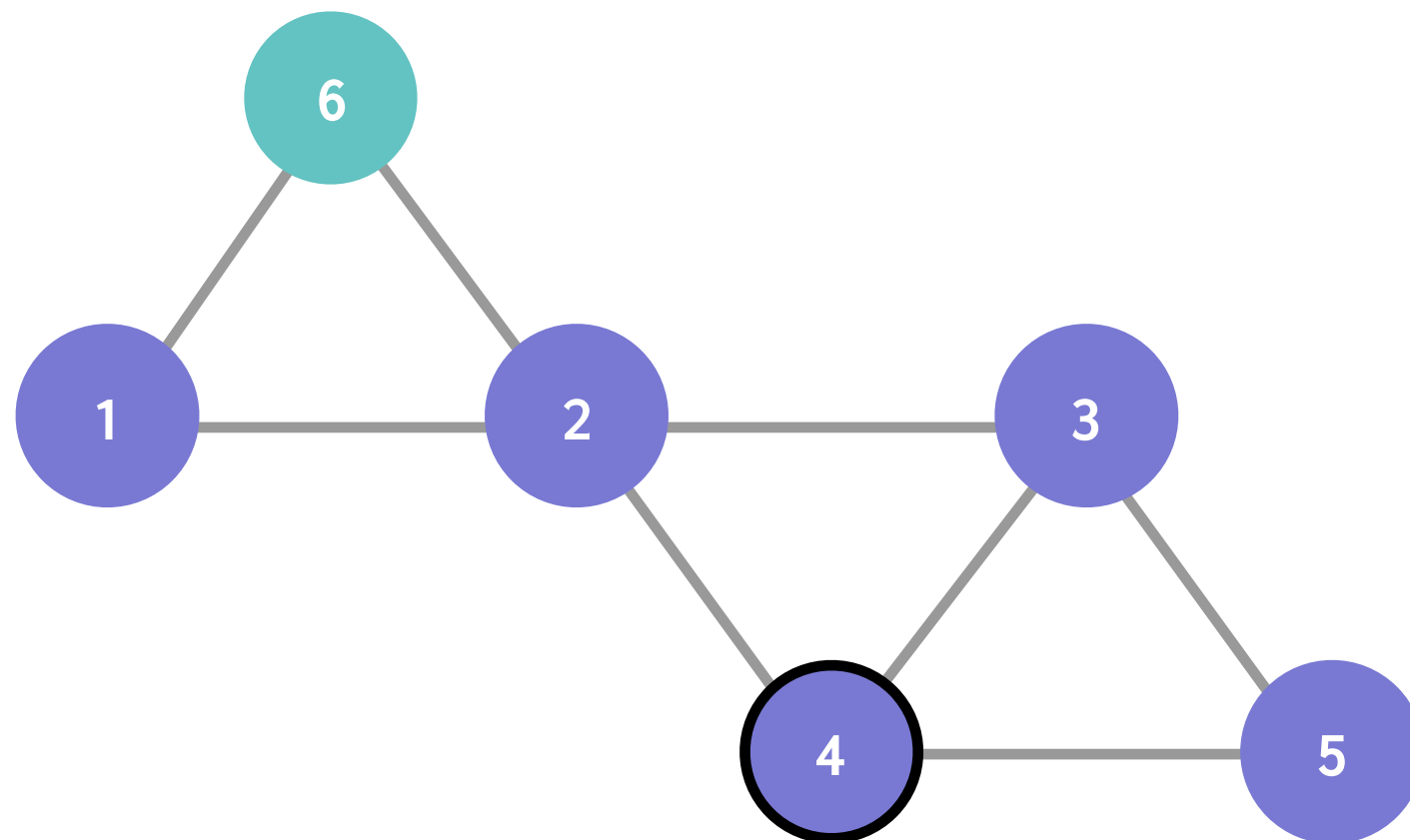
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



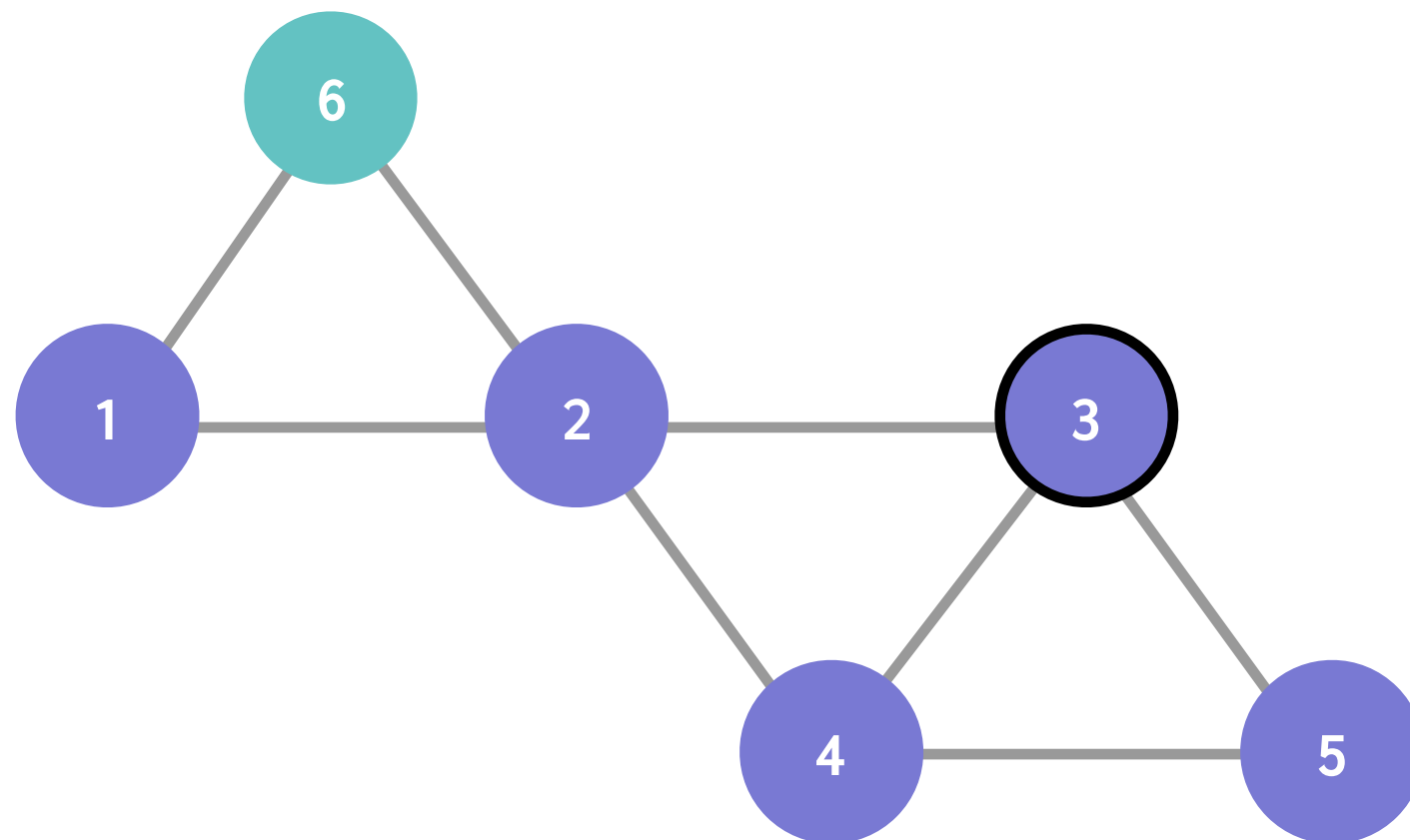
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



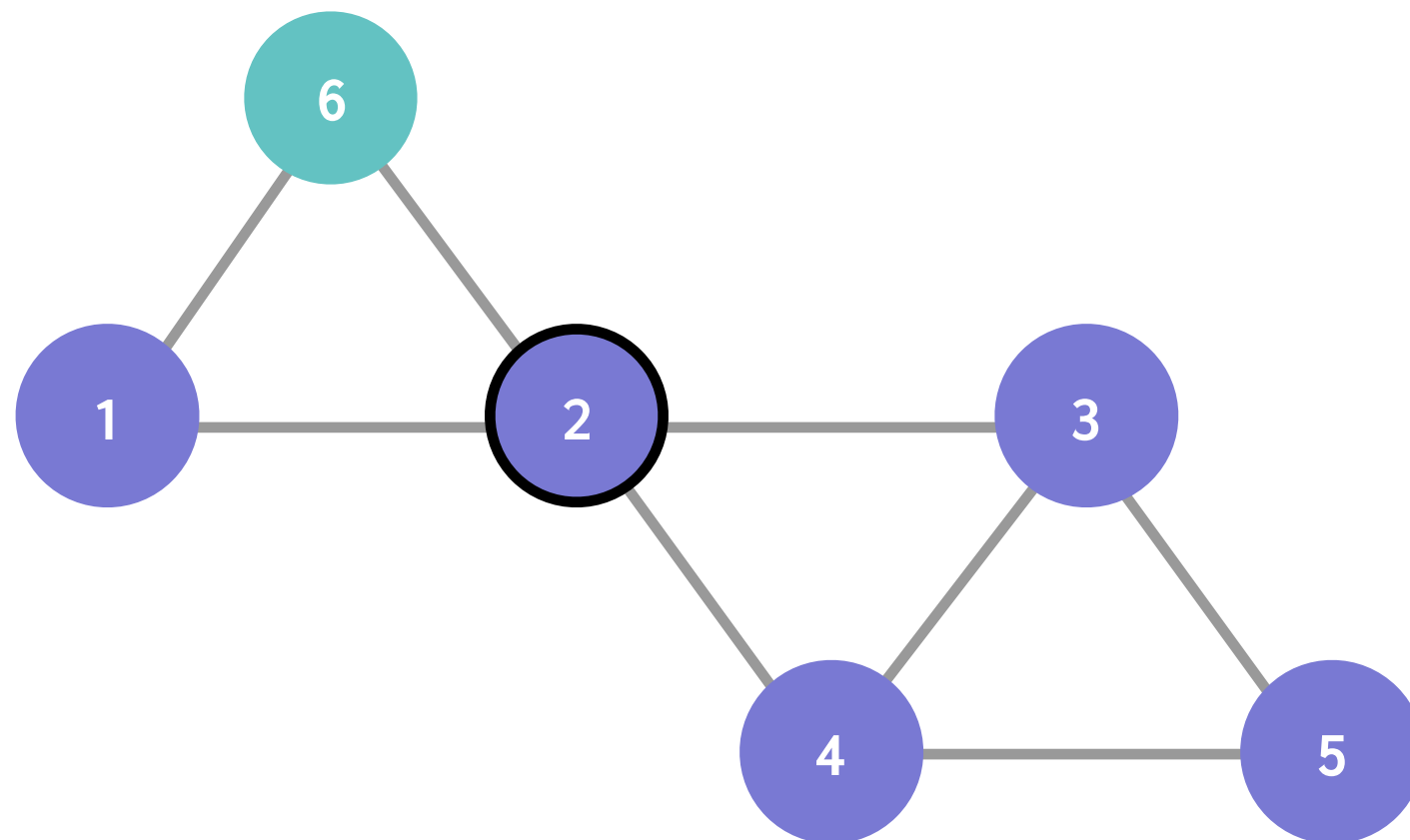
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



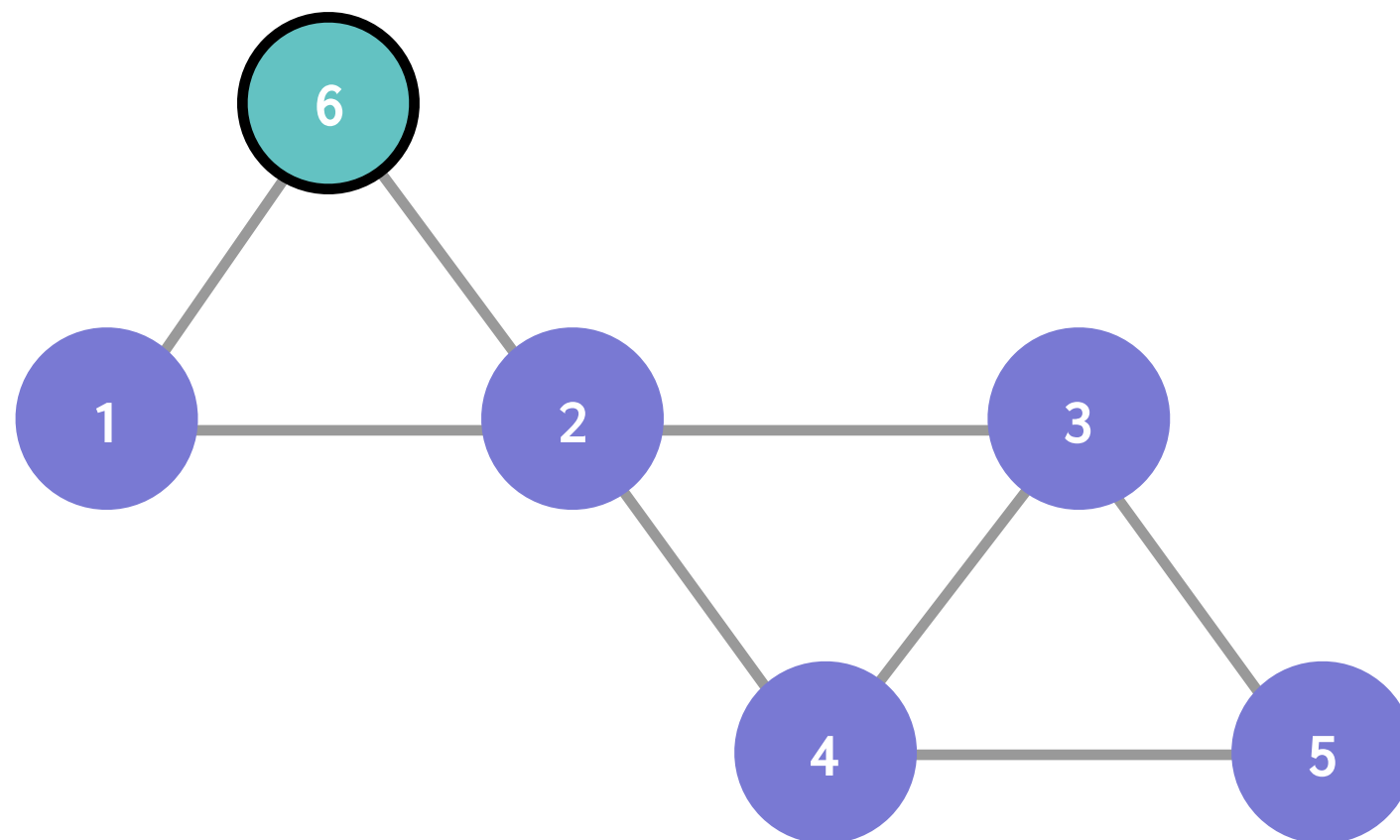
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



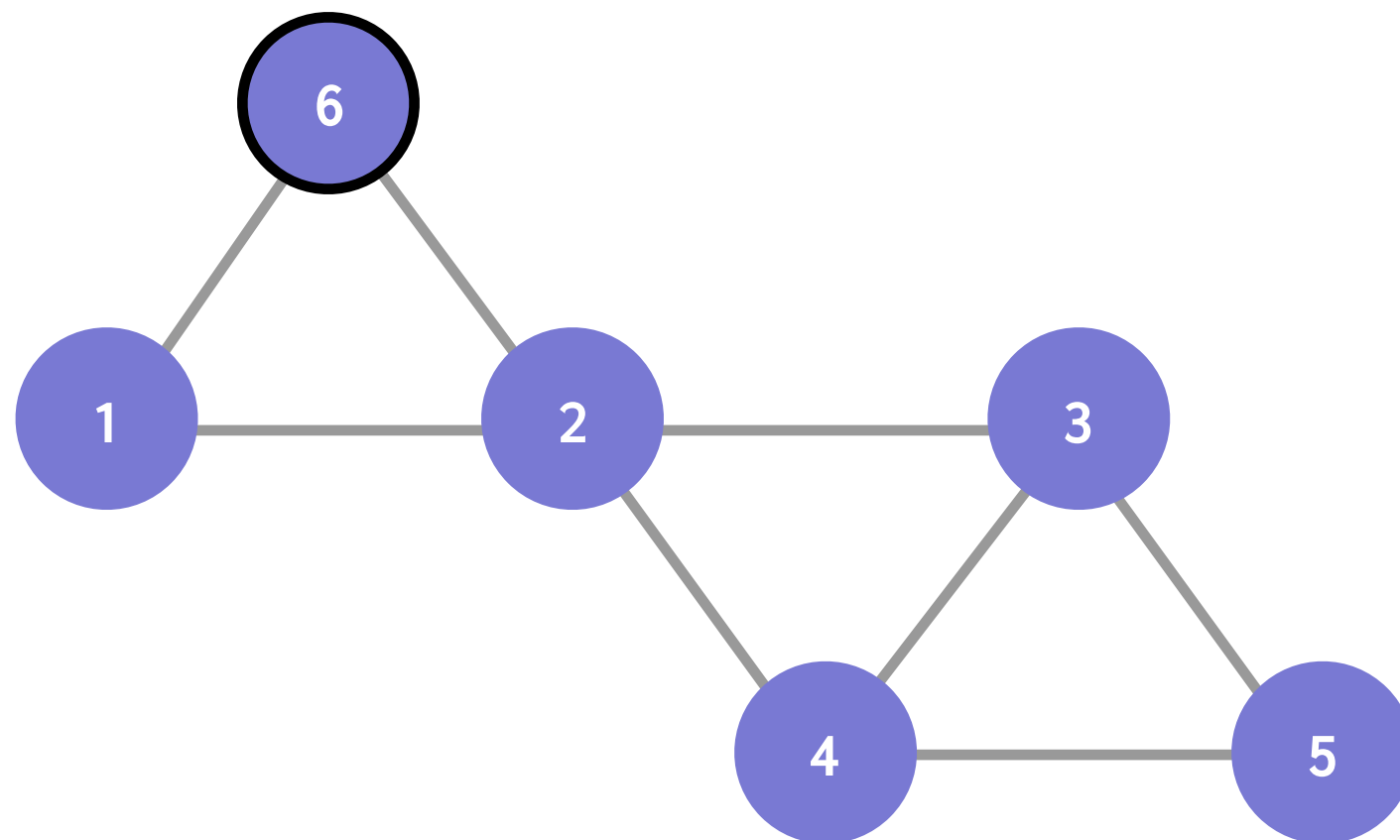
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



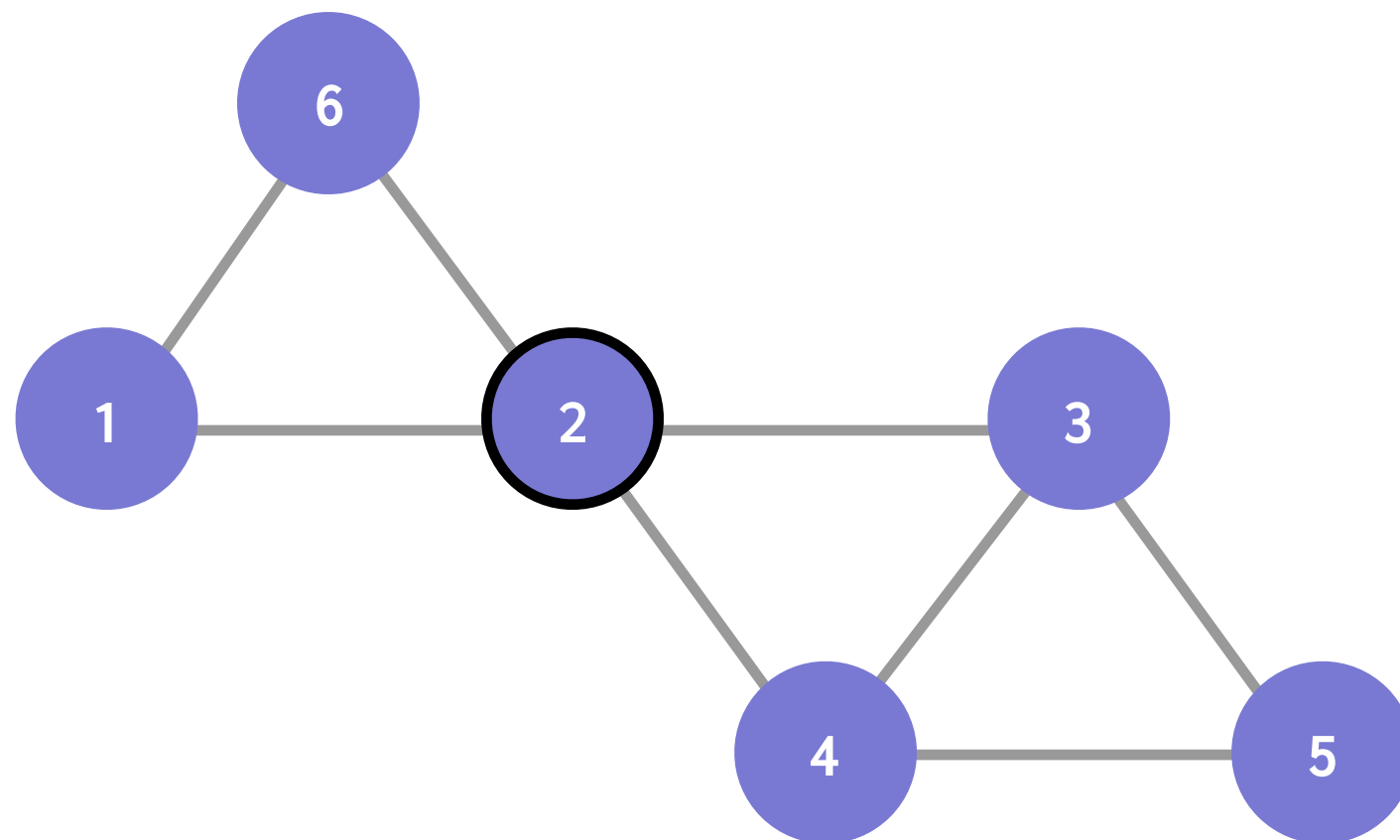
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



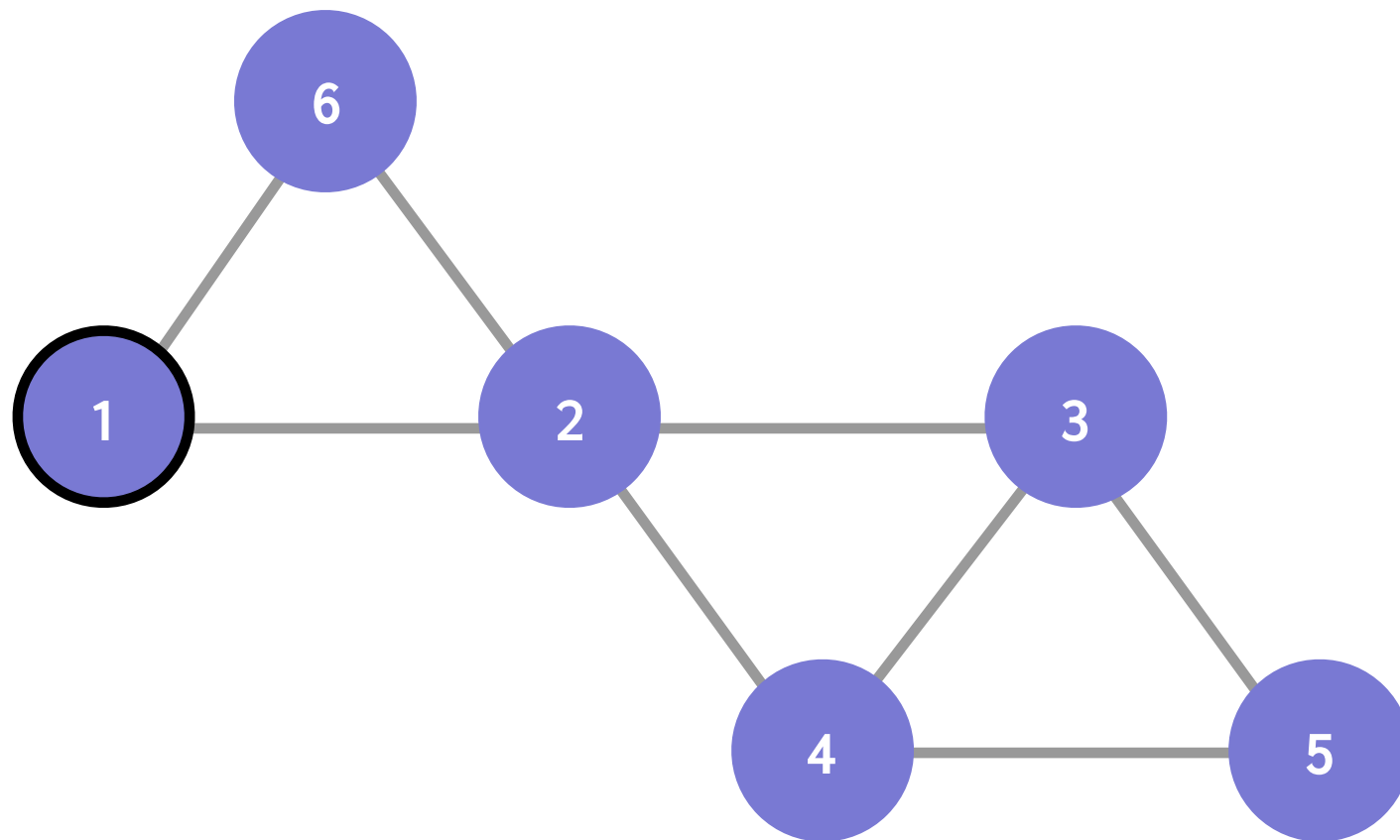
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



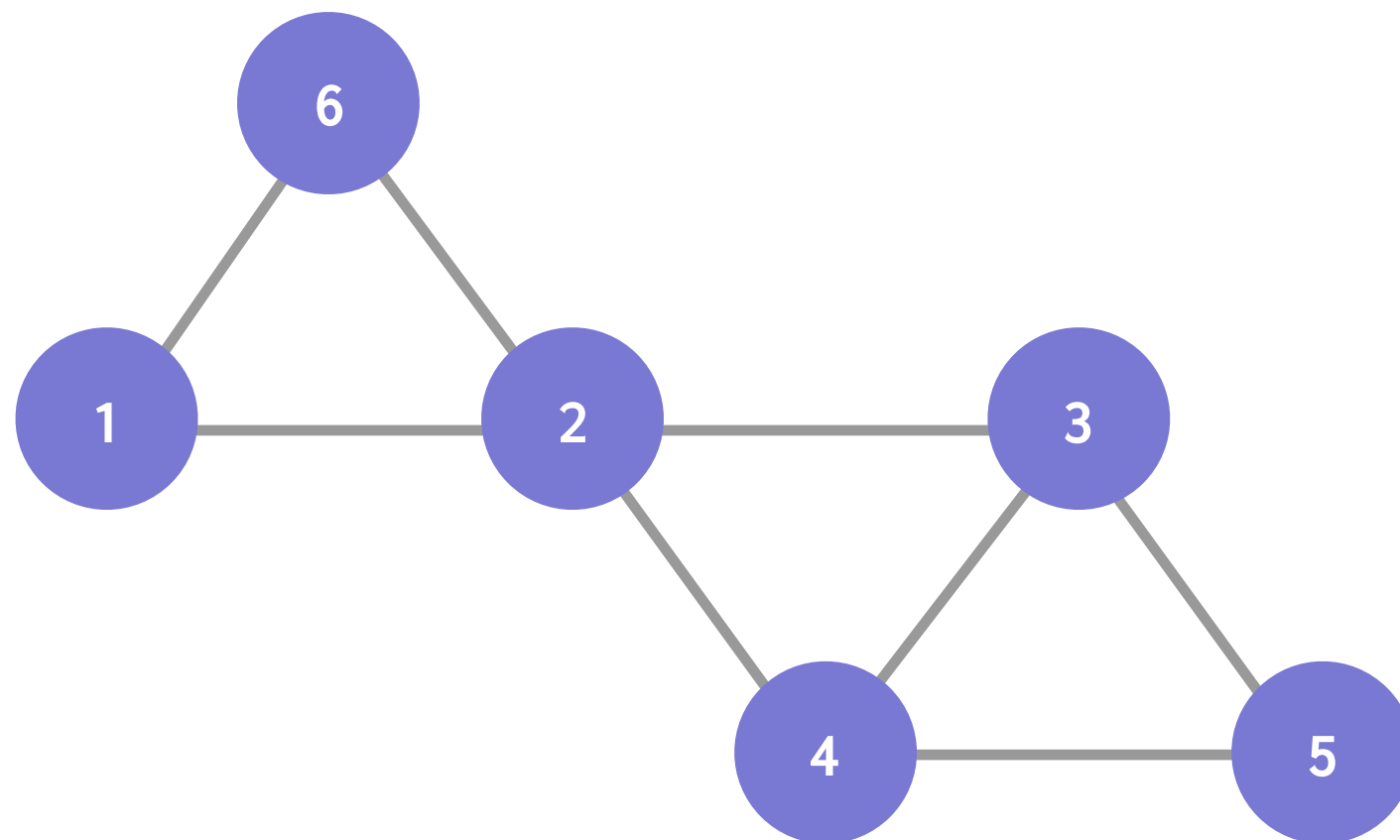
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



깊이우선탐색(DFS)의 예제

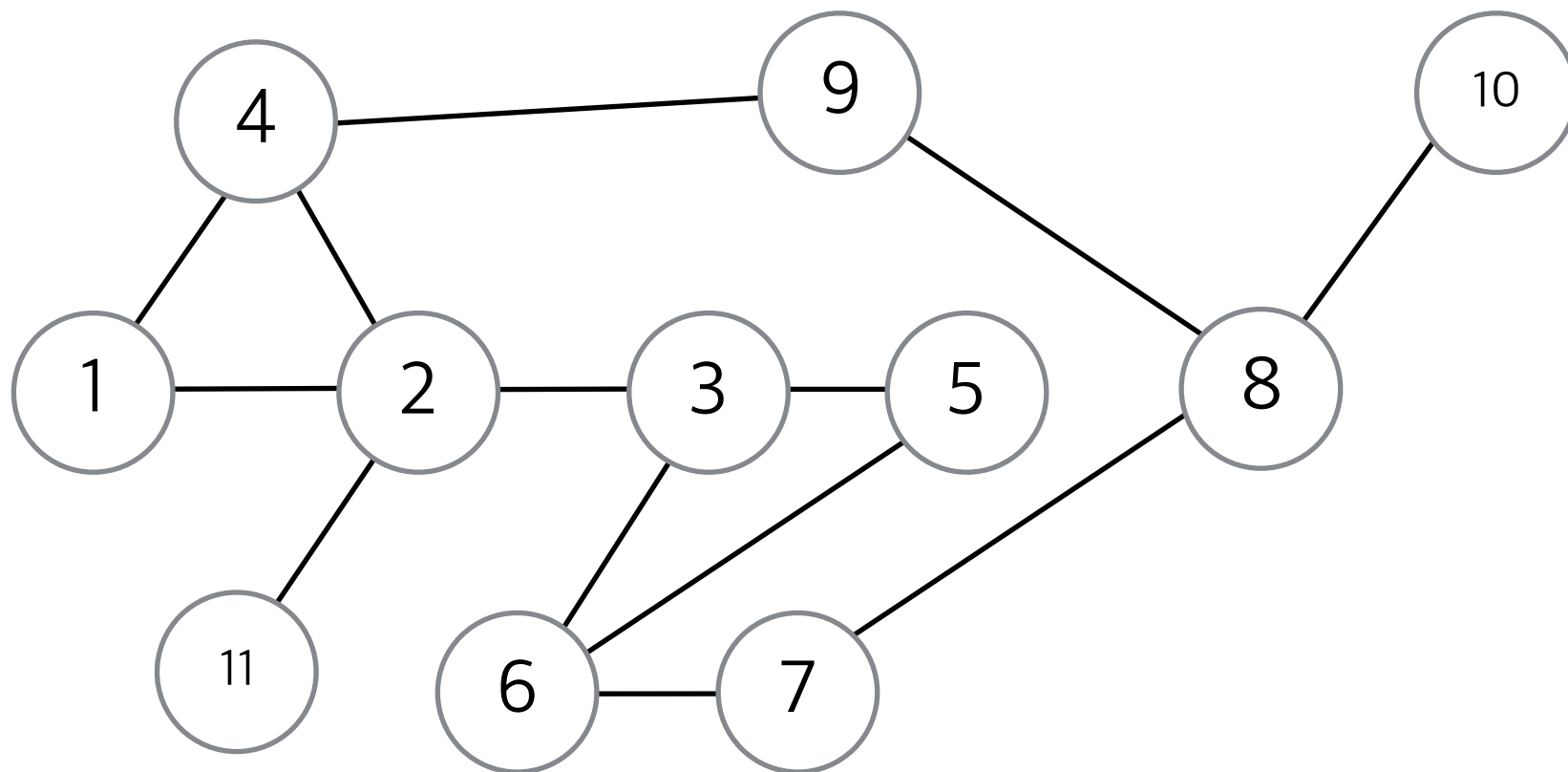
정점과 간선으로 이루어진 자료구조



깊이 우선 탐색 (DFS)

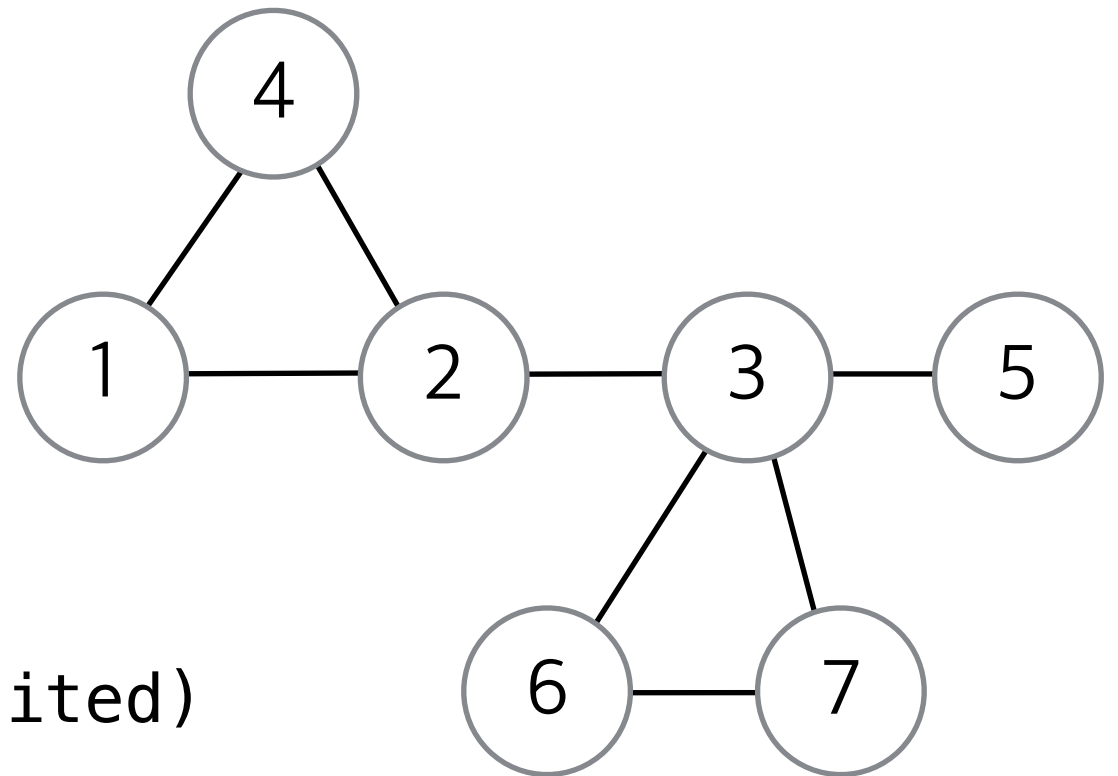
다음 그래프에 대하여 1을 시작으로 DFS한 결과는 ?

단, 인접한 노드가 여러개일 경우 노드 번호가 작은 노드부터 방문



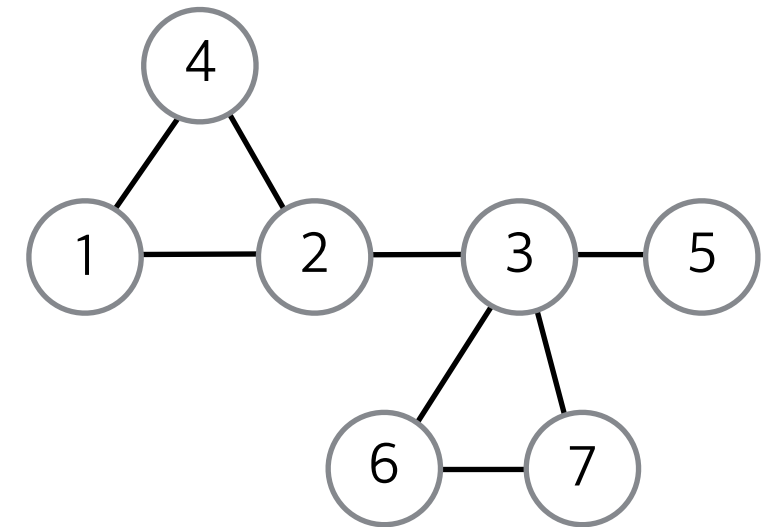
깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```



visited= [F, T, F, F, F, F, F, F]

```
graph[1] = [2, 4]  
graph[2] = [1, 3, 4]  
graph[3] = [2, 5, 6, 7]  
graph[4] = [1, 2]  
graph[5] = [3]  
graph[6] = [3, 7]  
graph[7] = [3, 6]
```

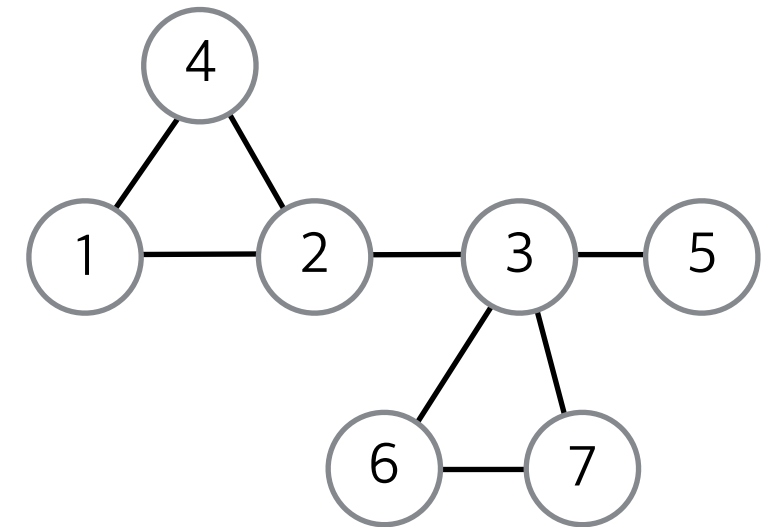
깊이 우선 탐색 (DFS) 구현

```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

```
    return result
```

```
DFS(graph, 1, visited)
```

```
    result = null
```



```
visited= [F, T, F, F, F, F, F, F]
```

```
graph[1] = [2, 4]
```

```
graph[2] = [1, 3, 4]
```

```
graph[3] = [2, 5, 6, 7]
```

```
graph[4] = [1, 2]
```

```
graph[5] = [3]
```

```
graph[6] = [3, 7]
```

```
graph[7] = [3, 6]
```

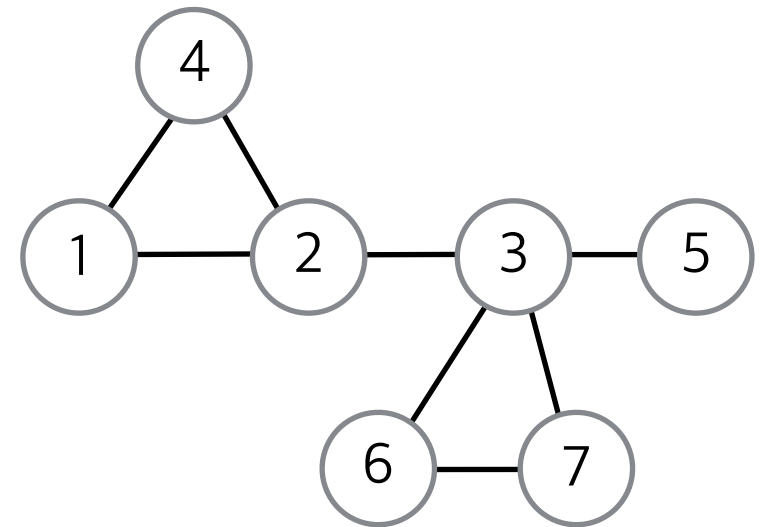
깊이 우선 탐색 (DFS) 구현

```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

```
    return result
```

```
DFS(graph, 1, visited)
```

```
    result = null
```



```
visited= [F, F, F, F, F, F, F, F]
```

```
graph[1] = [2, 4]
```

```
graph[2] = [1, 3, 4]
```

```
graph[3] = [2, 5, 6, 7]
```

```
graph[4] = [1, 2]
```

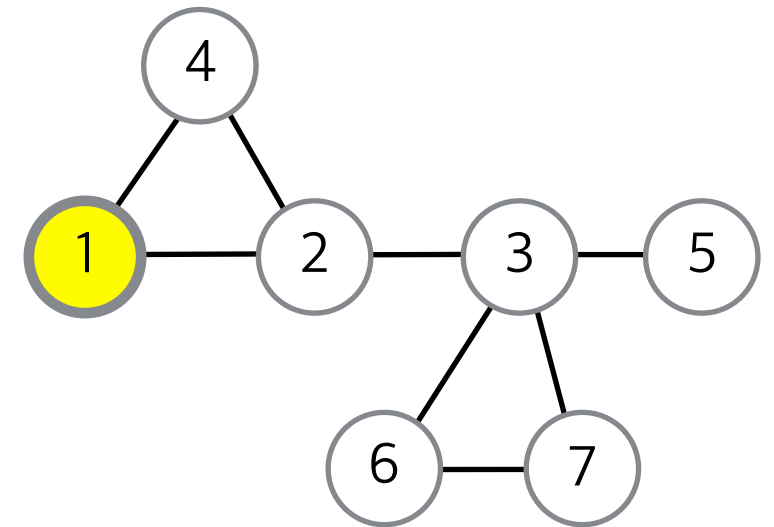
```
graph[5] = [3]
```

```
graph[6] = [3, 7]
```

```
graph[7] = [3, 6]
```

깊이 우선 탐색 (DFS) 구현

```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result  
  
DFS(graph, 1, visited)  
result = null
```

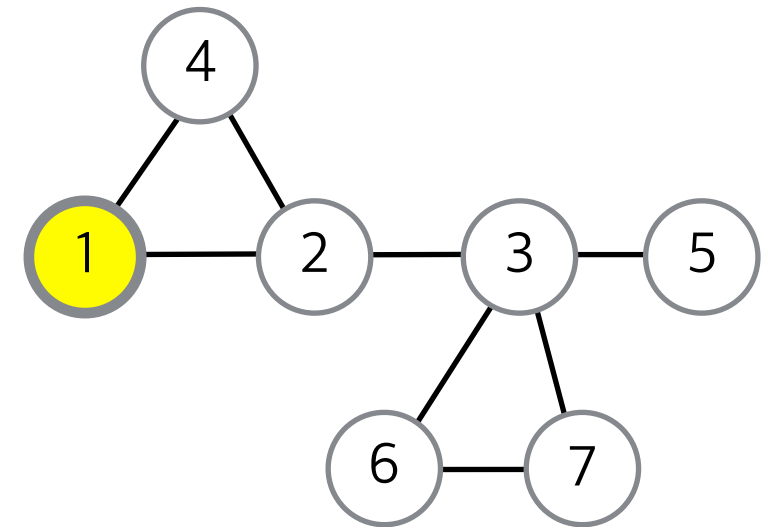


visited= [F, T, F, F, F, F, F, F]

```
graph[1] = [2, 4]  
graph[2] = [1, 3, 4]  
graph[3] = [2, 5, 6, 7]  
graph[4] = [1, 2]  
graph[5] = [3]  
graph[6] = [3, 7]  
graph[7] = [3, 6]
```


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
→   for v in graph[x] :  
       if visited[v] == False :  
           result = result +  
               DFS(graph, v, visited)  
  
    return result  
  
DFS(graph, 1, visited)  
result = [1]
```

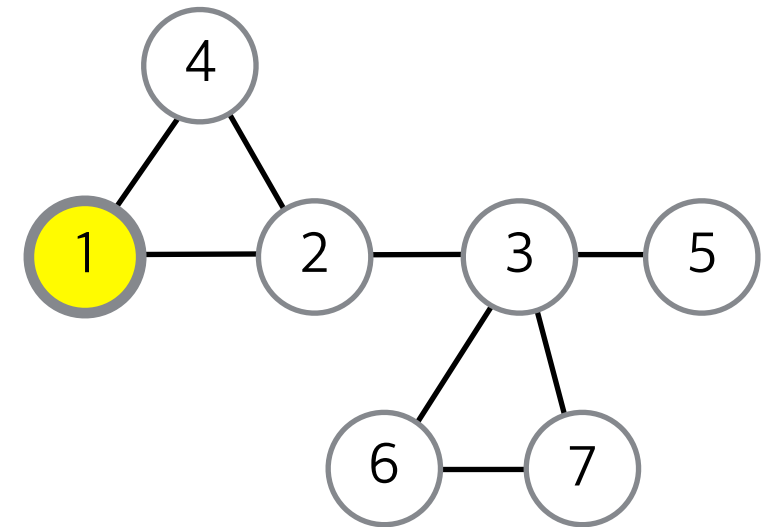


visited= [F, T, F, F, F, F, F, F]

```
graph[1] = [2, 4]  
graph[2] = [1, 3, 4]  
graph[3] = [2, 5, 6, 7]  
graph[4] = [1, 2]  
graph[5] = [3]  
graph[6] = [3, 7]  
graph[7] = [3, 6]
```

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result  
  
DFS(graph, 1, visited)  
result = [1], v = 2
```



visited= [F, T, F, F, F, F, F, F]

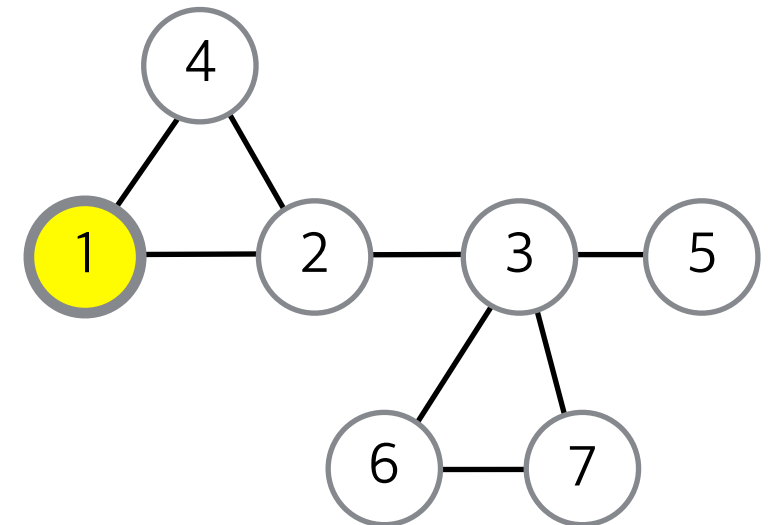
```
graph[1] = [2, 4]  
graph[2] = [1, 3, 4]  
graph[3] = [2, 5, 6, 7]  
graph[4] = [1, 2]  
graph[5] = [3]  
graph[6] = [3, 7]  
graph[7] = [3, 6]
```

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

→

```
return result  
DFS(graph, 1, visited)  
result = [1], v = 2
```



visited= [F, T, F, F, F, F, F, F]

```
graph[1] = [2, 4]  
graph[2] = [1, 3, 4]  
graph[3] = [2, 5, 6, 7]  
graph[4] = [1, 2]  
graph[5] = [3]  
graph[6] = [3, 7]  
graph[7] = [3, 6]
```

깊이 우선 탐색 (DFS) 구현

```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

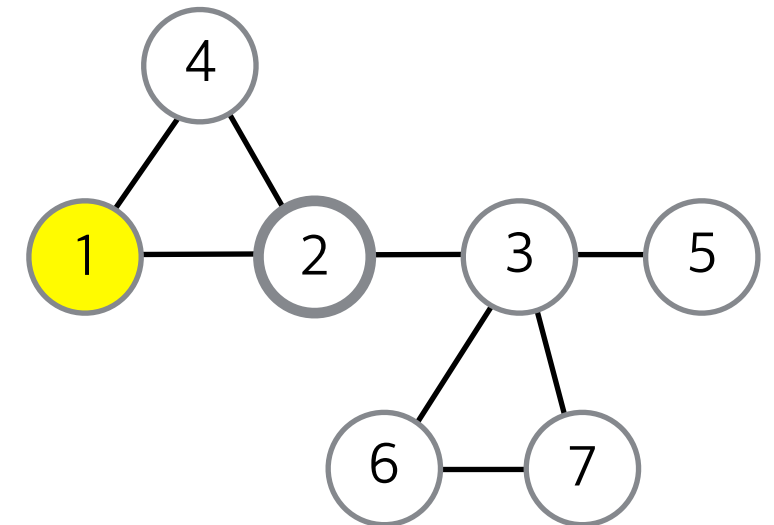
```
    return result
```

```
DFS(graph, 1, visited)
```

```
    result = [1], v = 2
```

```
DFS(graph, 2, visited)
```

```
    result = null
```



```
visited= [F, T, F, F, F, F, F, F]
```

```
graph[1] = [2, 4]
```

```
graph[2] = [1, 3, 4]
```

```
graph[3] = [2, 5, 6, 7]
```

```
graph[4] = [1, 2]
```

```
graph[5] = [3]
```

```
graph[6] = [3, 7]
```

```
graph[7] = [3, 6]
```

깊이 우선 탐색 (DFS) 구현

```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

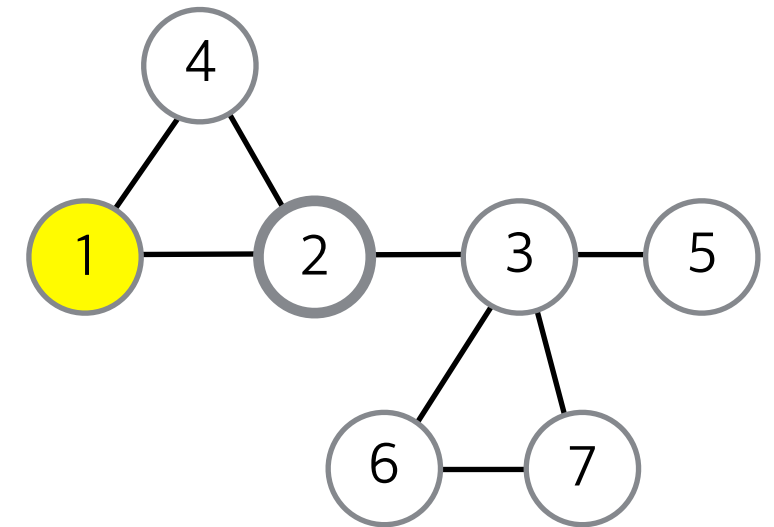
```
    return result
```

```
DFS(graph, 1, visited)
```

```
    result = [1], v = 2
```

```
DFS(graph, 2, visited)
```

```
    result = null
```



```
visited= [F, T, F, F, F, F, F, F]
```

```
graph[1] = [2, 4]
```

```
graph[2] = [1, 3, 4]
```

```
graph[3] = [2, 5, 6, 7]
```

```
graph[4] = [1, 2]
```

```
graph[5] = [3]
```

```
graph[6] = [3, 7]
```

```
graph[7] = [3, 6]
```

깊이 우선 탐색 (DFS) 구현

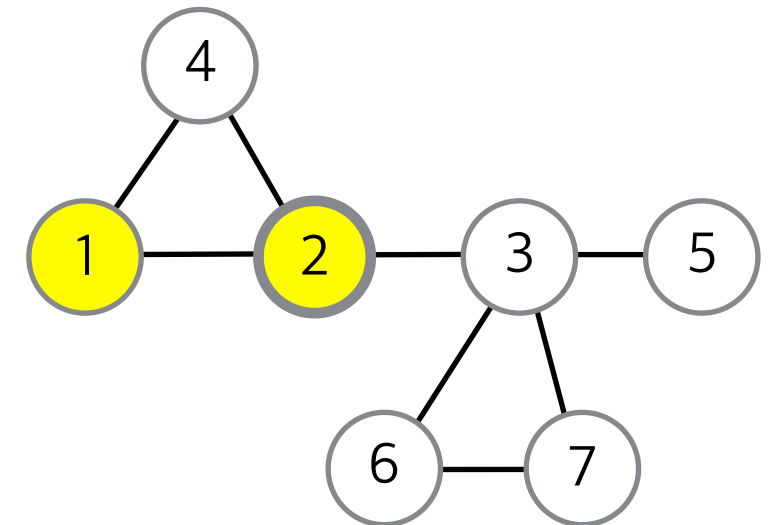
```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = null



visited= [F, T, **T**, F, F, F, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

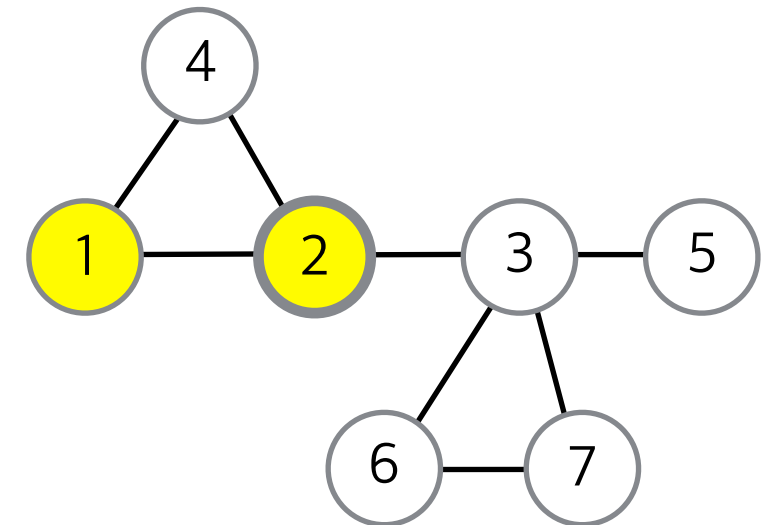
```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2]



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

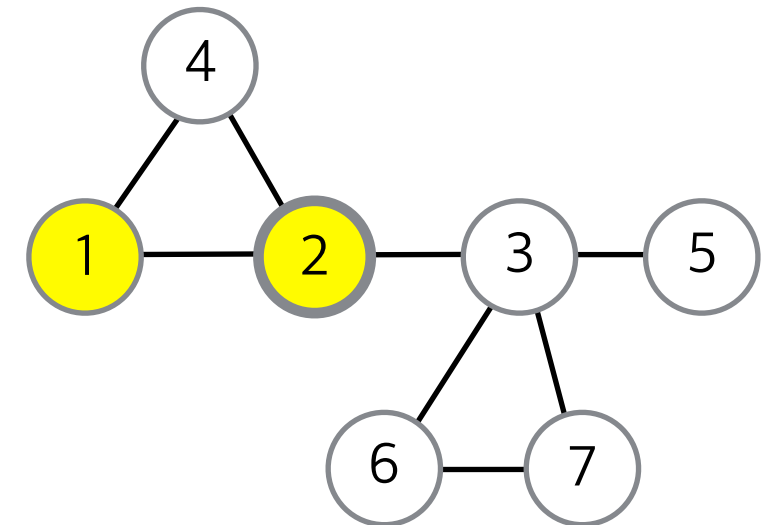
```
    return result
```

```
DFS(graph, 1, visited)
```

```
    result = [1], v = 2
```

```
DFS(graph, 2, visited)
```

```
    result = [2], v = 3
```



```
visited= [F, T, T, F, F, F, F, F]
```

```
graph[1] = [2, 4]
```

```
graph[2] = [1, 3, 4]
```

```
graph[3] = [2, 5, 6, 7]
```

```
graph[4] = [1, 2]
```

```
graph[5] = [3]
```

```
graph[6] = [3, 7]
```

```
graph[7] = [3, 6]
```


깊이 우선 탐색 (DFS) 구현

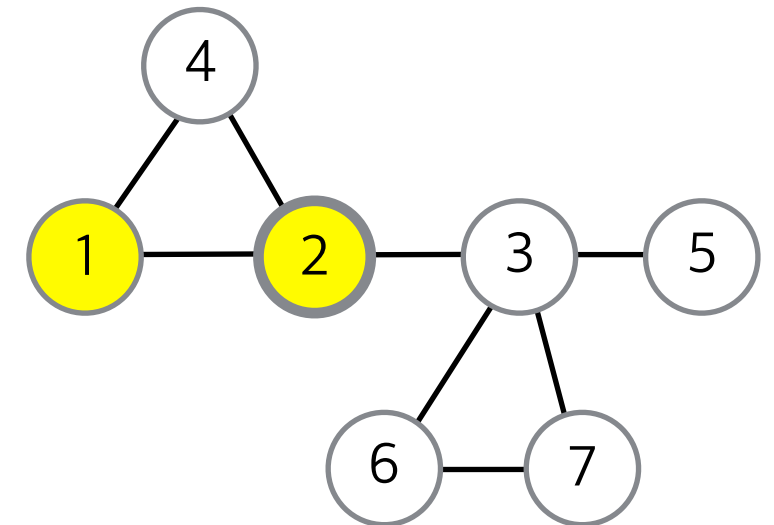
```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

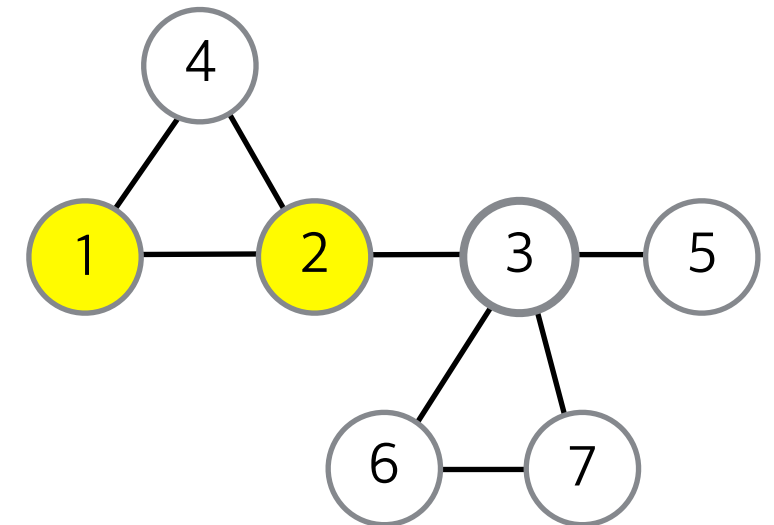
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = null



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

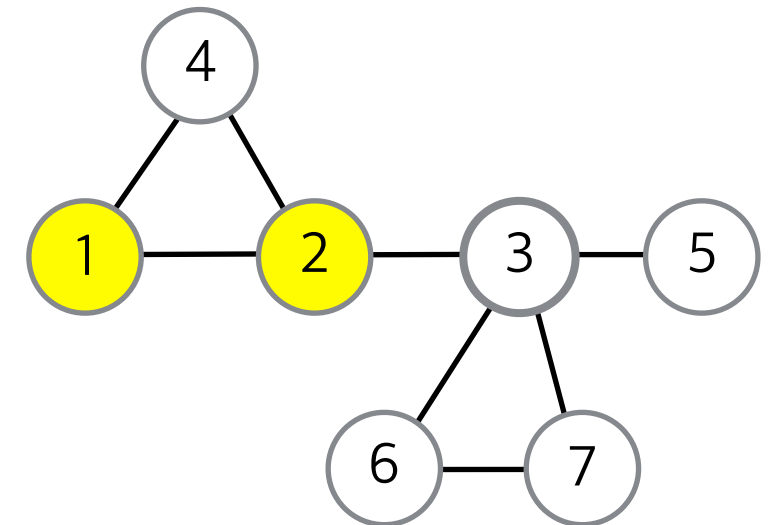
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = null



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

```
    return result
```

```
DFS(graph, 1, visited)
```

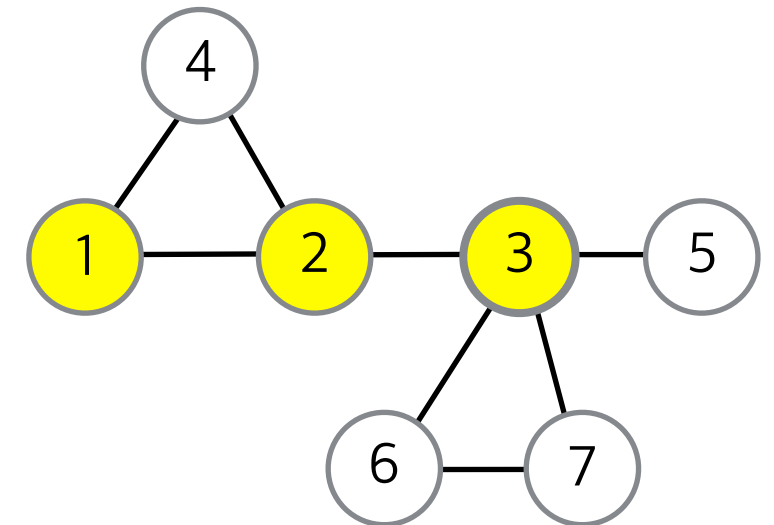
```
    result = [1], v = 2
```

```
DFS(graph, 2, visited)
```

```
    result = [2], v = 3
```

```
DFS(graph, 3, visited)
```

```
    result = null
```



```
visited= [F, T, T, T, F, F, F, F]
```

```
graph[1] = [2, 4]
```

```
graph[2] = [1, 3, 4]
```

```
graph[3] = [2, 5, 6, 7]
```

```
graph[4] = [1, 2]
```

```
graph[5] = [3]
```

```
graph[6] = [3, 7]
```

```
graph[7] = [3, 6]
```

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
→   for v in graph[x] :  
       if visited[v] == False :  
           result = result +  
               DFS(graph, v, visited)
```

```
    return result
```

DFS(graph, 1, visited)

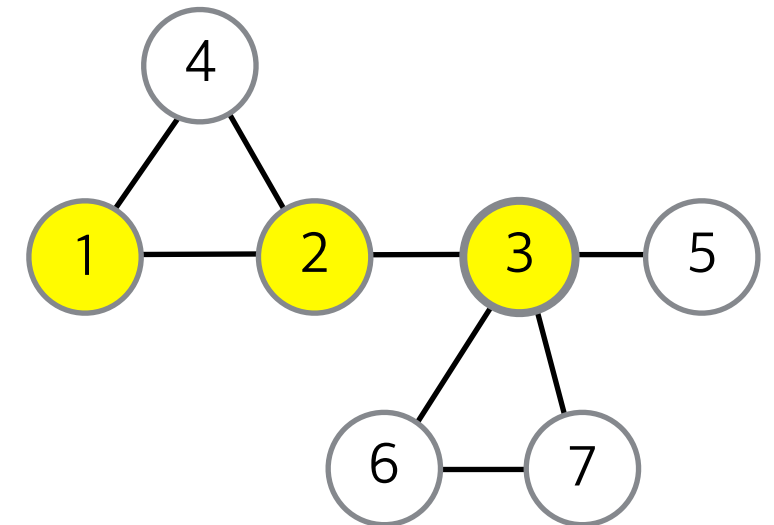
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3]



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

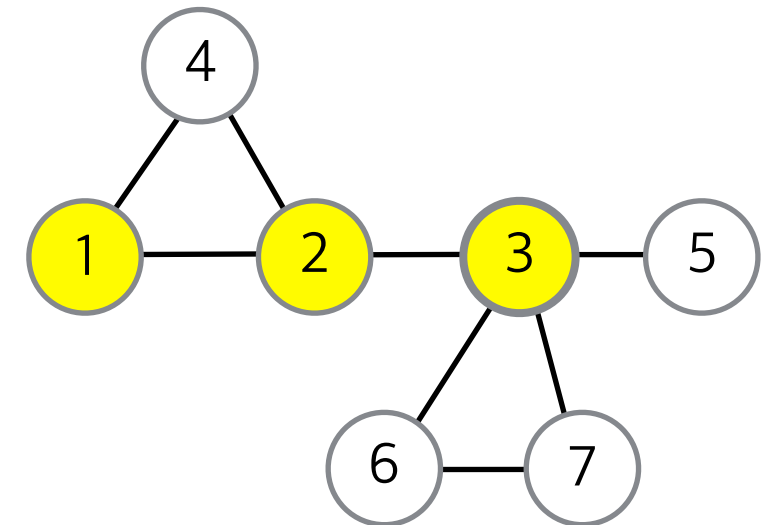
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 2



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

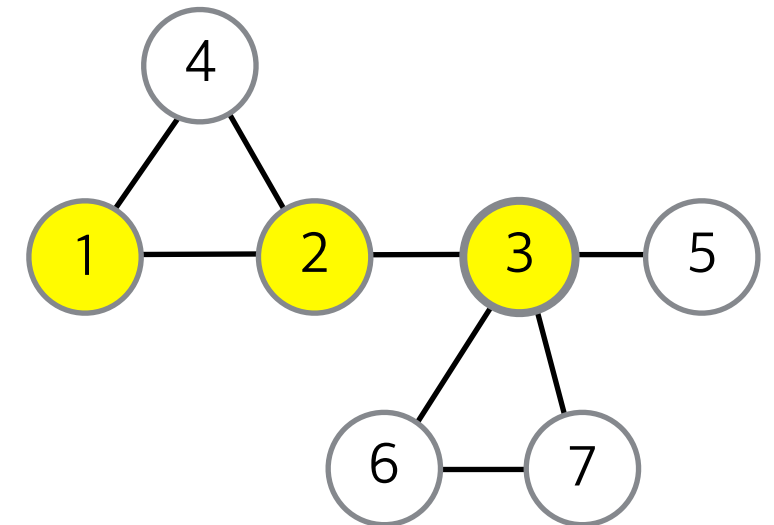
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 2



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

```
    return result
```

```
DFS(graph, 1, visited)
```

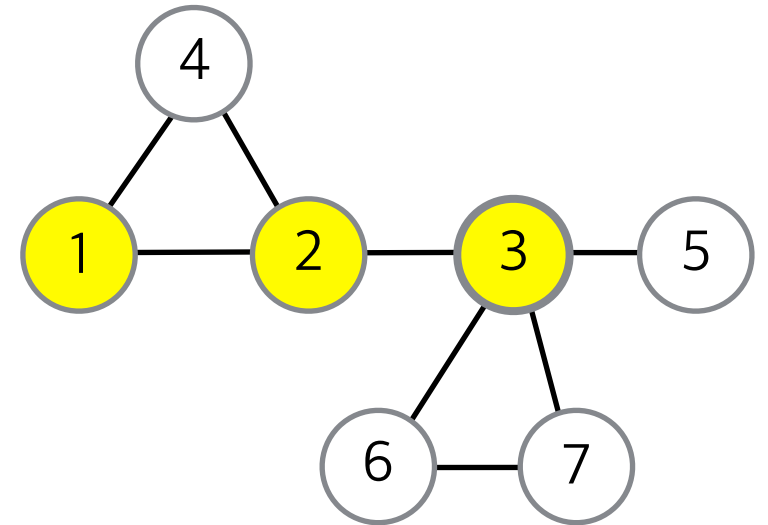
```
    result = [1], v = 2
```

```
DFS(graph, 2, visited)
```

```
    result = [2], v = 3
```

```
DFS(graph, 3, visited)
```

```
    result = [3], v = 5
```



```
visited= [F, T, T, T, F, F, F, F]
```

```
graph[1] = [2, 4]
```

```
graph[2] = [1, 3, 4]
```

```
graph[3] = [2, 5, 6, 7]
```

```
graph[4] = [1, 2]
```

```
graph[5] = [3]
```

```
graph[6] = [3, 7]
```

```
graph[7] = [3, 6]
```


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

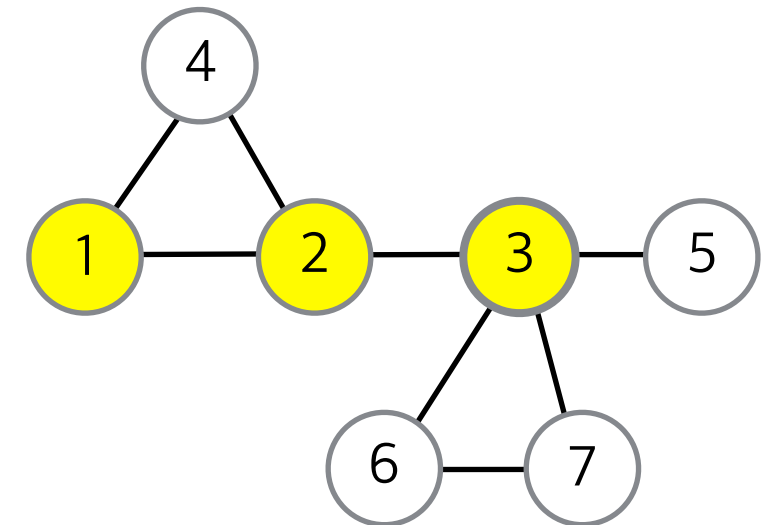
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

```
    return result
```

```
DFS(graph, 1, visited)
```

```
    result = [1], v = 2
```

```
DFS(graph, 2, visited)
```

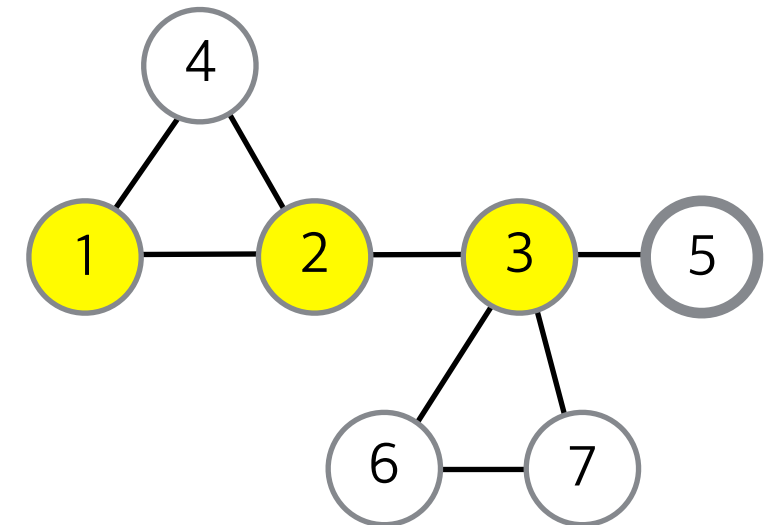
```
    result = [2], v = 3
```

```
DFS(graph, 3, visited)
```

```
    result = [3], v = 5
```

```
DFS(graph, 5, visited)
```

```
    result = null
```



```
visited= [F, T, T, T, F, F, F, F]
```

```
graph[1] = [2, 4]
```

```
graph[2] = [1, 3, 4]
```

```
graph[3] = [2, 5, 6, 7]
```

```
graph[4] = [1, 2]
```

```
graph[5] = [3]
```

```
graph[6] = [3, 7]
```

```
graph[7] = [3, 6]
```

깊이 우선 탐색 (DFS) 구현

```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

```
    return result
```

```
DFS(graph, 1, visited)
```

```
    result = [1], v = 2
```

```
DFS(graph, 2, visited)
```

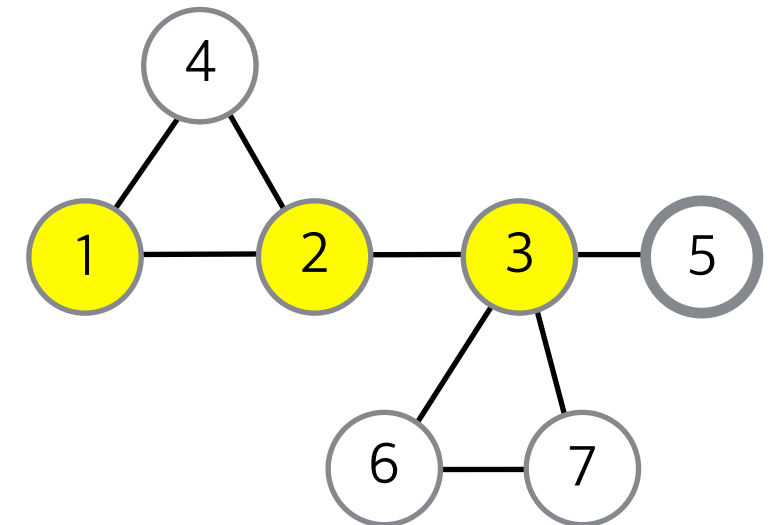
```
    result = [2], v = 3
```

```
DFS(graph, 3, visited)
```

```
    result = [3], v = 5
```

```
DFS(graph, 5, visited)
```

```
    result = null
```



```
visited= [F, T, T, T, F, F, F, F]
```

```
graph[1] = [2, 4]
```

```
graph[2] = [1, 3, 4]
```

```
graph[3] = [2, 5, 6, 7]
```

```
graph[4] = [1, 2]
```

```
graph[5] = [3]
```

```
graph[6] = [3, 7]
```

```
graph[7] = [3, 6]
```

깊이 우선 탐색 (DFS) 구현

```
→ def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

```
    return result
```

```
DFS(graph, 1, visited)
```

```
    result = [1], v = 2
```

```
DFS(graph, 2, visited)
```

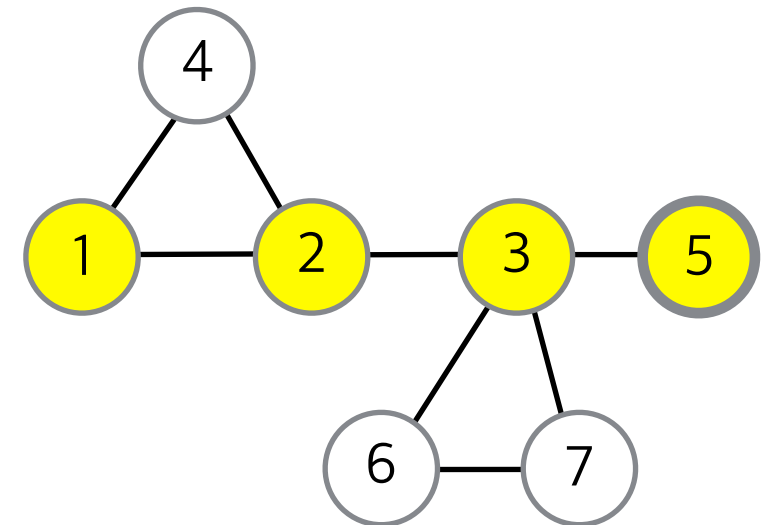
```
    result = [2], v = 3
```

```
DFS(graph, 3, visited)
```

```
    result = [3], v = 5
```

```
DFS(graph, 5, visited)
```

```
    result = null
```



```
visited= [F, T, T, T, F, T, F, F]
```

```
graph[1] = [2, 4]
```

```
graph[2] = [1, 3, 4]
```

```
graph[3] = [2, 5, 6, 7]
```

```
graph[4] = [1, 2]
```

```
graph[5] = [3]
```

```
graph[6] = [3, 7]
```

```
graph[7] = [3, 6]
```

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
→ for v in graph[x] :  
    if visited[v] == False :  
        result = result +  
            DFS(graph, v, visited)
```

```
return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

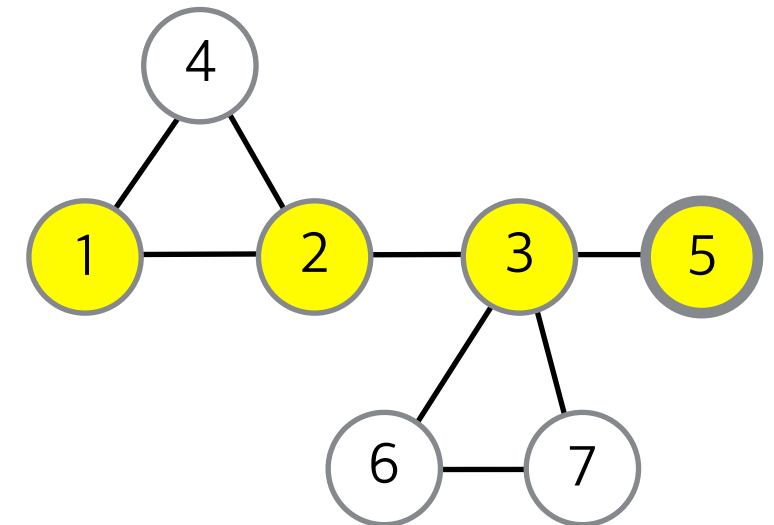
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5]



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

```
    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

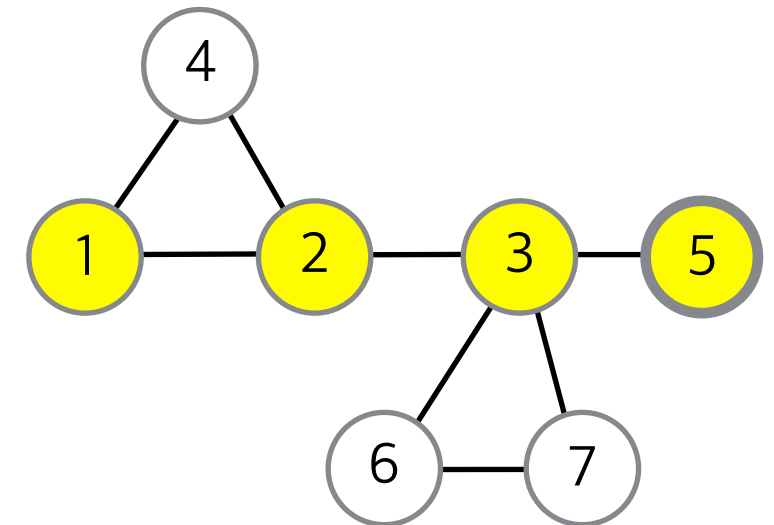
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
→ for v in graph[x] :  
    if visited[v] == False :  
        result = result +  
            DFS(graph, v, visited)
```

```
return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

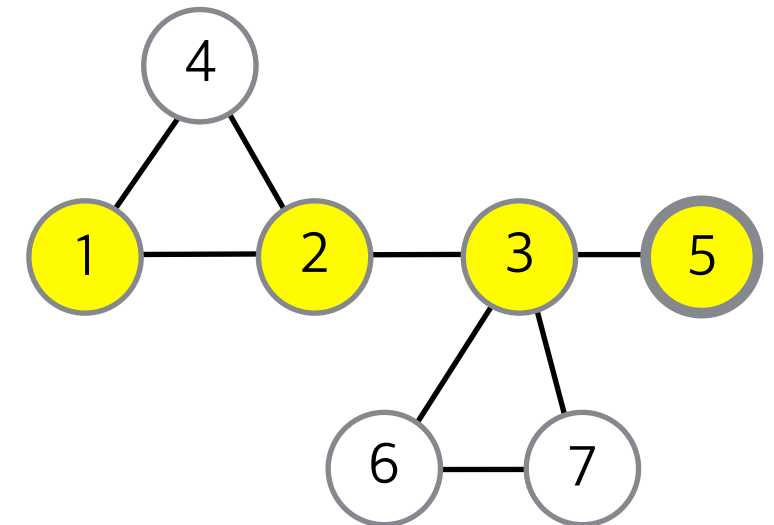
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

→ return result

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

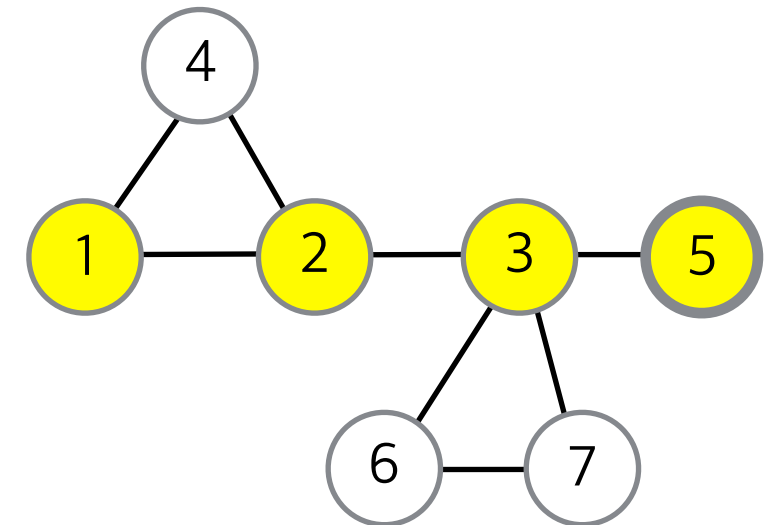
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

→ return result

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

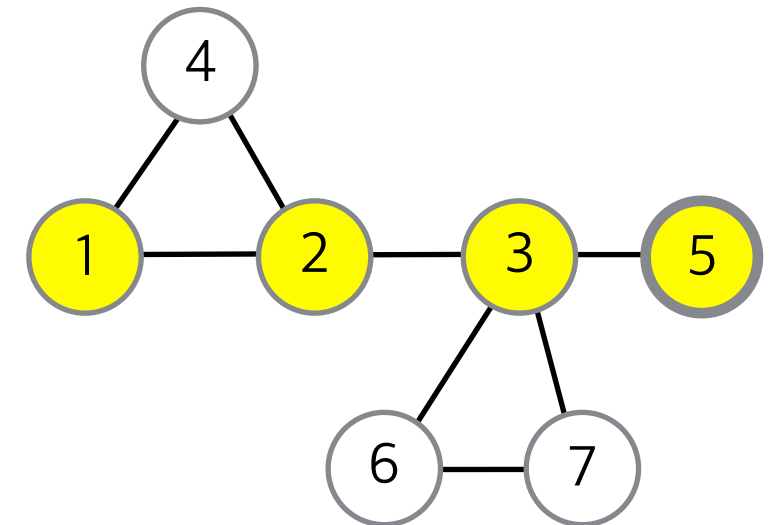
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

→ return result

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

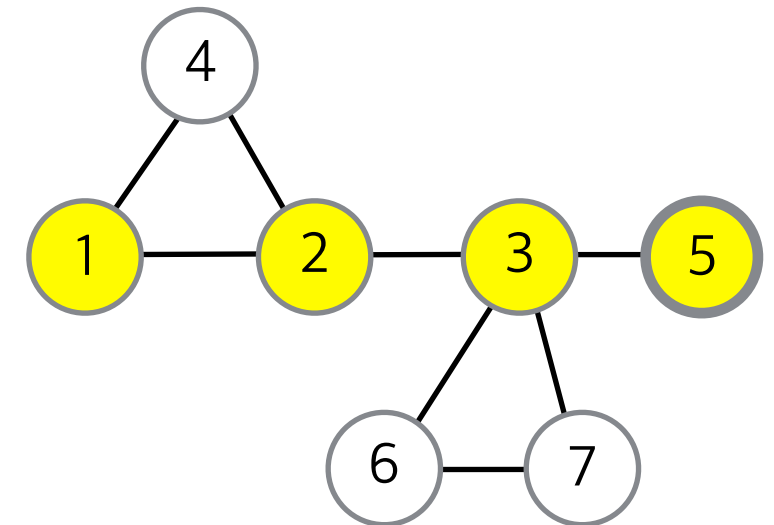
DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5], v = 3

[5]



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

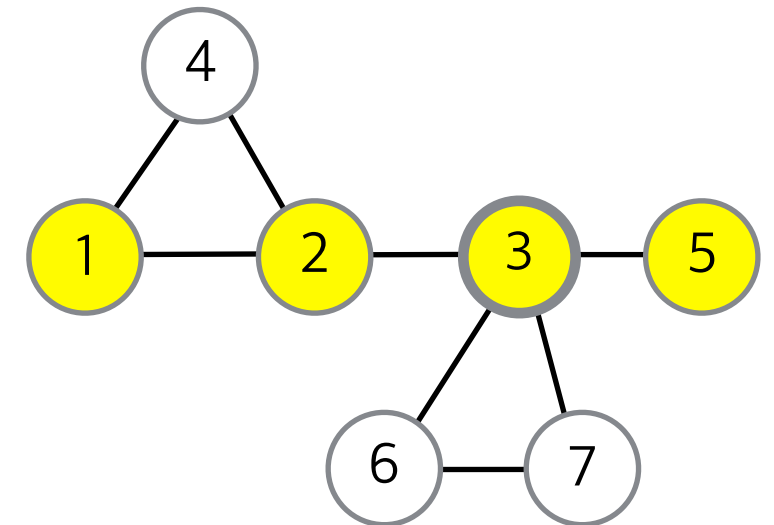
DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

[5]



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

return result

DFS(graph, 1, visited)

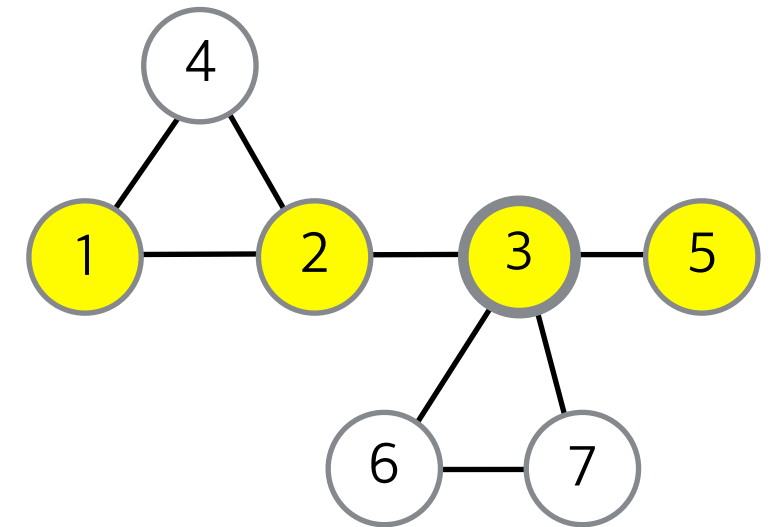
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 5



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

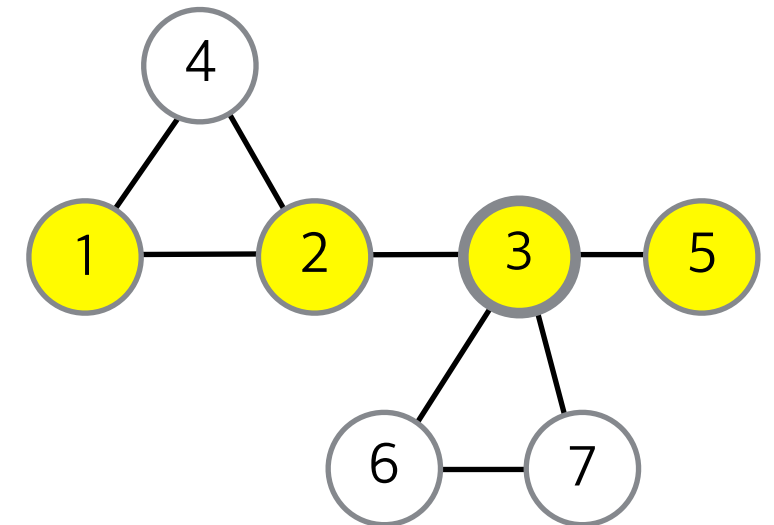
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 6



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

return result

DFS(graph, 1, visited)

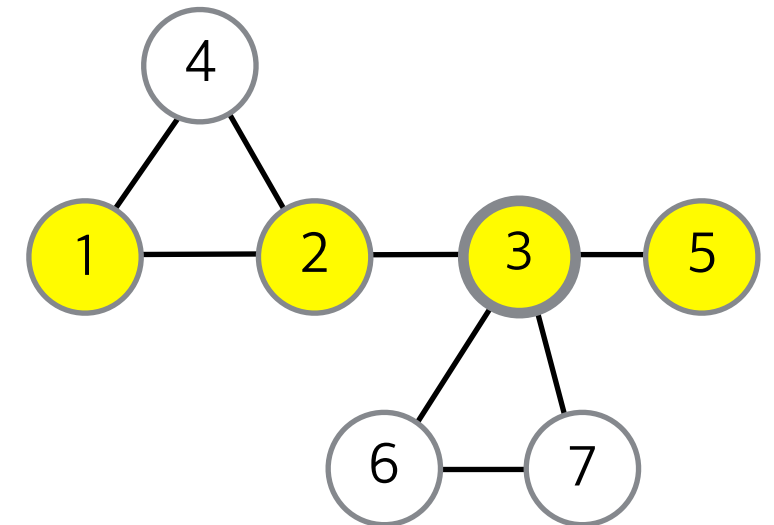
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 6



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

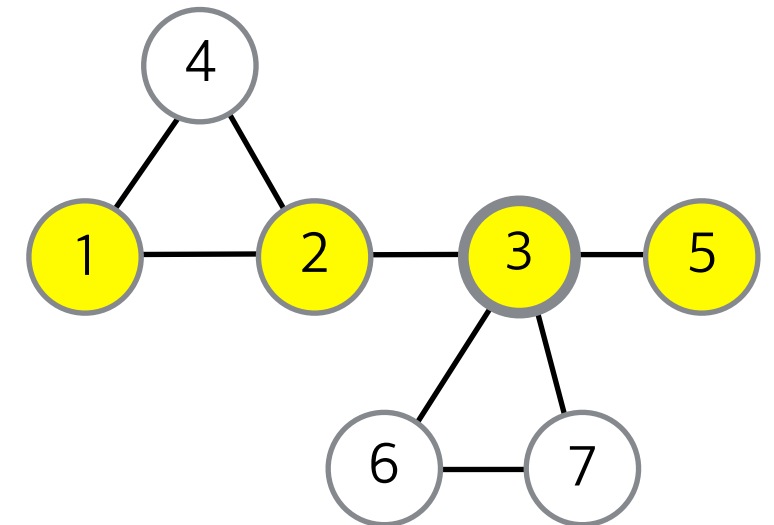
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 6



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
→  
  
    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

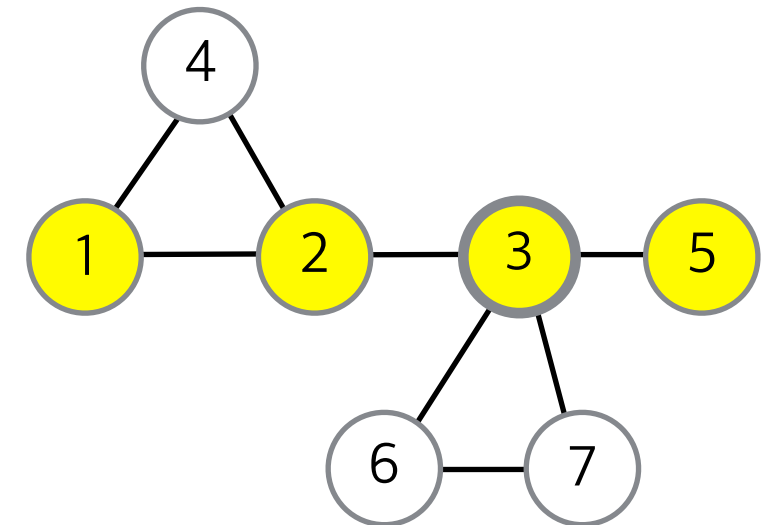
DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 6

DFS(graph, 6, visited)가 return하는 값은 ?



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
→  
  
    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

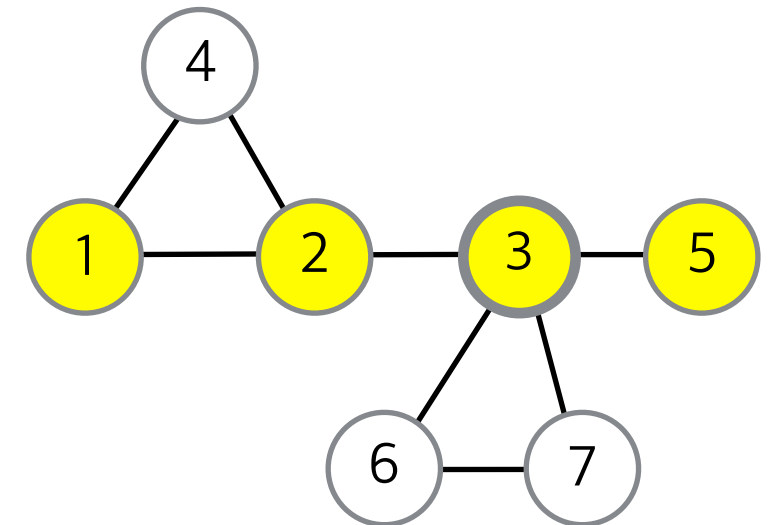
result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 6

DFS(graph, 6, visited)

[6, 7]



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

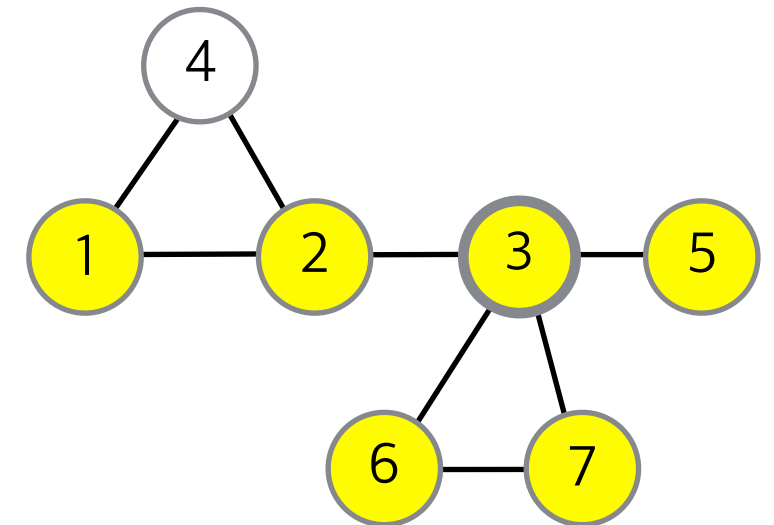
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 6



visited= [F, T, T, T, F, T, **T**, **T**]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

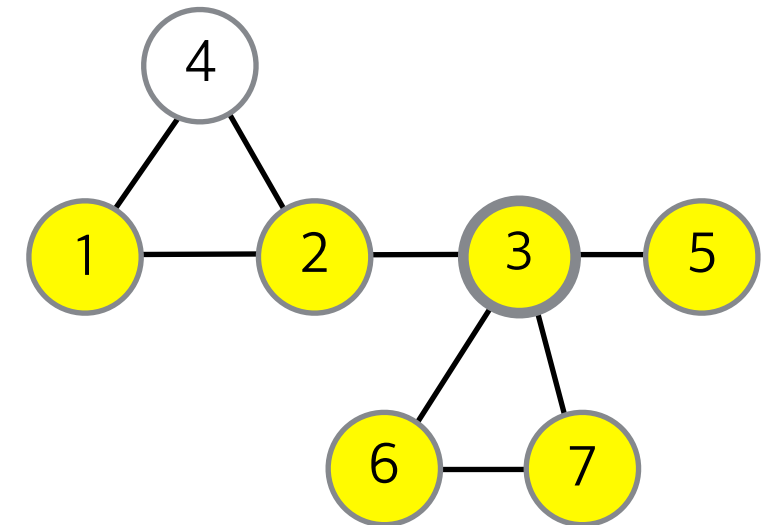
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 7



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

return result

DFS(graph, 1, visited)

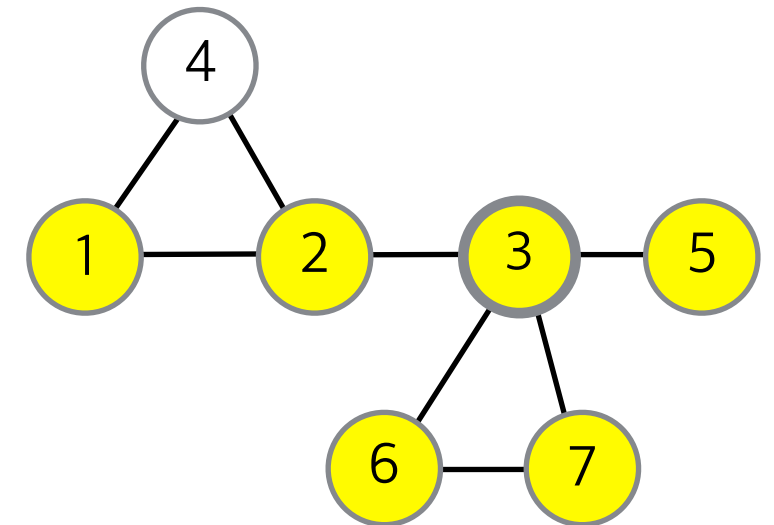
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 7



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

→ return result

DFS(graph, 1, visited)

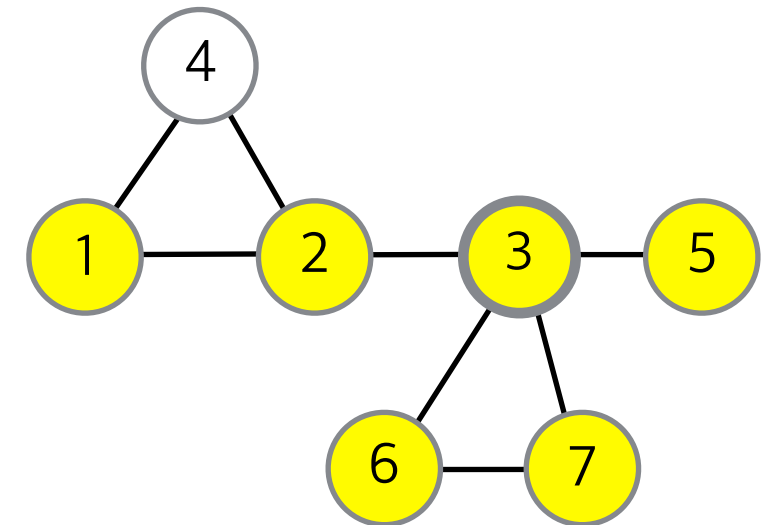
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 7



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

→ return result

DFS(graph, 1, visited)

result = [1], v = 2

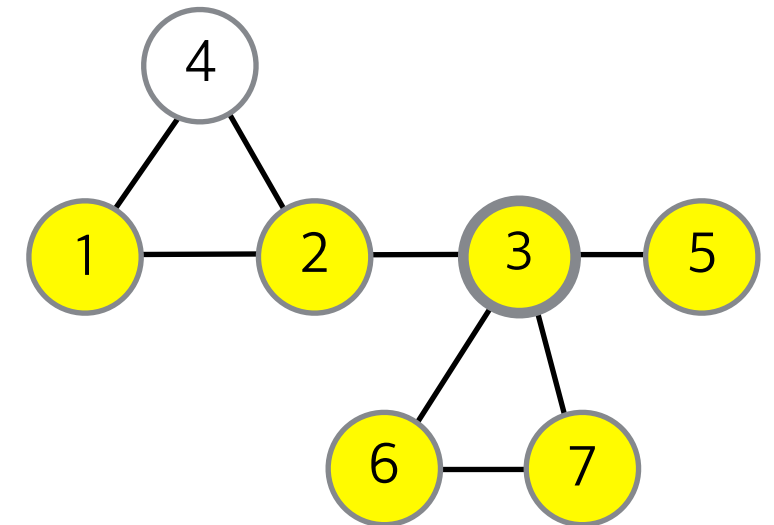
DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 7

[3, 5, 6, 7]



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

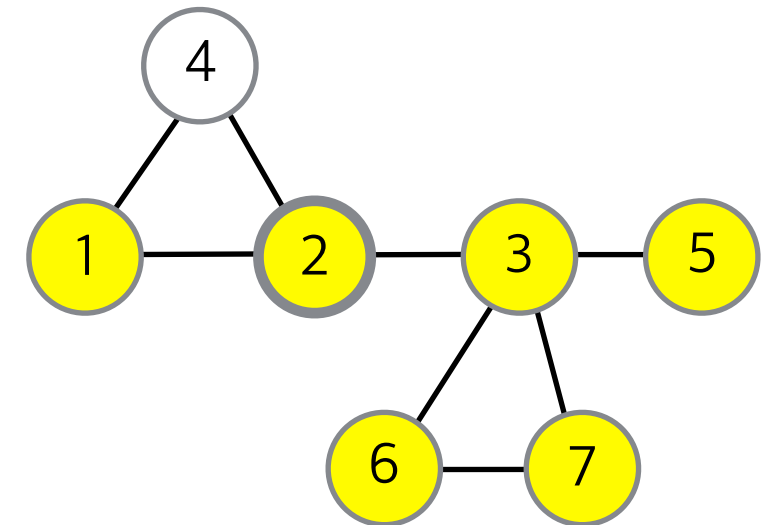
DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

[3, 5, 6, 7]



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

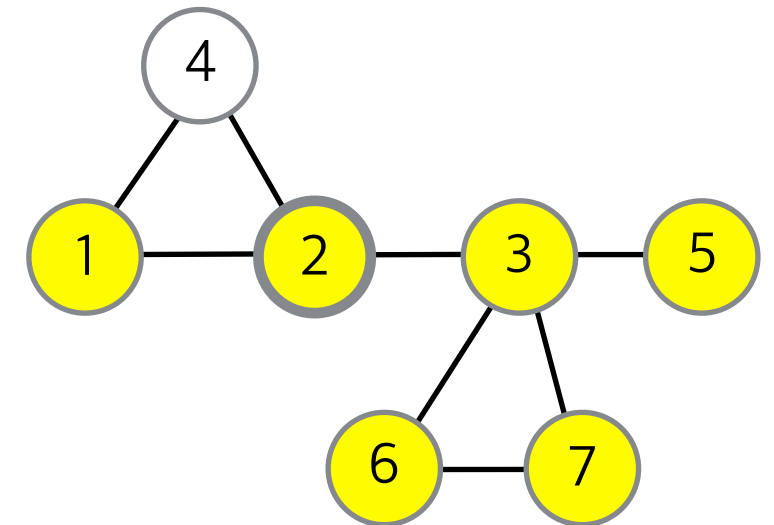
```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 3



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

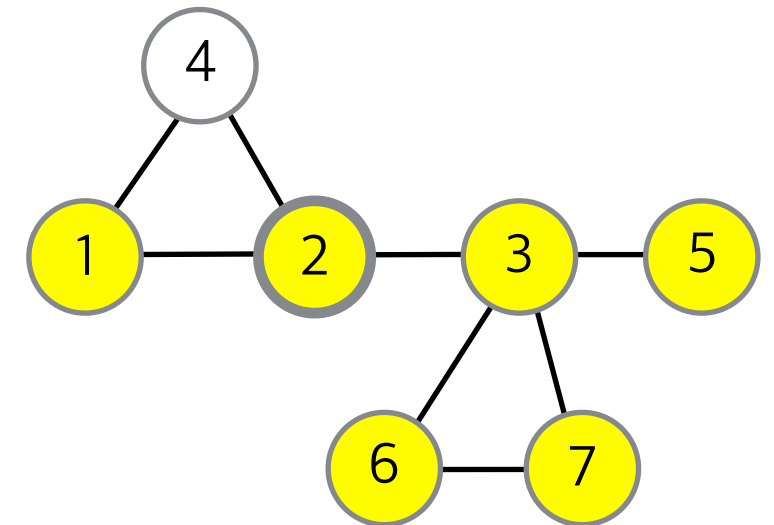
```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 4



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

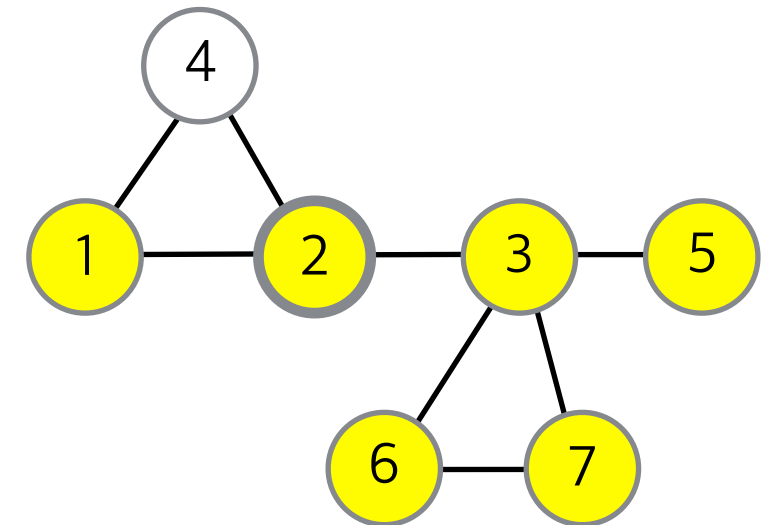
```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 4



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

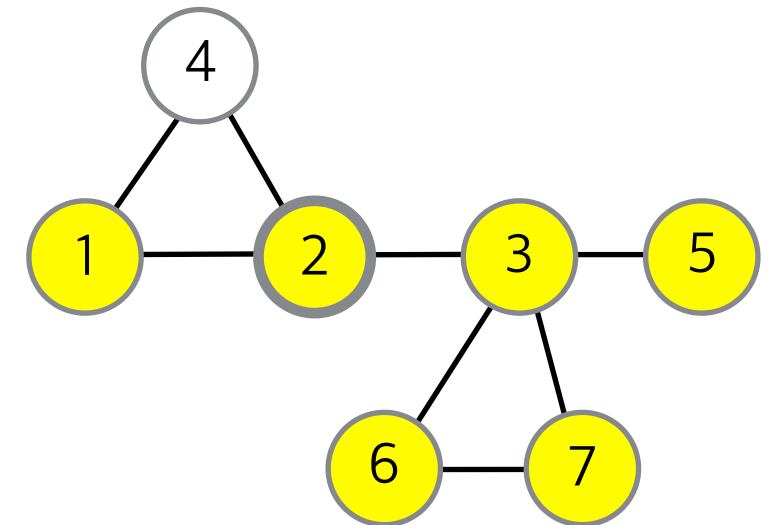
DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 4

DFS(graph, 4, visited)가 return하는 값은 ?



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

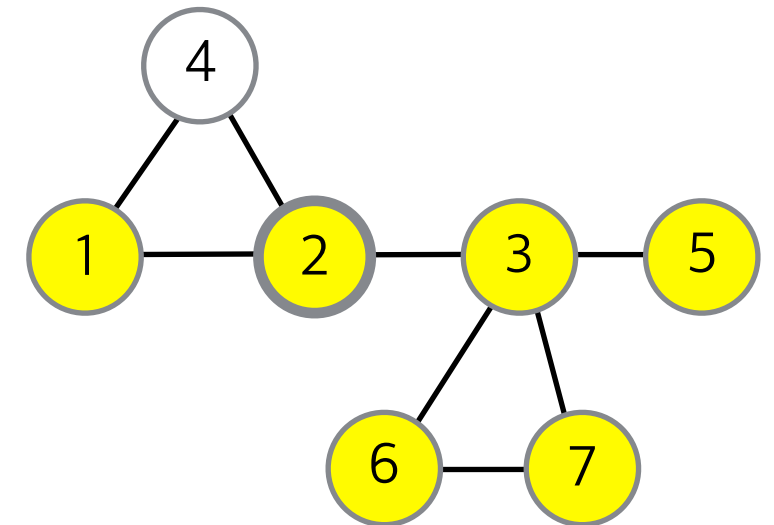
DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 4

[4]



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

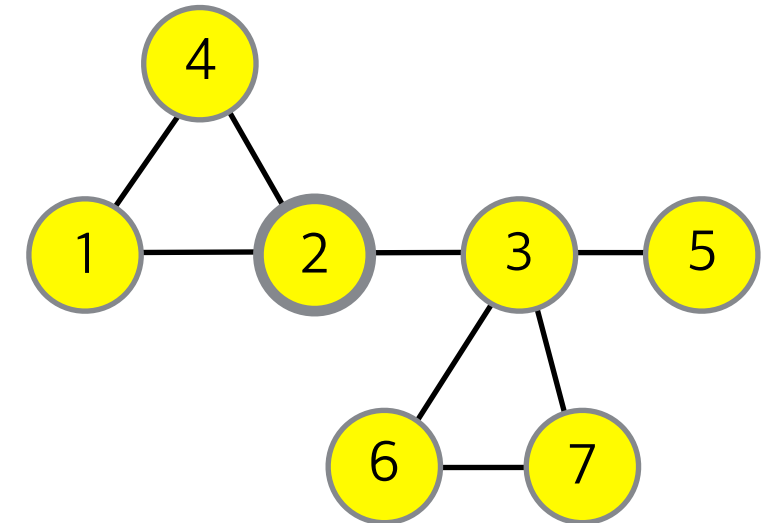
```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, **T**, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

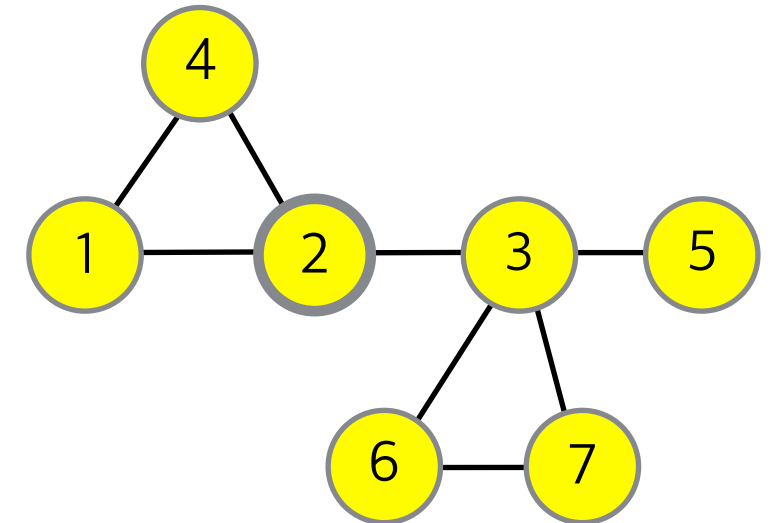
→ return result

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)
```

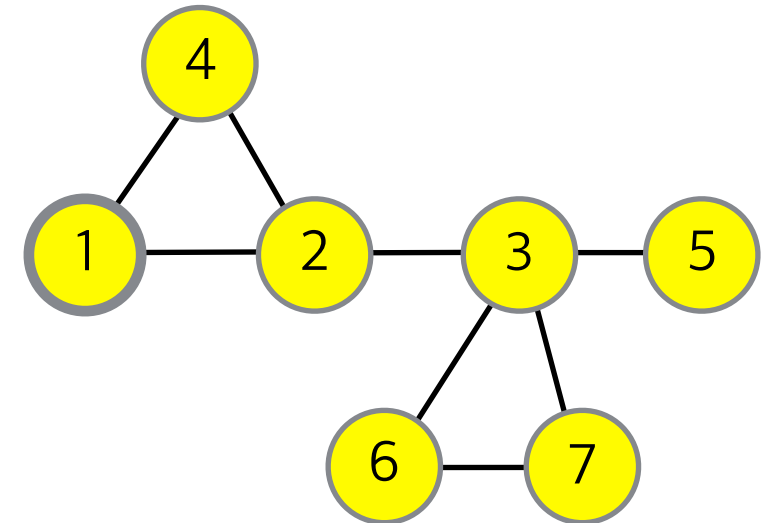
→

```
return result
```

DFS(graph, 1, visited)

result = [1], v = 2

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



visited= [F, T, T, T, T, T, T, T]

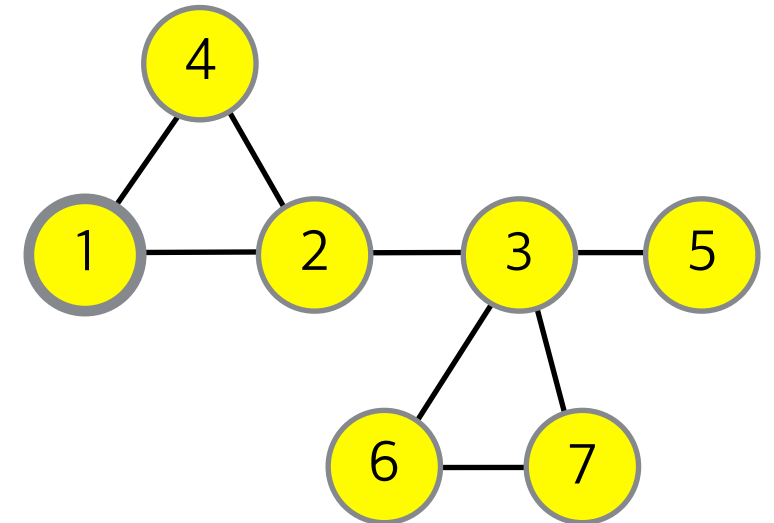
```
graph[1] = [2, 4]  
graph[2] = [1, 3, 4]  
graph[3] = [2, 5, 6, 7]  
graph[4] = [1, 2]  
graph[5] = [3]  
graph[6] = [3, 7]  
graph[7] = [3, 6]
```

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 2



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

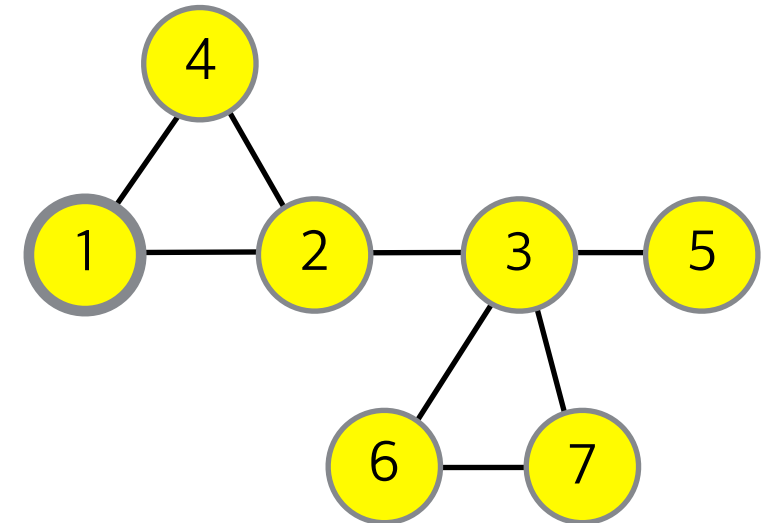
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

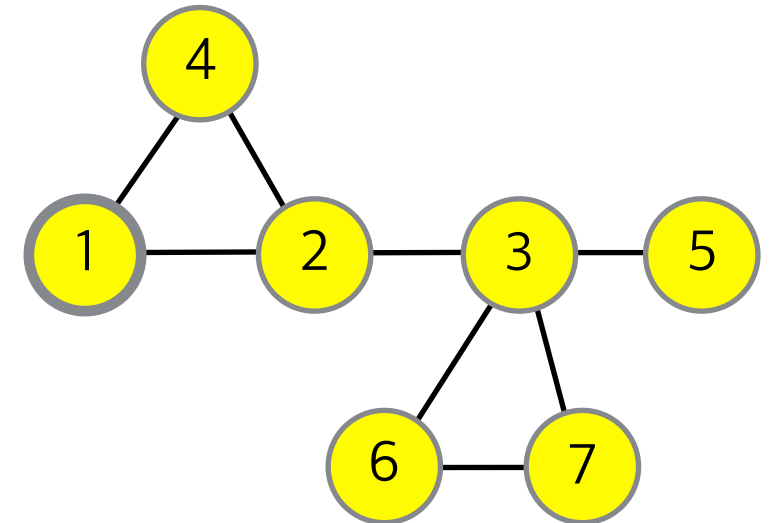
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    → for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
    return result
```

DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

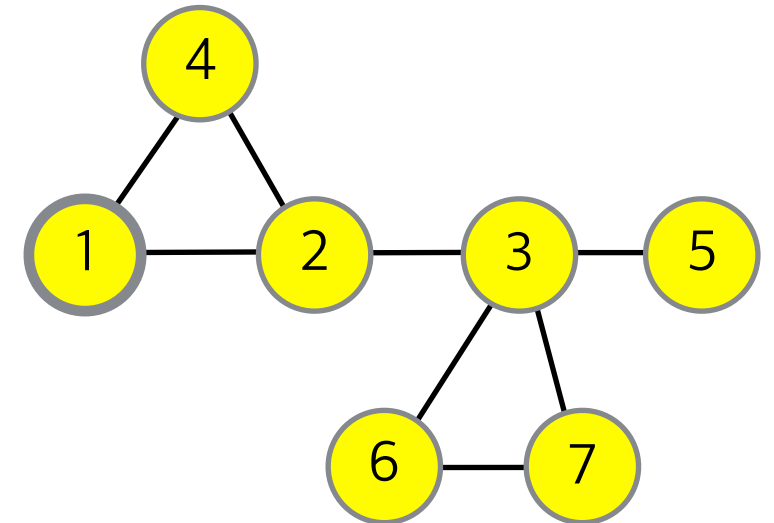
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result +  
                DFS(graph, v, visited)  
  
→ return result
```

DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

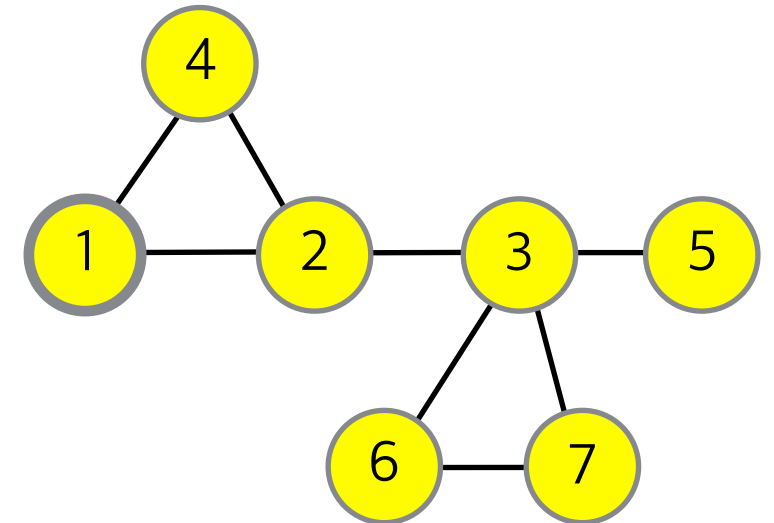
깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :  
    visited[x] = True  
    result = [x]  
  
    for v in graph[x] :  
        if visited[v] == False :  
            result = result + [1, 2, 3, 5, 6, 7, 4]  
            DFS(graph, v, visited)
```

→ return result

DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]

graph[2] = [1, 3, 4]

graph[3] = [2, 5, 6, 7]

graph[4] = [1, 2]

graph[5] = [3]

graph[6] = [3, 7]

graph[7] = [3, 6]

[예제 1] 깊이우선탐색

그래프를 깊이우선탐색한 결과를 출력하시오

시스템 입력

```
6 8
1 2
1 6
2 3
2 4
2 6
3 4
3 5
4 5
```

시스템 출력

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

[예제 1] 깊이우선탐색



`/* elice */`

DFS의 정확성증명 / 시간복잡도

정확성 증명

모든 정점을 방문한다

하나의 정점을 두번 이상 방문하지 않는다

DFS의 정확성증명 / 시간복잡도

정확성 증명

모든 정점을 방문한다

하나의 정점을 두번 이상 방문하지 않는다

시간복잡도

각 정점은 1번씩, 각 간선은 2번씩 방문함

$O(V+E)$

퀴즈 및 설문조사



감사합니다!

신현규

E-mail : hyungyu.sh@kaist.ac.kr

Kakao : yougatup

Facebook : yougatup

/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

blog.naver.com/elicer