

BFS 알고리즘

큐 자료구조

S.C.P

220113 이유경

Contents

1. DFS 복습

1-1. DFS 정의

1-2. DFS 동작 과정

2. 큐 자료구조

2-1. 큐 정의

2-2. 큐 동작 예시

2-3. 큐 구현 예제

3. BFS 알고리즘

3-1. BFS 정의

3-2. BFS 동작 예시

3-3. BFS 구현 예제

1-1. DFS 정의

DFS (깊이 우선 탐색)

- 루트 노드에서 시작해서 다음 분기로 넘어가기 전에 해당 분기를 완벽하게 탐색하는 방법
- 스택 자료구조(혹은 재귀 함수)를 이용
- 모든 정점을 방문하는 단순하고 고전적인 방법

동작 과정

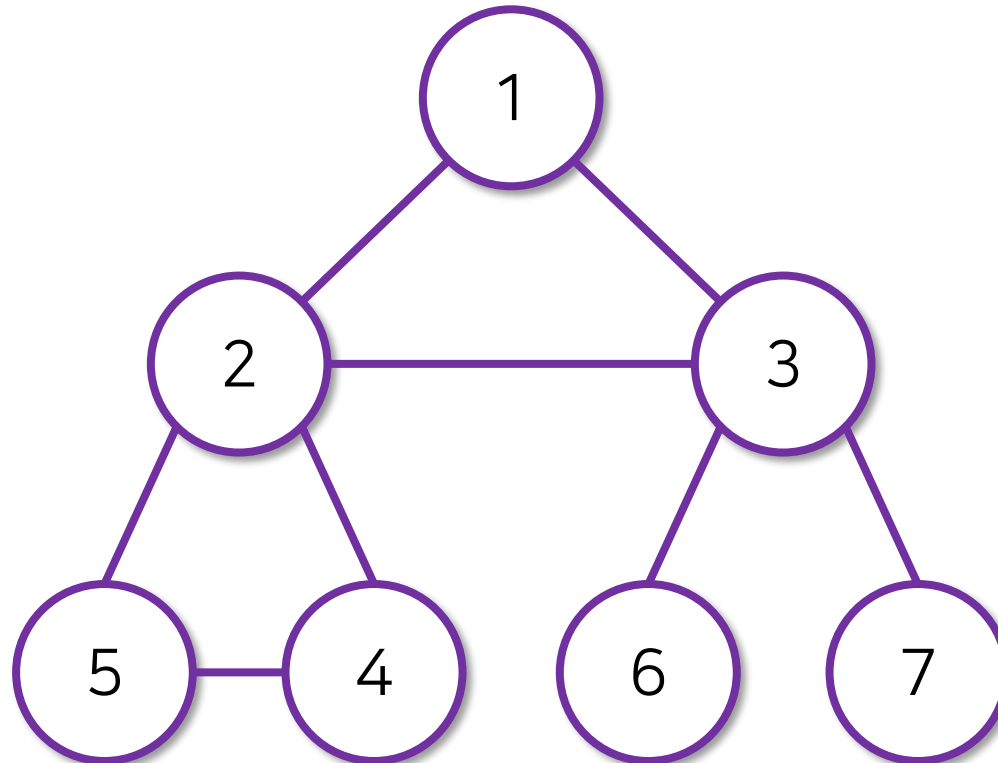
1. 스택의 최상단 노드 확인
2. 현재 정점과 인접한 간선들을 하나씩 검사
3. 아직 방문하지 않은 정점으로 향하는 간선이 있다면 그 간선을 무조건 방문
4. 더 이상 방문할 곳이 없다면, 마지막에 방문했던 간선을 따라 뒤로 돌아감

1-2. DFS 동작 과정



방문 기준

번호가 낮은 인접 노드부터

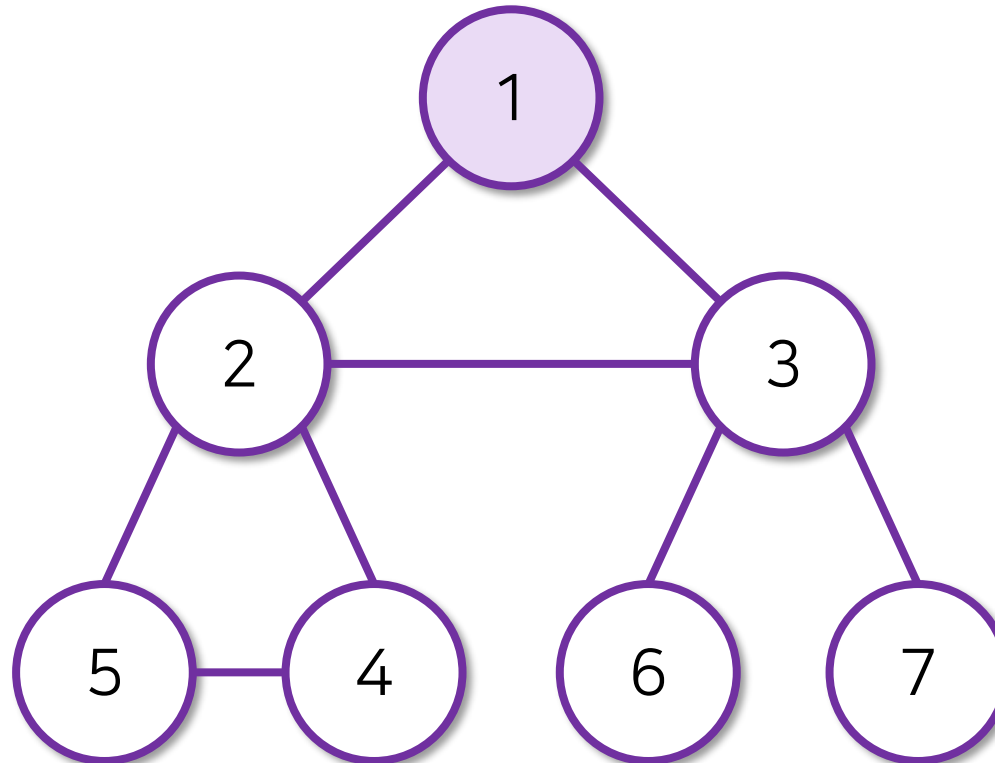


1-2. DFS 동작 과정



방문 기준

번호가 낮은 인접 노드부터

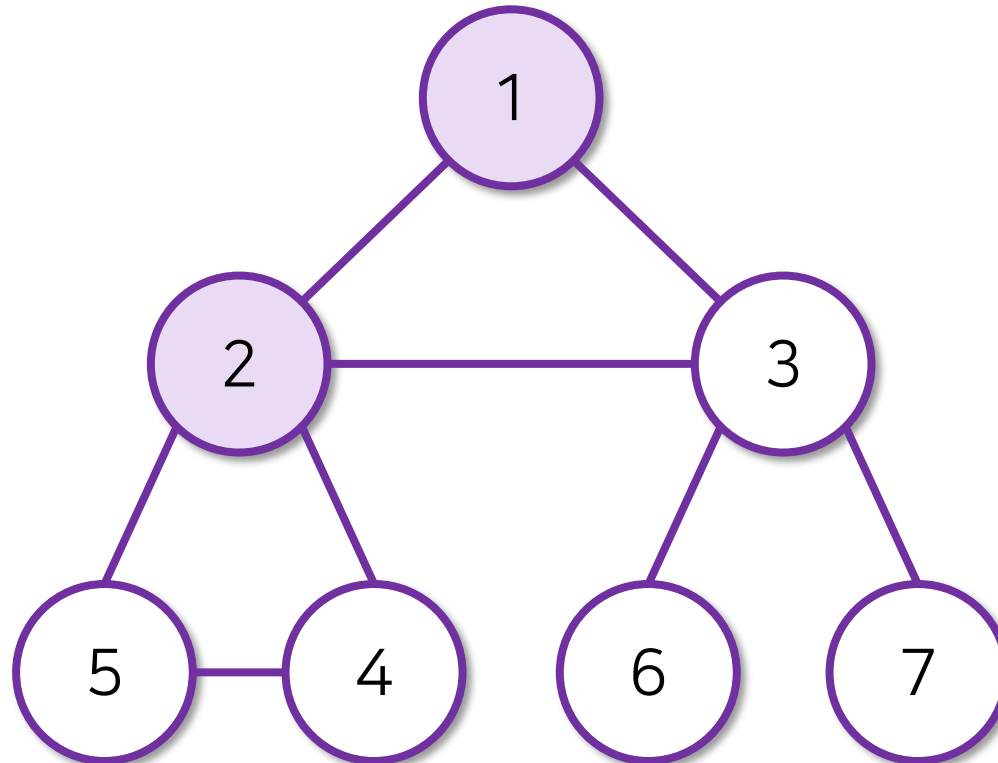


1-2. DFS 동작 과정

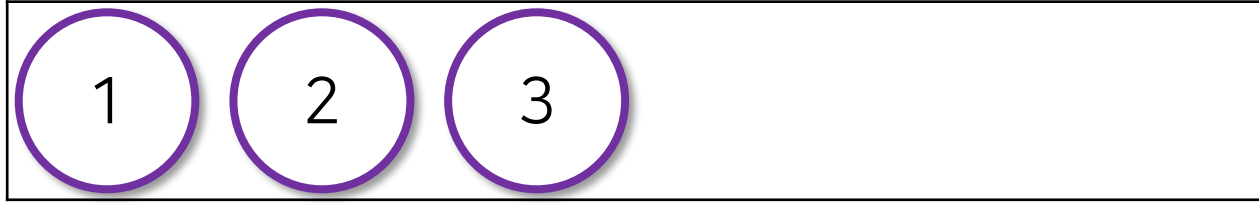


방문 기준

번호가 낮은 인접 노드부터

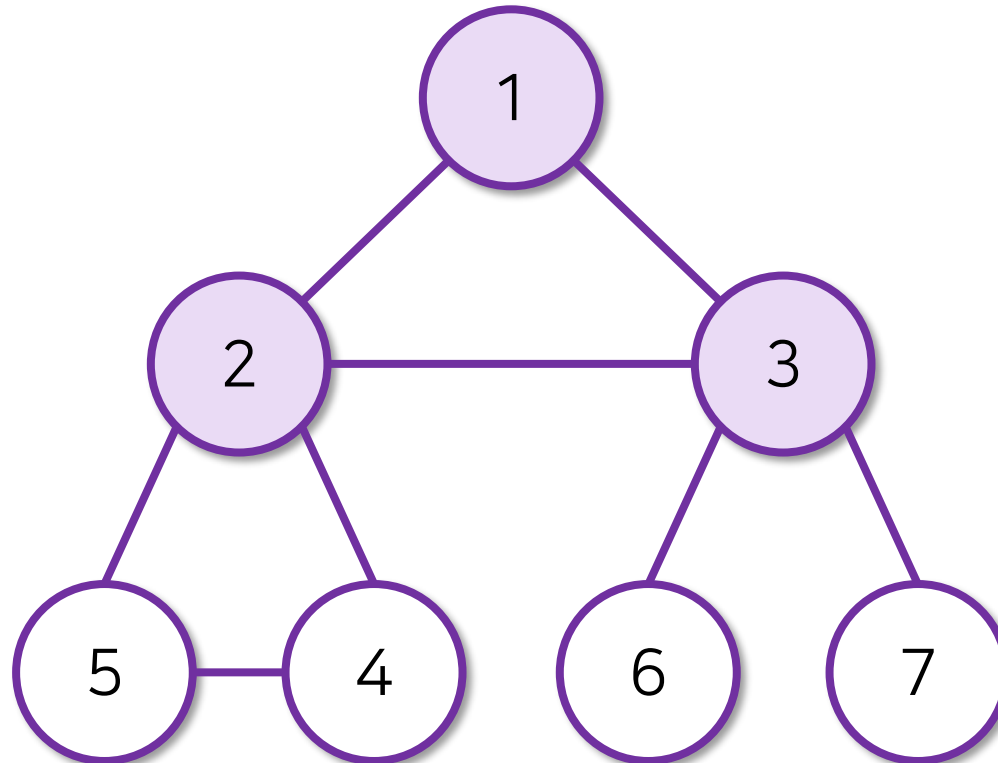


1-2. DFS 동작 과정

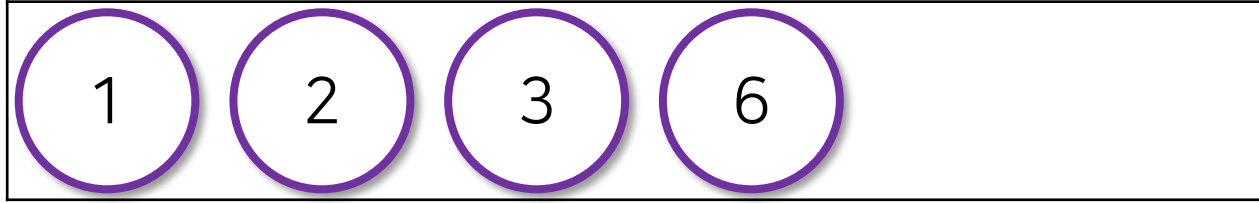


방문 기준

번호가 낮은 인접 노드부터

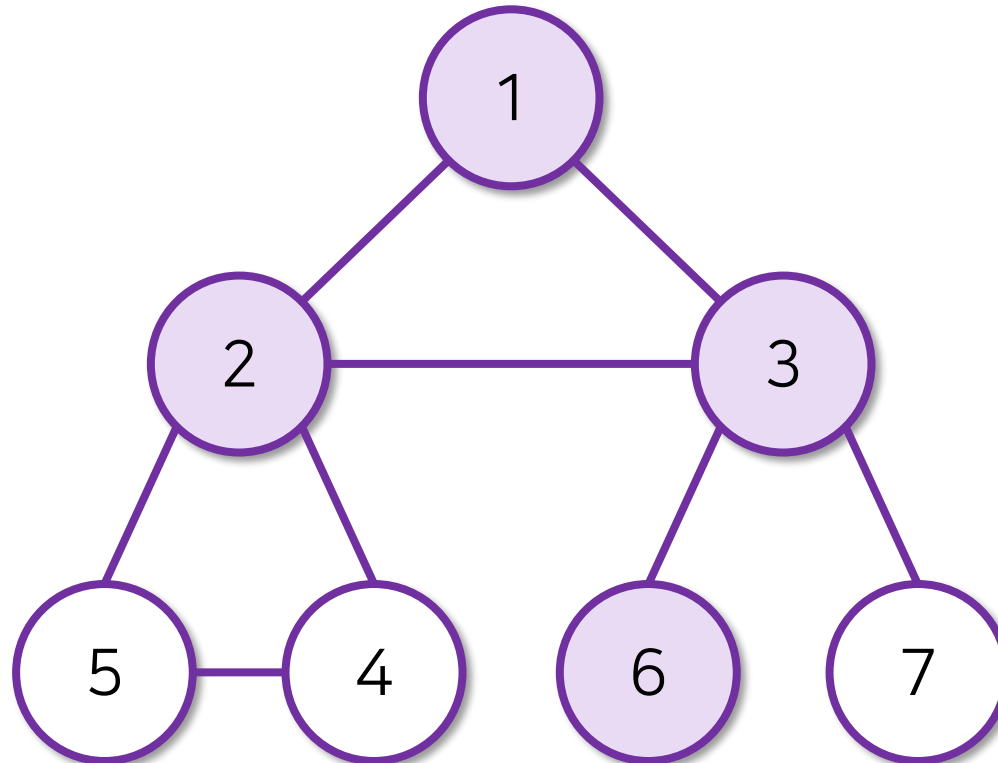


1-2. DFS 동작 과정

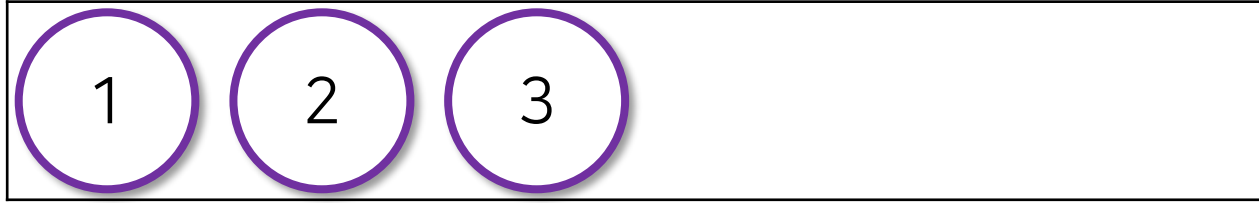


방문 기준

번호가 낮은 인접 노드부터

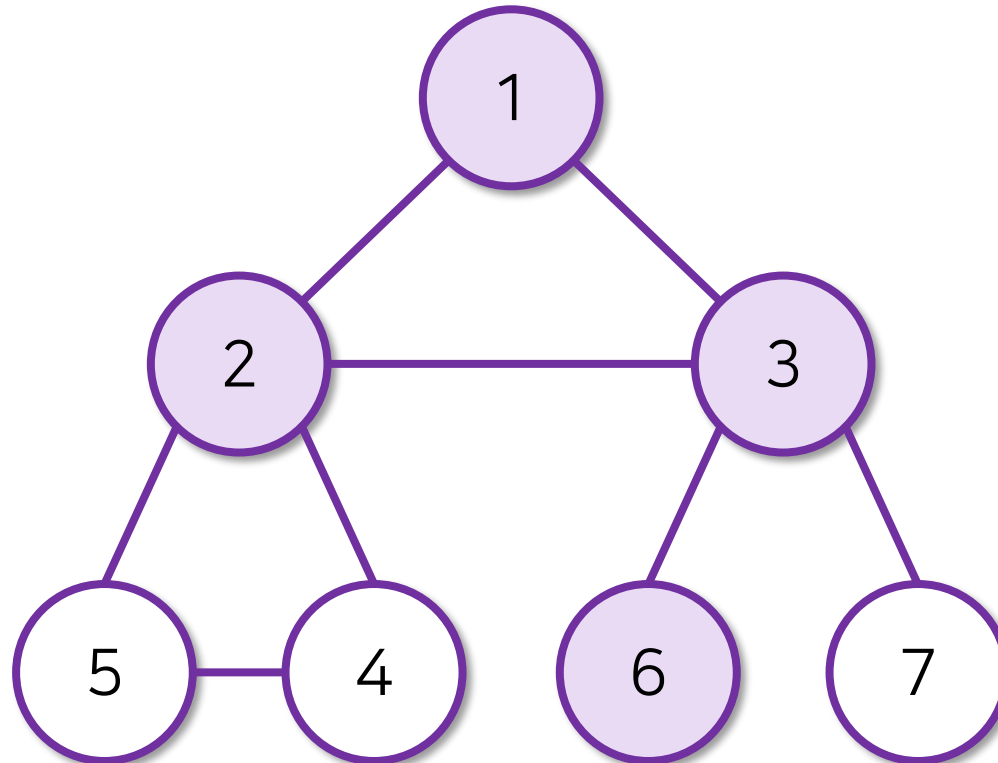


1-2. DFS 동작 과정

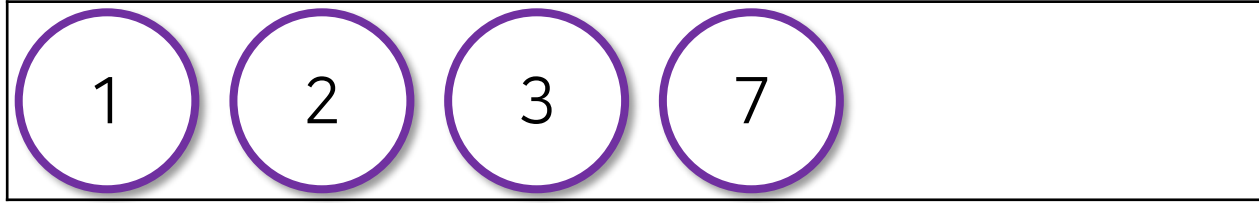


방문 기준

번호가 낮은 인접 노드부터

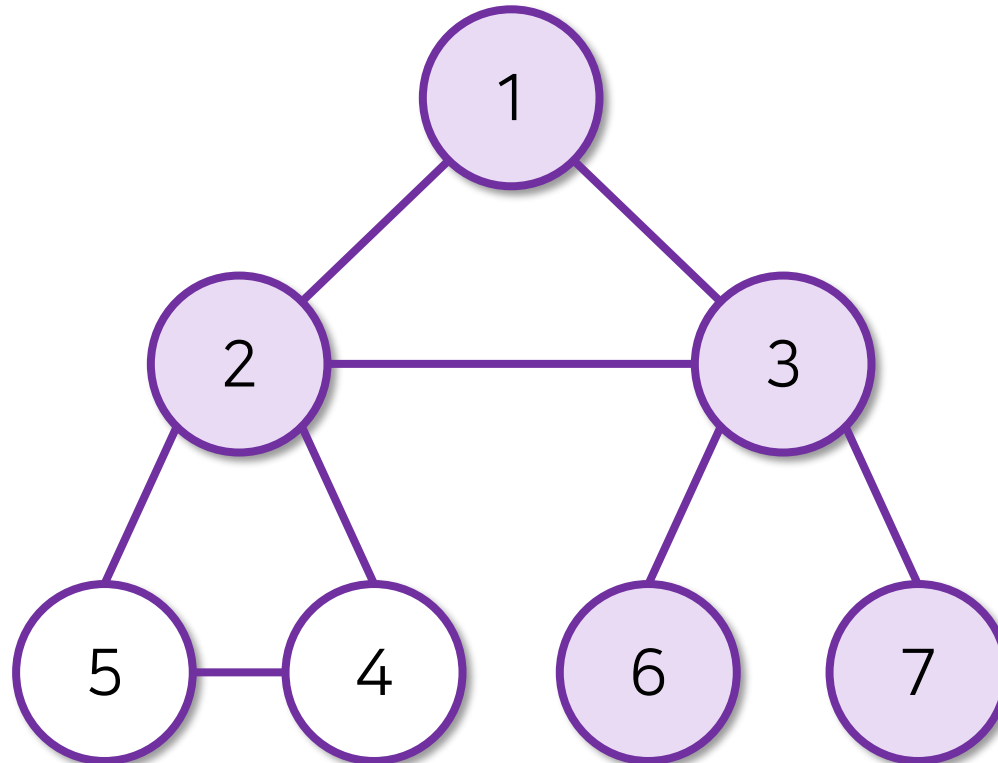


1-2. DFS 동작 과정

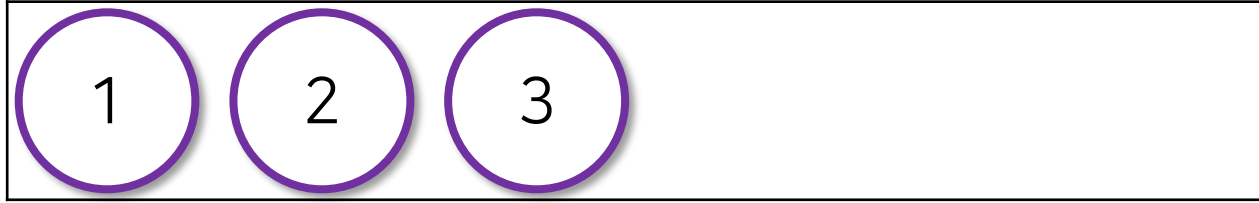


방문 기준

번호가 낮은 인접 노드부터

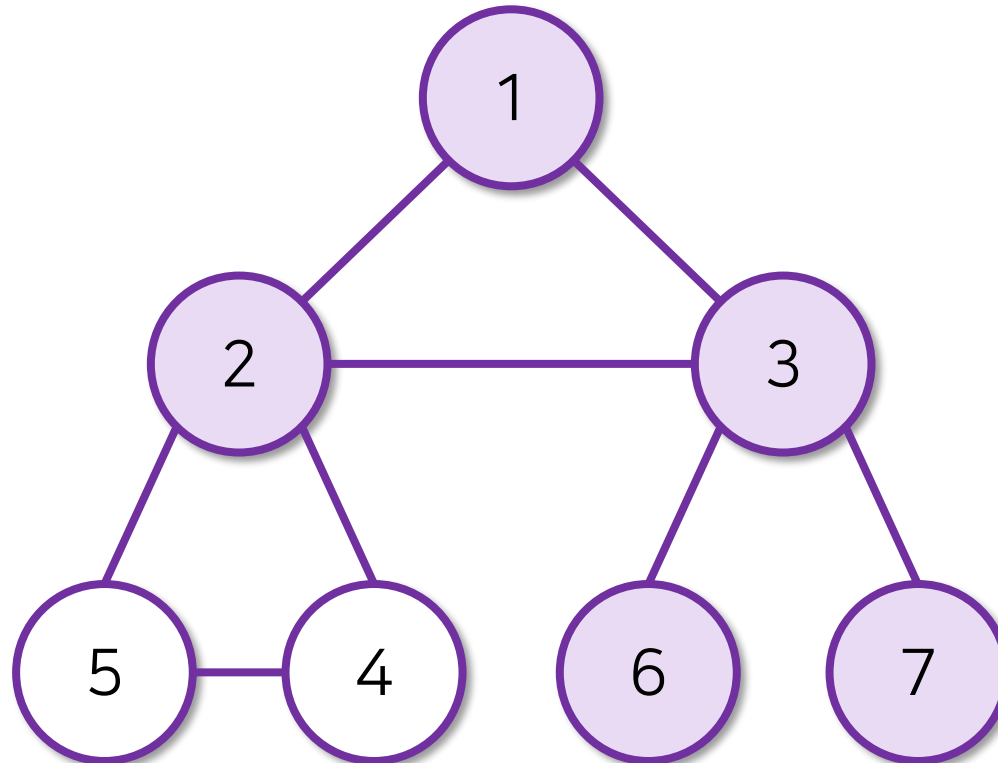


1-2. DFS 동작 과정



방문 기준

번호가 낮은 인접 노드부터

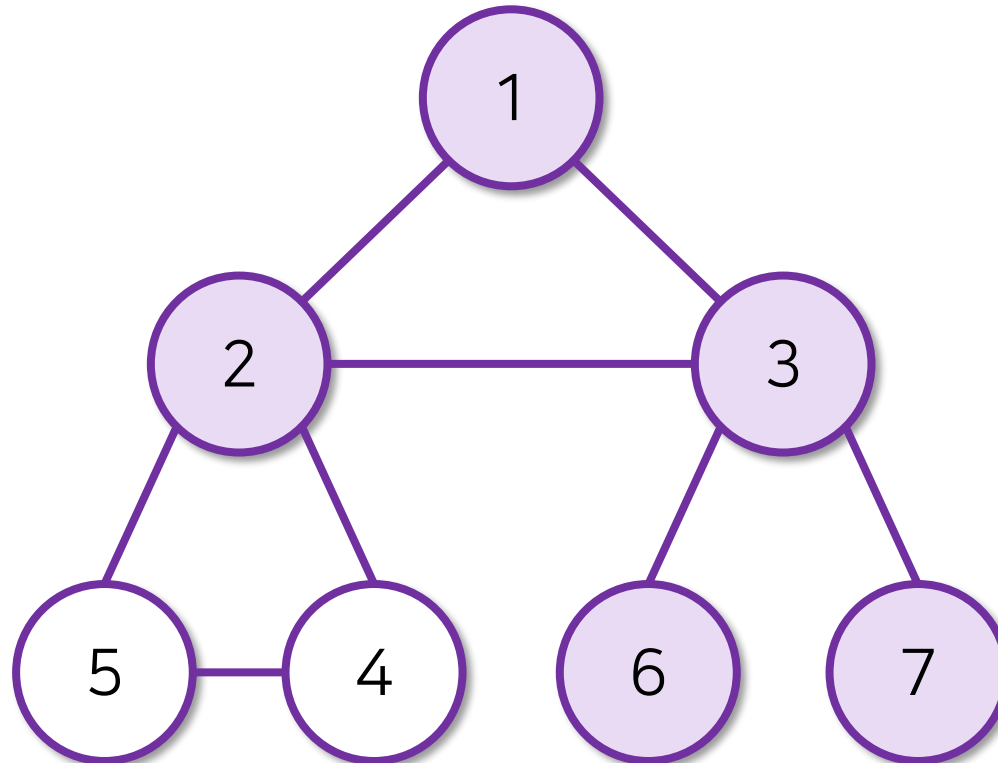


1-2. DFS 동작 과정



방문 기준

번호가 낮은 인접 노드부터

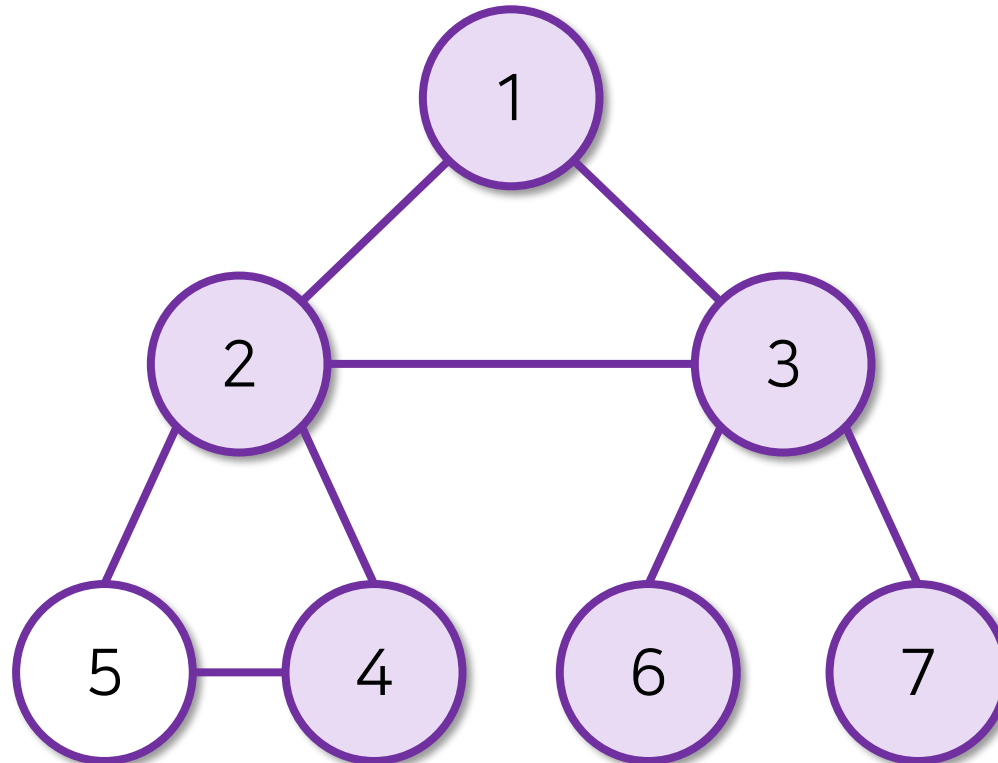


1-2. DFS 동작 과정



방문 기준

번호가 낮은 인접 노드부터

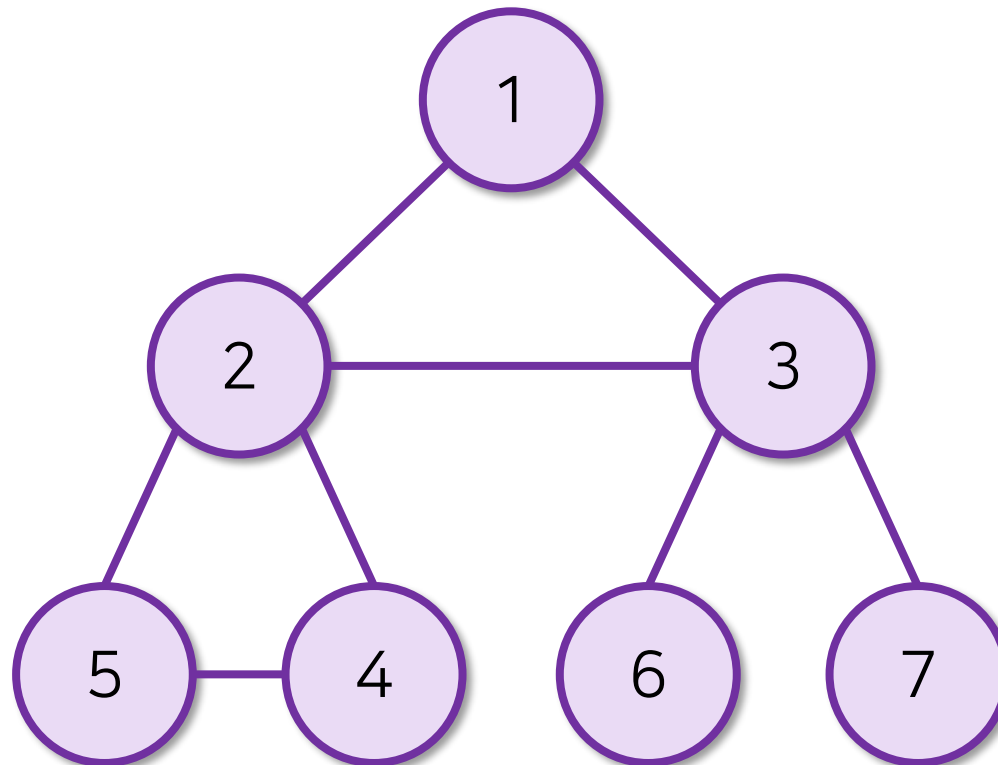


1-2. DFS 동작 과정



방문 기준

번호가 낮은 인접 노드부터

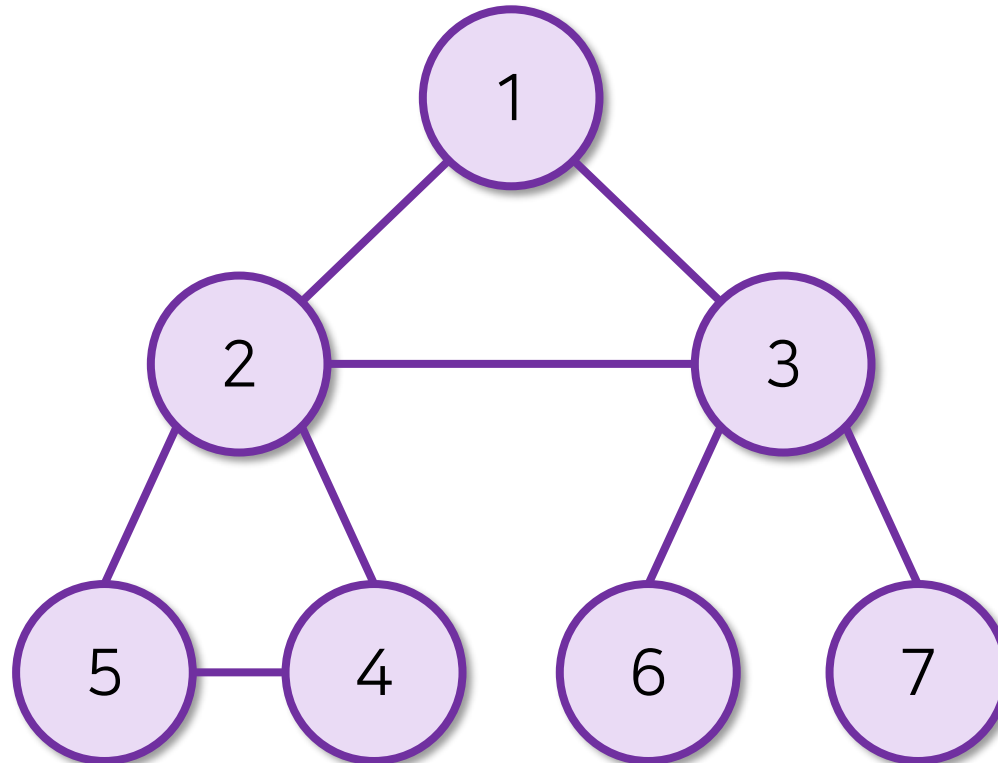


1-2. DFS 동작 과정

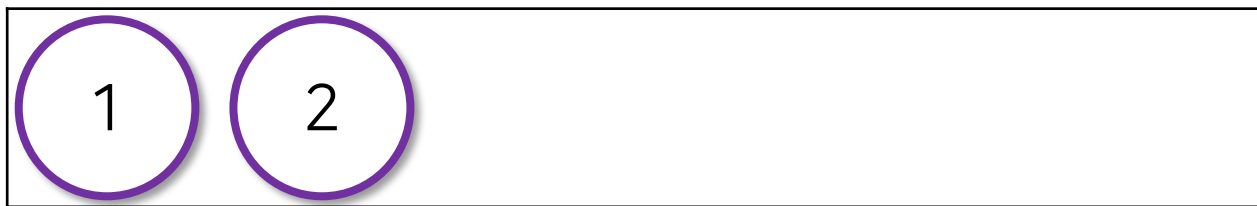


방문 기준

번호가 낮은 인접 노드부터

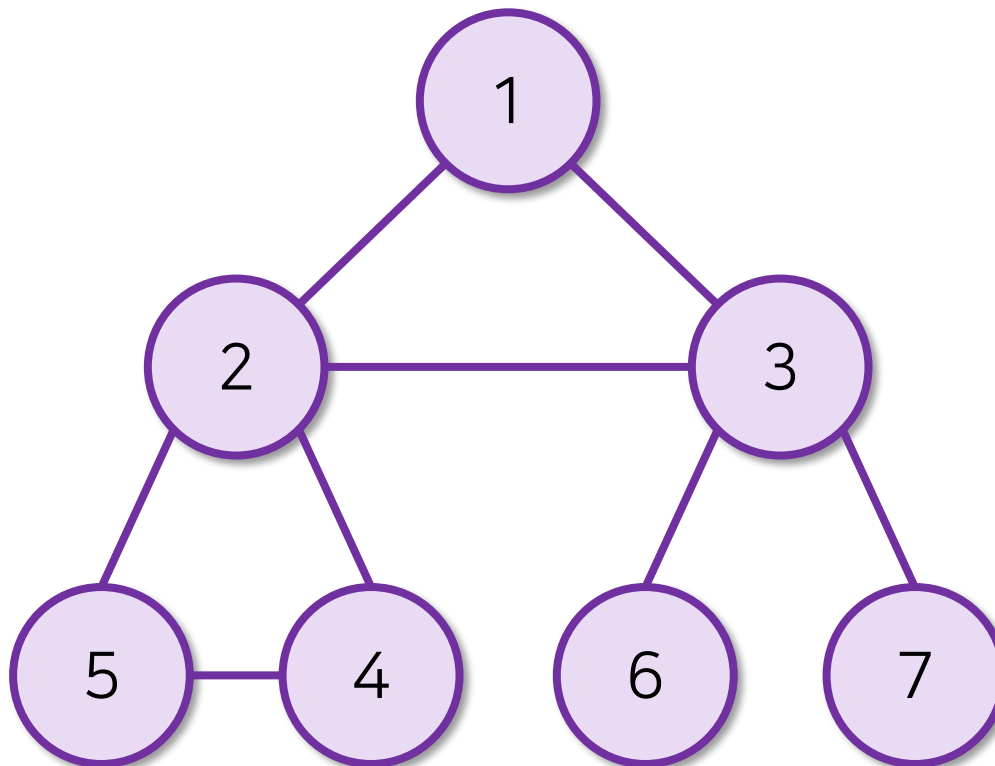


1-2. DFS 동작 과정



방문 기준

번호가 낮은 인접 노드부터

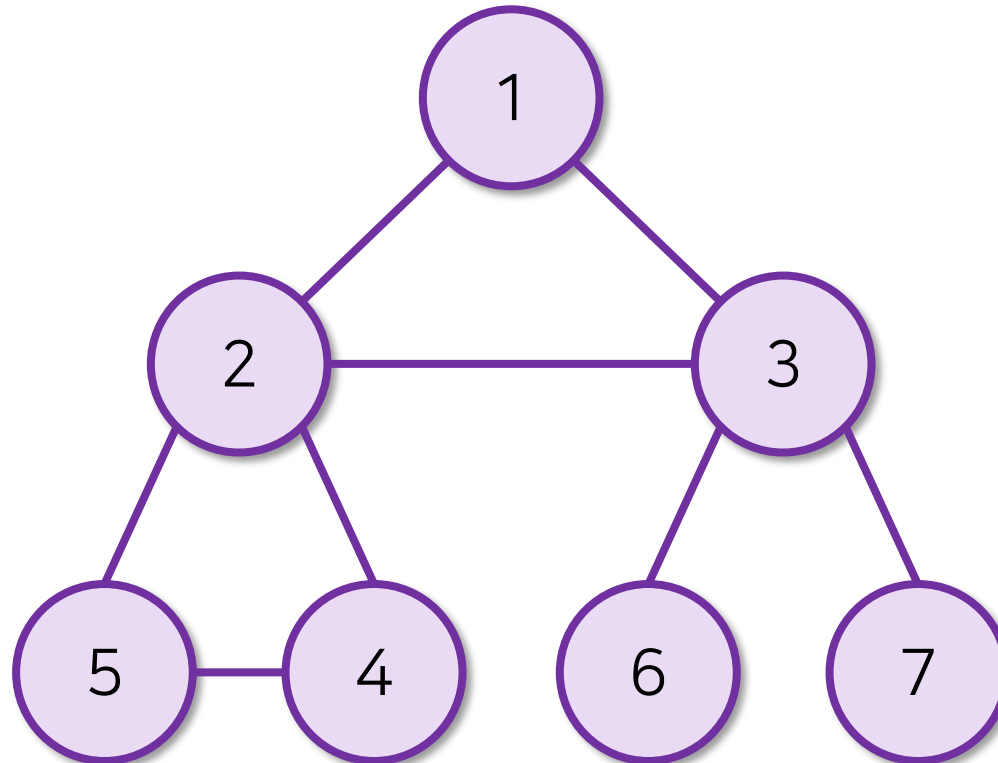


1-2. DFS 동작 과정



방문 기준

번호가 낮은 인접 노드부터

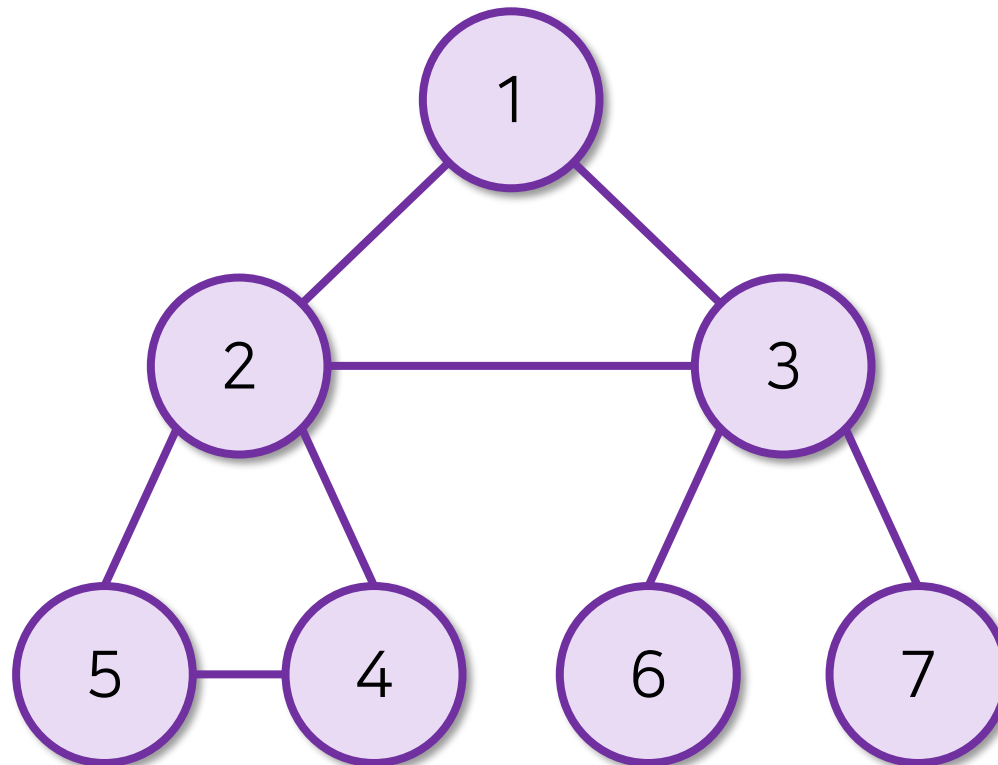


1-2. DFS 동작 과정



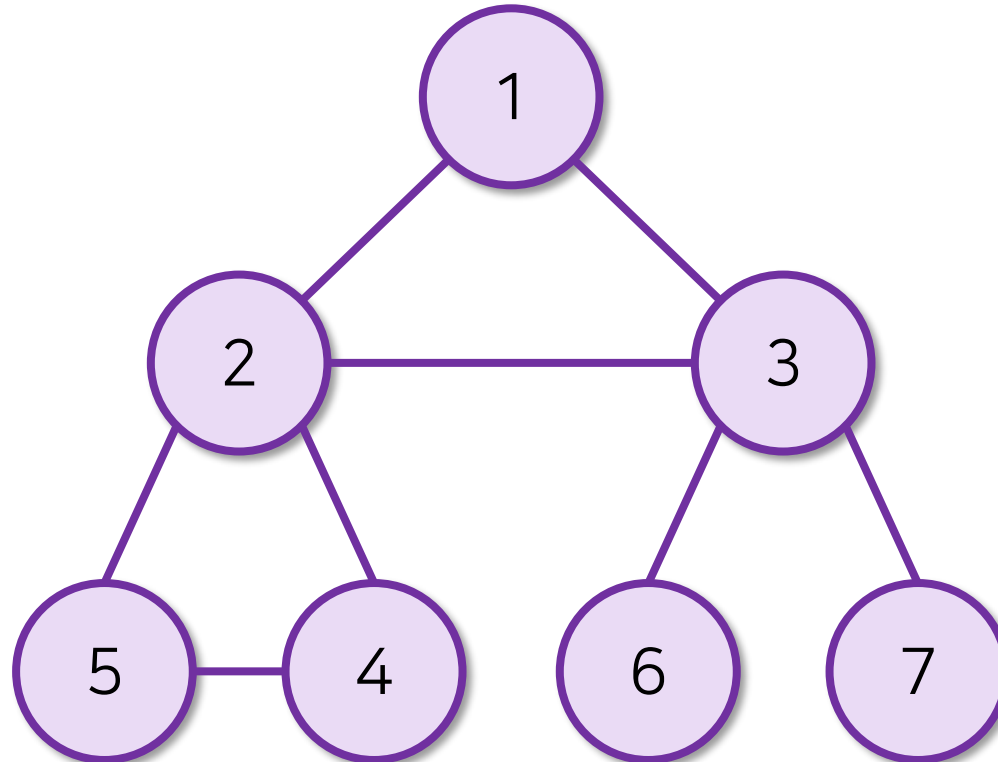
방문 기준

번호가 낮은 인접 노드부터



1-2. DFS 동작 과정

방문 경로



2-1. 큐 정의

먼저 집어 넣은 데이터가 먼저 나오는 구조

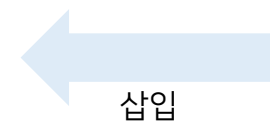
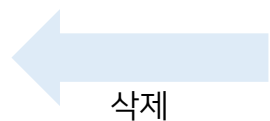
선입선출(*First In First Out*)

스택은 후입선출(*Last In First Out*)



2-2. 큐 동작 예시

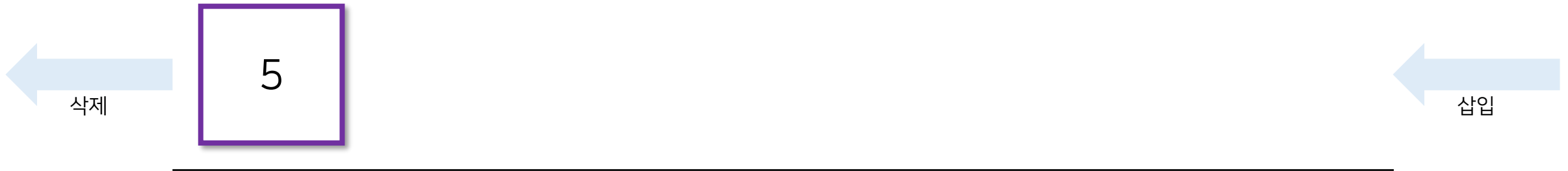
삽입(5) - 삽입(2) - 삽입(3) - 삽입(7) - 삭제() - 삽입(1) - 삽입(4) - 삭제()



↑ 큐 상자

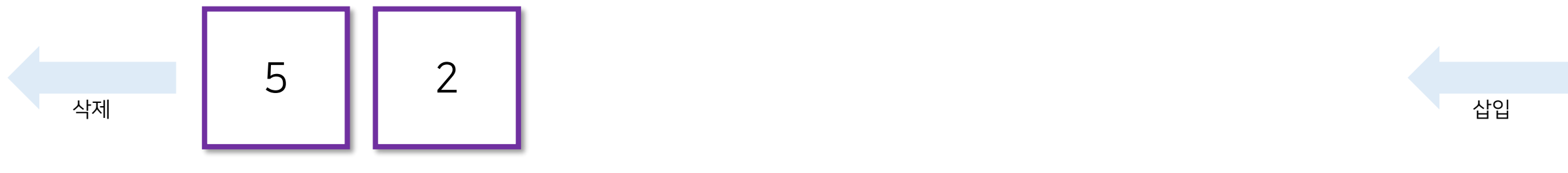
2-2. 큐 동작 예시

삽입(5) - 삽입(2) - 삽입(3) - 삽입(7) - 삭제() - 삽입(1) - 삽입(4) - 삭제()



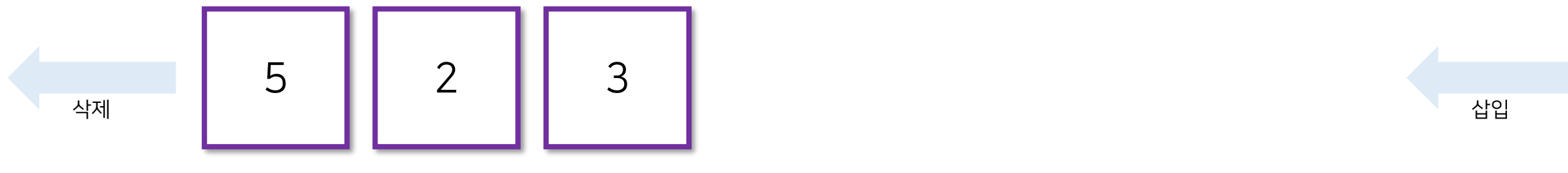
2-2. 큐 동작 예시

삽입(5) - **삽입(2)** - 삽입(3) - 삽입(7) - 삭제() - 삽입(1) - 삽입(4) - 삭제()



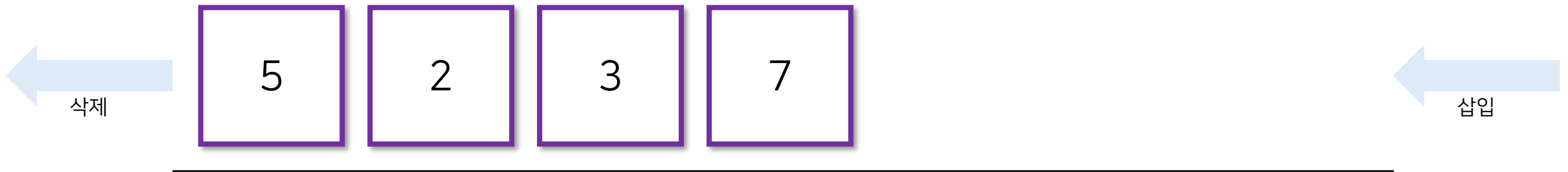
2-2. 큐 동작 예시

삽입(5) - 삽입(2) - **삽입(3)** - 삽입(7) - 삭제() - 삽입(1) - 삽입(4) - 삭제()



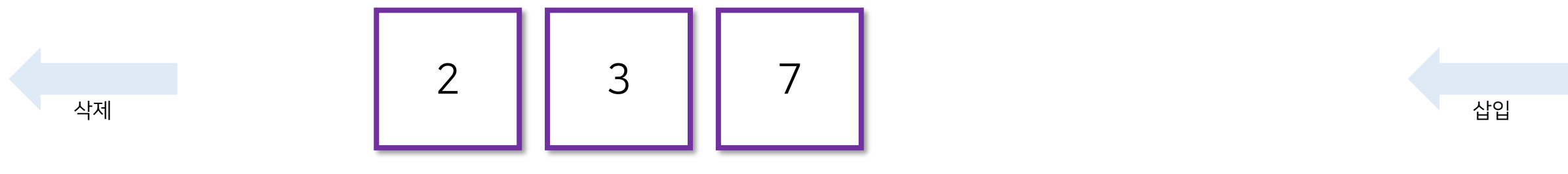
2-2. 큐 동작 예시

삽입(5) - 삽입(2) - 삽입(3) - **삽입(7)** - 삭제() - 삽입(1) - 삽입(4) - 삭제()



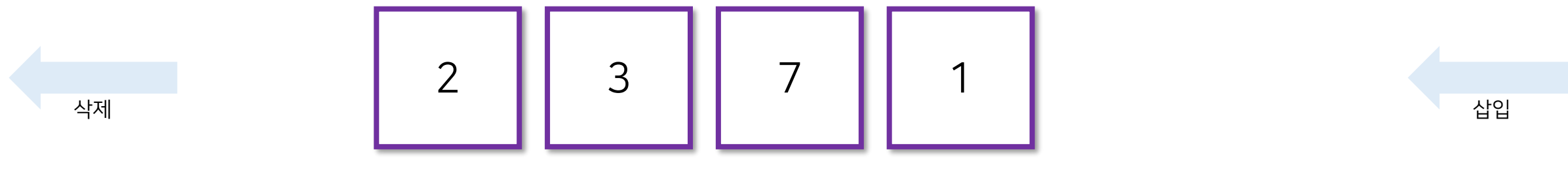
2-2. 큐 동작 예시

삽입(5) - 삽입(2) - 삽입(3) - 삽입(7) - 삭제() - 삽입(1) - 삽입(4) - 삭제()



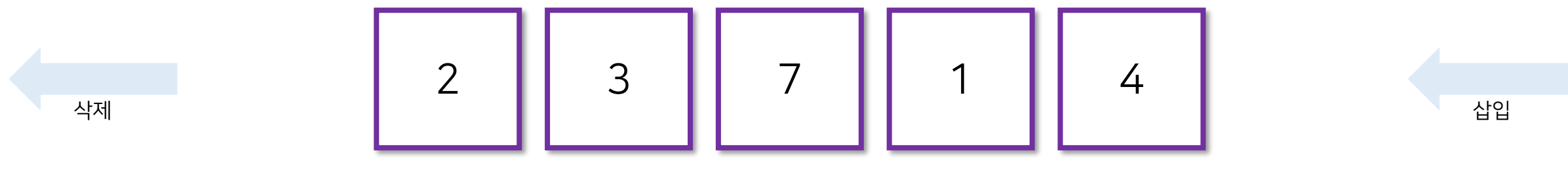
2-2. 큐 동작 예시

삽입(5) - 삽입(2) - 삽입(3) - 삽입(7) - 삭제() - **삽입(1)** - 삽입(4) - 삭제()



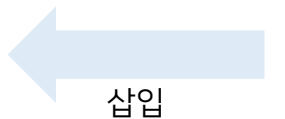
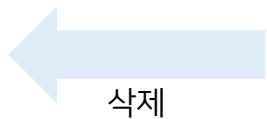
2-2. 큐 동작 예시

삽입(5) - 삽입(2) - 삽입(3) - 삽입(7) - 삭제() - 삽입(1) - **삽입(4)** - 삭제()



2-2. 큐 동작 예시

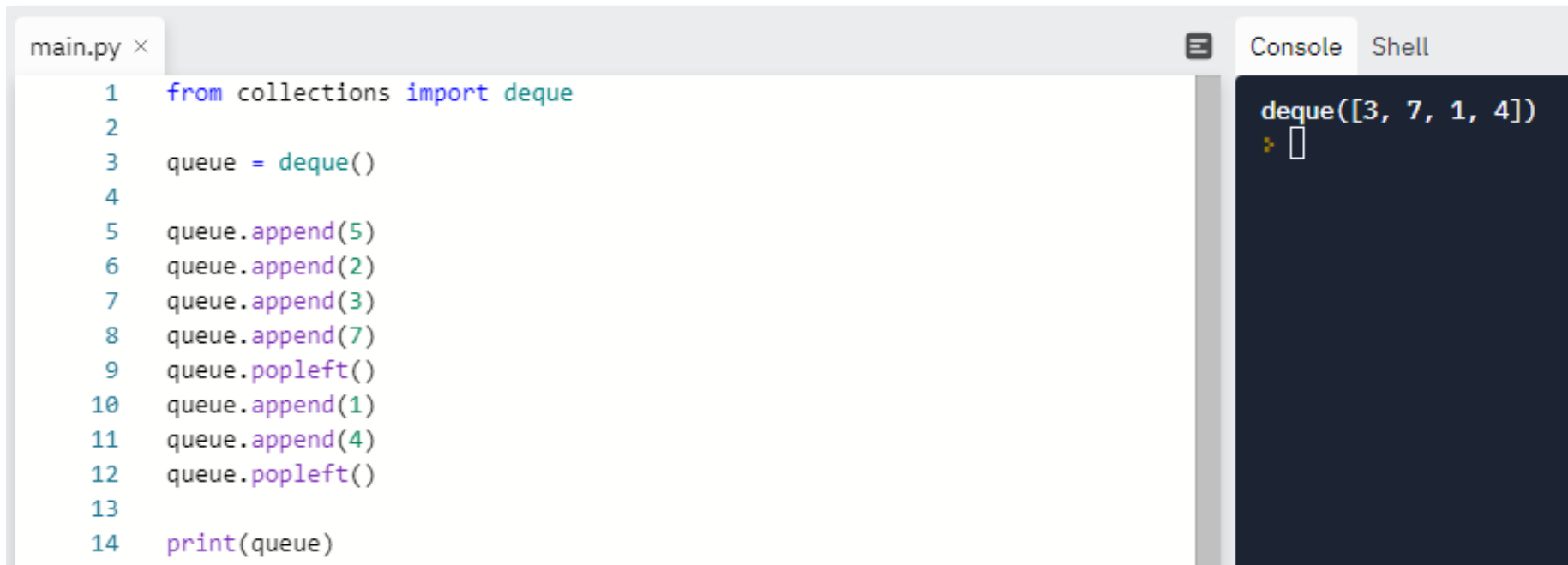
삽입(5) - 삽입(2) - 삽입(3) - 삽입(7) - 삭제() - 삽입(1) - 삽입(4) - 삭제()



2-3. 큐 구현 예제 (Python)

리스트 자료형으로도 구현할 수 있지만, 시간 복잡도를 봤을 때 비효율적

삽입(5) - 삽입(2) - 삽입(3) - 삽입(7) - 삭제() - 삽입(1) - 삽입(4) - 삭제()



```
main.py x
1  from collections import deque
2
3  queue = deque()
4
5  queue.append(5)
6  queue.append(2)
7  queue.append(3)
8  queue.append(7)
9  queue.popleft()
10 queue.append(1)
11 queue.append(4)
12 queue.popleft()
13
14 print(queue)
```

Console Shell

```
deque([3, 7, 1, 4])
>
```

3-1. BFS 정의

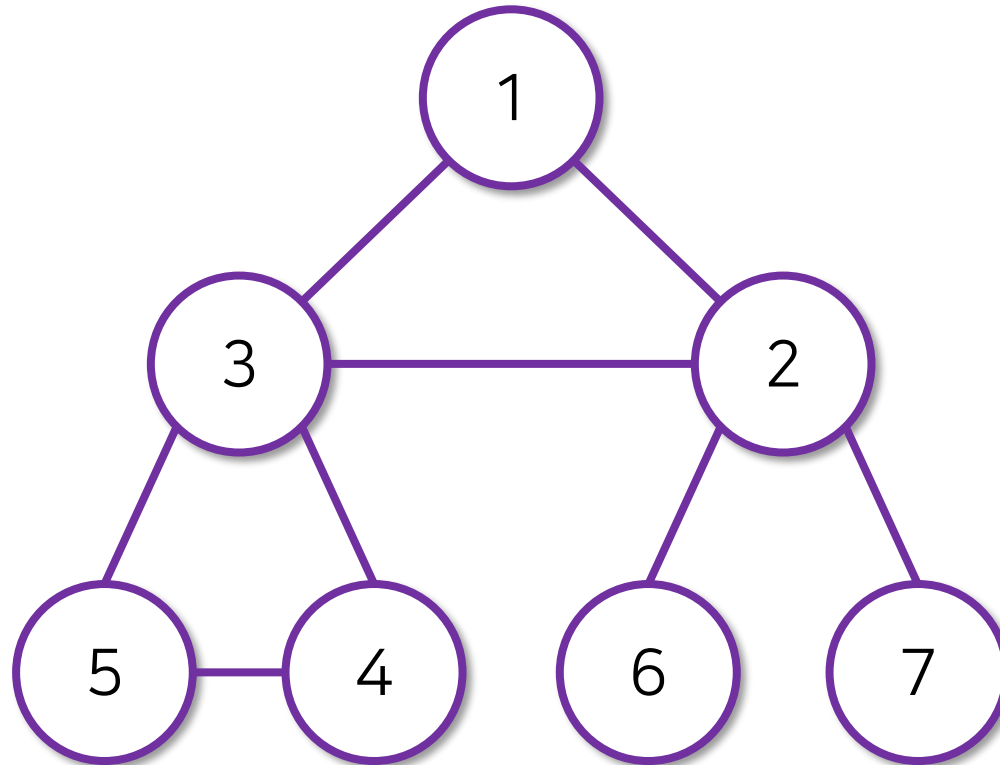
BFS (너비 우선 탐색)

- 루트 노드에서 시작해서 가까운 노드를 탐색하는 방법
- 가까운 정점을 먼저 방문하고 멀리 떨어져 있는 정점을 나중에 방문하는 '순회 방법'
- 모든 간선의 비용이 동일할 때 최단 경로를 구하는 목적으로 사용
- 큐 자료구조 활용

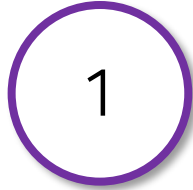
동작 과정

1. 탐색 시작 노드 정보를 큐에 삽입하고, 방문 처리
2. 큐에서 노드를 꺼내 방문하지 않은 인접 노드 정보를 모두 큐에 삽입하고, 방문 처리.
3. 더 이상 방문할 곳이 없을 때까지 반복

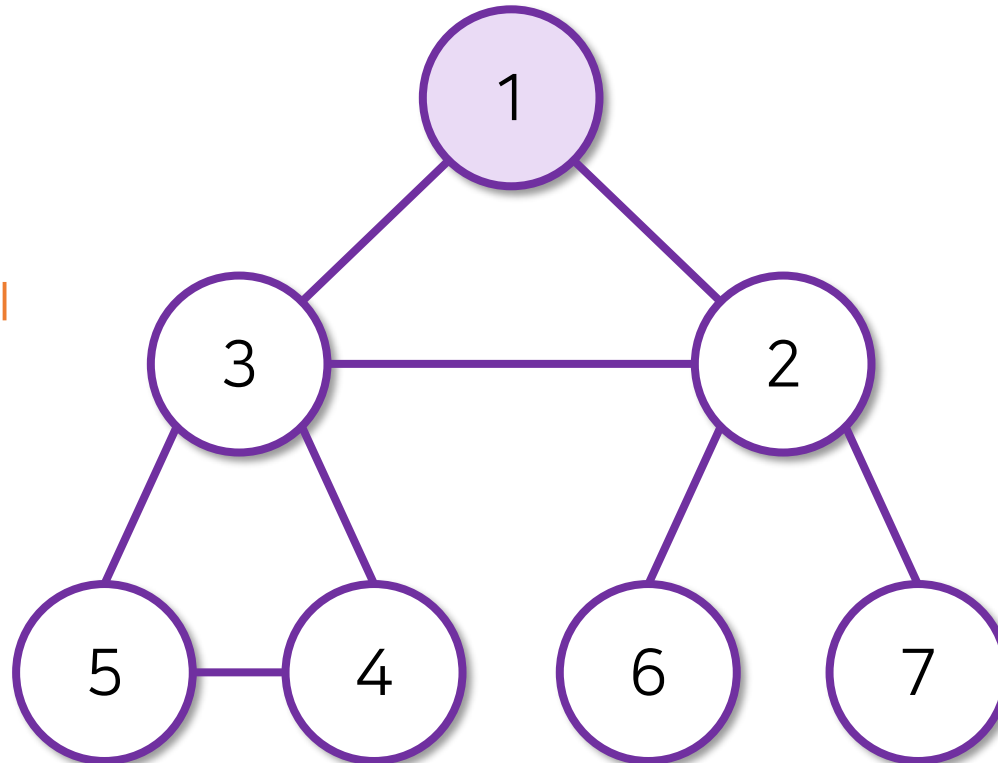
3-2. BFS 동작 예시



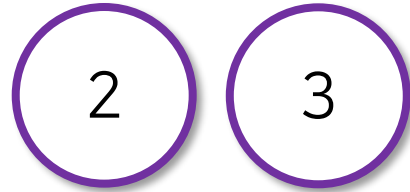
3-2. BFS 동작 예시



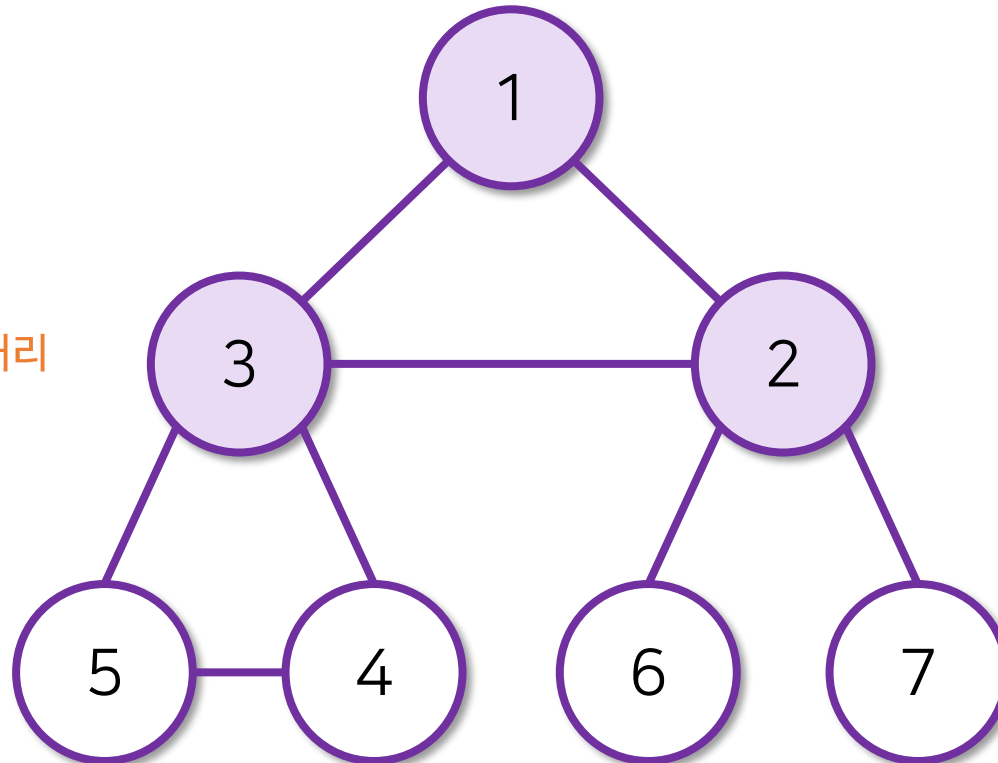
[Step 1]
시작 노드인 '1'을 큐에 삽입하고 방문 처리



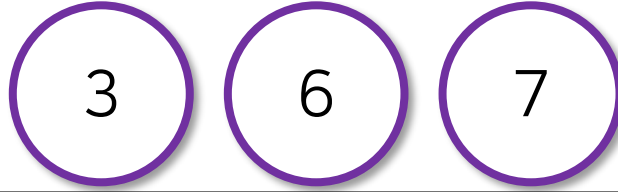
3-2. BFS 동작 예시



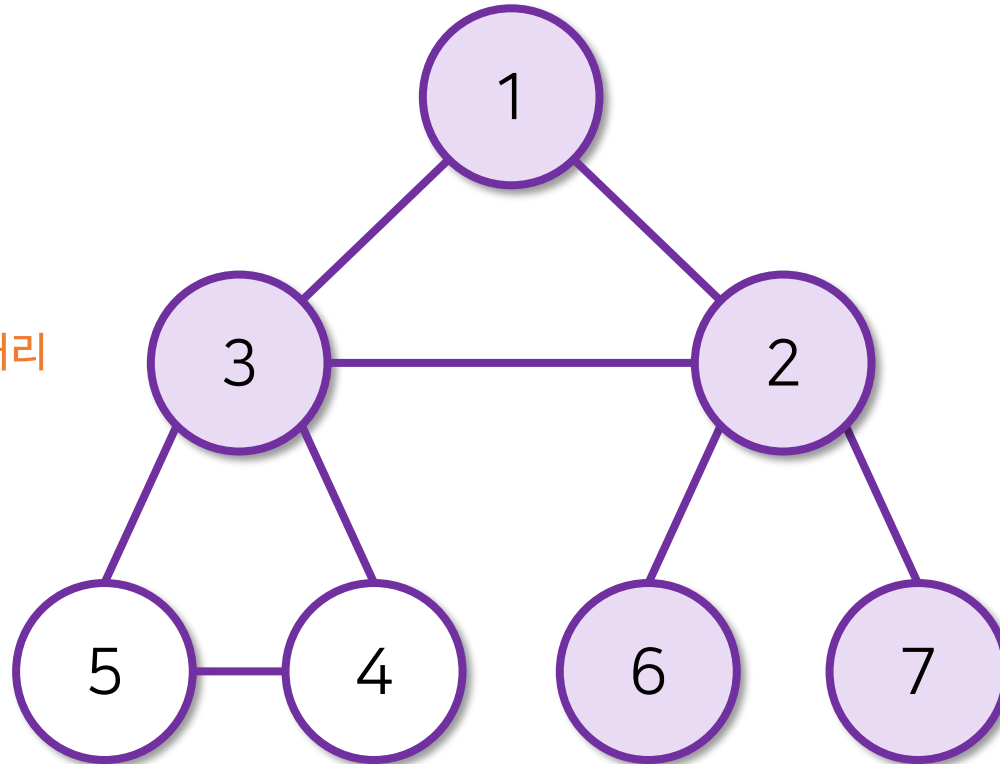
[Step 2]
큐에서 노드 '1'을 꺼내고, 방문하지 않은
인접 노드 '2', '3'을 큐에 삽입하고 방문 처리



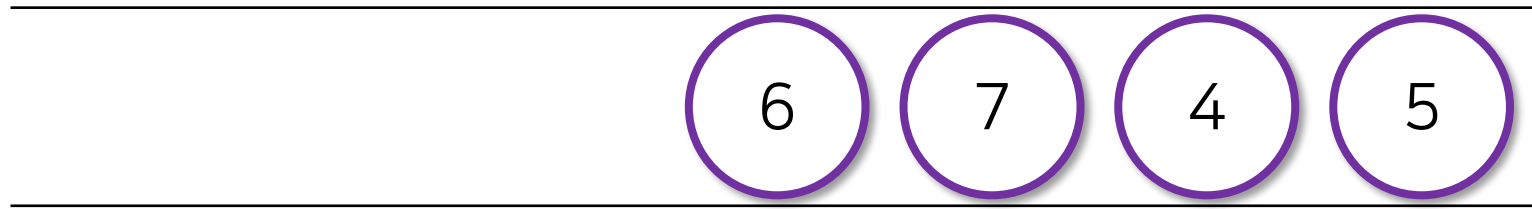
3-2. BFS 동작 예시



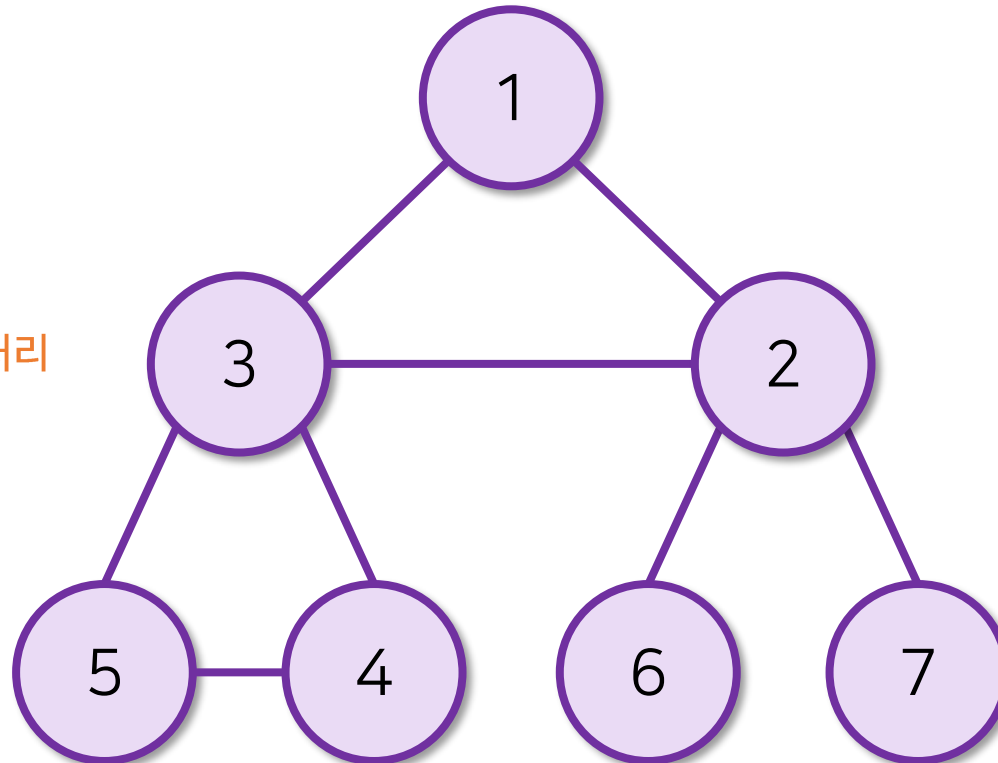
[Step 3]
큐에서 노드 '2'를 꺼내고, 방문하지 않은
인접 노드 '6', '7'을 큐에 삽입하고 방문 처리



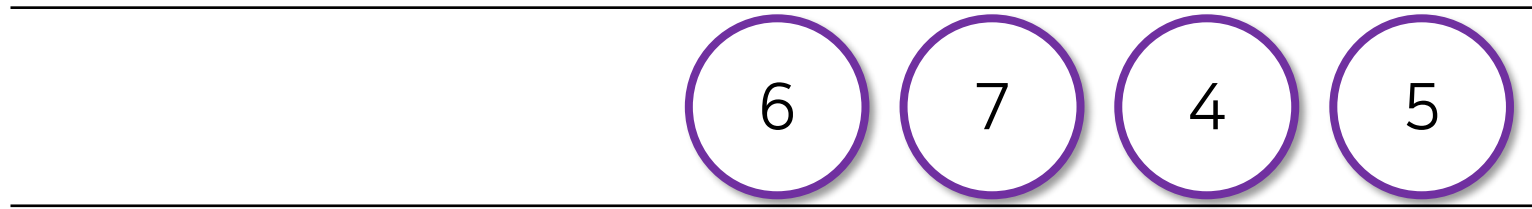
3-2. BFS 동작 예시



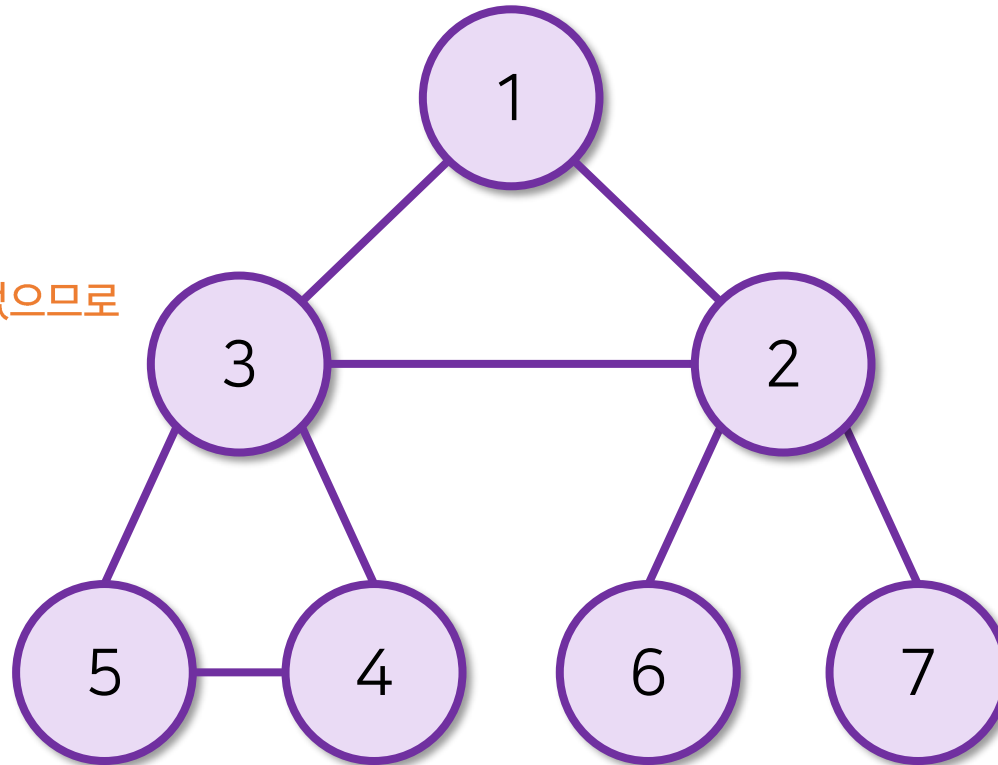
[Step 4]
큐에서 노드 '3'을 꺼내고, 방문하지 않은
인접 노드 '4', '5'를 큐에 삽입하고 방문 처리



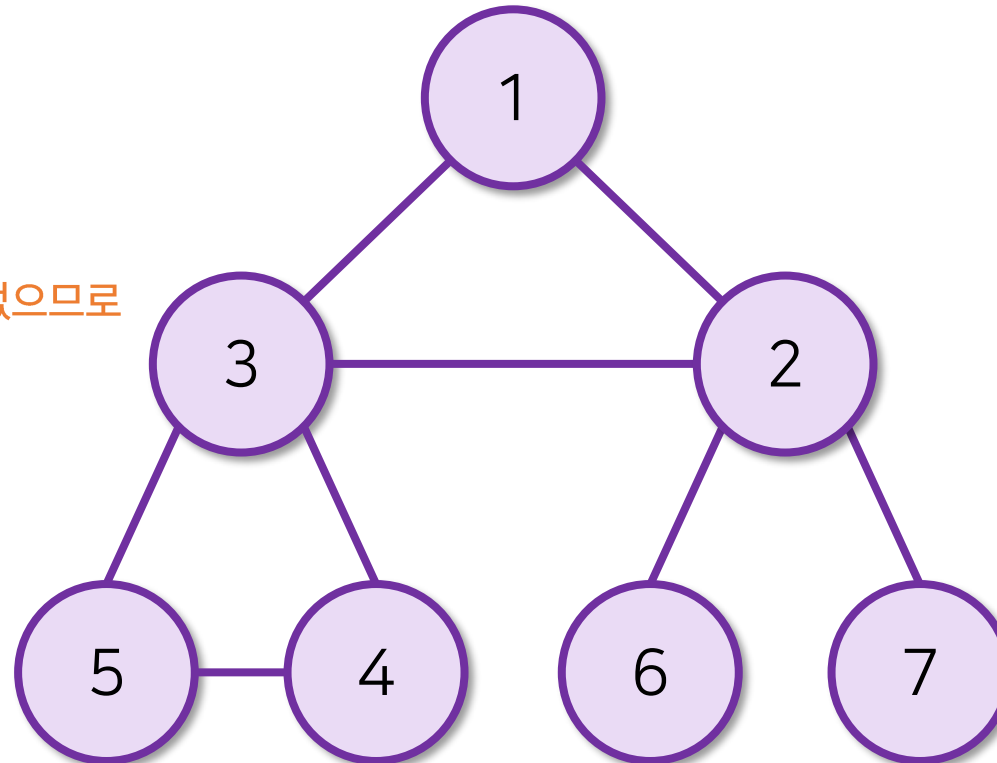
3-2. BFS 동작 예시



[Step 5]
그래프 내 노드에 방문하지 않은 노드가 없으므로
큐에서 모든 노드를 차례대로 제거



3-2. BFS 동작 예시

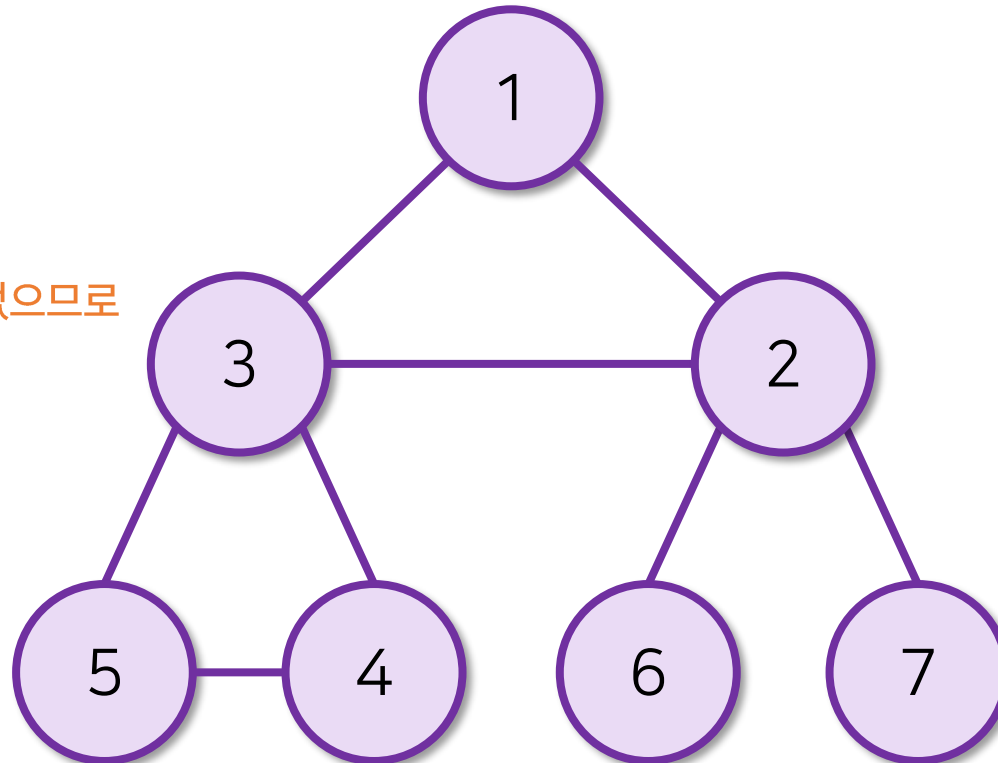


[Step 5]
그래프 내 노드에 방문하지 않은 노드가 없으므로
큐에서 모든 노드를 차례대로 제거

3-2. BFS 동작 예시



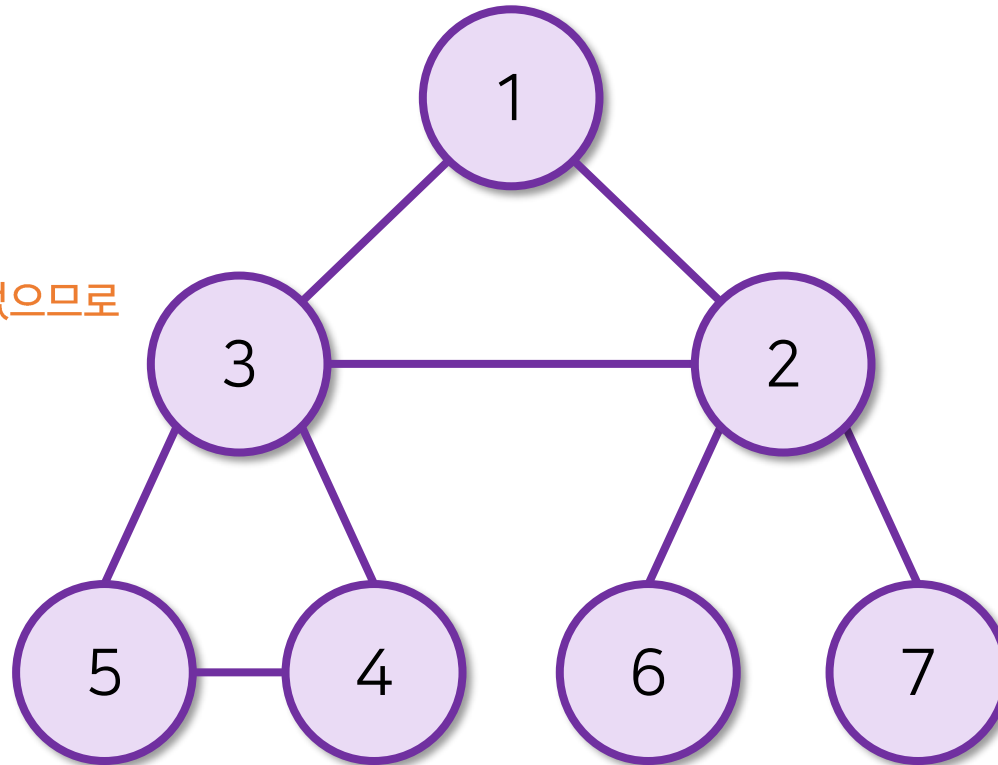
[Step 5]
그래프 내 노드에 방문하지 않은 노드가 없으므로
큐에서 모든 노드를 차례대로 제거



3-2. BFS 동작 예시

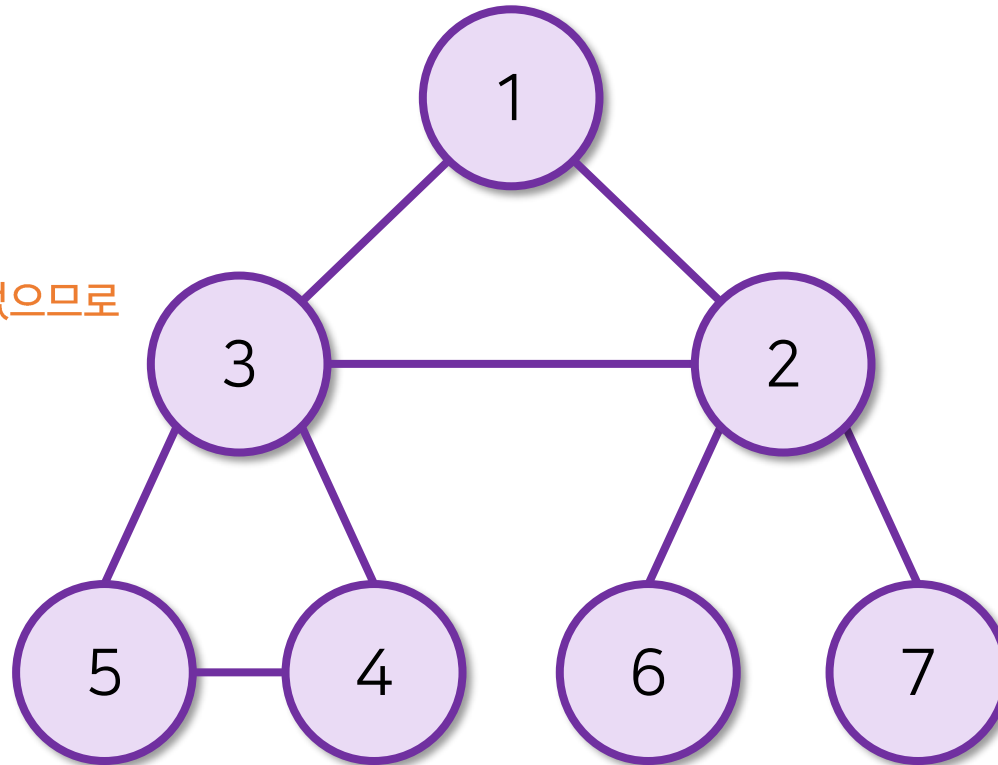
5

[Step 5]
그래프 내 노드에 방문하지 않은 노드가 없으므로
큐에서 모든 노드를 차례대로 제거



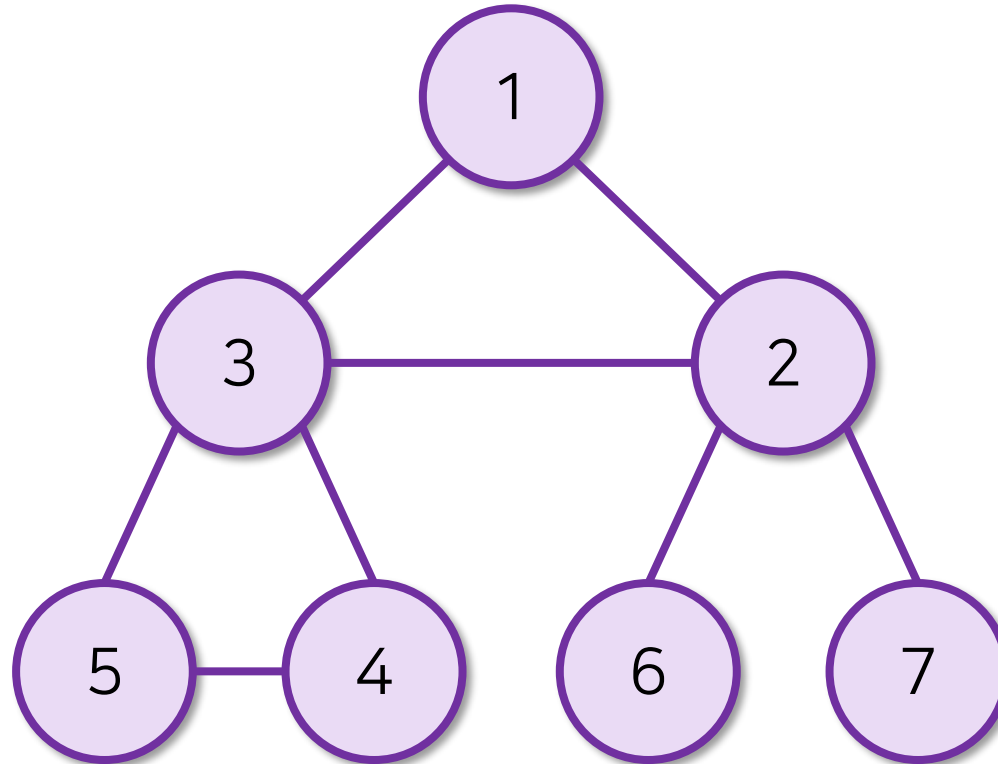
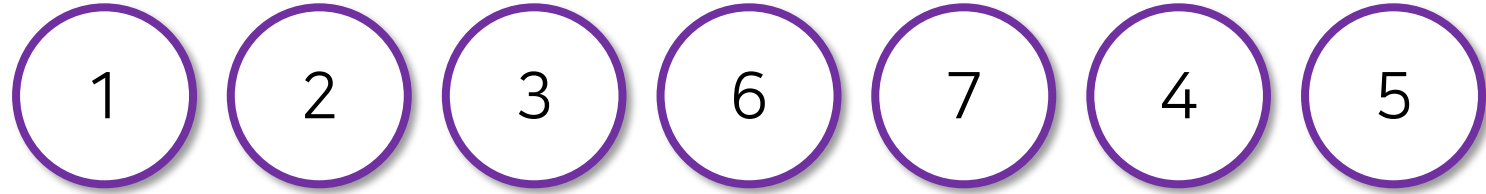
3-2. BFS 동작 예시

[Step 5]
그래프 내 노드에 방문하지 않은 노드가 없으므로
큐에서 모든 노드를 차례대로 제거

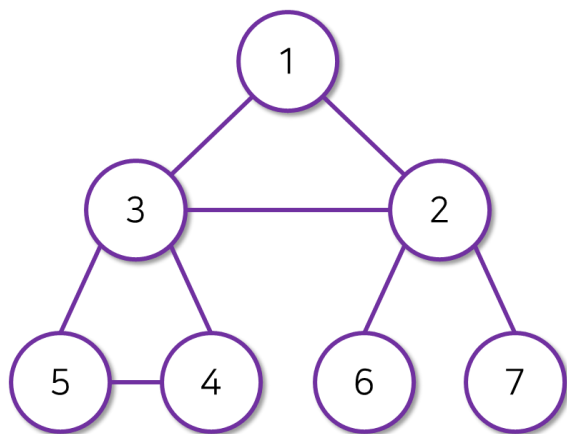


3-2. BFS 동작 예시

방문 경로



3-3. BFS 구현 예제 (Python)

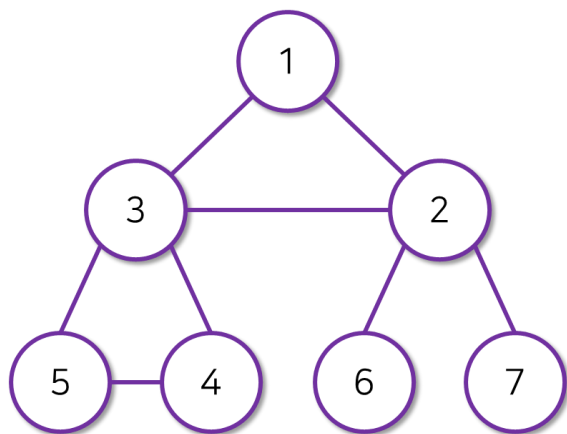


```
main.py × Console
1 from collections import deque
2
3 graph = [
4     [],
5     [2, 3],
6     [1, 3, 6, 7],
7     [1, 2, 4, 5],
8     [3, 5],
9     [4, 5],
10    [2],
11    [2]
12 ]
13
14 visited = [False] * len(graph)
15
16 def bfs (graph, node, visited):
17     queue = deque([node])
18     visited[node] = True
19     while queue:
20         v = queue.popleft()
21         print(v)
22         for i in graph[v]:
23             if not (visited[i]):
24                 queue.append(i)
25                 visited[i] = True
26
27 bfs(graph, 1, visited)
```

2차원 배열을 통해
노드 간의 연결 정보 표현

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

3-3. BFS 구현 예제 (Python)



```
1 def bfs (graph, node, visited):  
2     queue = deque([node])  
3     visited[node] = True  
4     while queue:  
5         v = queue.popleft()  
6         print(v)  
7         for i in graph[v]:  
8             if not (visited[i]):  
9                 queue.append(i)  
10            visited[i] = True
```

main.py ×

```
1 from collections import deque  
2  
3 graph = [  
4     [],  
5     [2, 3],  
6     [1, 3, 6, 7],  
7     [1, 2, 4, 5],  
8     [3, 5],  
9     [4, 5],  
10    [2],  
11    [2]  
12 ]  
13  
14 visited = [False] * len(graph)  
15  
16 def bfs (graph, node, visited):  
17     queue = deque([node])  
18     visited[node] = True  
19     while queue:  
20         v = queue.popleft()  
21         print(v)  
22         for i in graph[v]:  
23             if not (visited[i]):  
24                 queue.append(i)  
25                 visited[i] = True  
26  
27 bfs(graph, 1, visited)
```

Console

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

THANK YOU

QUESTION
