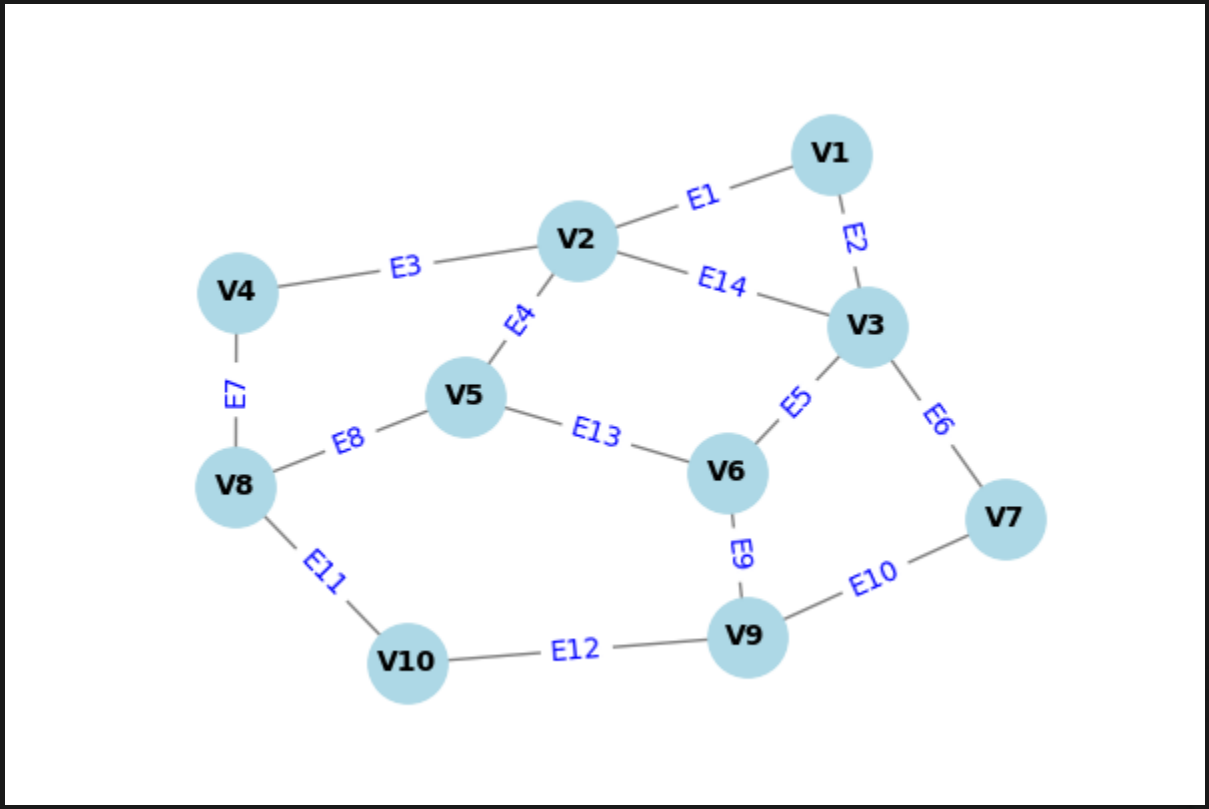
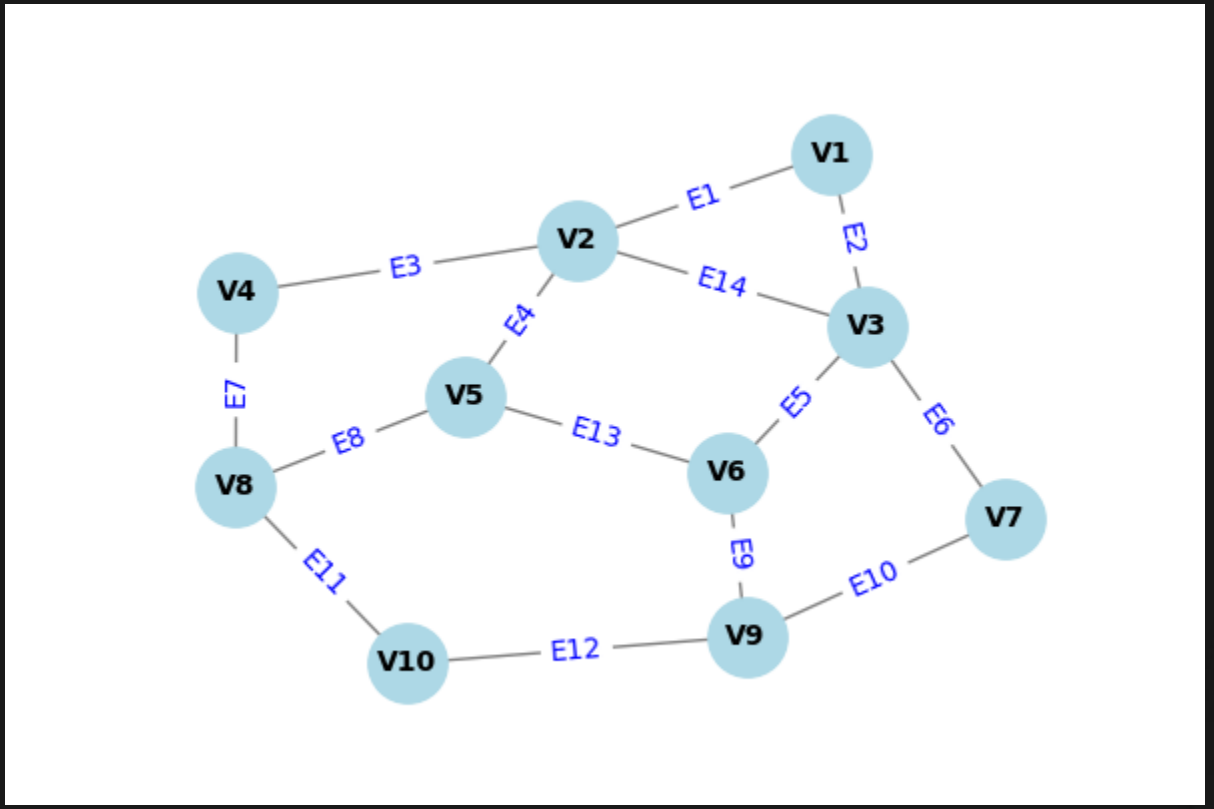
3-1 그래프

그래프 (Graph)


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
E1
E2
E3
E4
E5
E6
E7
E8
E9
E10
E11
E12
E13
E14

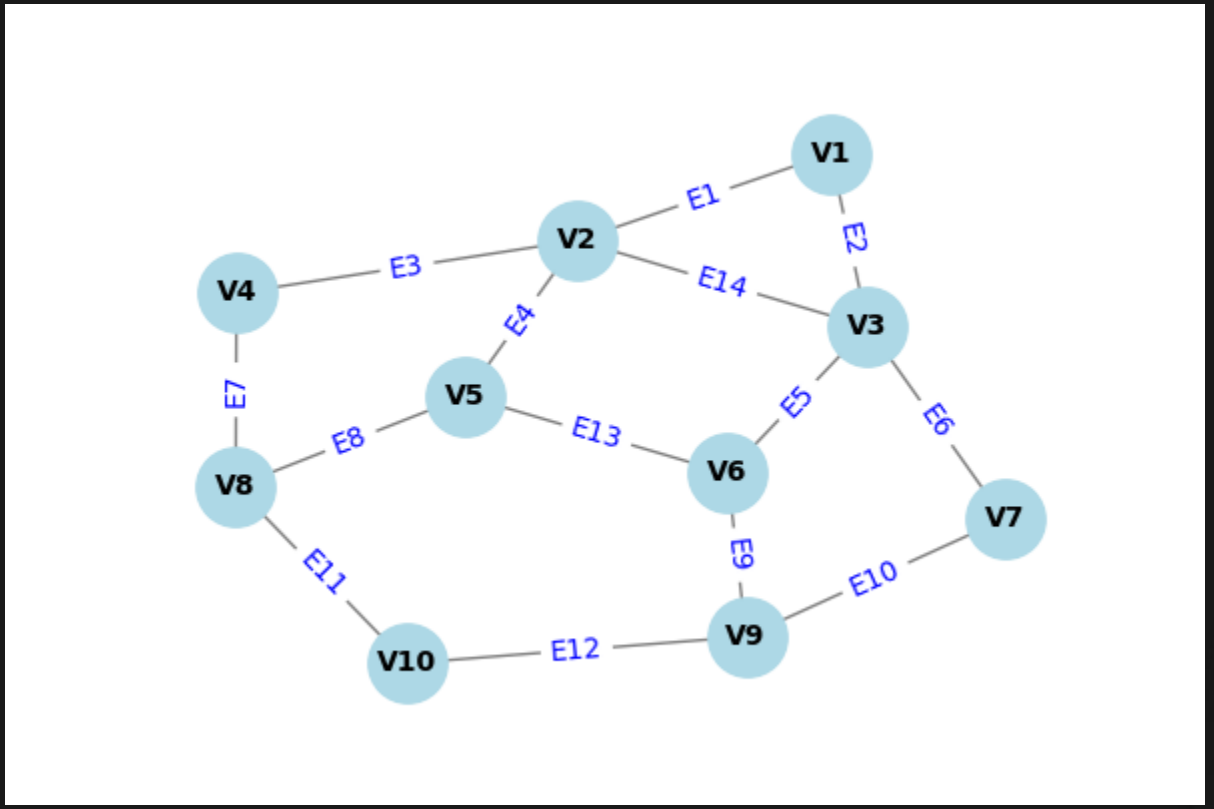
그래프 (Graph)

- 정의: Vertex (V) 와 Edge (E) 로 이루어진 순서쌍 (V,E)



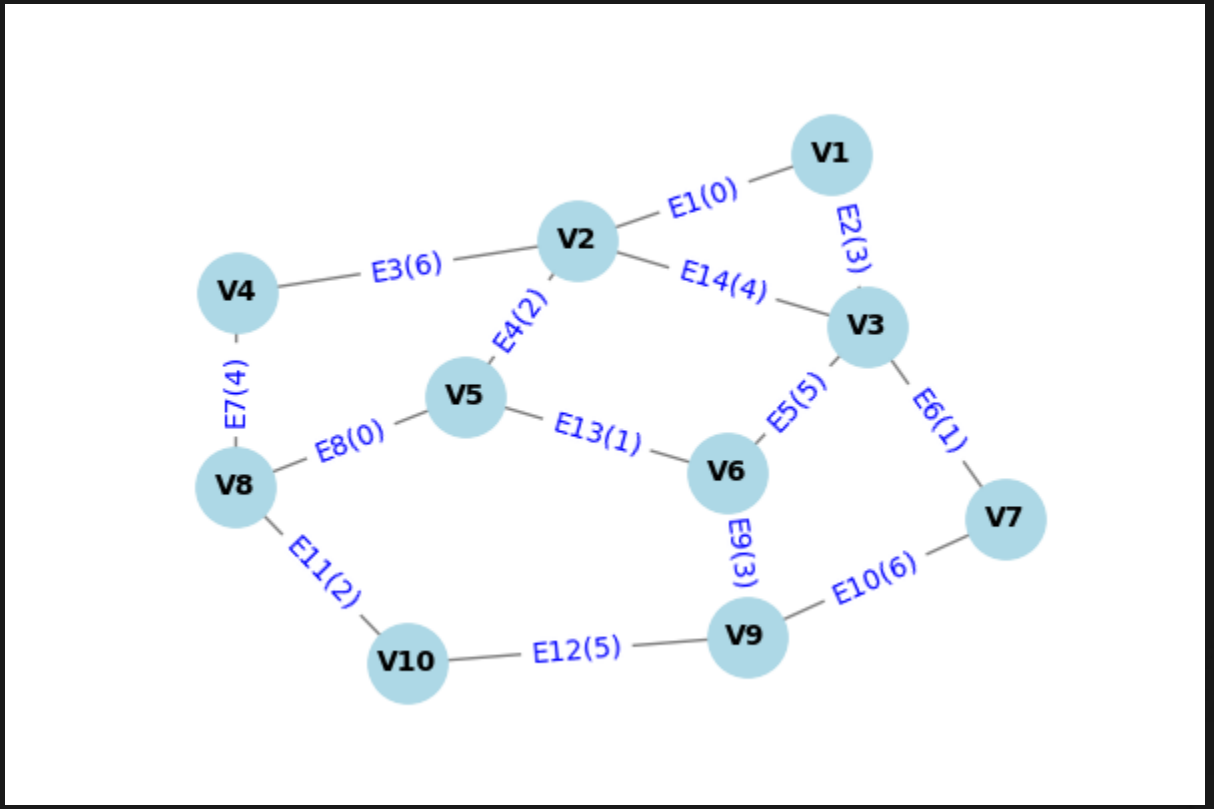
그래프 (Graph) - 예시 1

- 정의: Vertex (V) 와 Edge (E) 로 이루어진 순서쌍 (V,E)



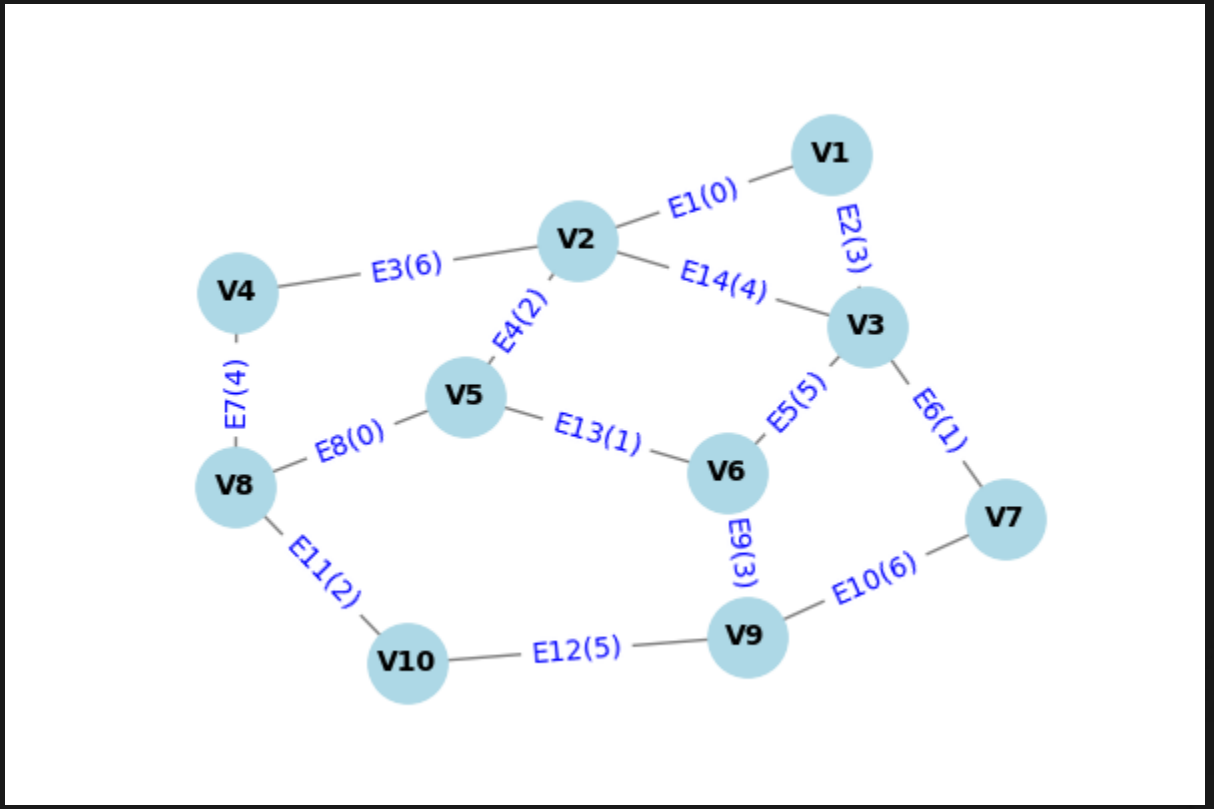
그래프 (Graph)

- 정의: Vertex (V) 와 Edge (E) 로 이루어진 순서쌍 (V,E)



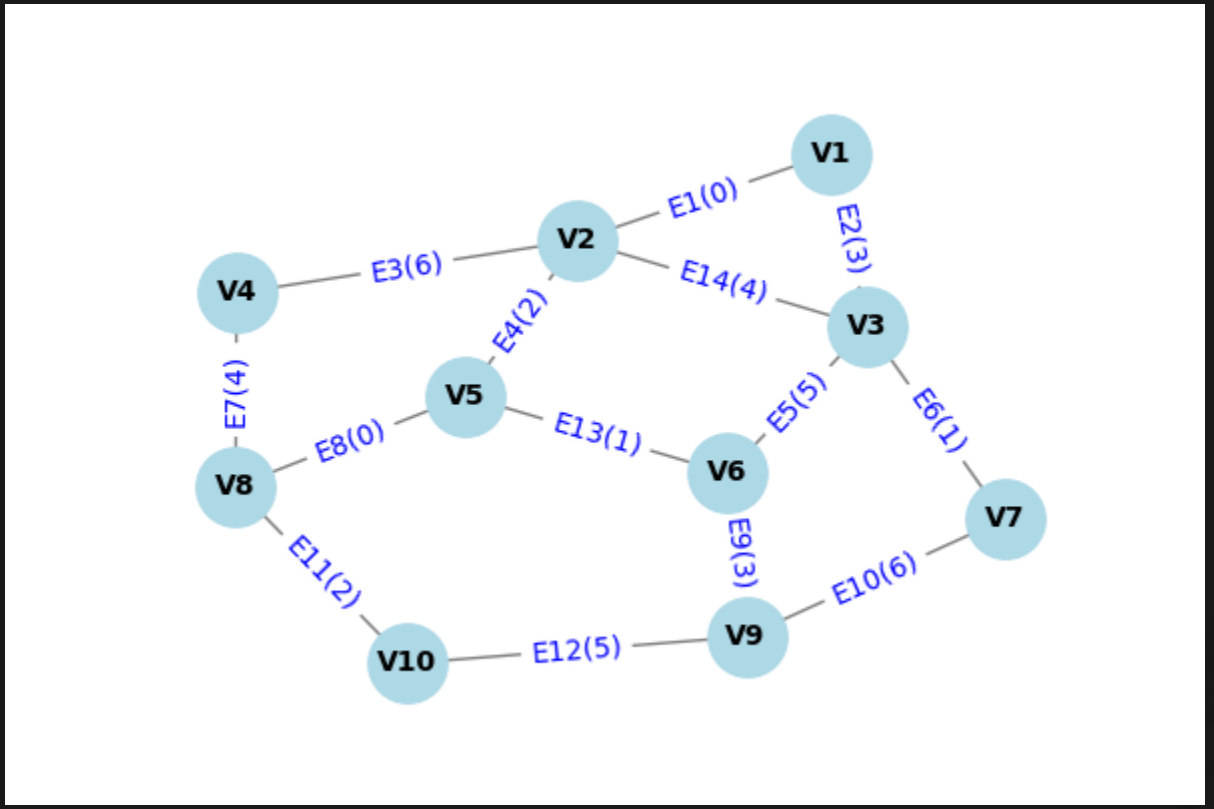
그래프 (Graph) - 예시 2

- 정의: Vertex (V) 와 Edge (E) 로 이루어진 순서쌍 (V,E)

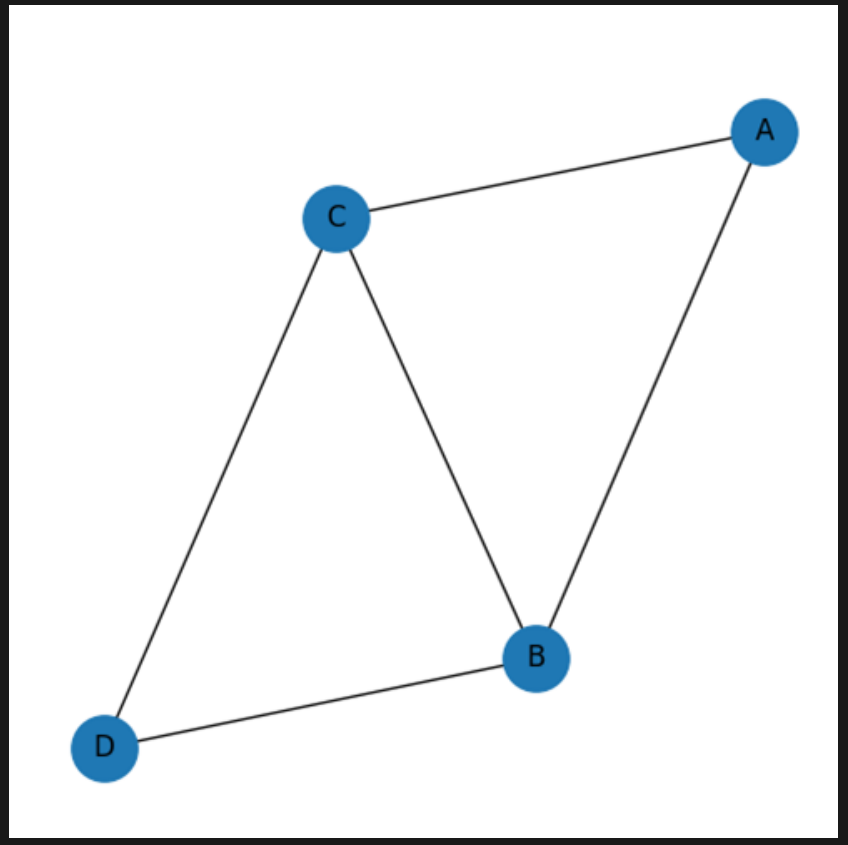


그래프 (Graph)

- 정의: Vertex (V) 와 Edge (E) 로 이루어진 순서쌍 (V,E)
- 필수 용어
  - 가중치 (weight)
  - 차수 (degree)
  - 경로 (path)



그래프의 데이터 구조



Adjacency List

A: ['B', 'C']

B: ['A', 'D', 'C']

C: ['A', 'D', 'B']

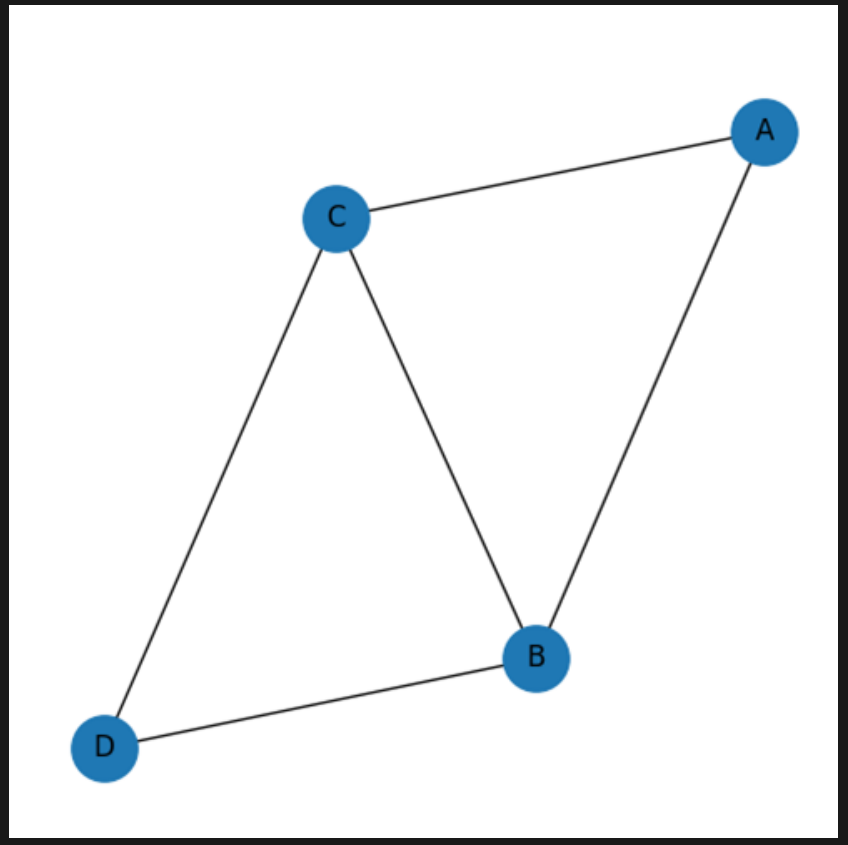
D: ['B', 'C']

```
graph_undir = {
    'A': ['B', 'C'],
    'B': ['A', 'C', 'D'],
    'C': ['A', 'B', 'D'],
    'D': ['B', 'C']
}
```

Adjacency Matrix

	A	B	C	D
A	0	1	1	0
B	1	0	1	1
C	1	1	0	1
D	0	1	1	0

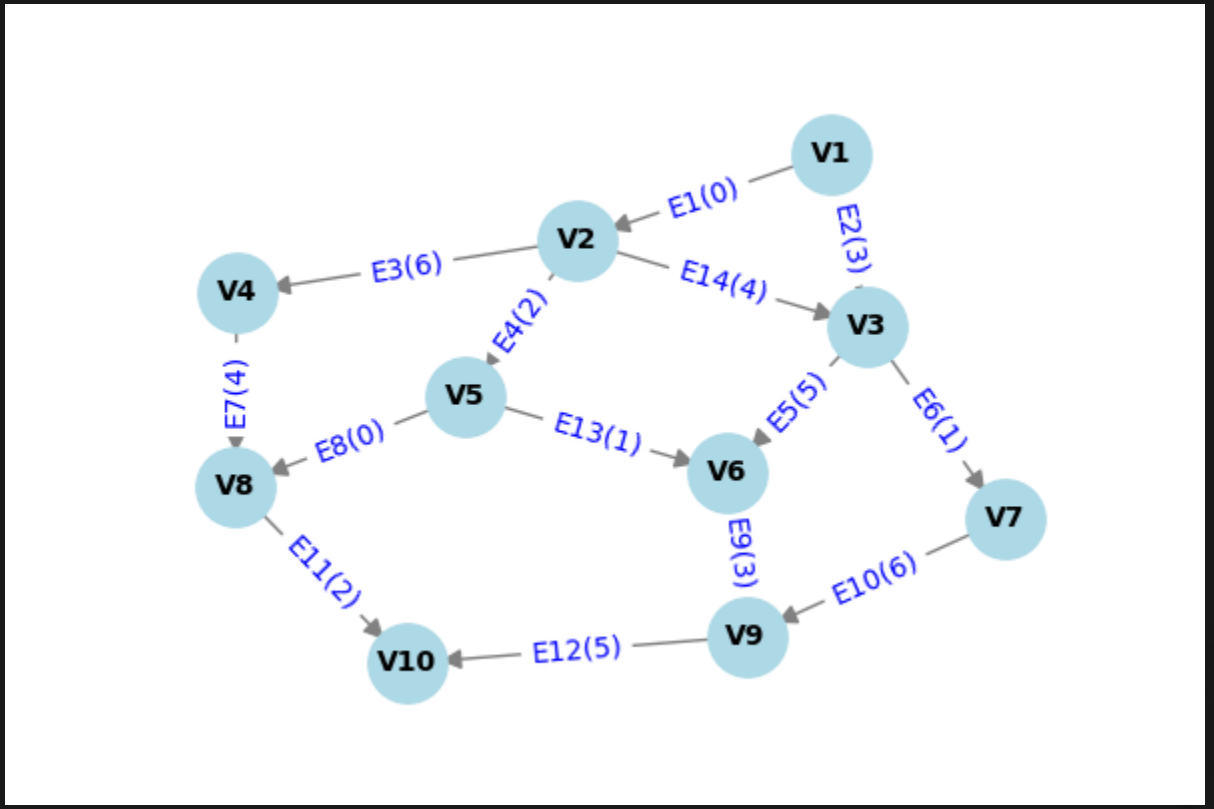
그래프 코드




```python
import networkx as nx

G = nx.Graph()
nodes = ['A', 'B', 'C', 'D']
edges = [
    ('A', 'B'), ('A', 'C'),
    ('B', 'D'), ('C', 'D'),
    ('B', 'C')
]
G.add_nodes_from(nodes)
G.add_edges_from(edges)
```

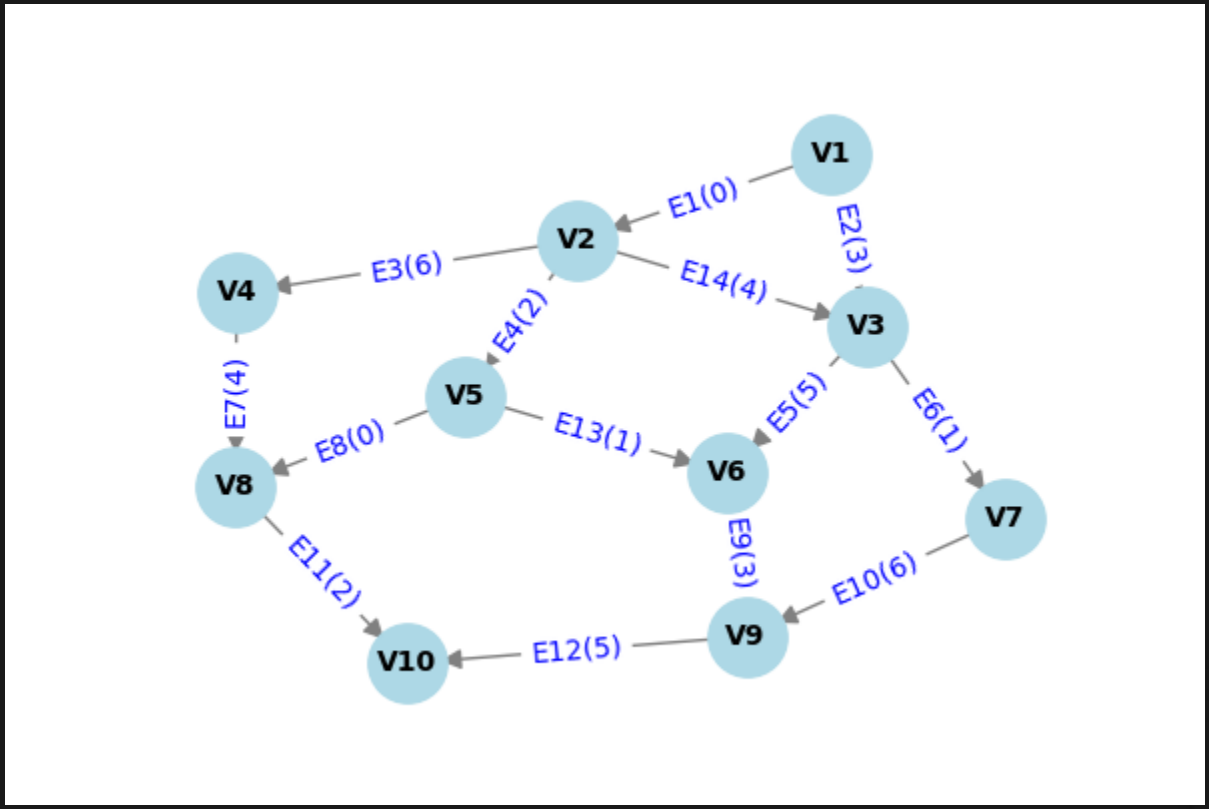
방향성 그래프 (Directed-Graph)

- 정의: Vertex (V) 와 Edge (E) 로 이루어진 순서쌍 (V,E)



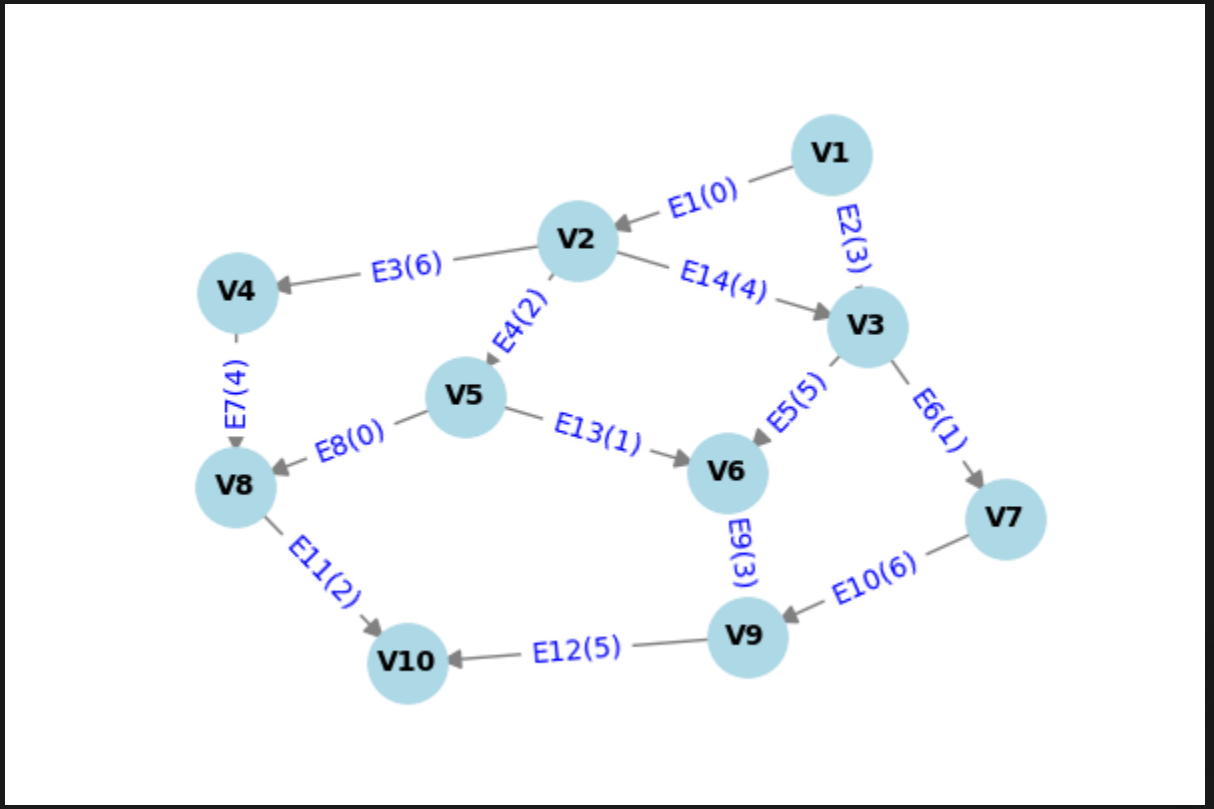
방향성 그래프 (Directed-Graph) - 예시 3

- 정의: Vertex (V) 와 Edge (E) 로 이루어진 순서쌍 (V,E)



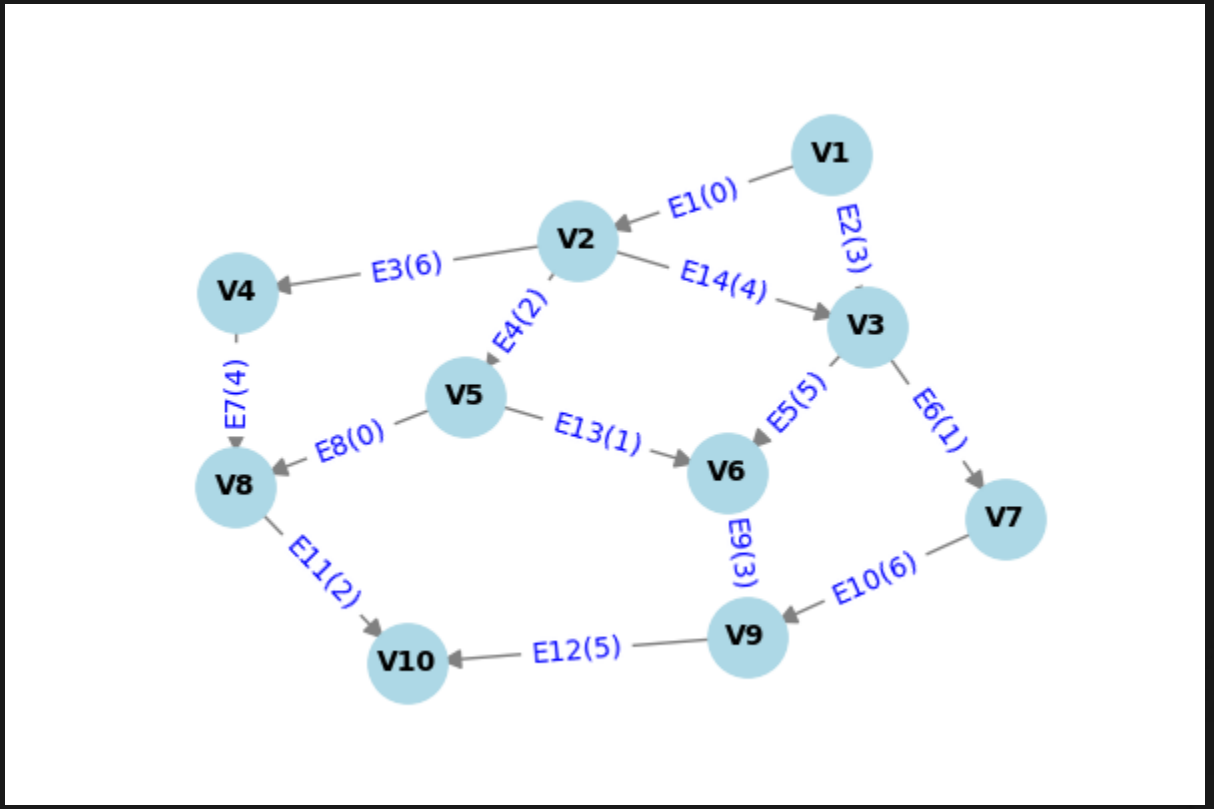
방향성 그래프 (Directed-Graph)

- 정의: Vertex (V) 와 Edge (E) 로 이루어진 순서쌍 (V,E)
- 필수 용어
  - 차수 (degree)
  - 경로 (path)
  - 사이클 (cycle)

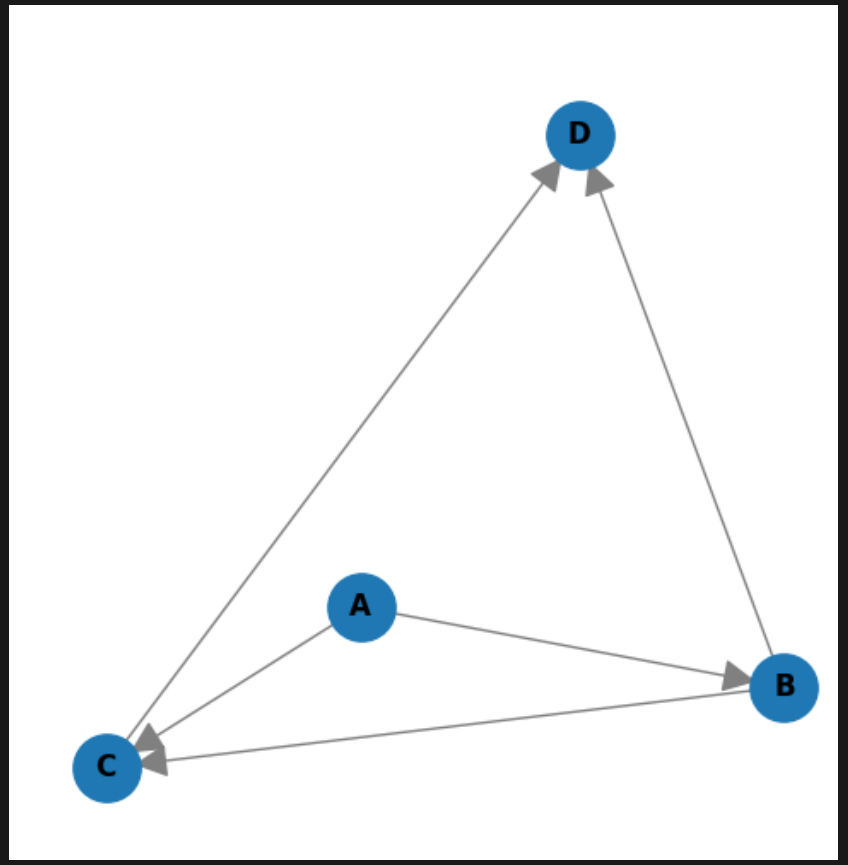


방향성 그래프 (Directed-Graph)

- 정의: Vertex (V) 와 Edge (E) 로 이루어진 순서쌍 (V,E)
- 실제 적용사례로는 수천, 수만개의 vertex 와 edge들로 이루어짐



방향성 그래프의 데이터 구조



Adjacency List

A: ['B', 'C']

B: ['C', 'D']

C: ['D']

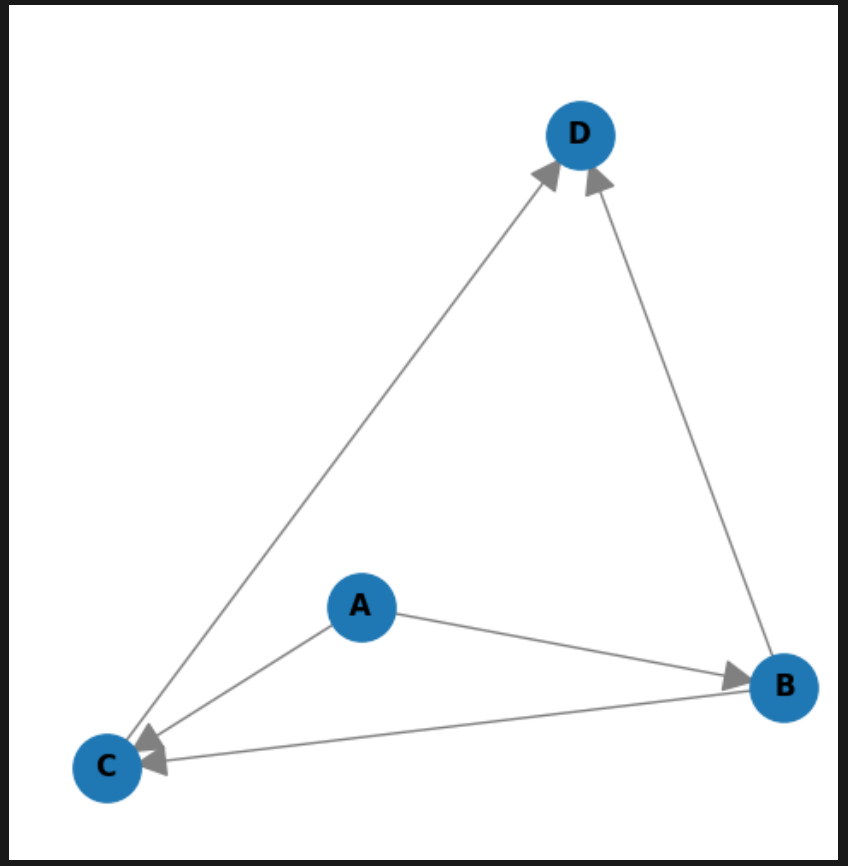
D: []

```
graph_dir = {
    'A': ['B', 'C'],
    'B': ['C', 'D'],
    'C': ['D'],
    'D': []
}
```

Adjacency Matrix

	A	B	C	D
A	0	1	1	0
B	0	0	1	1
C	0	0	0	1
D	0	0	0	0

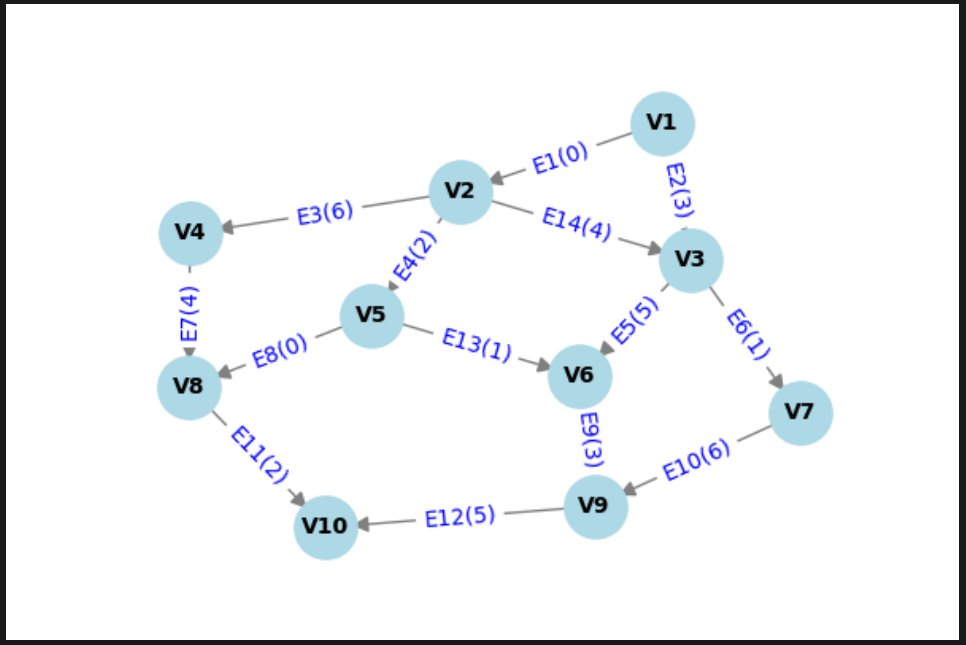
방향성 그래프 코드



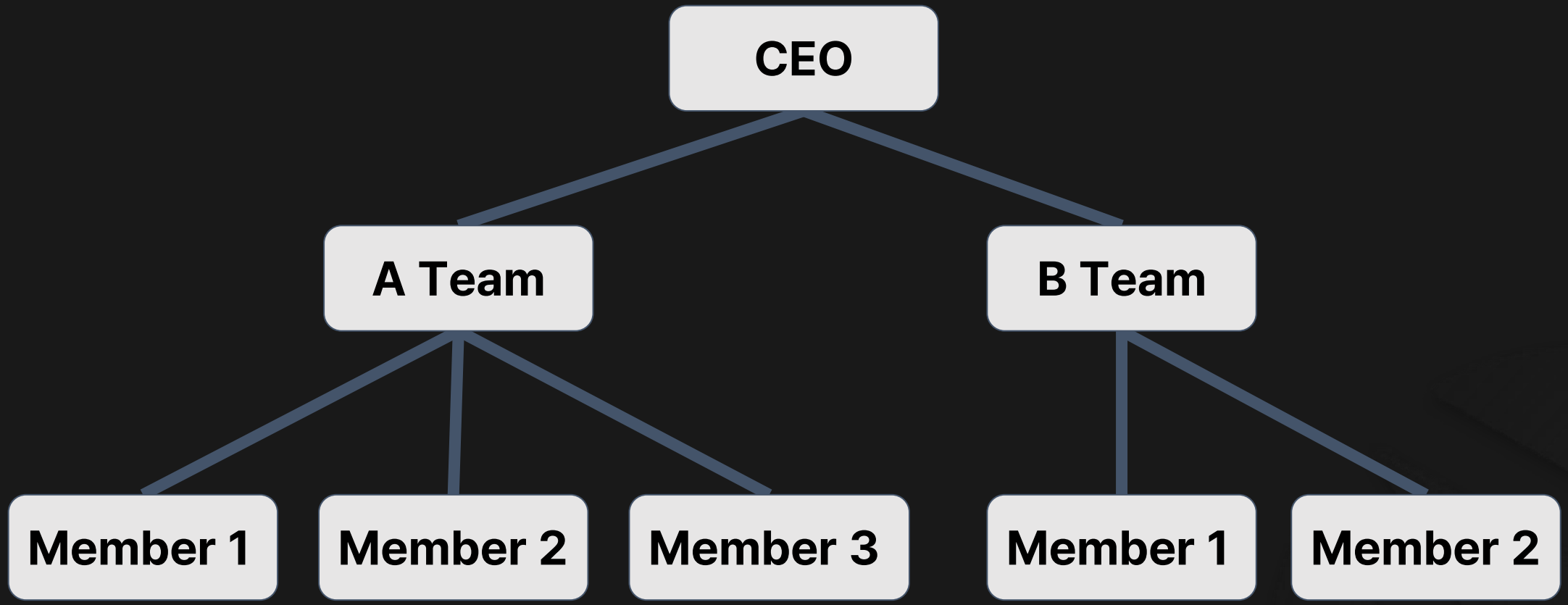

```python
import networkx as nx

G = nx.DiGraph()
nodes = ['A', 'B', 'C', 'D']
edges = [
    ('A', 'B'), ('A', 'C'),
    ('B', 'C'), ('B', 'D'),
    ('C', 'D')
]
G.add_nodes_from(nodes)
G.add_edges_from(edges)
```

그래프 탐색 알고리즘의 필요성

테스크 → 데이터 구조로 문제를 정의 (그래프) → 경로 탐색 → 경로

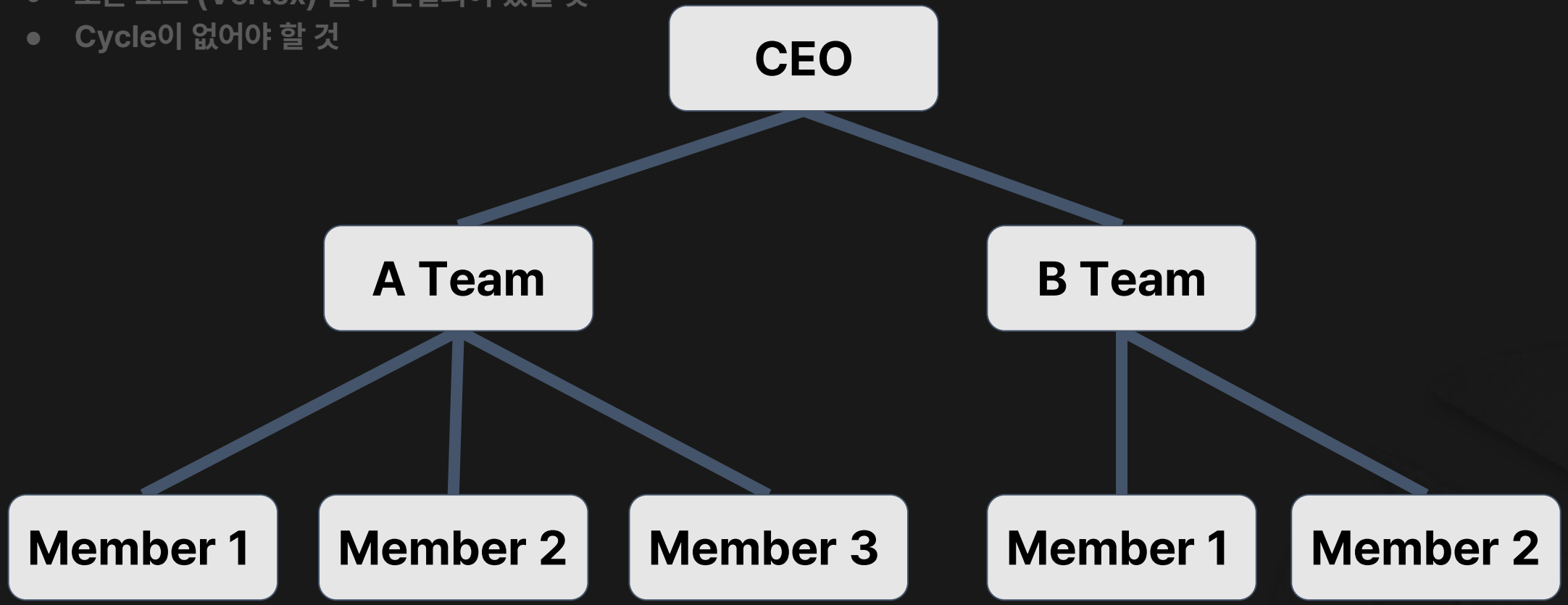


트리 (Tree)



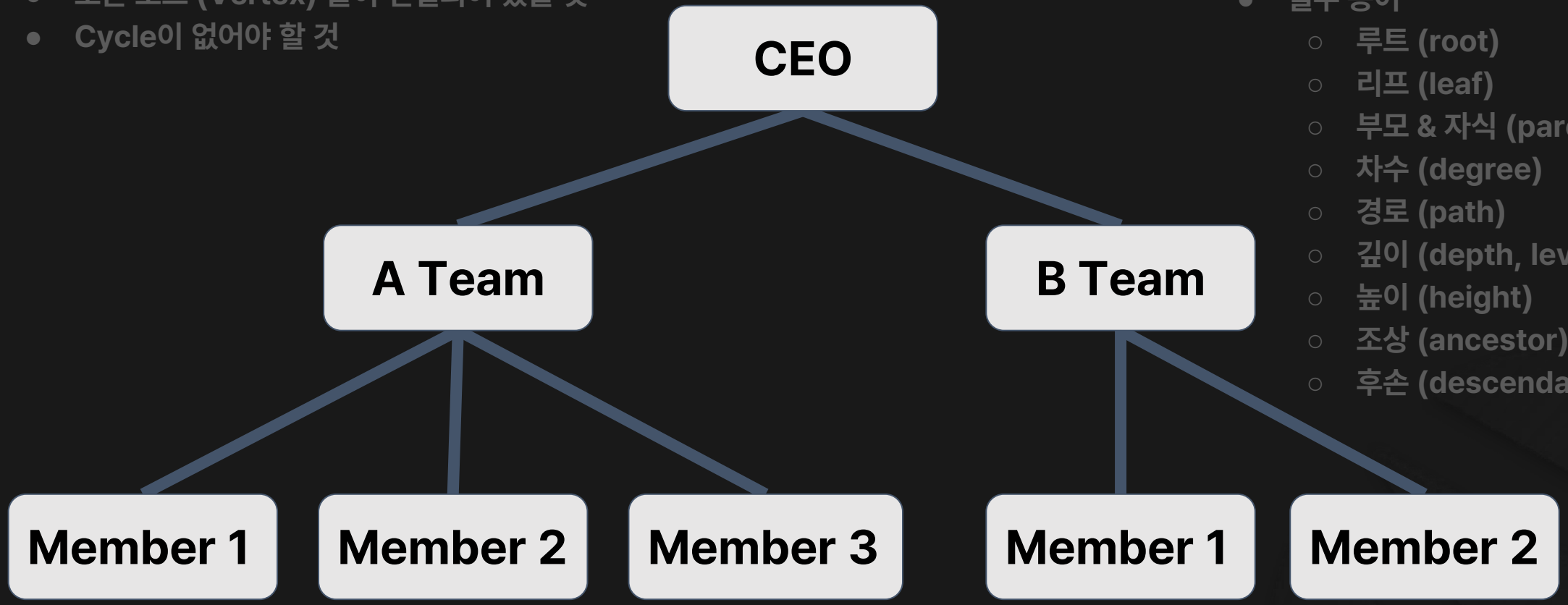
트리 (Tree)

- 모든 노드 (Vertex) 들이 연결되어 있을 것
- Cycle이 없어야 할 것



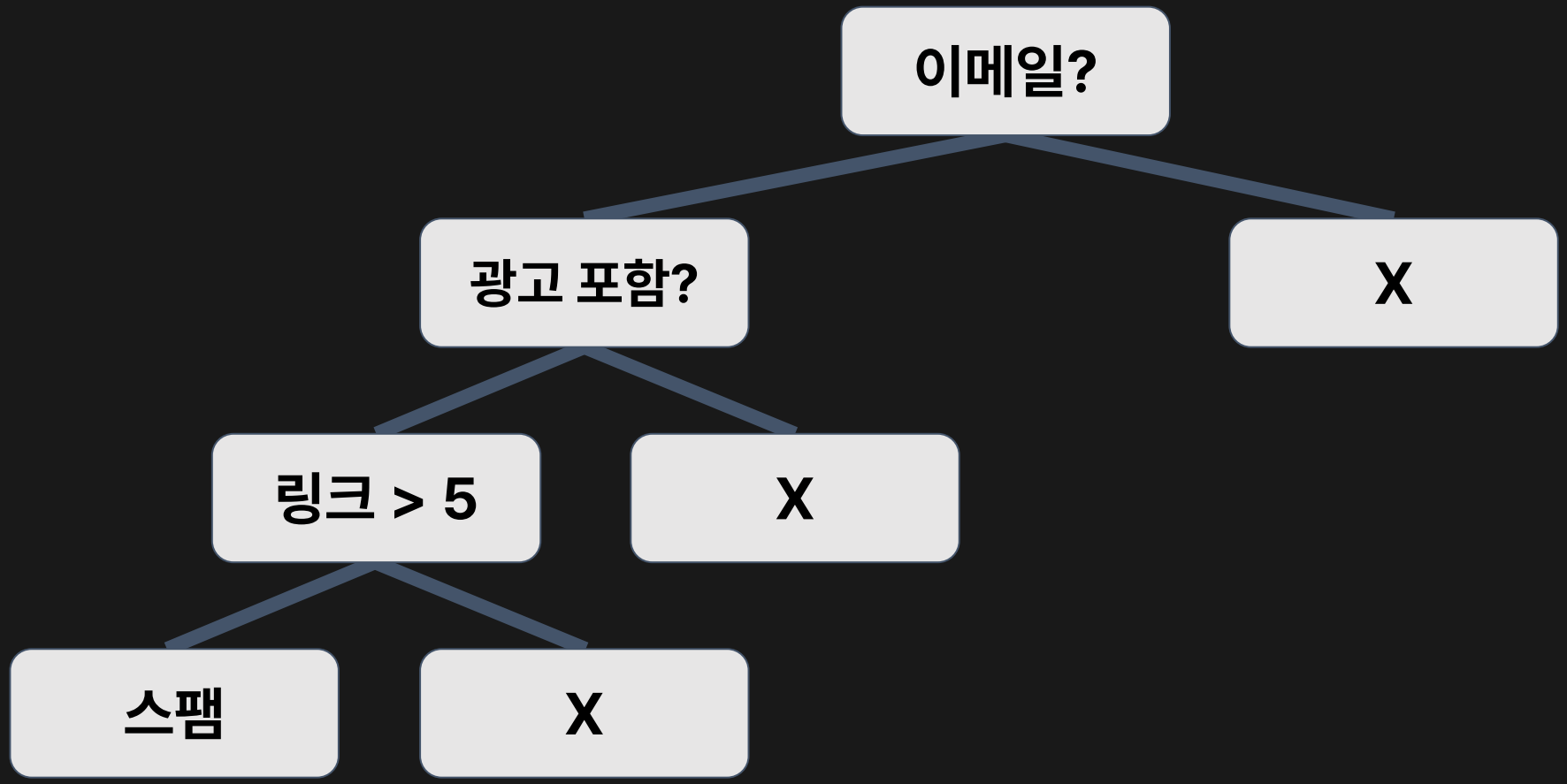
트리 (Tree)

- 모든 노드 (Vertex) 들이 연결되어 있을 것
- Cycle이 없어야 할 것

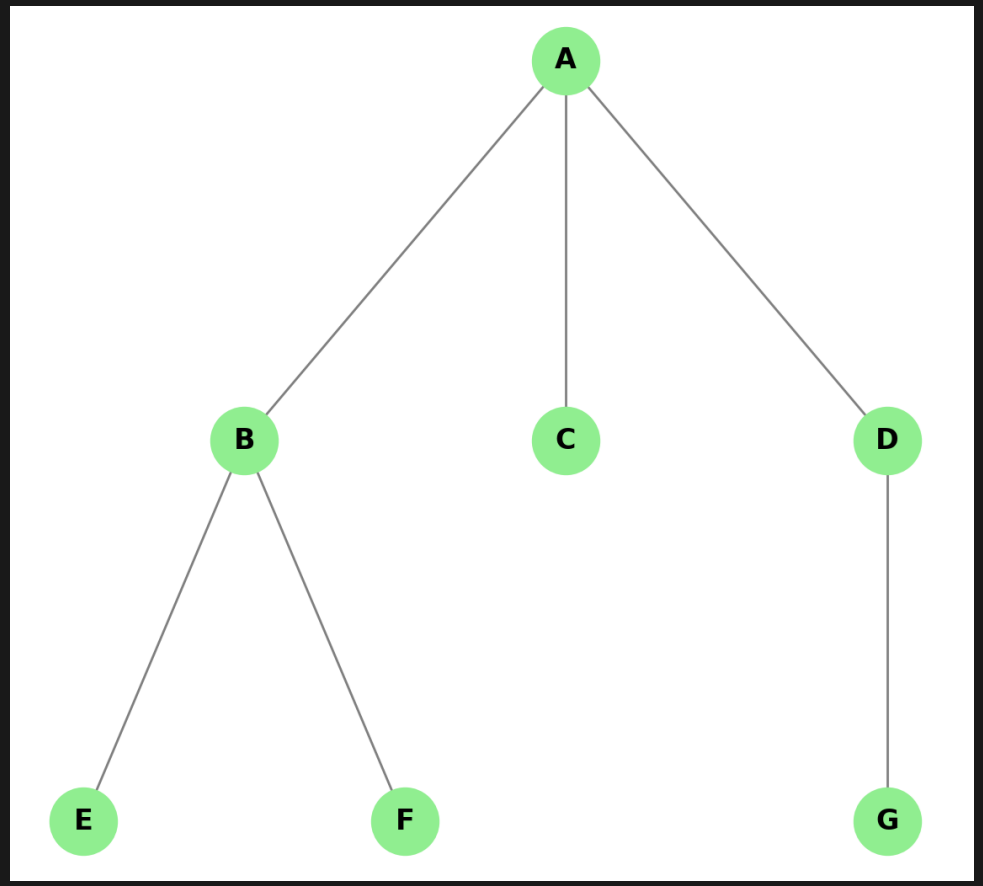


- 필수 용어
  - 루트 (root)
  - 리프 (leaf)
  - 부모 & 자식 (parent & child)
  - 차수 (degree)
  - 경로 (path)
  - 깊이 (depth, level)
  - 높이 (height)
  - 조상 (ancestor)
  - 후손 (descendant)

트리 (Tree) - 예시 (Decision Tree)



트리의 데이터 구조

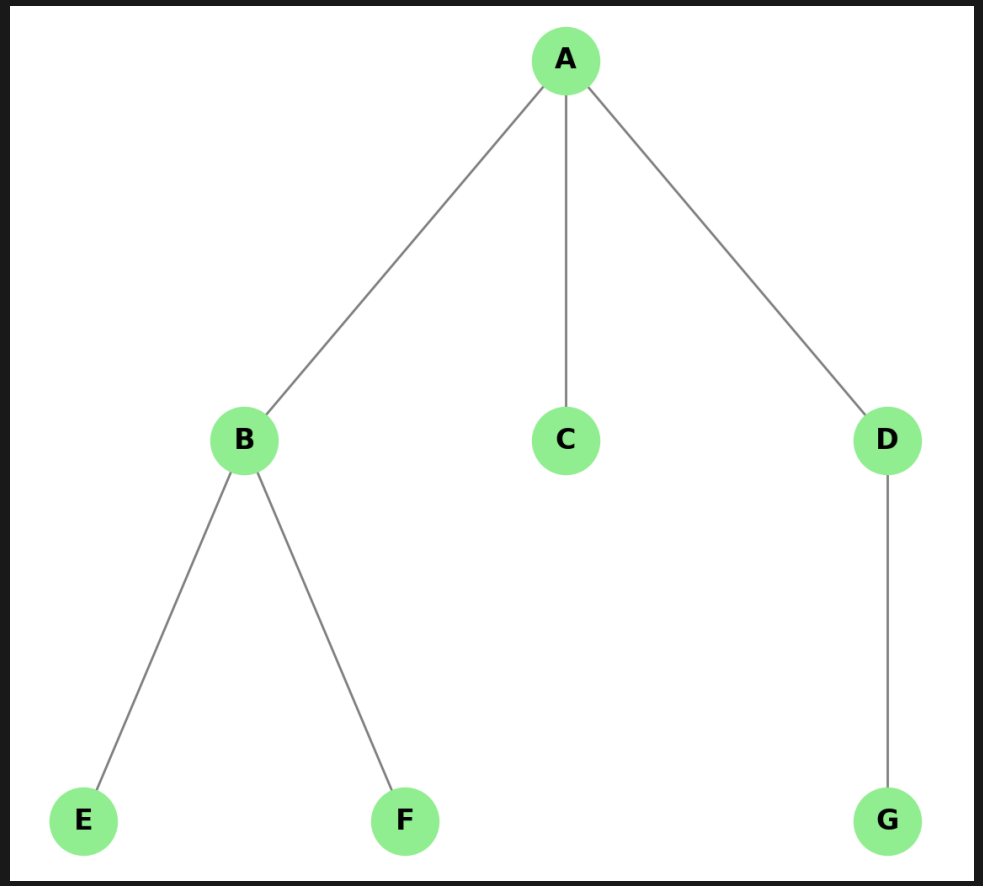


Adjacency List

```
A: ['B', 'C', 'D']

B: ['E', 'F']

C: []

D: ['G']

E: []

F: []

G: []
```

```
tree = {
    'A': ['B', 'C', 'D'],
    'B': ['E', 'F'],
    'C': [],
    'D': ['G'],
    'E': [],
    'F': [],
    'G': []
}
```

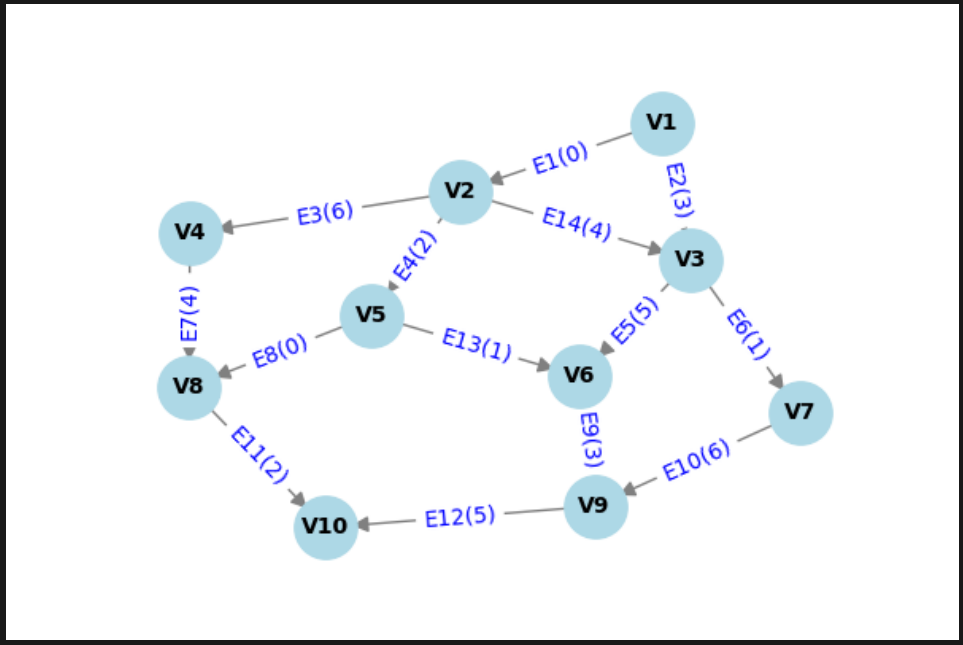
트리의 데이터 구조




```python
import networkx as nx

G = nx.DiGraph()
tree_nodes = ['A', 'B', 'C', 'D', 'E', 'F', 'G']
tree_edges = [
    ('A', 'B'), ('A', 'C'), ('A', 'D'),
    ('B', 'E'), ('B', 'F'),
    ('D', 'G')
]
G.add_nodes_from(tree_nodes)
G.add_edges_from(tree_edges)
```

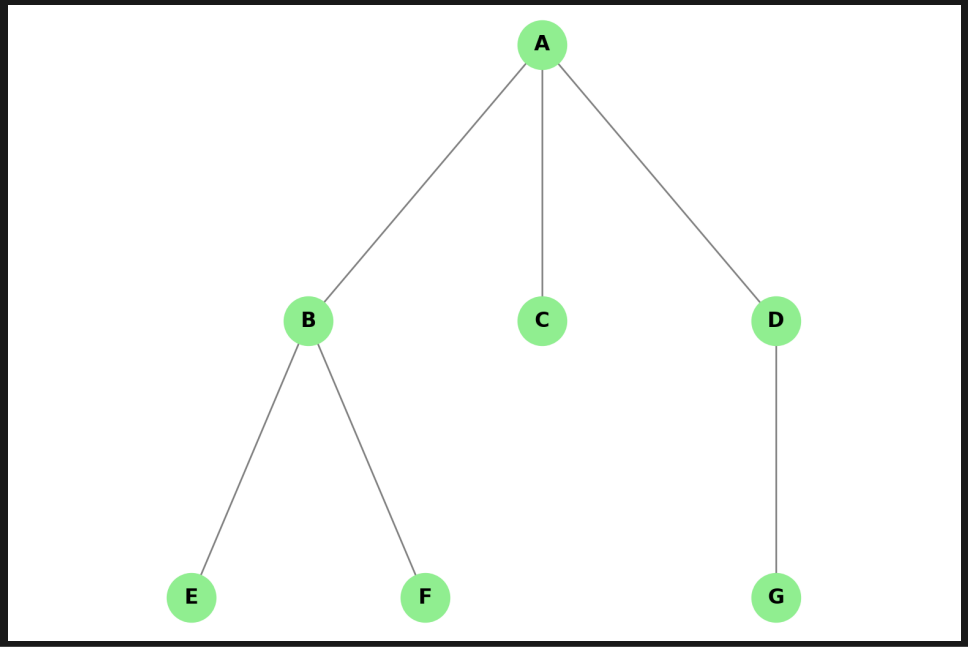
그래프 탐색 알고리즘의 필요성

테스크 → 데이터 구조로 문제를 정의 (그래프) → 경로 탐색 (목적에 따라) → 경로



그래프 탐색 알고리즘의 필요성

테스크 → 데이터 구조로 문제를 정의 (트리) → 경로 탐색 (목적에 따라) → 경로



강의 요약

01

그래프

그래프

방향성 그래프

데이터 구조

02

트리

디시전 트리

데이터 구조

03

서치 알고리즘의 필요성

04

그래프와 트리의 코드 구조