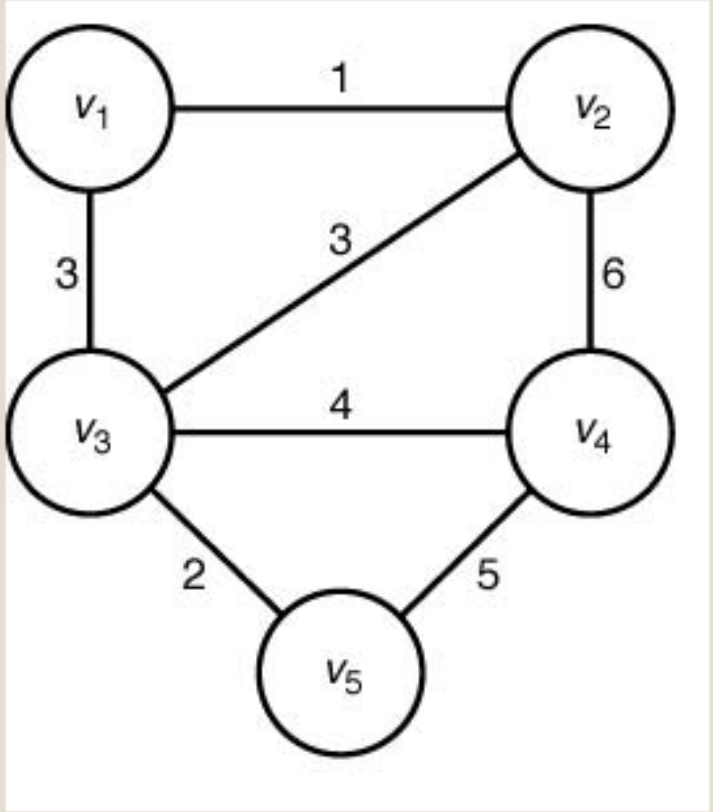
GRAPH ALGORITHM

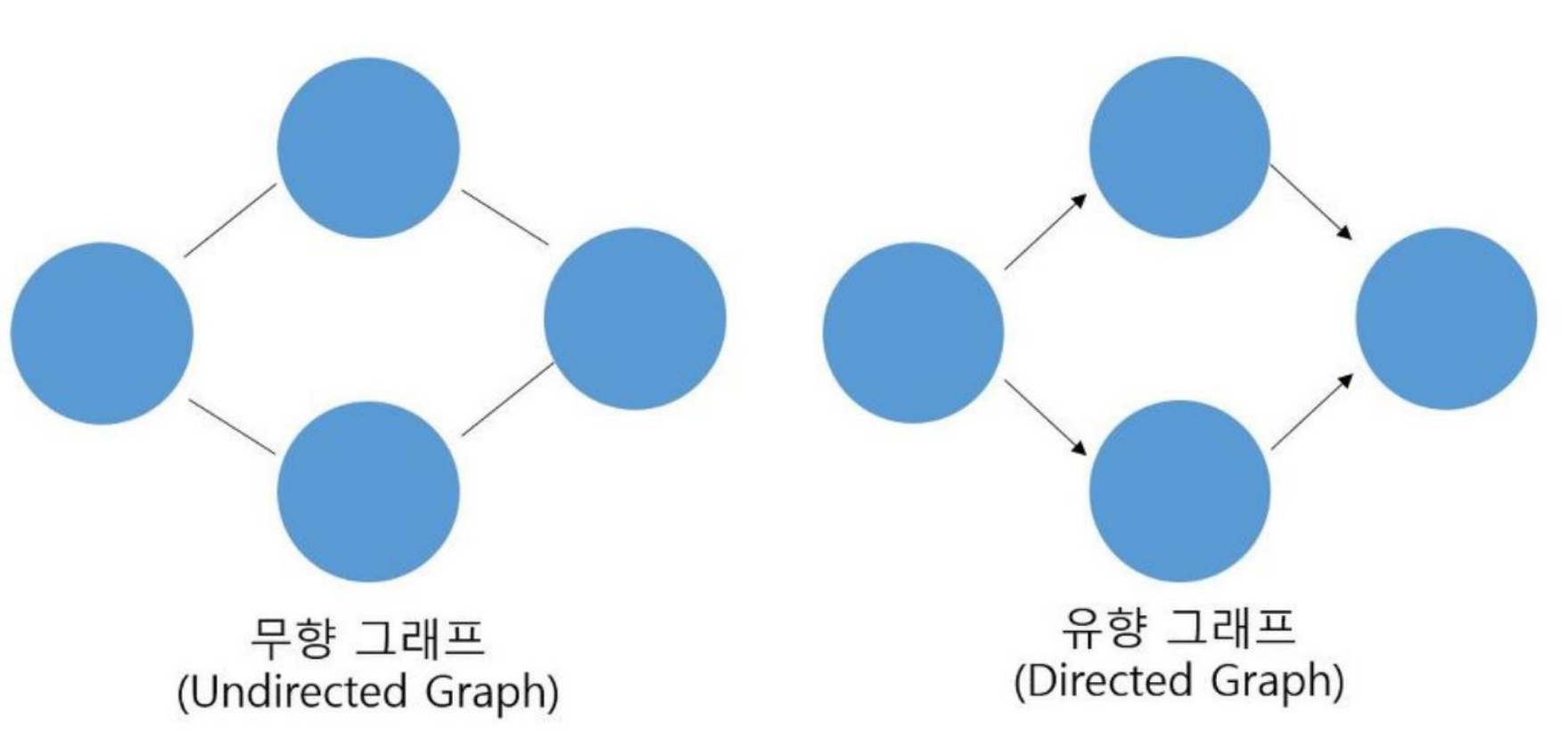
기초 용어

- **Vertex / Node**
- **Edge / Link**
- **Weight**
- Degree
- Path
- Cycle
- Adjacency
- 그리고 특성에 따라 분류되는..

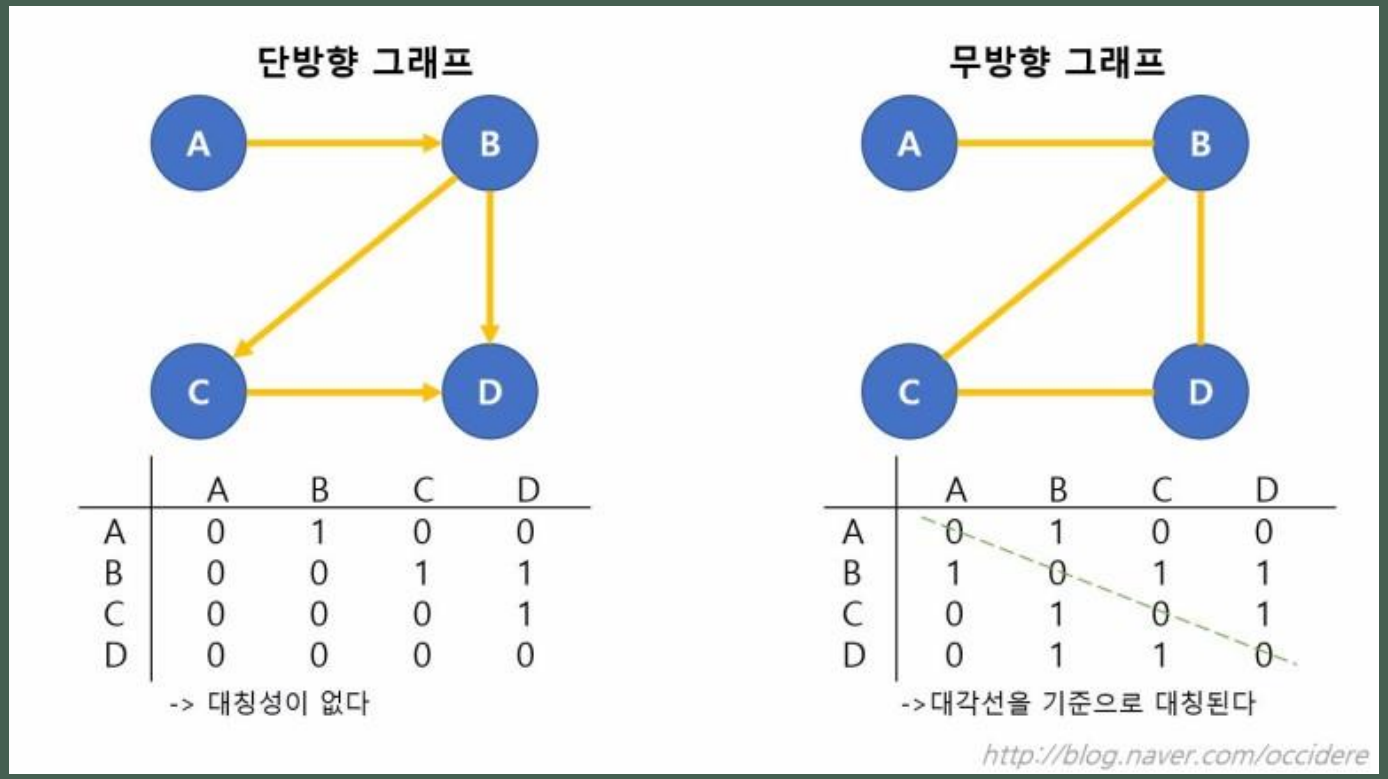
(Connected, Forest, Tree, Dag, Etc..)



종류



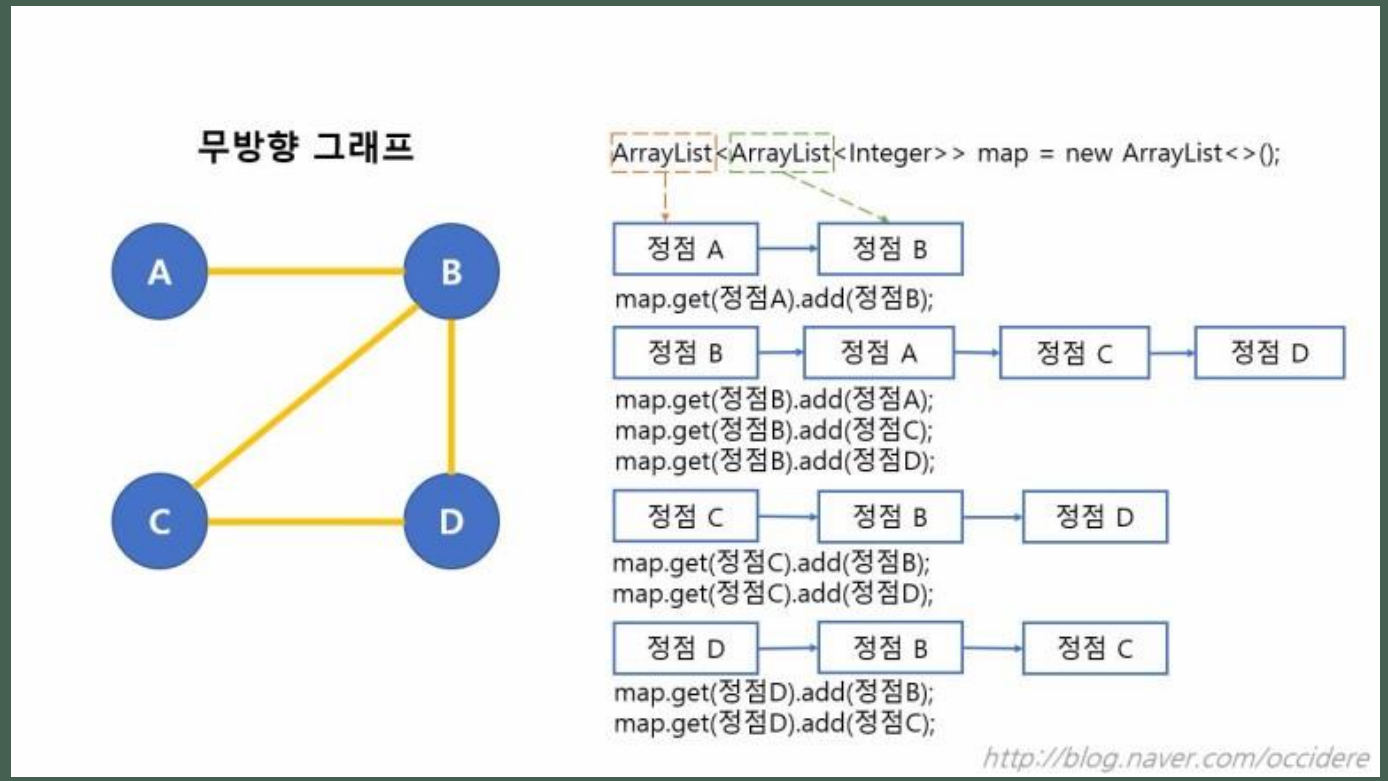
표현 및 구현



- **인접 행렬 (Adjacency Matrix)**

- 장점 : 직관적이며 쉽게 구현 가능
- 단점 : 불필요한 정보의 저장이 많으며, 그래프의 크기가 커지면 메모리 초과가 발생할 수 있음
- 구현 : int형의 2차원 배열을 주로 이용하며, 이동할 수 있으면 1, 없으면 0으로 표기함

표현 및 구현



- **인접 리스트 (Adjacency List)**

- 장점: 필요한 정보만 저장하여 메모리 절약 가능

- 단점: 인접행렬에 비해 다소 어려움

- 구현: 리스트(List)나 벡터(Vector)등의 자료구조를 이용하여 각 정점에서 이동 가능한 정점들을 저장(List나 Vector를 이용한 2차원 배열이라 생각하면 이해하기 쉬움)

탐색 기법

	너비 우선 탐색 (BFS)	깊이 우선 탐색 (DFS)	다익스트라 (Dijkstra)	플로이드 와샬 (Floyd Washall)
탐색 방식	자신과 연결된 주변 정점부터 탐색해 나감	자신과 연결된 정점을 선택해, 그 정점에서 연결된 모든 정점을 파고들어가며 끝날 때 까지 들어가 탐색	간선에 **음의 가중치가 없는** 그래프에서, 어느 한 정점에서 각 정점까지 **최소 가중치를 갖는** 루트 탐색 (BFS + 최단경로 찾기)	간선에 **양,음의 가중치**가 있는 그래프에서, **모든 정점에 대해 각 정점까지 최소 가중치를 갖는 루트 탐색** (모든 정점+최단경로 찾기)
특 징	깊이가 깊은 그래프에 대해 높은 성능	넓이가 넓은 그래프에 대해 높은 성능	비교적 빠르게 최단경로 탐색 가능	모든 정점에서의 최단 경로와, 양, 음수 값의 모든 가중치에 대해 탐색 가능 코드가 단순함
제약 조건	너비가 넓은 그래프에 대해 낮은 성능	깊이가 깊은 그래프에 대해 낮은 성능	한 정점에 대해서만 가능하며, 음수 가중치는 불가	속도가 느림
이용되는 자료구조	큐(Queue)	스택(Stack)	BFS 응용(+ 최소 힙)	3중 for문
시간 복잡도	인접행렬: $O(V^2)$ 인접리스트: O(V+E) (V: 정점의 개수, E: 간선의 개수)		일반: $O(V^2)$ 최소 힙 사용: $O(E+V\log V)$	$O(V^3)$

탐색 유형

- 미로 탐색 유형
  - 현재 좌표를 중심으로 상하좌우로 이동하면서 길을 찾는 유형
  - 주로 인접행렬 형태로 그래프가 주어짐
  - 그래프보다는 지도나 그림 자체로 이해하는 것이 좋음
  - 대표 문제 : https://www.acmicpc.net/problem/2178
- 정적 탐색 유형
  - 주어진 정보로부터 인접행렬/리스트를 생성해서 연결된 길을 찾는 유형
  - 주로 인접 노드로 갈 수 있는지에 대한 정보가 주어짐
  - 특정 노드 방문 기록을 boolean 배열로 체크해 나가는 방식으로 탐색하는 것이 좋음
  - 대표 문제 : https://www.acmicpc.net/problem/1260

활용

- 지하철 A역에서 B역으로 가는 최단 경로
- 최단거리 (다익스트라, 벨만포드)
- 네트워크 플로우,
- 최소 스패닝 트리 (Minimum Spanning Tree / MST)

  프림 알고리즘 / 크루스칼 알고리즘
- BFS/ DFS
- 등등등등 많습니다..