Tree Initialization

알린이월드 – 알형

Tree vs Graph

트리는 그래프의 한 종류입니다.

그래프(Graph):

정점과 간선으로 이루어진 자료구조

방향성이 있거나 없을 수 있음

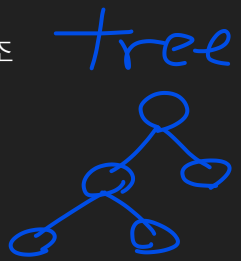
순환 경로 포함 가능

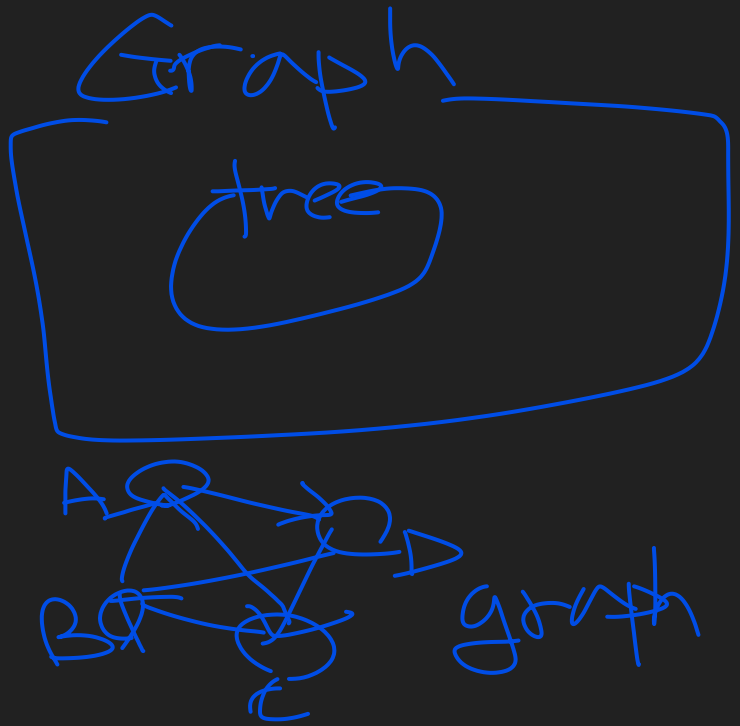
트리(Tree):

그래프의 특수한 형태로, 연결되어 있으면서

사이클(순환 경로)이 없음

즉, 모든 두 정점 사이에 단 하나의 경로만 존재





Sparse Graph vs Dense Graph

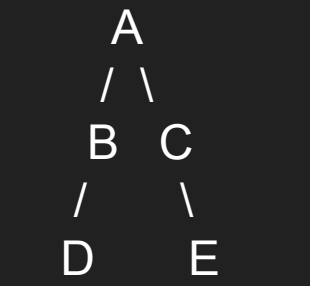
희소 그래프 (Sparse Graph)

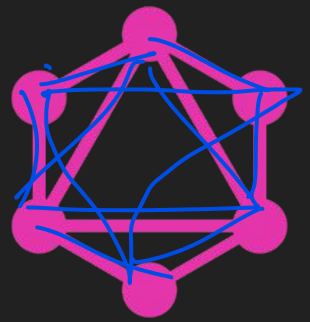
- 간선의 수가 노드 수의 제곱에 비해 매우 적은 그래프
- 대부분의 노드 쌍이 서로 연결되어 있지 않음
- 예시: 트리 구조 (n개의 노드에 대해 n−1개의 간선)
- 공식: 간선 수 $E \approx O(V)$, V는 노드의 수

밀집 그래프 (Dense Graph)

- 간선의 수가 노드 수의 제곱에 가까운 그래프
- 대부분의 노드가 서로 연결되어 있음
- 공식: 간선 수 $E \approx O(V^2)$, V는 노드의 수

[희소 그래프 - 트리]





트리 초기화 차이

구분	Node	인접리스트(Adjacency List)	인접행렬(Adjacency Matrix)
장점	1)노드와 관련된 추가 정보를 함께 저장 가능 2)객체지향적 설계 3)코드가 더 명확하고 유지보수가 쉬움	1)메모리 효율적 (희소 그래프에 적합) 2)자식 노드 추가/삭제가 O(1) 3)특정 노드의 자식들을 순회하기 쉬움	1)두 노드 간 연결 확인이 O(1) 2)구현이 단순하고 직관적
단점	1)메모리 사용량이 더 큼 2)객체 생성/접근 오버헤드	1)특정 두 노드 간 연결 여부 확인이 O(degree) 시간소요	1)공간복잡도가 $O(n^2)$로 비효율적 2)희소 그래프에서 메모리 낭비 3)자식 노드 탐색 시 항상 O(n) 시간 필요

예시	class Node {
	int data;
	Node left;
	Node right;
	public Node(int data) {
	this.data = data;
	this.left = null;
	this.right = null;
	}
	}
		List<Integer>[] graph = new ArrayList[n];	int[][] graph = new int[n][n];