**트리**

트리는 계층적 관계(Hierarchical Relationship)를 표현하는 자료구조이다

트리의 예1)



트리의 예2) –의사결정 트리



트리는 단순한 데이터의 저장을 넘어서 데이터의 표현을 위한 도구

****

**트리 관련 용어**

∙ 노드: node

트리의 구성요소에 해당하는 A, B, C, D, E, F와 같은 요소

∙ 간선: edge

노드와 노드를 연결하는 연결선

∙ 루트 노드: root node

트리 구조에서 최상위에 존재하는 A와 같은 노드

∙ 단말 노드: terminal node

아래로 또 다른 노드가 연결되어 있지 않은 E, F, C, D와 같은 노드

∙ 내부 노드: internal node

단말 노드를 제외한 모든 노드로 A, B와 같은 노드

트리의 노드갂 관계

∙ 노드 A는 노드 B, C, D의 부모 노드(parent node)이다.

∙ 노드 B, C, D는 노드 A의 자식 노드(child node)이다.

∙ 노드 B, C, D는 부모 노드가 같으므로, 서로가 서로에게 형제 노드(sibling node)이다.



하나의 트리를 구성하는 왼쪽과 오른쪽의 작은 트리를 가리켜 서브 트리라 한다.

\*서브 트리 역시 서브 트리로 이뤄져 있으며, 그 서브 트리 역시 또 다른 서브 트리로 이뤄져 있다. 이렇듯 트리는 그 구조가 재귀적이다



서브 트리 역시 또 다른 서브 트리를 갖는다

 

공집합(empty set)도 이진 트리에서는 노드로 간주한다

레벨과 높이, 그리고 포화, 완전 이진 트리



\*트리의 높이와 레벨의 최대 값은 같다



모든 레벨에 노드가 꽉 찬 포화 이진 트리



빈 틈 없이 차곡차곡 채워진 완전 이진 트리

완전 이진 트리는 위에서 아래로 왼쪽에서 오른쪽으로 채워진 트리를 의미한다.

따라서 포화 이진 트리는 동시에 완전 이진 트리이지만 그 역은 성립하지 않는다.



∙ 노드에 번호를 부여하고 그 번호에 해당하는 값을

배열의 인덱스 값으로 활용한다.

∙ 편의상 배열의 첫 번째 요소는 사용하지 않는다.



연결 리스트 기반에서는 트리의 구조와 리스트의 연결 구조가 일치한다.

따라서 구현과 관련된 직관적인 이해가 더 좋은 편이다.

∙ BTreeNode \* MakeBTreeNode(void); // 노드의 생성

∙ BTData GetData(BTreeNode \* bt); // 노드에 저장된 데이터를 반환

∙ void SetData(BTreeNode \* bt, BTData data); // 노드에 데이터를 저장

∙ BTreeNode \* GetLeftSubTree(BTreeNode \* bt); // 왼쪽 서브 트리의 주소 값 반환

∙ BTreeNode \* GetRightSubTree(BTreeNode \* bt); // 오른쪽 서브 트리의 주소 값 반환

∙ void MakeLeftSubTree(BTreeNode \* main, BTreeNode \* sub); // main의 서브 왼쪽 서브 트리로 sub를 연결

∙ void MakeRightSubTree(BTreeNode \* main, BTreeNode \* sub); //main의 오른쪽 서브 트리로 sub를 연결