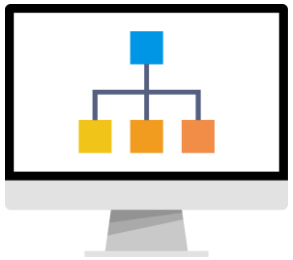


Data Structure & Algorithm
자료구조 및 알고리즘

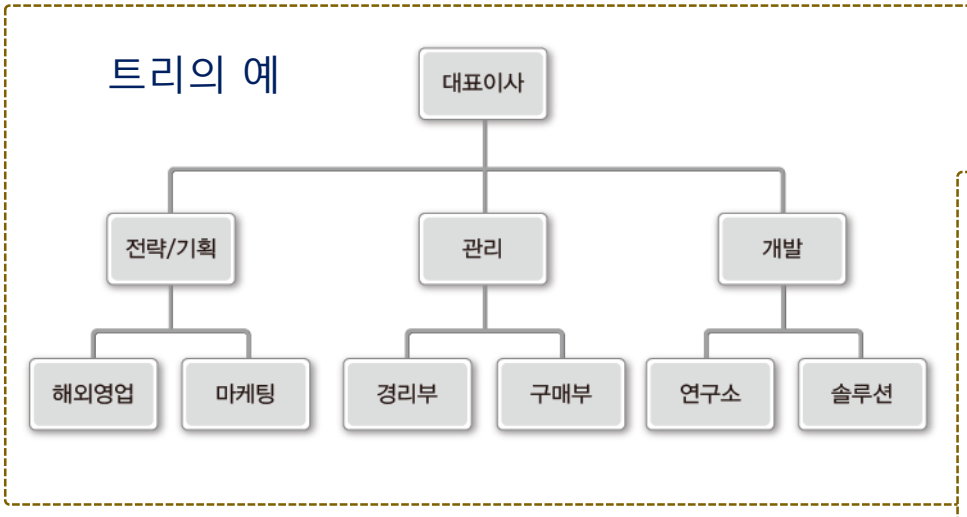
13. 트리 (Tree, Part 1)

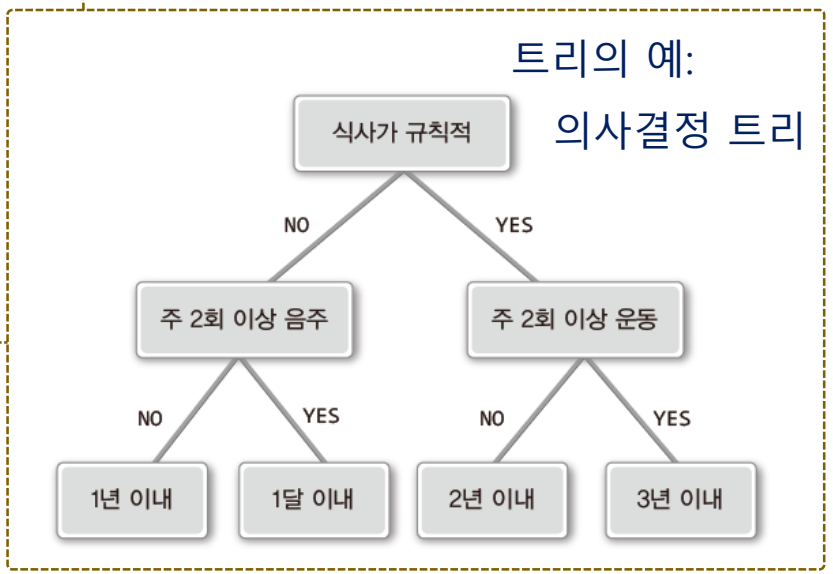
11강 보고서 돌아보기

- 입력으로 주어진 수식에 세 종류의 괄호만 있다고 하자. 괄호의 짝이 맞는지 어떻게 알 수 있을까?
  - (): 맞음
  - {[]()}: 맞음
  - {[](}: 틀림
  - [([])]: 틀림

트리의 접근과 이해

"트리는 계층적 관계(Hierarchical Relationship)를 표현하는 자료구조이다."





트리는 단순한 데이터의 저장을 넘어서
데이터의 표현을 위한 도구이다!

트리 관련 용어의 소개

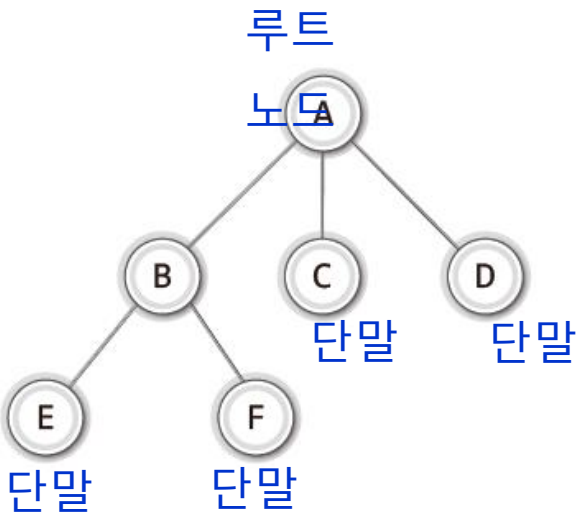
- 노드: node
  트리의 구성요소에 해당하는 A, B, C, D, E, F와 같은 요소

- 간선: edge
  노드와 노드를 연결하는 연결선

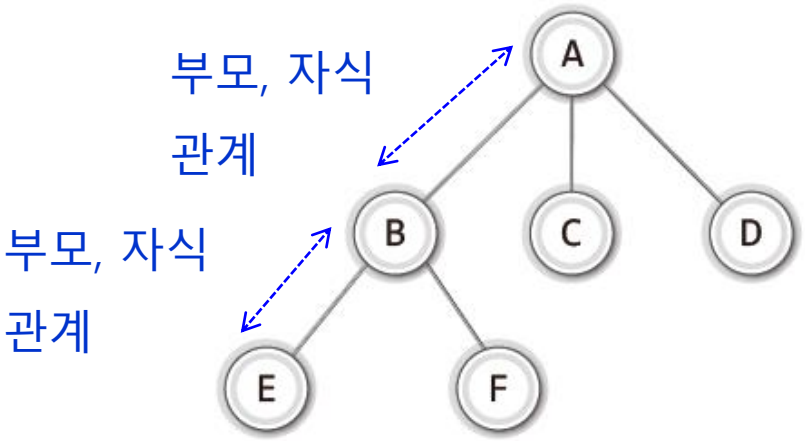
- 루트 노드: root node
  트리 구조에서 최상위에 존재하는 A와 같은 노드

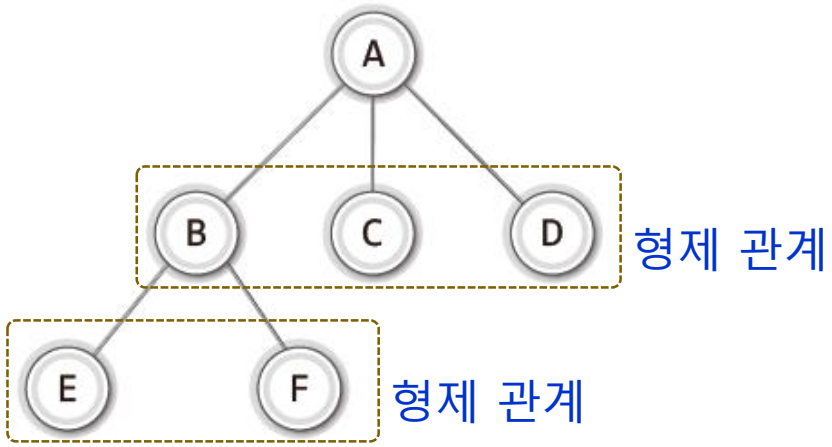
- 단말 노드, 말단 노드: terminal node
  아래로 또 다른 노드가 연결되어 있지 않은 E, F, C, D와 같은 노드

- 내부 노드: internal node
  단말 노드를 제외한 모든 노드로 A, B와 같은 노드



트리의 노드간 관계

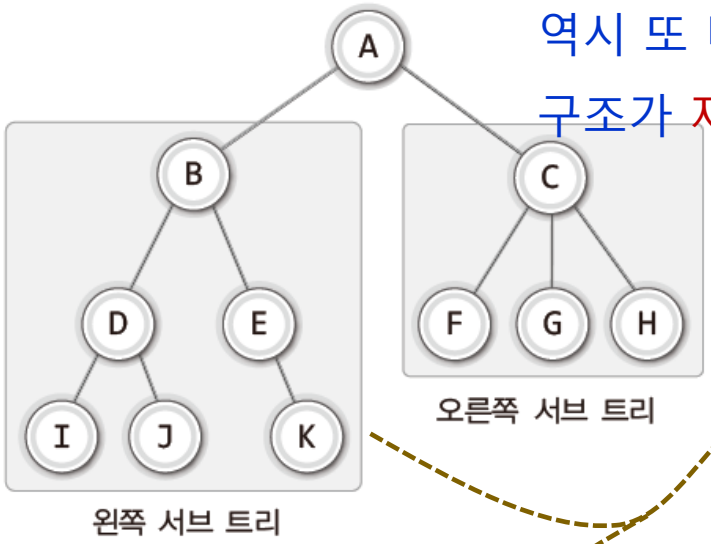




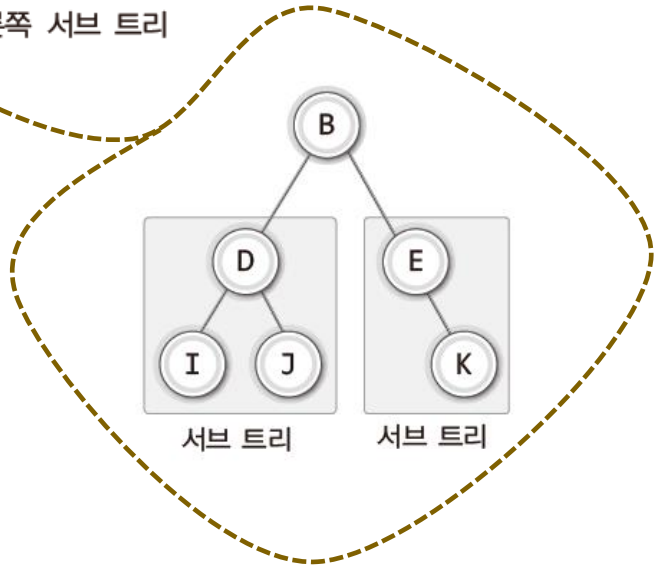
- 노드 A는 노드 B, C, D의 부모 노드(parent node)이다.
- 노드 B, C, D는 노드 A의 자식 노드(child node)이다.
- 노드 B, C, D는 부모 노드가 같으므로, 서로가 서로에게 형제 노드(sibling node)이다.
- 노드 A, B는 노드 E의 선조 노드(ancestor)이다.
- 노드 B, C, D, E, F는 노드 A의 자손 노드(descendant)이다.

서브 트리의 이해

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



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

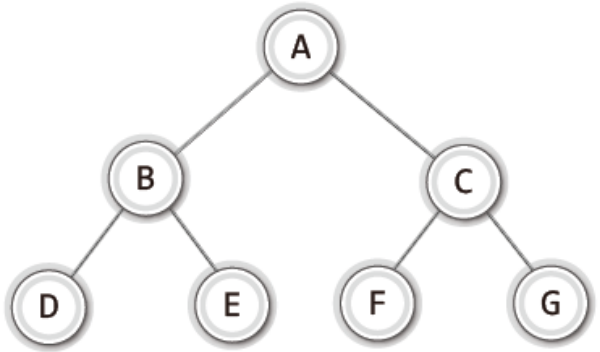


서브 트리 역시 또 다른 서브 트리를 갖는다!

이진 트리의 이해

이진 트리의 조건

- 루트 노드를 중심으로 두 개의 서브 트리로 나뉘어진다.
- 나뉘어진 두 서브 트리도 모두 이진 트리이어야 한다.



이진 트리의 예

"이진 트리가 되려면, 루트 노드를 중심으로 둘로 나뉘는 두 개의 서브 트리도 이진 트리이어야 하고, 그 서브 트리의 모든 서브 트리도 이진 트리이어야 한다."

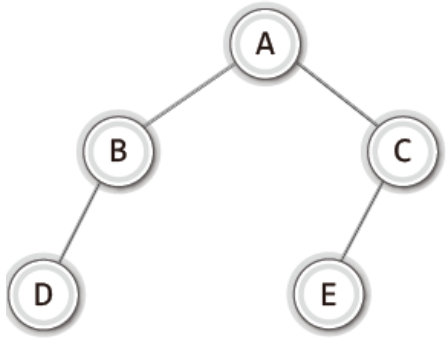
재귀적인 성향을 담아내지 못한 완전하지 않은 표현

트리 그리고 이진 트리는 그 구조가 재귀적이다!

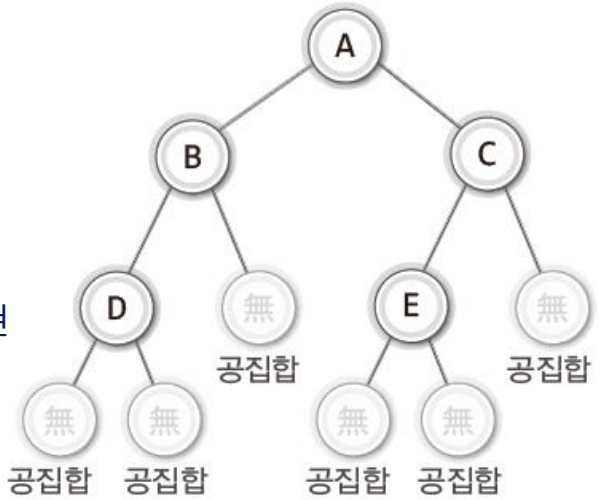
따라서 트리와 관련된 연산은 재귀적으로 사고하고 재귀적으로 구현할 필요가 있다!

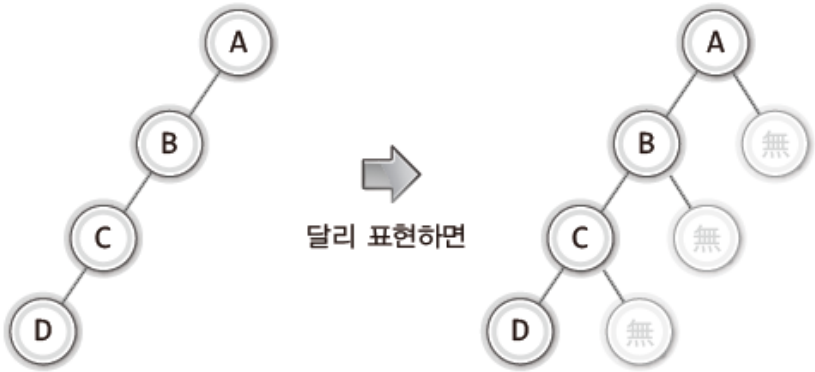
이진 트리와 공집합 노드

공집합(empty set)도 이진 트리에서는 노드로 간주한다!



다른 표현

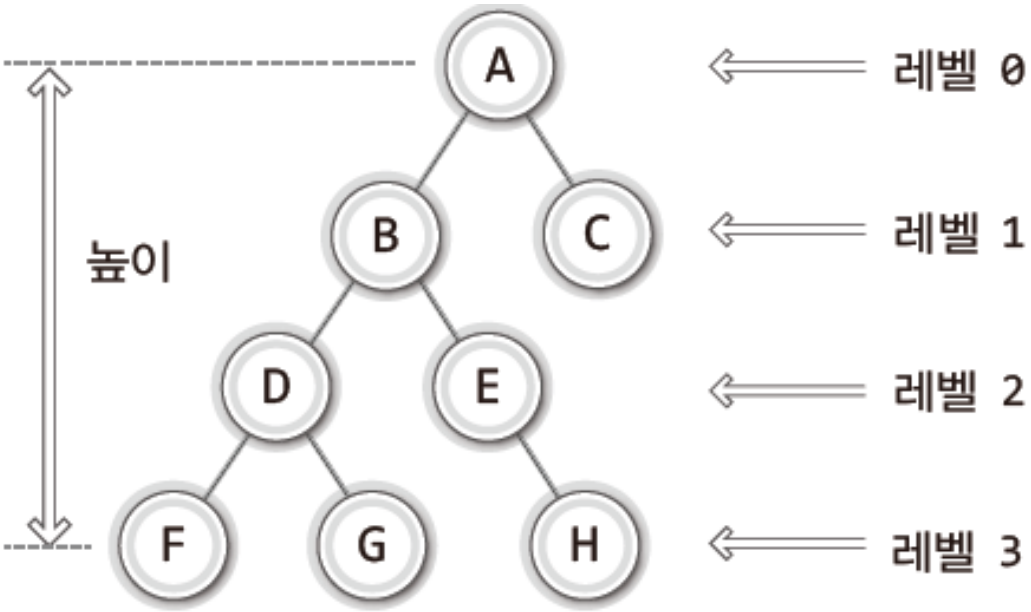




하나의 노드에 두 개의 노드가 달리는 형태의 트리는 모두 이진 트리이다.

크기, 레벨, 그리고 높이

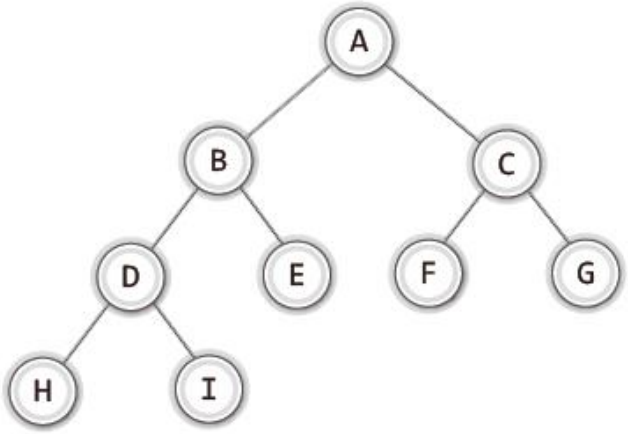
높이는 3!



트리의 높이와 레벨의 최대 값은 같다!

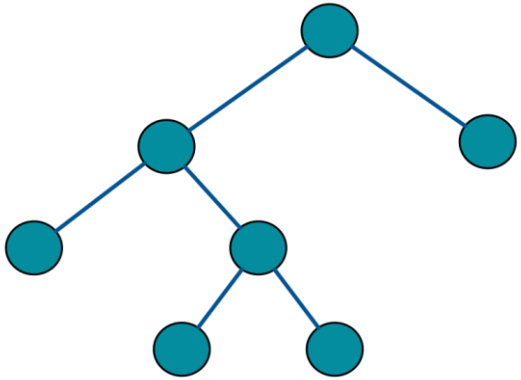
크기 = 노드의 수

이진 트리의 비교

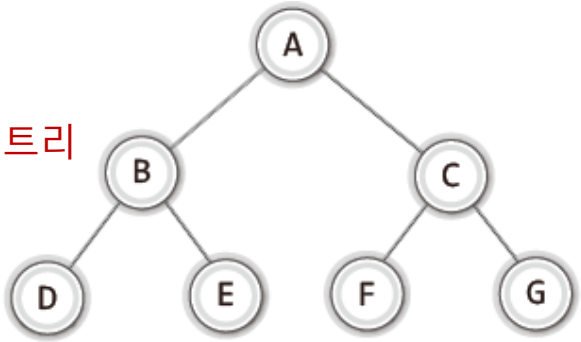


Complete Binary Tree
완전 이진 트리

빈 틈 없이 차곡차곡 채워진! 완전 이진 트리



Full Binary Tree
정 이진 트리



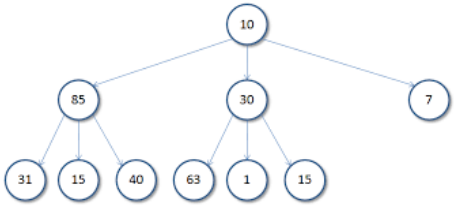
Perfect Binary Tree
포화 이진 트리

모든 레벨에 노드가 꽉 찬! 포화 이진 트리

이진 트리의 관계

이진 트리 (binary tree)

삼진 트리 (ternary tree)



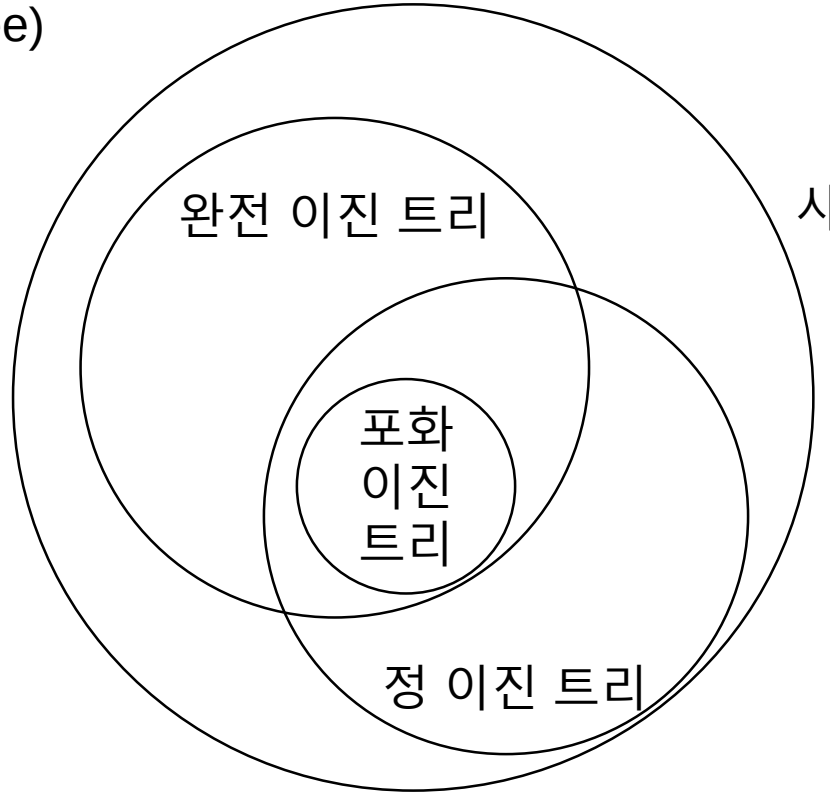
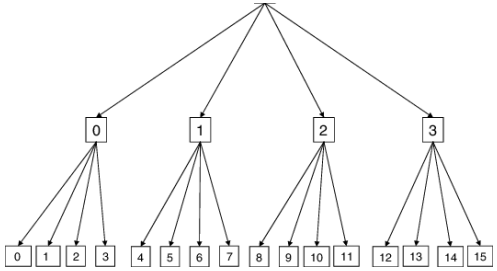
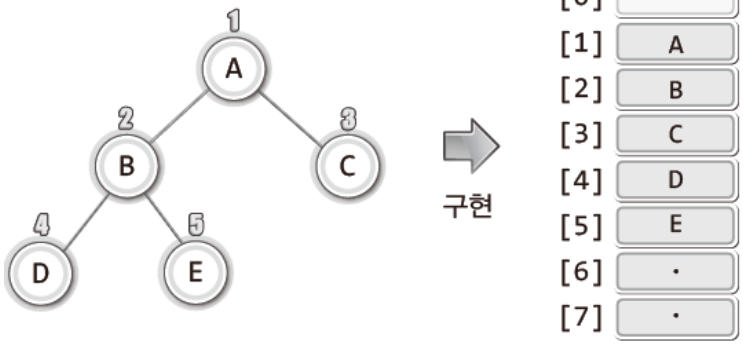


사진 트리 (quaternary tree)



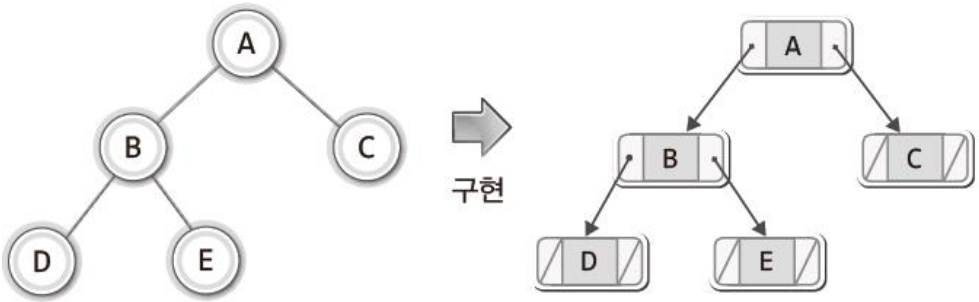
이진 트리의 구현

이진 트리 구현의 두 가지 방법



- 노드에 번호를 부여하고 그 번호에 해당하는 값을 배열의 인덱스 값으로 활용한다.
- 편의상 배열의 첫 번째 요소는 사용하지 않는다.

배열의 기본적인 장점! 접근이 용이하다라는 특성이 트리에서도 그대로 반영이 된다!
뿐만 아니라 배열을 기반으로 했을 때 완성하기 용이한 트리 관련 연산도 존재한다.



연결 리스트 기반에서는 트리의 구조와 리스트의 연결 구조가 일치한다.
따라서 구현과 관련된 직관적인 이해가 더 좋은 편이다.

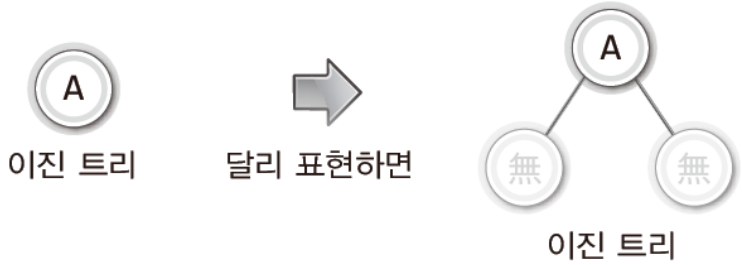
헤더파일에 정의된 구조체의 이해

```
// 이진 트리의 노드를 표현한 구조체
typedef struct _bTreeNode
{
    BTData data;
    struct _bTreeNode * left;
    struct _bTreeNode * right;
} BTreeNode;
```

이것이 노드이자 이진 트리를 표현한 구조체의 정의이다!

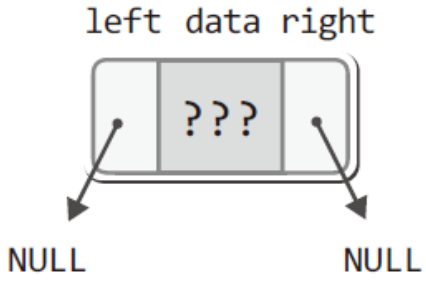
이진 트리의 모든 노드는 직/간접적으로 연결되어 있다. 따라서 루트 노드의 주소 값만 기억하면, 이진 트리 전체를 가리키는 것과 다름이 없다.

논리적으로도 하나의 노드는 그 자체로 이진 트리이다. 따라서 노드를 표현한 구조체는 실제로 이진 트리를 표현한 구조체가 된다.



헤더파일에 선언된 함수들1

- BTreeNode * MakeBTreeNode(void); // 노드의 생성



이러한 형태의 노드를 동적으로 할당하여 생성한다!
유효한 데이터는 SetData 함수를 통해서 채우되
포인터 변수 left와 right는 NULL로 자동 초기화 된다.

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

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

노드에 직접 접근하는 것보다. 함수를 통한 접근이 보다 칭찬받을 수 있는 구조!

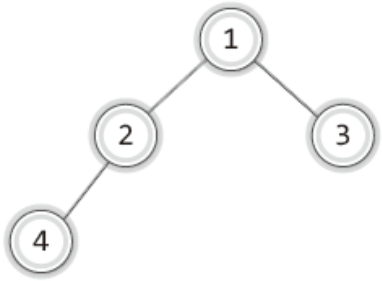
헤더파일에 선언된 함수들2

- BTreeNode * GetLeftSubTree(BTreeNode * bt);

  왼쪽 서브 트리의 주소 값 반환!

- BTreeNode * GetRightSubTree(BTreeNode * bt);

  오른쪽 서브 트리의 주소 값 반환!



√ 루트 노드를 포함하여 어떠한 노드의 주소 값도 인자로 전달될 수 있다.

√ 전달된 노드의 왼쪽, 오른쪽 '서브 트리의 루트 노드 주소 값' 또는 그냥 '노드의 주소 값'이 반환된다.

- 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!

하나의 노드도 일종의 이진 트리이다! 따라서 위와 같이 함수를 이름을 짓는 것이 타당하다!

위의 함수들은 단순히 노드가 아니라 트리를 대상으로도 그 결과를 보인다는 사실을 기억하자!

정의된 함수들의 이해를 돕는 main 함수

```
int main(void)
{
    BTreeNode * ndA = MakeBTreeNode();      // 노드 A 생성

    BTreeNode * ndB = MakeBTreeNode();      // 노드 B 생성

    BTreeNode * ndC = MakeBTreeNode();      // 노드 C 생성

    ndB, ndB, ndC를 이용한 SetData 함수 호출을 통해 유용한 데이터 채운 후...

    // 노드 A의 왼쪽 자식 노드로 노드 B 연결
    MakeLeftSubTree(ndA, ndB);

    // 노드 A의 오른쪽 자식 노드로 노드 C 연결
    MakeRightSubTree(ndA, ndC);

    . . . .
}
```

앞서 정의한 이진 트리 관련 함수들은 이진 트리를 만드는 도구이다!

이진 트리의 구현

```
BTreeNode * MakeBTreeNode(void)
{
    BTreeNode * nd = (BTreeNode*)malloc(sizeof(BTreeNode));
    nd->left = NULL;
    nd->right = NULL;
    return nd;
}

BTData GetData(BTreeNode * bt)
{
    return bt->data;
}

void SetData(BTreeNode * bt, BTData data)
{
    bt->data = data;
}

BTreeNode * GetLeftSubTree(BTreeNode * bt)
{
    return bt->left;
}
```

구현 자체는 크게 어려움이 없다!

```
BTreeNode * GetRightSubTree(BTreeNode * bt)
{
    return bt->right;
}

void MakeLeftSubTree(BTreeNode * main, BTreeNode * sub)
{
    if(main->left != NULL)
        free(main->left);

    main->left = sub;
}

void MakeRightSubTree(BTreeNode * main, BTreeNode * sub)
{
    if(main->right != NULL)
        free(main->right);

    main->right = sub;
}
```

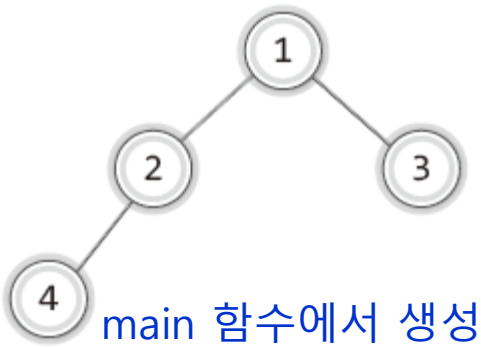
기존에 연결된 노드는 삭제되게 구현!

기존에 연결된 노드는 삭제되게 구현!

이진 트리 관련 main 함수

```
int main(void)
{
    BTreeNode * bt1 = MakeBTreeNode();     // 노드 bt1 생성
    BTreeNode * bt2 = MakeBTreeNode();     // 노드 bt2 생성
    BTreeNode * bt3 = MakeBTreeNode();     // 노드 bt3 생성
    BTreeNode * bt4 = MakeBTreeNode();     // 노드 bt4 생성

    SetData(bt1, 1);    // bt1에 1 저장
    SetData(bt2, 2);    // bt2에 2 저장
    SetData(bt3, 3);    // bt3에 3 저장
    SetData(bt4, 4);    // bt4에 4 저장

    MakeLeftSubTree(bt1, bt2);          // bt2를 bt1의 왼쪽 자식 노드로
    MakeRightSubTree(bt1, bt3);         // bt3를 bt1의 오른쪽 자식 노드로
    MakeLeftSubTree(bt2, bt4);          // bt4를 bt2의 왼쪽 자식 노드로

    // bt1의 왼쪽 자식 노드의 데이터 출력
    printf("%d \n", GetData(GetLeftSubTree(bt1)));

    // bt1의 왼쪽 자식 노드의 왼쪽 자식 노드의 데이터 출력
    printf("%d \n", GetData(GetLeftSubTree(GetLeftSubTree(bt1))));

    return 0;
}
```

트리를 완전히 소멸시키는 방법은? ☞ 순회!



main 함수에서 생성하는 트리

BinaryTree.h
BinaryTree.c
BinaryTreeMain.c

실행결과

```
2
4
```

요약

- 트리는 계층이 있는 자료 구조
  - 노드, 간선, 루트 노드, 단말 노드, 내부 노드, 부모 노드, 자식 노드, 선조 노드, 자손 노드, 형제 노드, 레벨, 높이
  - 레벨은 0부터 시작한다!

- 이진 트리(binary tree): 노드가 최대 두 개의 자식만 가지는 트리
  - 포화 이진 트리(Perfect): 모든 내부 노드가 두개의 자식을 가지는 트리
  - 완전 이진 트리(Complete): 마지막 레벨을 제외하고 모두 채워져 있으며 마지막 레벨의 노드도 가능한 왼쪽에 있는 트리
  - 정 이진 트리(Full): 모든 노드의 자식 노드의 개수가 0 또는 2인 트리

출석 인정을 위한 보고서 작성

- A4 반 장 이상으로 아래 질문에 답한 후 포털에 있는 과제 제출 란에 PDF로 제출

- 노드가 n개인 트리의 간선 수의 최솟값/최댓값은?
- 높이가 n인 포화 이진 트리의 개수는?
- 높이가 n인 포화 이진 트리의 노드 수는?
- 높이가 n인 완전 이진 트리의 개수는?
- 높이가 n인 완전 이진 트리의 내부 노드 수는?
- 높이가 n인 정 이진 트리의 노드의 수의 최솟값/최댓값은?