DATA
STRUCTURES
USING C

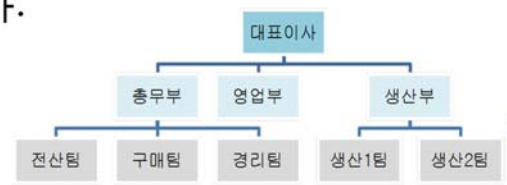
- 트리의 개념과 용어들을 이해한다.
- 이진트리의 표현 방법 2가지를 이해한다.
- 이진트리의 순회방법을 이해한다.
- 이진트리의 연산들을 이해한다.
- 이진트리를 이용한 문제해결 능력을 배양한다.

CHAPTER 08

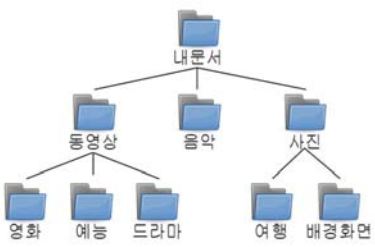
트리

트리(TREE)

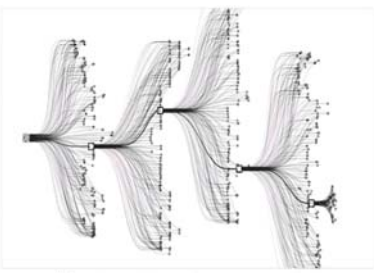
- 트리: 계층적인 구조를 나타내는 자료구조
- 트리는 부모-자식 관계의 노드들로 이루어짐
- 응용분야:



(a) 회사의 조직도



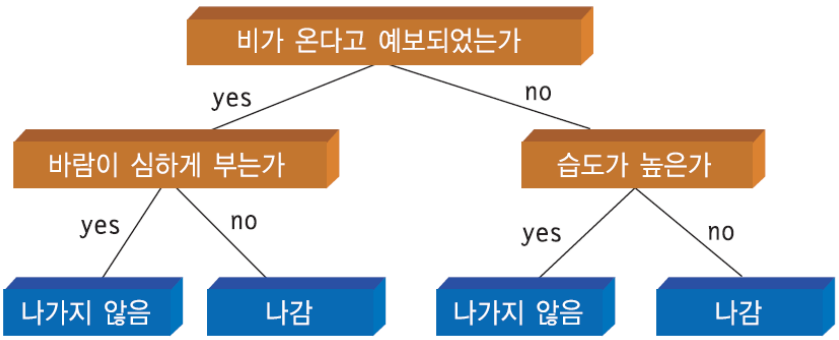
(b) 컴퓨터의 폴더 구조



(c) 인공지능 바둑 프로그램의 거대한 결정 트리(decision tree)

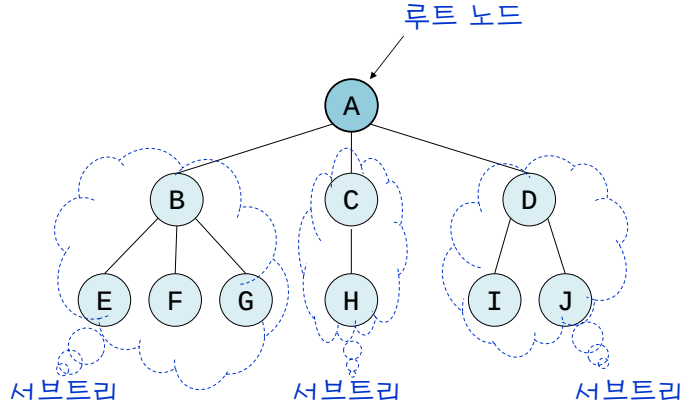
결정 트리

- (예) 등산에 대한 결정 트리



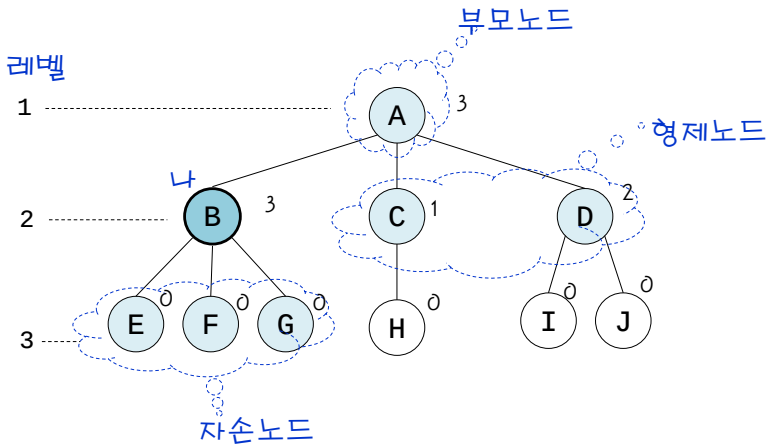
트리의 용어

- 노드(node) : 트리의 구성요소
- 루트(root) : 부모가 없는 노드(A)
- 서브트리(subtree) : 하나의 노드와 자손들로 이루어짐
- 단말노드(terminal) : 자식이 없는 노드(A,B,C,D)
- 비단말노드 : 자식을 가지는 노드(E,F,G,H,I,J)

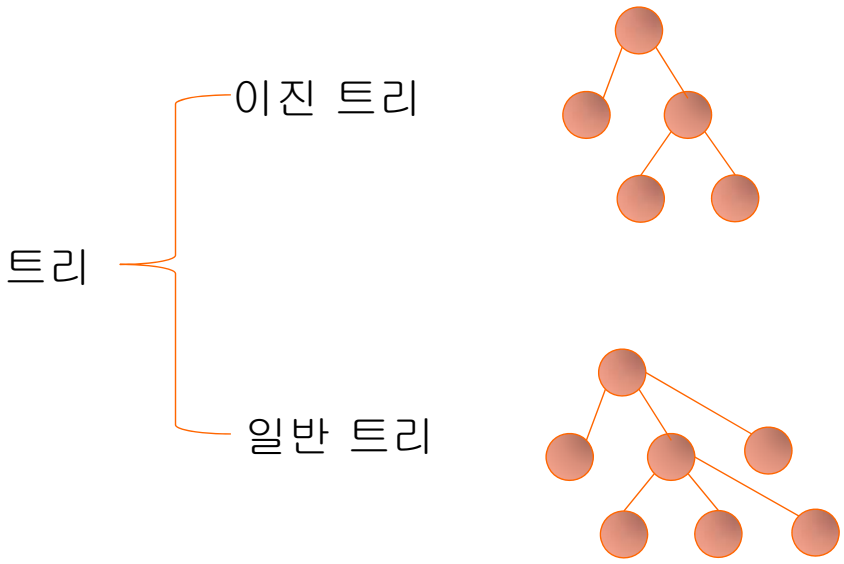


트리의 용어

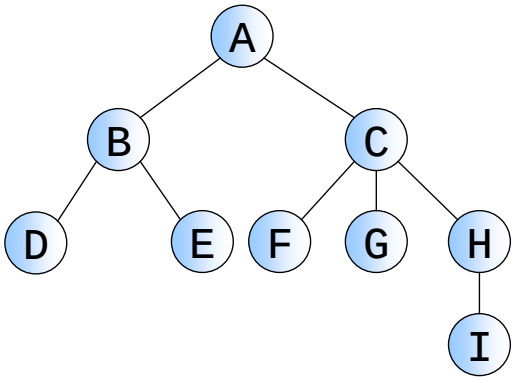
- 자식, 부모, 형제, 조상, 자손 노드 : 인간과 동일
- 레벨(level) : 트리의 각층의 번호
- 높이(height) : 트리의 최대 레벨(3)
- 차수(degree) : 노드의 자식 노드수



트리의 종류

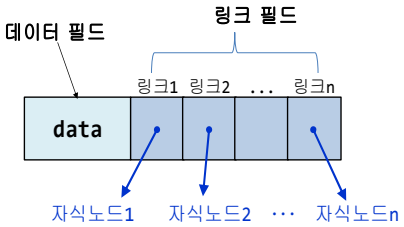


- 자식 노드가 없는 노드를 ( )라고 한다.
- 어떤 노드가 가지고 있는 자식 노드의 개수를 ( )라고 한다.
- 다음 트리에서 루트 노드는 ( )이고 F의 부모 노드는 ( )이며 B의 형제 노드는 ( )이다. 노드 F의 차수는 ( )이며 트리의 높이는 ( )이다

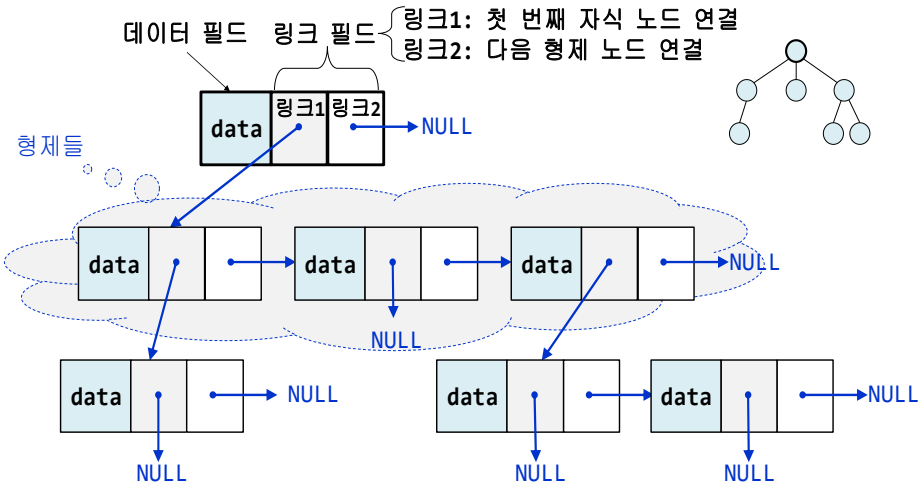


일반 트리의 표현

- 일반 트리

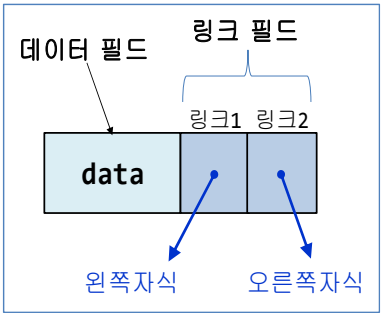


- 노드의 예: 너무 복잡함



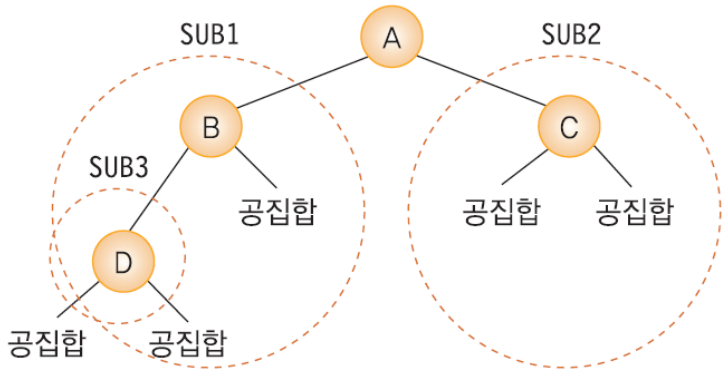
8.2 이진트리(binary tree)

- 모든 노드가 2개의 서브 트리를 가지고 있는 트리
  - 서브트리는 공집합일수 있다.



- 이진 트리(binary tree)
  - 각 노드에는 최대 2개까지의 자식 노드가 존재
  - 모든 노드의 차수가 2 이하가 된다-> 구현하기가 편리함
  - 서브 트리간의 순서가 존재 (왼쪽, 오른쪽)

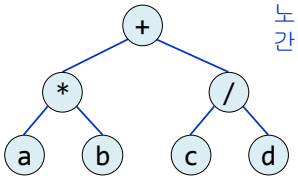
이진 트리 검증



- 이진 트리는 공집합이거나
- 루트와 왼쪽 서브 트리, 오른쪽 서브 트리로 구성된 노드들의 유한 집합으로 정의된다. 이진 트리의 서브 트리들은 모두 이진 트리이어야 한다.

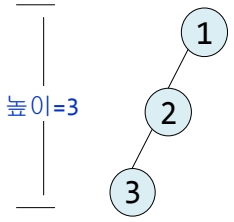
이진트리의 성질

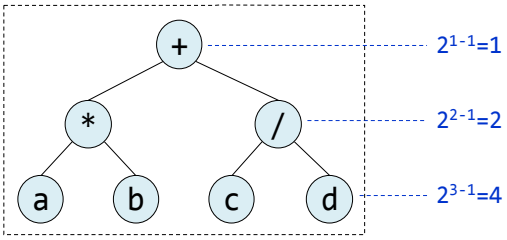
- 노드의 개수가 n개이면 간선의 개수는 n-1



노드의 개수: 7
간선의 개수: 6

- 높이가 h: 최소 h개 ~ 최대 2^h-1개의 노드를 가짐

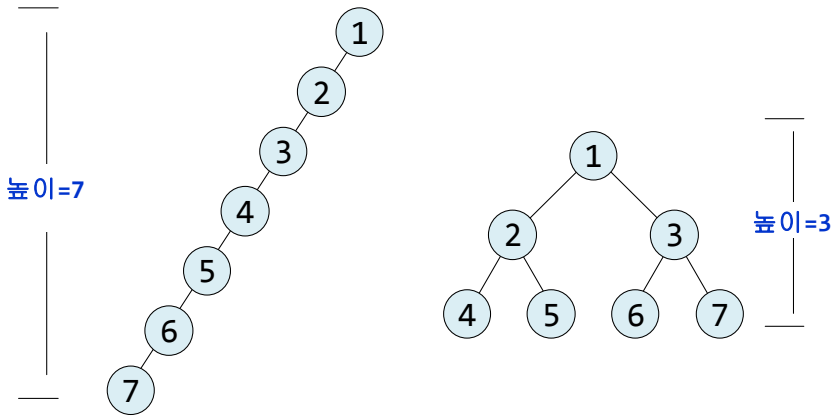




최소 노드 개수= 3 최대 노드 개수= $2^{1-1}+2^{2-1}+2^{3-1}=1+2+4=7$

이진트리의 성질

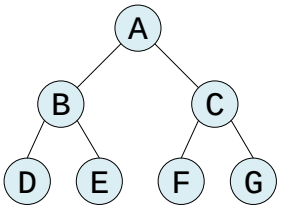
- n개 노드의 이진트리 높이: $\lceil log_2(n+1) \rceil$ ~ n



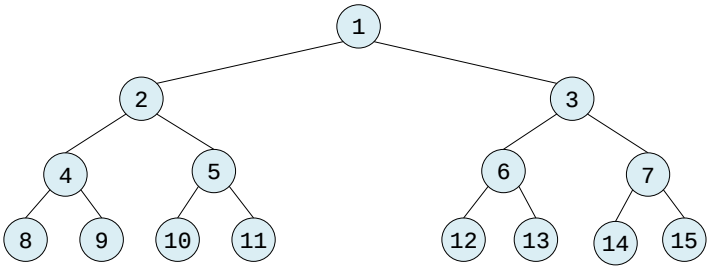
- 이진 트리에서 높이가 5일 때, 최대 몇 개의 노드를 가질 수 있는가?

이진트리의 분류

- 포화 이진 트리(full binary tree)
  - 트리의 각 레벨에 노드가 꽉 차있는 이진트리

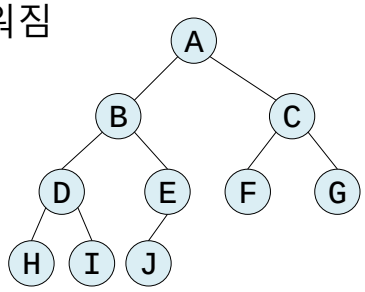


  - 노드의 번호

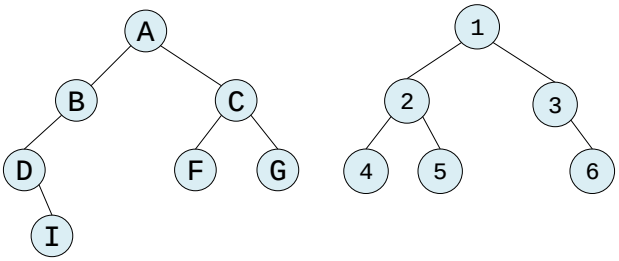


이진트리의 분류

- 완전 이진 트리(complete binary tree)
  - 높이가 h일 때 레벨 1부터 h-1까지는 노드가 모두 채워짐
  - 마지막 레벨 h에서는 노드가 순서대로 채워짐



- 기타 이진 트리



- 이진 트리에서 간선 수 e와 노드 수 n의 관계는?
- 높이가 h인 이진 트리에서 최대 노드 수는?
- 노드가 n개인 트리에서 최소 높이와 최대 높이는?

- 20개의 노드로 구성되어 있는 완전 이진 트리에서 루트가 레벨 1이면 레벨 5에는 몇 개의 노드가 있는가?

이진트리의 추상 자료형

데이터:

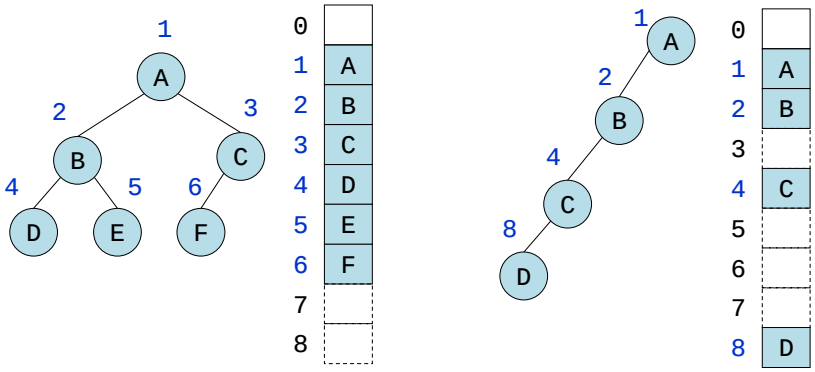
노드의 집합. 공집합이거나, 루트노드와 왼쪽 서브트리, 오른쪽 서브트리로 구성됨. 모든 서브트리도 이진트리이어야 함.

연산:

- **init()**: 이진트리를 초기화한다.
- **is_empty()**: 이진트리가 공백 상태인지 확인한다.
- **create_tree(e, left, right)**: 이진트리 left와 right를 받아 e를 루트로 하는 이진트리를 생성한다.
- **get_root()**: 이진트리의 루트 노드를 반환한다.
- **get_count()**: 이진트리의 노드의 수를 반환한다.
- **get_leaf_count()**: 이진트리의 단말 노드의 수를 반환한다.
- **get_height()**: 이진트리의 높이를 반환한다.

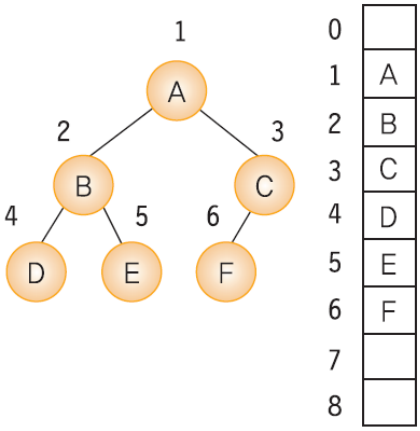
8.3 이진트리의 표현 : 배열표현법

- 모든 이진트리를 포화이진트리라고 가정
  - 각 노드에 번호를 붙여서 그 번호를 배열의 인덱스로 삼아 노드의 데이터를 배열에 저장

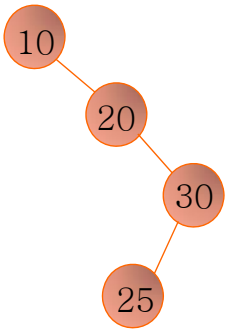


부모와 자식 인덱스 관계

- 노드 i의 부모 노드 인덱스 = $\lfloor i/2 \rfloor$
- 노드 i의 왼쪽 자식 노드 인덱스 = 2i
- 노드 i의 오른쪽 자식 노드 인덱스 = 2i+1

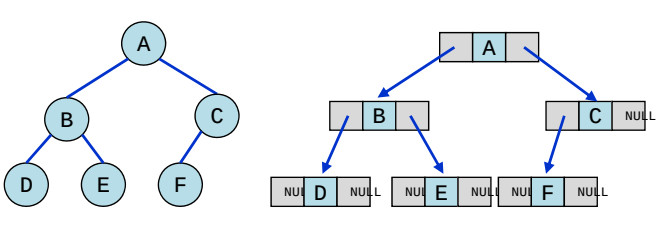


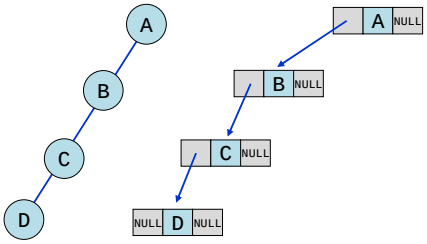
- 다음과 같은 트리를 배열 표현법으로 저장된 모습을 그려라. 각 노드는 어떤 인덱스에 저장되는가?



이진트리의 표현: 링크 표현법

- 부모노드가 자식노드를 가리키게 하는 방법
  - 포인터를 이용





8.4 링크 표현법을 이용한 이진트리

- 노드 구조체

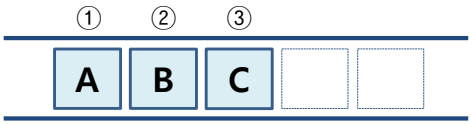
```
typedef char TElement;          // 트리에 저장할 데이터의 자료형
typedef struct BinTrNode {
  TElement      data;           // 노드에 저장할 데이터
  struct BinTrNode* left;       // 왼쪽 자식 노드의 포인터
  struct BinTrNode* right;      // 오른쪽 자식 노드의 포인터
} TNode;
```

- 이진트리 데이터

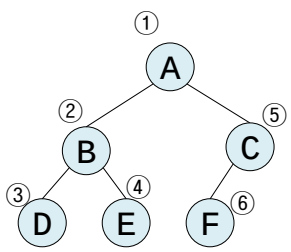
```
TNode* root = NULL;          // 루트노드의 포인터
```

8.5 이진트리의 순회

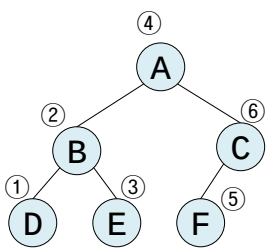
- 순회(traversal)
  - 트리의 노드들을 체계적으로 방문하는 것
- 선형 자료구조(큐)에서의 순회
  - 하나의 방법만 존재함



- 이진 트리에서의 순회
  - 다양한 순회 방법이 존재함 (비선형 자료구조)



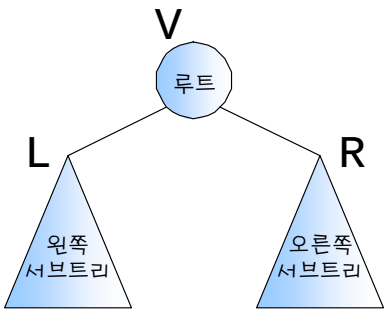
순회방법 1



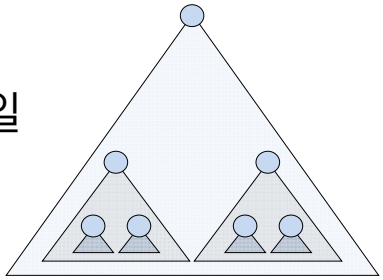
순회방법 2

이진트리의 기본 순회

- 전위순회(preorder traversal)
  - 루트→왼쪽 자식→오른쪽 자식
- 중위순회(inorder traversal)
  - 왼쪽 자식→루트 →오른쪽 자식
- 후위순회(postorder traversal)
  - 왼쪽 자식→오른쪽 자식→루트



- 전체 트리나 서브 트리나 구조는 동일

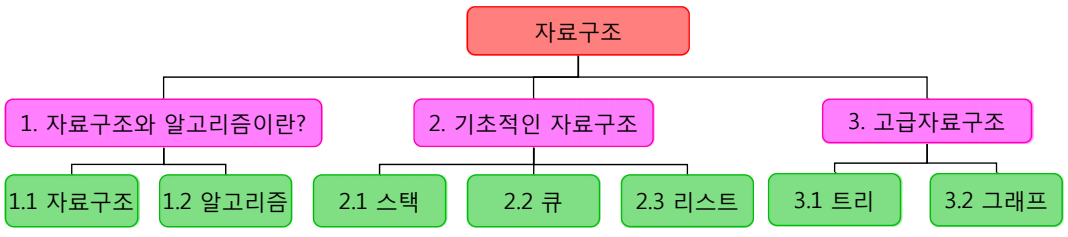


전위순회 (preorder)

- 루트 → 왼쪽 서브트리 → 오른쪽 서브트리

```
preorder(x)
if x≠NULL                          // x가 NULL이 아닐 때만 처리
   then print DATA(x);             // ① 루트(x) 노드 처리
        preorder(LEFT(x));         // ② 왼쪽 서브트리 방문
        preorder(RIGHT(x));        // ③ 오른쪽 서브트리 방문
```

- 응용분야: (예) 구조화된 문서출력

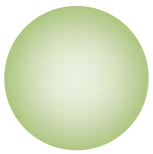


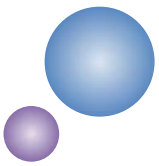
중위순회(inorder)

- 왼쪽 서브트리 → 루트 → 오른쪽 서브트리

```
inorder(x)
if x≠NULL                          // x가 NULL이 아닐 때만 처리
    then inorder(LEFT(x));         // ① 왼쪽 서브트리 방문
         print DATA(x);            // ② 루트(x) 노드 처리
         inorder(RIGHT(x));        // ③ 오른쪽 서브트리 방문
```

```
void inorder(TNode *n)
{
  if( n != NULL ) {
    inorder(n->left);
    printf("[%c] ", n->data);
    inorder(n->right);
  }
}
```





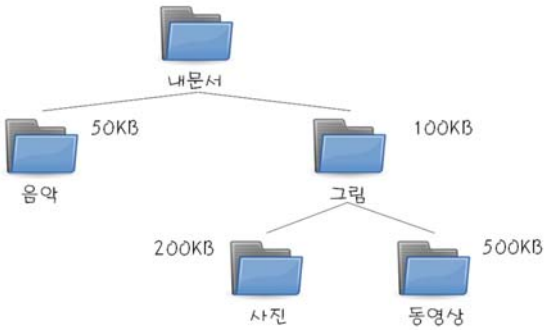
- 다음의 전위 순회와 중위 순회 결과를 생성할 수 있는 이진 트리를 그려라.
  - 전위 순회: A, B, D, E, C, F, G, H
  - 중위 순회: E, D, B, A, G, F, H, C

후위순회(postorder)

- 왼쪽 서브트리 → 오른쪽 서브트리 → 루트

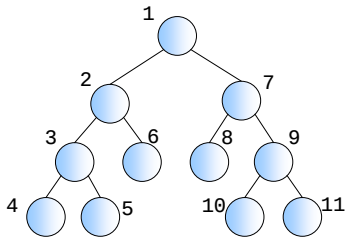
```
postorder(x)
if x≠NULL                              // x가 NULL이 아닐 때만 처리
    then  postorder(LEFT(x));          // ① 왼쪽 서브트리 방문
          postorder(RIGHT(x));         // ② 오른쪽 서브트리 방문
          print DATA(x);               // ③ 루트(x) 노드 처리
```

- 응용 예: 폴더 용량 계산

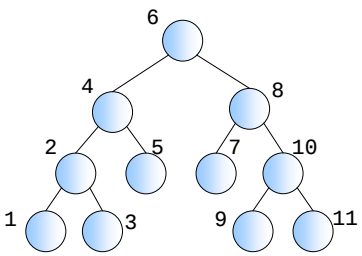


순회 방법에 따른 노드 방문 순서

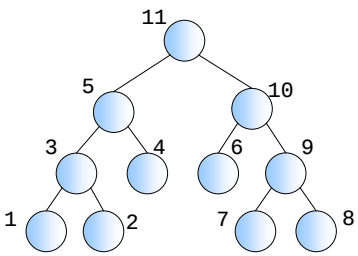
- 전위순회



- 중위순회

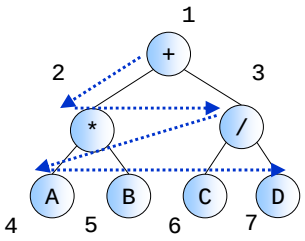


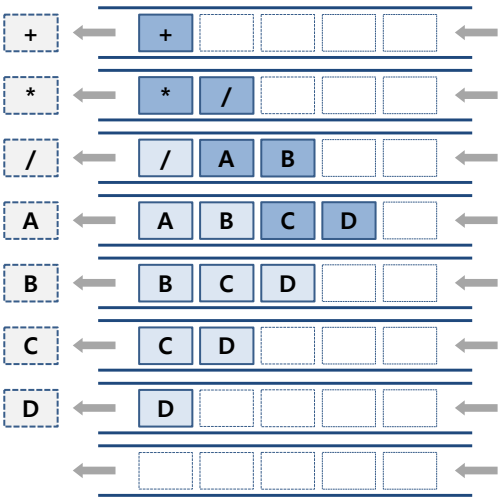
- 후위순회



레벨 순회

- 노드를 레벨 순으로 검사하는 순회방법
  - 큐를 사용해 구현

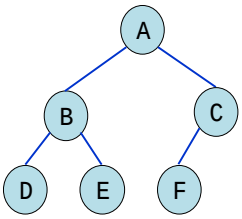


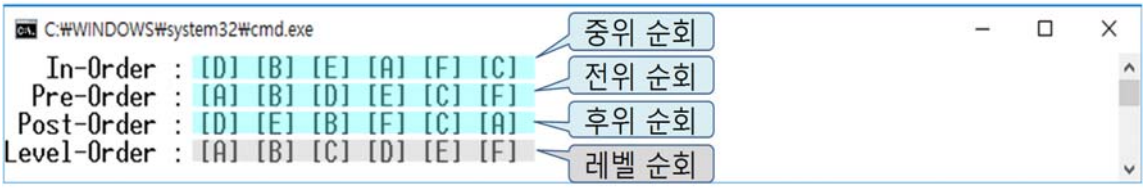


레벨 순회 알고리즘

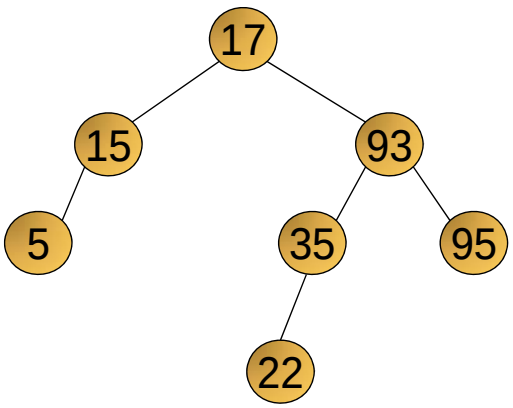
```
level_order()
initialize queue;
if(root == NULL) then return;
enqueue(root);
while isEmpty(queue)≠TRUE do
        x ← dequeue(queue);
        if( x≠NULL) then
                print DATA(x);
                enqueue(LEFT(x));
                enqueue(RIGHT(x));
```

순회 프로그램 비교





- 아래의 트리를 전위, 중위, 후위 순회할 경우, 노드의 방문 순서는?
- 아래의 트리를 레벨 순회할 경우, 노드의 방문 순서는?



- 다음과 같은 함수가 이진트리의 루프 노드에 대해 호출된다고 하자. 함수가 반환하는 값은 무엇인가?

```
int mystery(Tnode* p) {
        if (p == NULL)
                return 0;
        else if (p->left == NULL && p->right == NULL)
                return p->data;
        else
                return max(mystery(p->left), mystery(p->right));
}
```

8.6 이진트리연산: 노드 개수

- 트리의 노드 개수를 계산

```
count_node(x)
if x = NULL
      then return 0;
      else return 1+count_node(x.left)+count_node(x.right);
```

```
int count_node(TNode *n)
{
  if( n == NULL ) return 0;
  return 1 + count_node(n->left) + count_node(n->right);
}
```

이진트리연산: 단말 노드 개수

- 트리의 단말 노드 개수를 계산

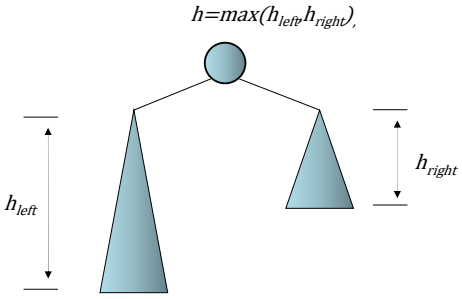
```
count_leaf(x)
if x = NULL then return 0;
if x.left=NULL and x.right=NULL
then return 1;
else return count_leaf(x.left) + count_leaf(x.right);
```

```
int count_leaf(TNode *n)
{
        if( n == NULL ) return 0;
        if(n->left == NULL && n->right == NULL ) return 1;
        else return count_leaf(n->left) + count_leaf(n->right);
}
```

이진트리연산 : 트리 높이

- 서브 트리들의 높이 중에서 최대값을 구하여 반환

```
getHeight(x)
if x = NULL
  then return 0;
  else return 1 + max(getHeight(x.left),
                      getHeight(x.right);
```

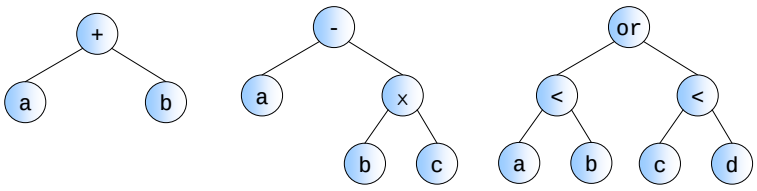



```
int calc_height(TNode *n)
{
  int hLeft, hRight;
  if( n == NULL ) return 0;
  hLeft = calc_height(n->left);
  hRight = calc_height(n->right);
  return (hLeft>hRight)
        ? hLeft+1 : hRight+1;
}
```

- 이진 트리의 노드 개수가 n개라면 NULL 링크의 개수는?
- 일반적인 이진 트리에서 최대값과 최소값을 탐색하기 위한 함수를 작성하고 실행 예를 보이시오. (힌트: 순환호출을 사용하라.)

수식 트리

- 산술식을 트리형태로 표현한 것
  - 비단말노드 : 연산자(operator)
  - 단말노드 : 피연산자(operand)
- 예)

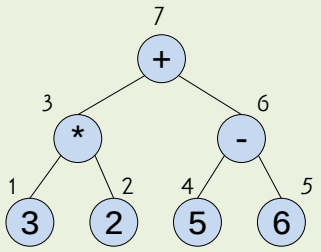


수식	a + b	a - (b × c)	(a < b) or (c < d)
전위순회	+ a b	- a × b c	or < a b < c d
중위순회	a + b	a - b × c	a < b or c < d
후위순회	a b +	a b c × -	a b < c d < or

수식 트리 계산

- 후위순회를 사용
  - 서브트리의 값을 순환호출로 계산
  - 비단말노드: 양쪽서브트리의 값을 연산자를 이용해 계산

```
evaluate(exp)
if exp = NULL
  then return 0;
  else x←evaluate(LEFT(exp));
       y←evaluate(RIGHT(exp));
       op←DATA(exp);
       return (x op y);
```

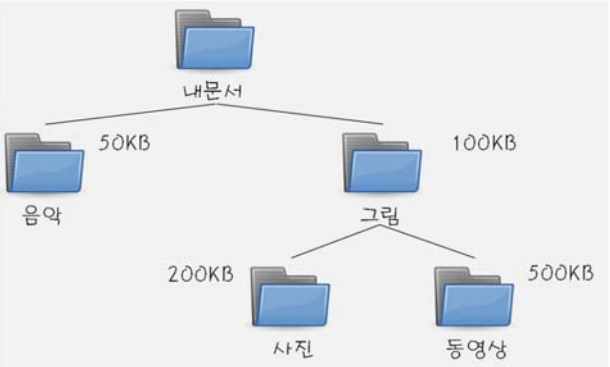


폴더 용량 계산

- 후위 순회

```
int calc_size(TNode *n)
{
  if( n == NULL ) return 0;
  return n->data
    + calc_size(n->left)
    + calc_size(n->right);
}

void main()
{
  TNode *m2, *m3, *m4, *m5;
  m4 = create_tree ( 200, NULL, NULL );
  m5 = create_tree ( 500, NULL, NULL );
  m3 = create_tree ( 100, m4, m5 );
  m2 = create_tree ( 50, NULL, NULL );
  root = create_tree ( 0, m2, m3 );
}
```



- 이진트리가 완전이진트리인지를 검사하는 다음 연산을 구현하라.
  int is_complete_binary_tree();
- 임의의 node의 레벨을 구하는 연산을 구현하라. 만약 node가 트리 안에 있지 않으면 0을 반환하라.
  int level(Tnode* node);
- 이진트리의 모든 노드에서 왼쪽 서브트리와 오른쪽 서브트리의 높이의 차이가 2보다 작으면 이 트리를 "균형 잡혀 있다(balanced)"라고 한다. 현재 이진트리가 균형 잡혀 있는지를 검사하는 다음 연산을 구현하라.
  int is_balanced();
- 이진트리에서 경로의 길이(path length)를 루트에서부터 모든 자식 노드까지의 경로의 길이의 합이라고 하자. 경로의 길이를 구하는 다음 연산을 구현하라.
  int path_length();
- 이진트리를 좌우로 대칭시키는 다음 연산을 구현하라.
  void reverse();