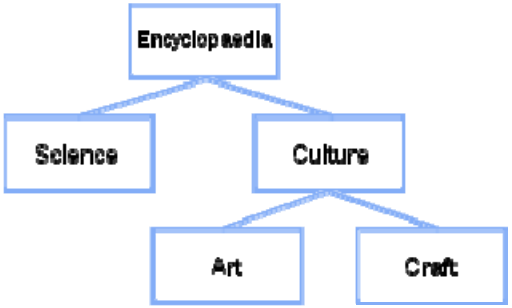
제 8 강의 . 트리(Tree) 자료구조

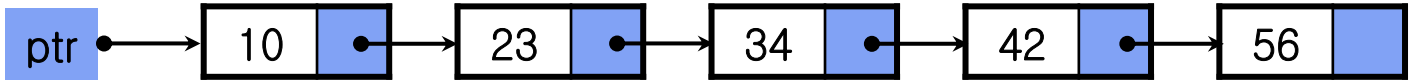
1. 트리의 개념
2. 이진 트리
3. 이진 트리의 저장



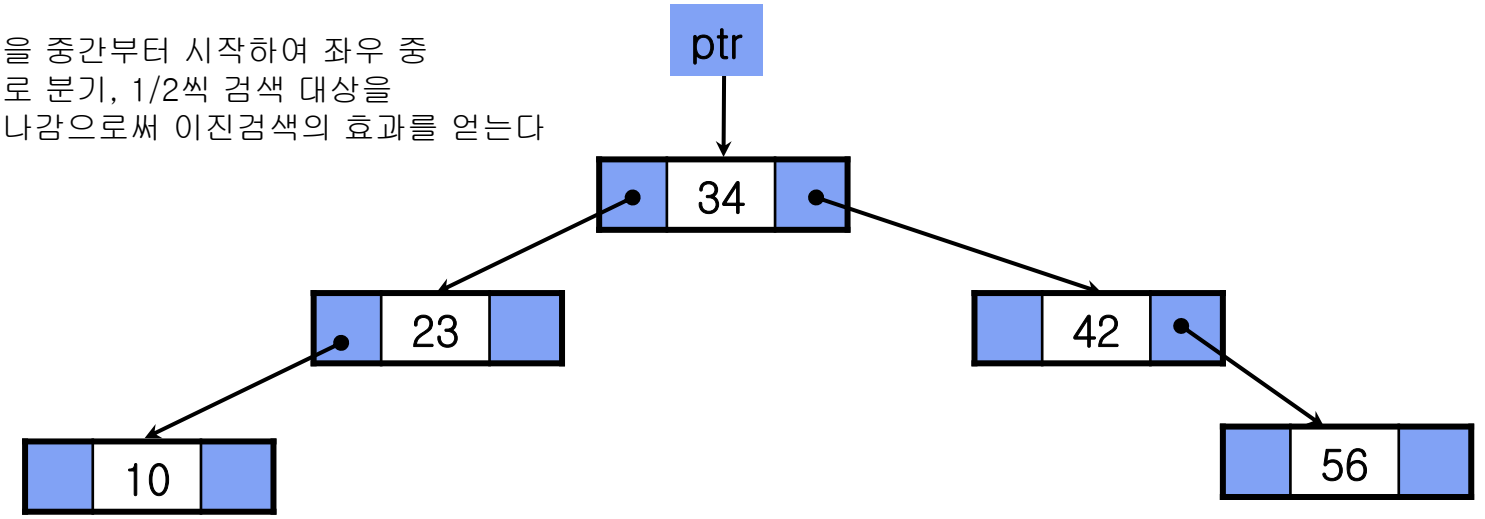
트리 자료구조 필요성

연결리스트의 삽입삭제 시 데이터를 이동하지 않는 장점을 살리자.
연결리스트의 검색시 노드의 처음부터 찾아가야하는 단점을 보완하자.

⇒ 데이터를 중간부터 찾아가는 이진검색의 장점을 이용하자.
⇒ 연결리스트의 포인터를 리스트의 중간에 두는 방법?



⇒ 검색을 중간부터 시작하여 좌우 중 하나로 분기, 1/2씩 검색 대상을 줄여나감으로써 이진검색의 효과를 얻는다



1. 트리의 개념

트리의 정의

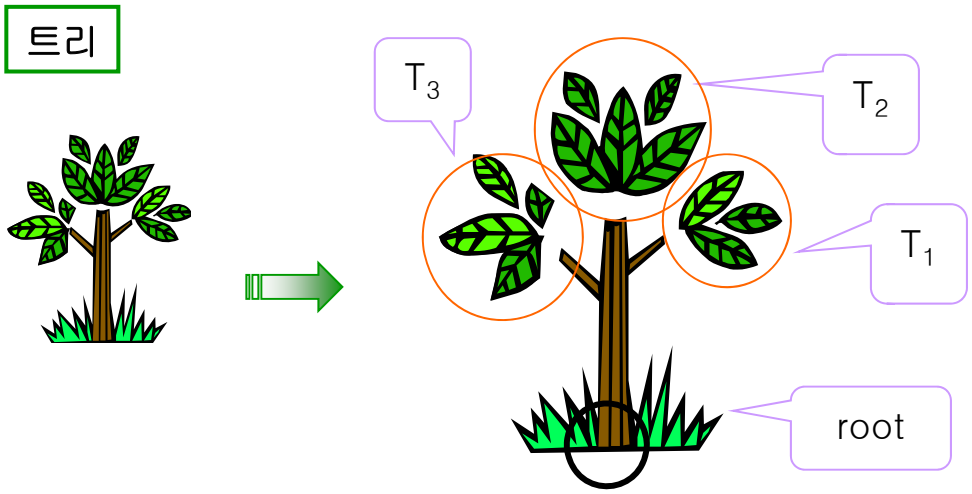
트리는 1개 이상의 노드를 갖는 집합으로 노드들은 다음 조건을 만족한다.

1) 트리에는 루트(root)라고 부르는 특별한 노드가 있다.

2) 다른 노드들은 원소가 중복되지 않는 n개의 부속 트리 (subtree)

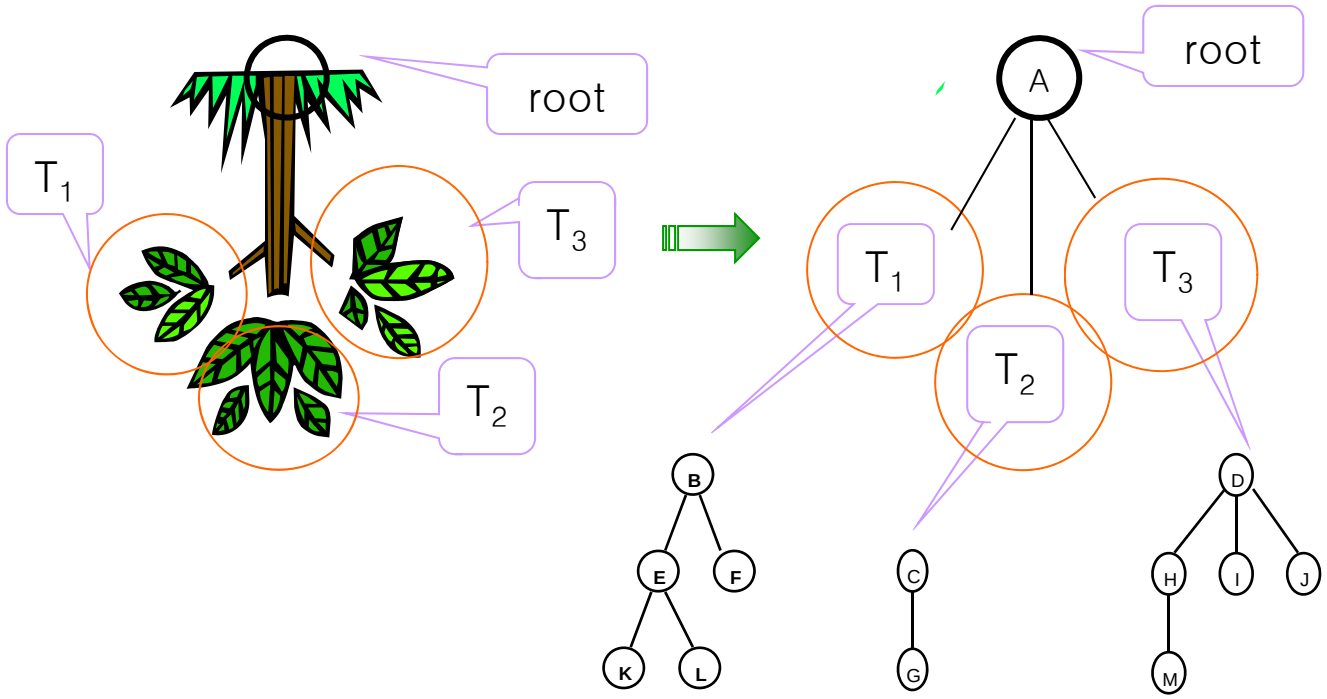
$T_1, T_2, \cdots, T_n$으로 나누어지며 T_i 각각은 루트의 부속 트리라고 부른다.

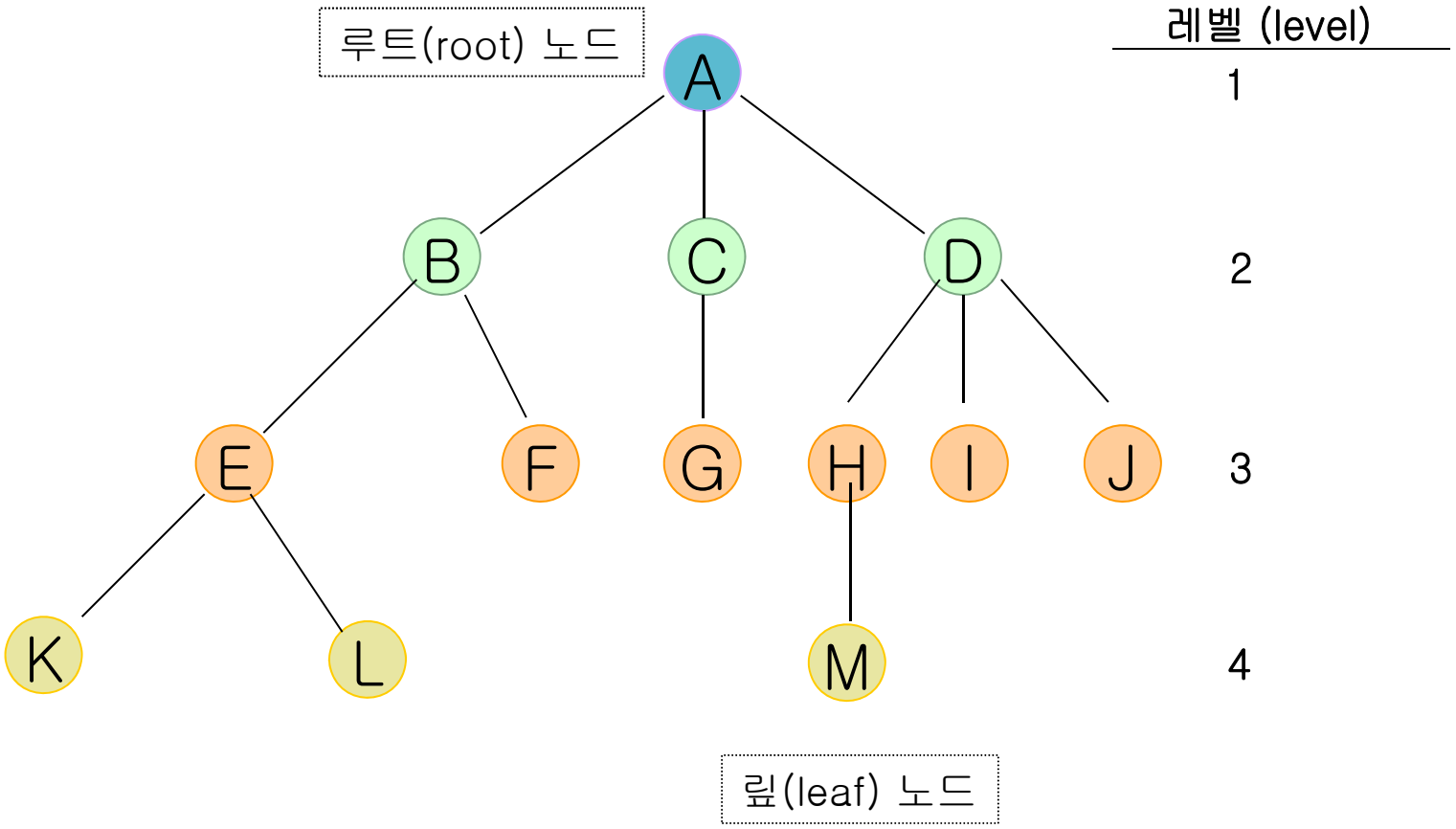
(트리는 사이클이 없는 그래프 (acyclic graph)이며 계층 구조를 이룬다.)



자료구조 트리(tree)는 나무를 거꾸로 그린다.

부속 트리 T_1, T_2, T_3는 다시 트리 구조를 이룬다.




루트(root) 노드
레벨 (level)
1
2
3
4
A
B
C
D
E
F
G
H
I
J
K
L
M
립(leaf) 노드

트리 자료구조는 왜 필요할 까요?

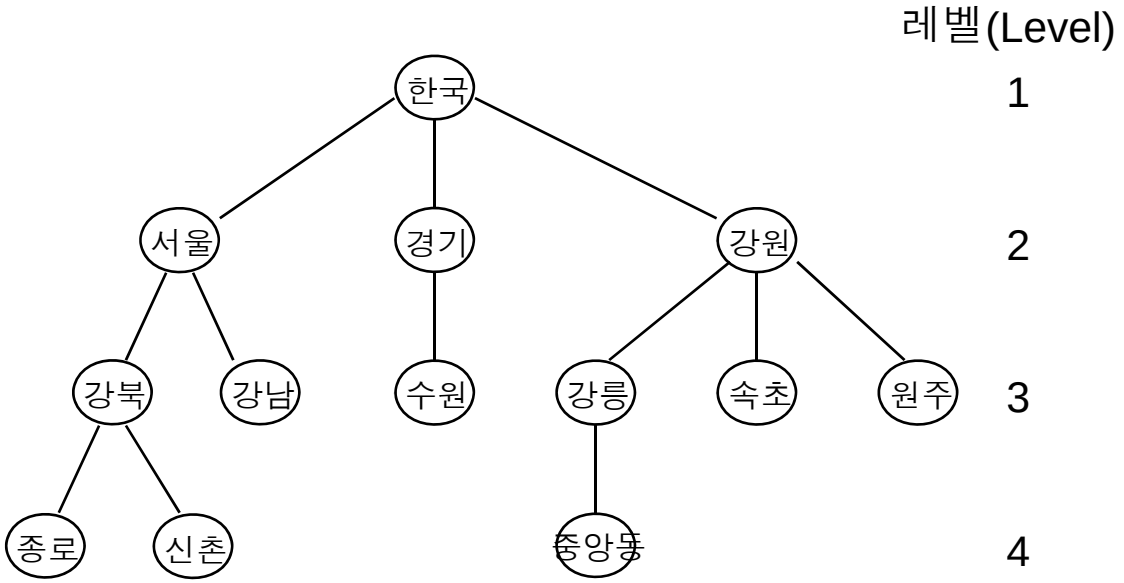
- 트리 구조에 저장하면 더 효율적인 자료들이 있기 때문이다.

예를 들면, 계층적인 데이터 형태들은 트리에 저장하면 자연스럽게 표현된다.

1) 회사나 정부의 조직 구조,

2) 나라, 지방, 시•군별, 계층적인 데이터의 저장,

3) 인덱스 (인덱스는 계층적 자료 구조로 검색을 쉽게해준다.)

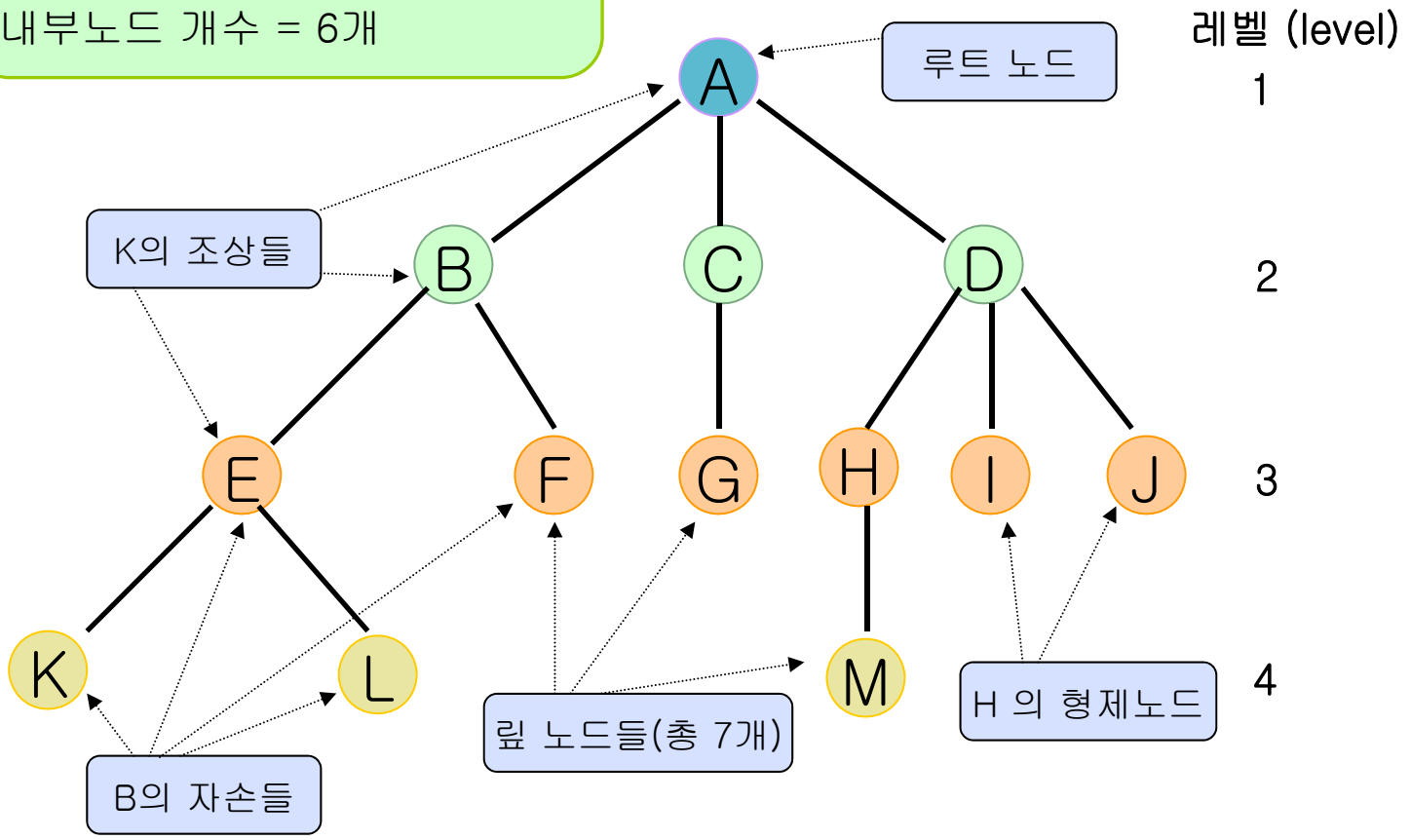


트리에 관한 용어

- 노드의 차수(degree) : 노드의 부속 트리의 개수
- 트리의 차수(degree of tree) : 트리의 최대 차수
- 릾(leaf, 단말, terminal) 노드: 차수가 0인 노드, 즉 맨 끝에 달린 노드들이다.
- 내부(internal, non-terminal) 노드 : 차수가 1 이상인 노드
- 부모(parent) : 부속 트리(subtree)를 가진 노드
- 자식(child) : 부모에 속하는 부속노드
- 형제(sibling) : 부모가 같은 자식 노드들
- 조상(ancestor) : 노드의 부모 노드들의 총 집합
- 자손(descendant) : 노드의 부속 트리에 있는 모든 노드들
- 레벨(level) : 루트 노드들로부터 깊이(루트 노드의 레벨 = 1)
- 트리의 깊이(depth of tree) : 트리에 속한 노드의 최대 레벨

* 트리의 예
전체 노드 개수 = 13개
단말(릮) 노드 개수 = 7개
내부노드 개수 = 6개

트리의 차수 = 3
트리의 깊이 = 4



트리 구조를 컴퓨터 내부에 저장하는 방법

1) n-링크 표현법

- 노드에 n 개의 링크를 두고 자식의 개수만큼 링크에 저장한다. 모든 노드는 자식 노드 수에 관계없이 최대 n개의 링크를 갖는다. 각 링크는 부속 트리가 저장된 곳을 링크한다.

* 차수가 n인 경우 표현된 노드

참고 : 트리를 리스트 형태로 표현하는 방법 - 괄호를 사용하여 같은 레벨에 있는 노드들을 같은 괄호로 묶는다.

예) (A(B(E(K,L),F),C(G),D(H(M),I,J)))

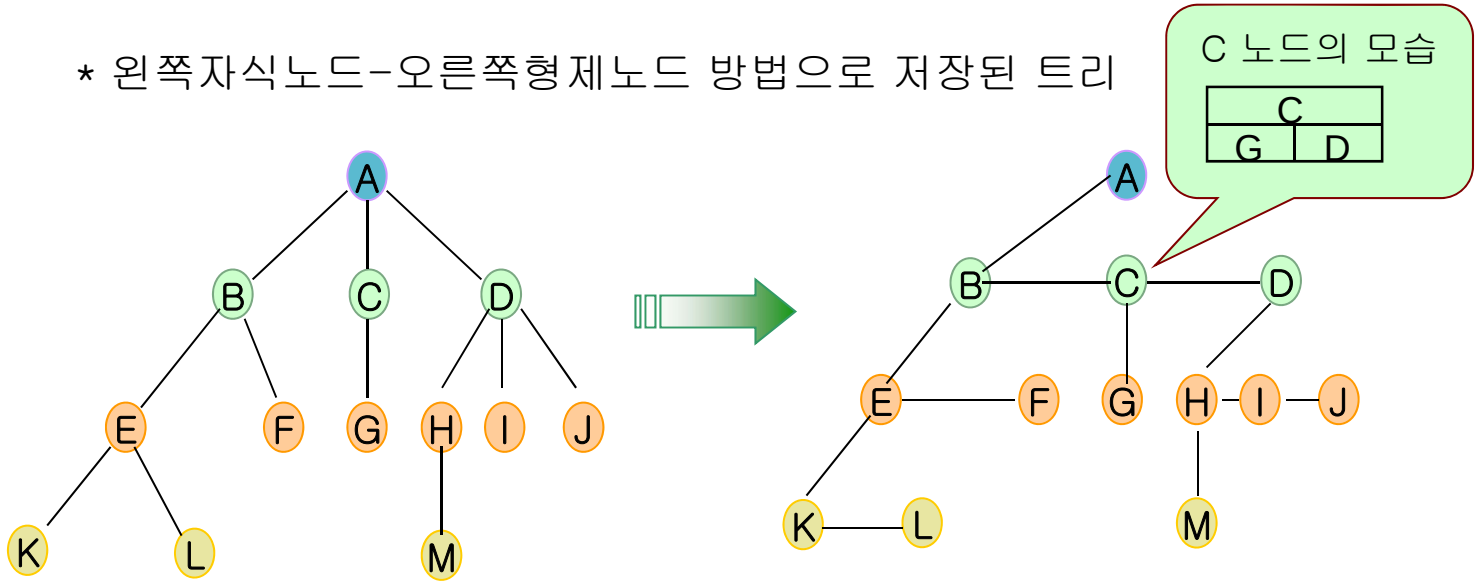
2) 왼쪽자식노드-오른쪽형제노드 표현 방법

- 모든 노드에 링크를 2개를 둔다.
  첫째 링크는 첫번째 자식노드를 표현하고
  둘째 링크는 자신의 오른쪽 형제 노드를 표현한다.
- 노드의 길이가 2개로 고정되기 때문에 n-링크 방법보다 간편하다.

* 왼쪽자식노드-오른쪽형제노드 표현법

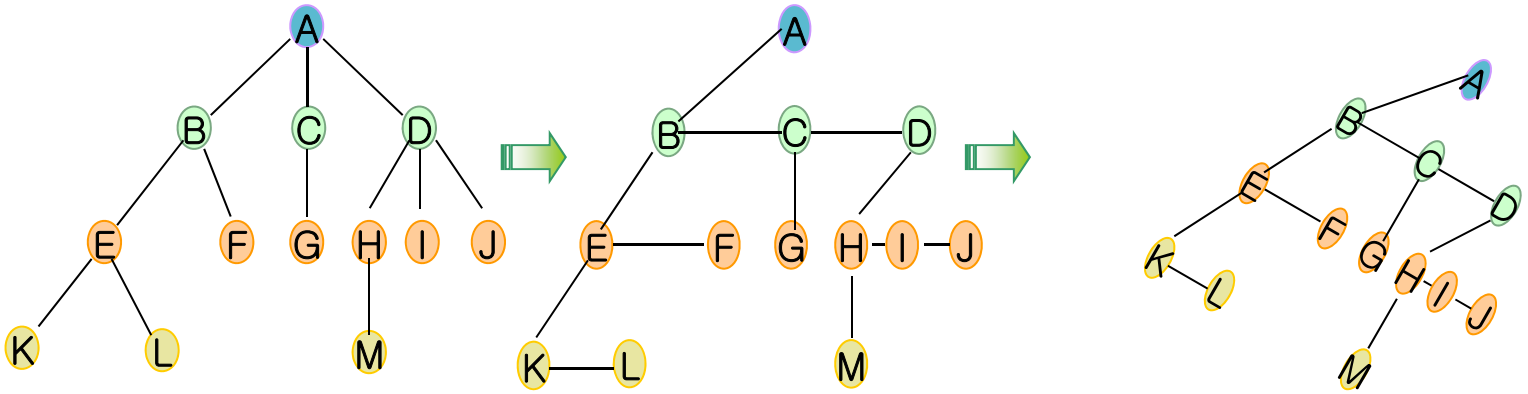
data	
left child	right sibling

* 왼쪽자식노드-오른쪽형제노드 방법으로 저장된 트리

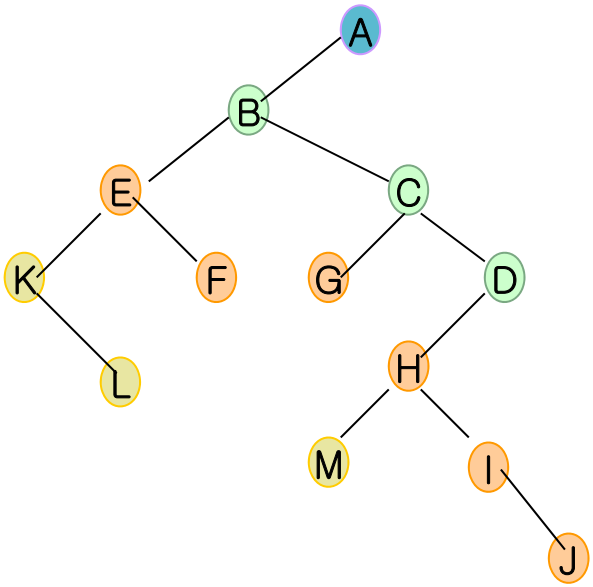


차수가 n인 트리를 차수가 2인 트리로 저장하는 방법

- 트리를 왼쪽자식노드-오른쪽형제노드로 변환하여 그린 다음 45도 시계 방향으로 돌리면 된다.(약간 수정은 필요)
- 모든 노드가 2개 이하의 자식 노드를 갖는다.
- 차수가 n인 트리가 차수가 2인 트리가 된다.(이진 트리)



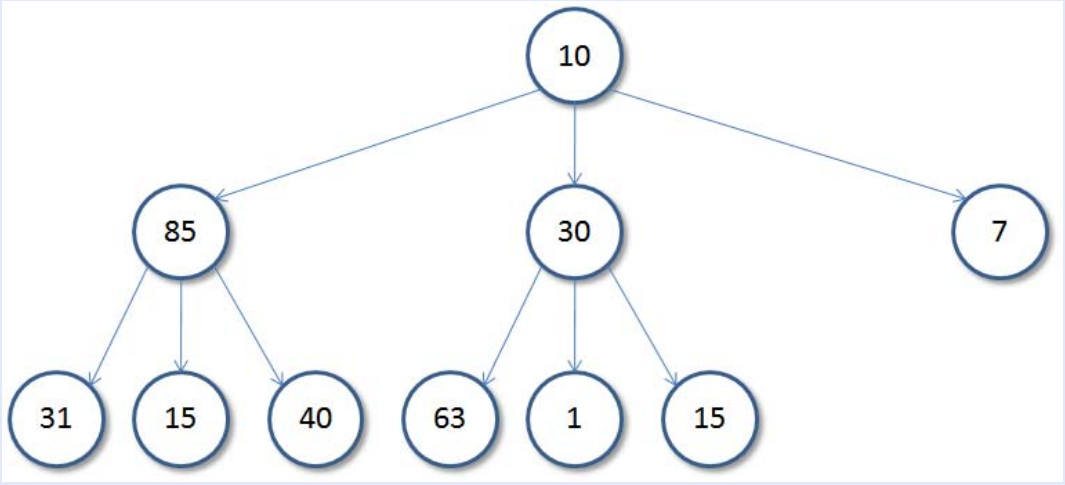
일반 트리 -> 왼쪽자식-오른쪽형제 표현 -> 이진트리모양으로 회전



왼쪽 자식노드-오른쪽 형제노드 표현 방법 -> 이진 트리 모양

Q/A

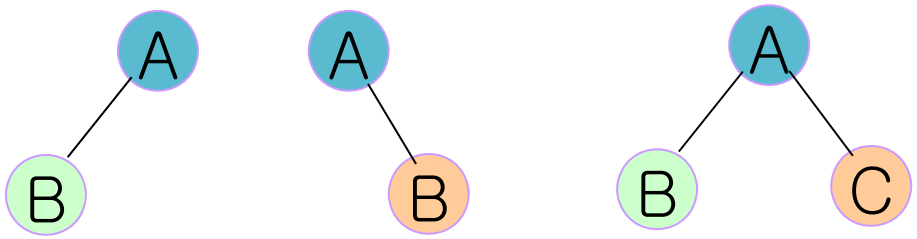
1. 다음 차수가 3인 트리를 왼쪽자식-오른쪽 형제노드 표현 방법으로 바꾸어 보아라.



2. 이진(Binary) 트리

자식 노드의 수가 2개 이하인 것을 이진 트리(binary tree)라고 한다.
대부분 응용에서 일반 트리 보다는 원래부터 이진 트리로 표현된 문제가 많다.

정의) 이진 트리(a binary tree)는 유한개의 노드로 구성된 트리로
1) 비어있거나 혹은
2) 루트노드와 2개의 부속 트리로 구성된다. 2개의 부속 트리는
왼쪽 부속 트리, 오른쪽 부속 트리라고 부른다.
* 주의 : 노드가 없는 경우도 이진 트리의 일종이다.



이진 트리의 예 : 왼쪽 두 트리는 서로 다른 이진 트리이다.

이진 트리의 성질

✓이진 트리는 자식의 개수가 2개 이하인 일반트리와 약간 다르다.

✓ 노드가 없는 트리도(empty node)도 이진 트리가 된다.

✓ 부속 트리(자식노드)에 순서가 있다.

✓ 이진 트리의 최대 레벨이 i 인 경우 트리의 최대 노드의 개수는 2^i-1 이다.

레벨 1의 최대 노드 개수 = 1

레벨 2의 최대 노드 개수 = 2

…

레벨 i의 최대 노드 개수 = 2^{i-1}

레벨 i인 트리의 최대 노드는 = 1 + 2 + 4 + 8 …+ 2^{i-1} = 2^i-1

✓ 이진 트리의 릮 노드의 개수 n_0 와 차수가 1인 노드의 개수 n_1, 차수가 2인 노드의 개수 n_2 에 관한 다음의 등식이 성립한다.

$n_0 = n_2 + 1$

(증명) 전체 노드 개수를 n이라고 하면

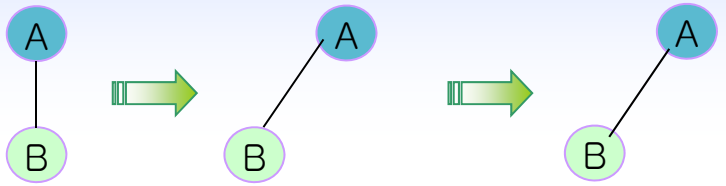
① $n= n_2 + n_1 + n_0$ (노드 개수는 세가지 경우를 합한 것이다)

② $n= 2 * n_2 + 1 * n_1 + 0 * n_0 + 1$

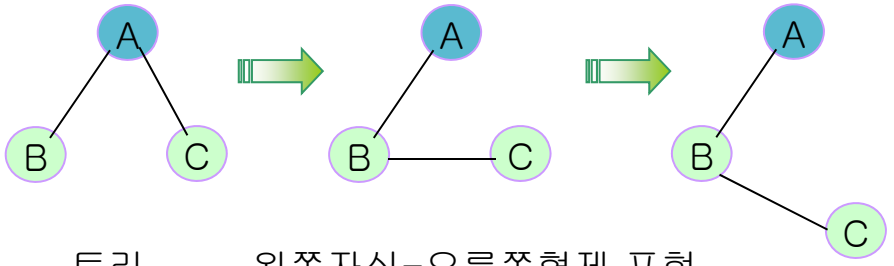
(전체노드수는 자식노드를 곱한 식에다, 루트 노드 1을 더한다.)

두 식을 계산하면 n과 n_1 이 소거되어 증명 성립

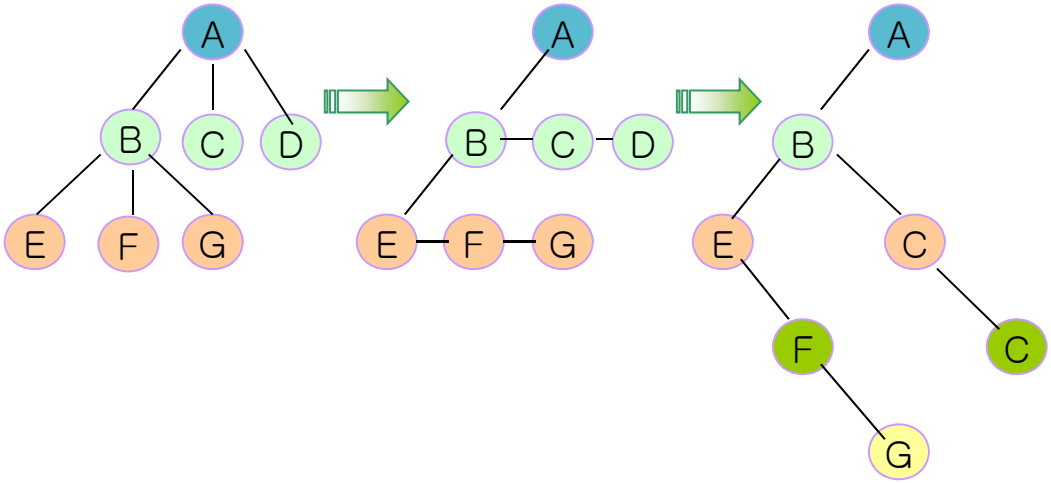
- 트리와 이진(Binary) 트리의 예



트리 왼쪽자식-오른쪽형제 표현 이진 트리



트리 왼쪽자식-오른쪽형제 표현 이진 트리



트리 왼쪽자식-오른쪽형제 트리 이진 트리

다양한 이진 트리

스큐(skewed) 이진 트리

(정의) 트리의 노드가 왼쪽이나 오른쪽으로 한쪽으로만 노드가 있는 트리이다.

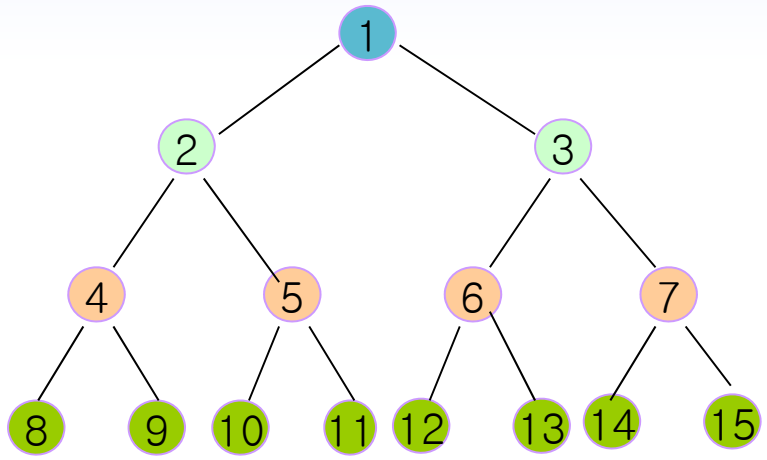
포화(full) 이진 트리(full binary tree of depth k)

(정의) 트리의 깊이가 k(레벨을 1로 시작), $2^k - 1$ 노드를 가진 이진 트리
(트리 깊이가 1이면 노드가 1개, 2이면 3개, 3이면 7개, 4이면 15개, ...
로 트리에 노드가 꽉 차있는 이진 트리이다.)

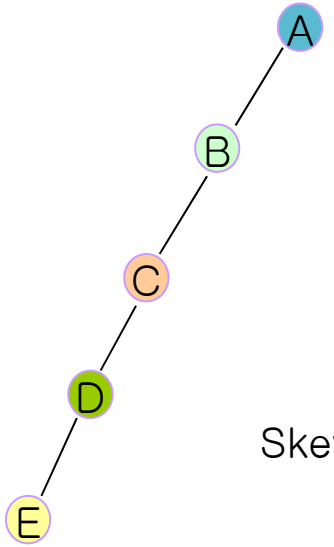
완전(complete) 이진트리(complete binary tree)

(정의) n개의 노드를 가진 complete(완전) 이진트리는 포화 이진트리에서 노드에 1부터 n까지 번호를 붙였을 때 만들어진 트리이다.

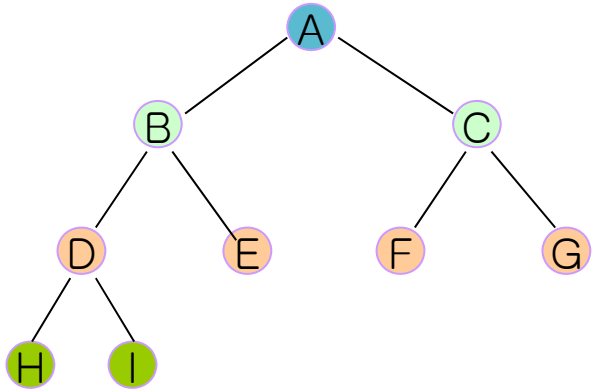
- 이진(Binary) 트리의 예



포화(full) 이진 트리(깊이=4)



Skewed 이진 트리



완전(complete) 이진 트리

Q/A

* 다음 트리에 대하여 답을 구해보자

(1) 깊이가 5인 이진트리의 노드의 최소개수는?
(2) 깊이가 5인 이진트리의 노드의 최대 개수는?
(3) 깊이가 5인 포화이진트리의 노드의 최소 개수는?
(4) 깊이가 5인 포화이진트리의 노드의 최대 개수는?
(5) 깊이가 5인 완전이진트리의 노드의 최소 개수는?
(6) 깊이가 5인 완전이진트리의 노드의 최대 개수는?
(7) 이진트리에서 자식이 2인 노드의 개수가 4개 일때 릮노드의 개수는?

3. 이진(Binary) 트리의 저장

- 이진 트리의 저장

3-1 배열을 이용한 저장

트리의 레벨 순서대로 배열에 저장을 해두는 방법이다.

트리의 노드를 레벨순서대로 왼쪽에서 오른쪽으로 저장한다.

배열이름을 T[]라고 하면 루트노드는 T[0]에, 레벨2의 첫째노드는 T[1]에, 둘째 노드는 T[2]에,…, 저장하는 방식이다.

배열에 저장할 경우 임의의 배열 원소의 부모와 왼쪽, 오른쪽 자식을 찾는 함수는 다음과 같이 된다.(첨자가 0부터 시작할 경우)

1) 부모 노드의 위치 –

parent(i) = (i –1) / 2 if i $\neq$ 0
parent(i) = 0 if i = 0 : (루트노드는 부모가 없다.)

2) 왼쪽자식의 위치 – left_child(i) = 2·i + 1

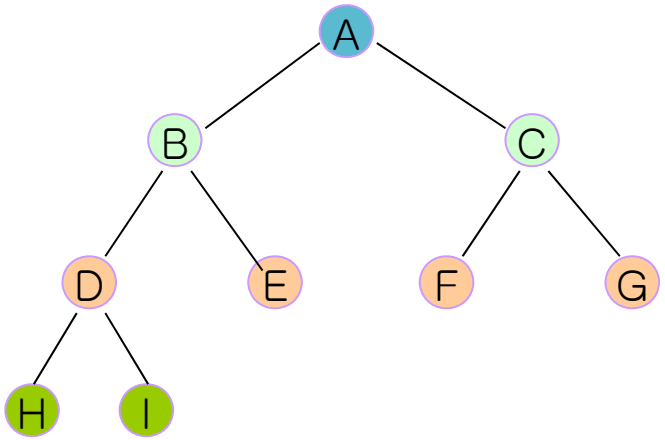
3) 오른쪽자식의 위치 – right_child(i) = 2·i + 2

* 첨자가 1부터 시작하면?

배열에 저장된 이진(Binary) 트리 예 1

예를 들어 T[6]의 부모는 T[(6-1)/2]이고 T[3]
노드의 왼쪽자식은 T[3*2 +1]이다.

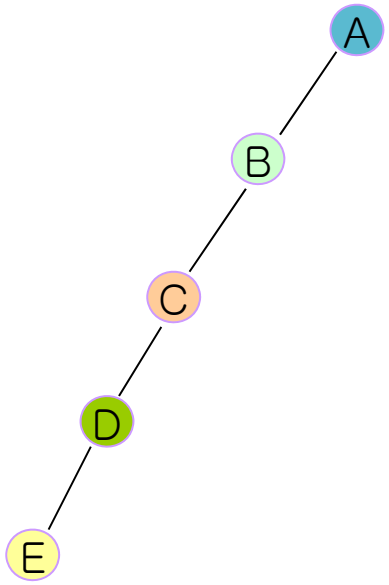
T[1]의 오른쪽 자식노드의 위치를 찾아 보아라.(소수점이하는 버린다.)



[0]	A
[1]	B
[2]	C
[3]	D
[4]	E
[5]	F
[6]	G
[7]	H
[8]	I

배열 T[ ]

배열에 저장된 이진(Binary) 트리 예 2



skew 이진 트리

[0]	A
[1]	B
[2]	-
[3]	C
[4]	-
[5]	-
[6]	-
[7]	D
[8]	-
.	.
.	.
.	.
[15]	E

배열에서 데이터가 비어있는 곳이 많다.

배열을 이용한 이진 트리 저장의 문제점

(1) 배열의 이용하지 않는 저장 공간이 많다.

깊이가 k인 트리에서 전체 필요한 공간은 $S(k) = 2^k - 1$ 이지만 skewed 이진 트리 처럼 깊이에 비하여 노드 수가 적은 경우 기억 공간 활용율이 낮다.

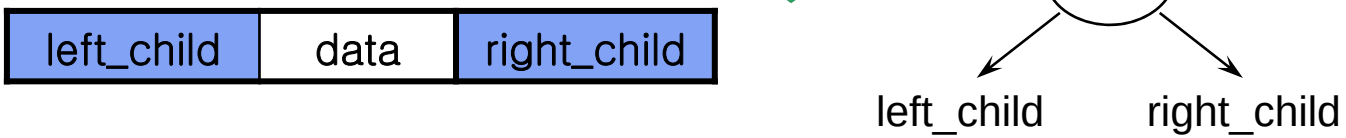
- complete(완전) 이진 트리의 저장에는 효과적이다

(2) 트리의 최대 깊이를 대비하여 많은 기억장소를 확보하고, 트리가 예상보다 커지면 프로그램 수행을 종료해야 한다.

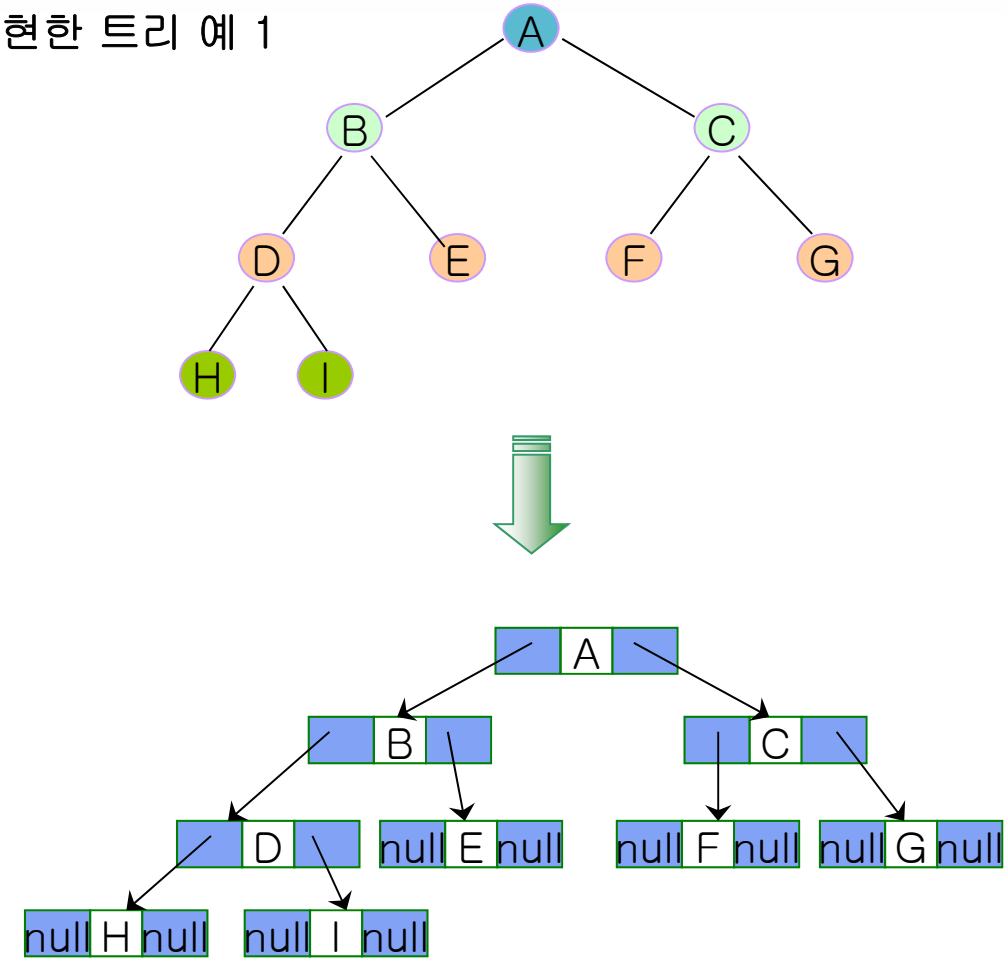
3-2 연결리스트를 이용한 트리의 표현

(1) 데이터
(2) 왼쪽자식에 대한 포인터
(3) 오른쪽자식에 대한 포인터

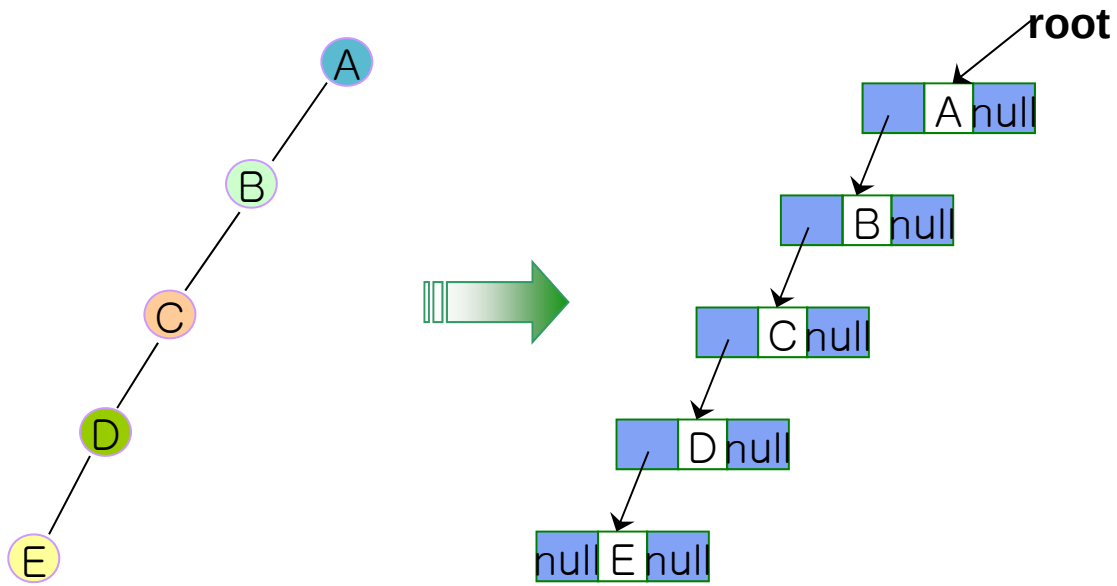
```
struct tnode {
          int data;
          struct tnode * left_child;
          struct tnode * right_child;
};
typedef struct tnode node;
typedef node * tree_ptr;
```



- 연결리스트로 표현한 트리 예 1



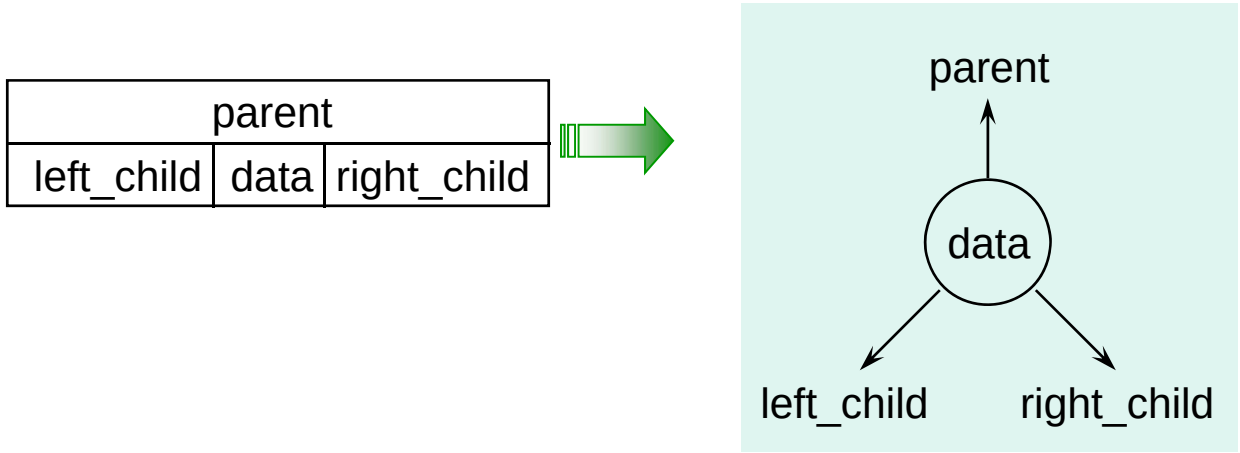
- 연결리스트로 표현한 트리 예 2



skewed 이진 트리 경우

참고 이진 트리를 연결리스트로 표현하는 다른 방법

필드를 한 개 더 추가하여 부모노드에 대한 포인터를 저장한다.
그러나 트리를 다룰 때 링크 3개를 관리해야 하므로 복잡하여 잘 사용하지 않는다.



이진 트리 저장 방법의 비교

(1) 배열

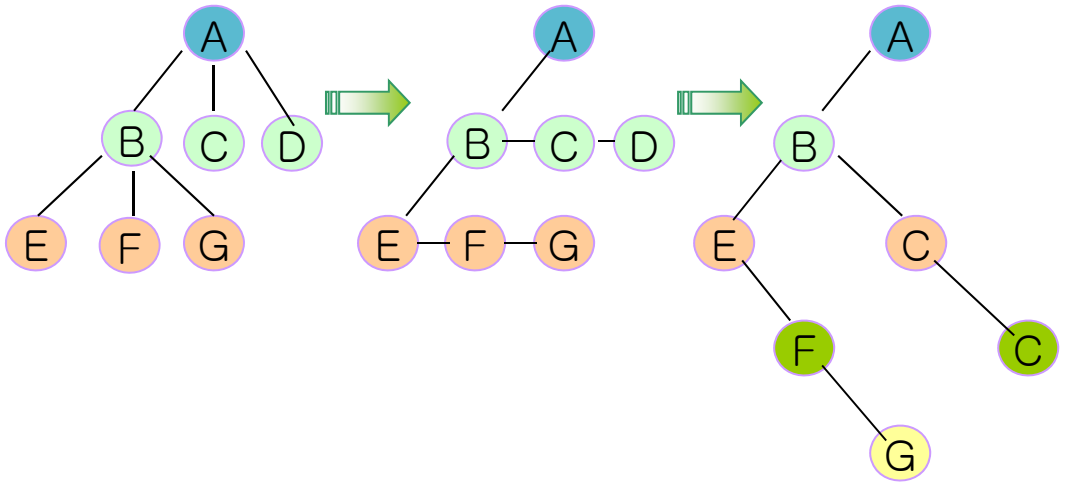
- 저장이 편하다. 배열 선언 int T[MAX];
- 노드의 위치를 쉽게 알 수 있다. 첨자가 0부터 시작한다고 가정
  m 레벨의 k번째 노드의 인덱스 위치는 ? -> $(2^{m-1} - 1) + (k - 1)$
  노드는 $T[(2^{m-1} - 1) + (k - 1)]$에 위치

(2) 연결리스트

- 트리의 크기와 깊이에 관계없이 노드 수 만큼의 데이터를 저장한다.
- 트리의 크기가 증가하여도 기억장소에서 노드를 확보하여 트리를 구성
- 트리 전체를 탐색하는 알고리즘이 배열보다는 복잡하다.

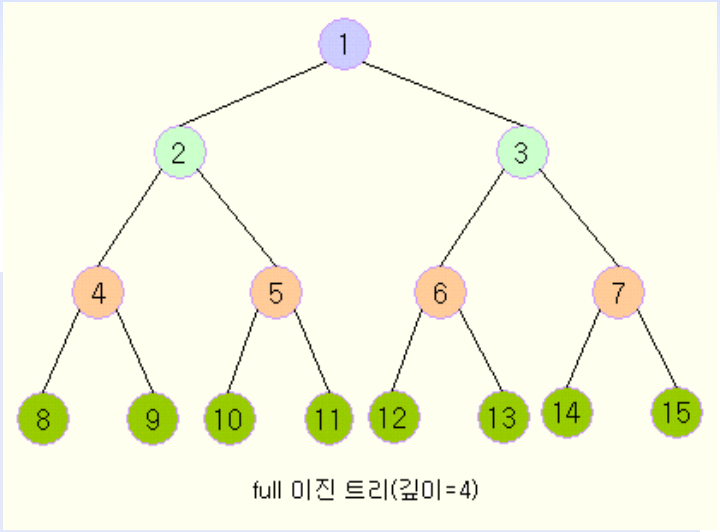
일반 트리도 이진 트리로 변형하여 저장하는 것이 더 효율적이다.

(일반 트리에서 자식노드의 개수가 변하기 때문에 저장 방법이 쉽지 않다.)



트리 왼쪽자식-오른쪽형제 트리 이진 트리

Q/A



full 이진 트리(깊이=4)

다음 트리에 대하여 답을 구해보자

(1) 트리의 깊이는?
(2) 노드번호 6의 형제노드는?
(3) 노드번호 6의 조상노드들은?
(4) 노드번호 6의 오른쪽 자식 노드는?
(5) 트리를 배열 X[ ]에 저장하였을 때(첨자 0부터 시작) X[6]에 저장되는 데이터는 무엇인가?
(6) 연결리스트로 저장한다면 전체 노드의 수는?
(7) 연결리스트로 저장할 경우, 링크필드의 전체 개수는?
(8) 연결리스트로 저장할 경우, 전체 링크필드 중 링크에 값이 있는 필드의 수는?(NULL이 아닌 필드)
(9) 연결리스트로 저장할 경우, 링크의 값이 NULL인 필드의 수는?

Review

⦿ 트리는 중요한 자료구조 중의 하나이다.

이 장에서는 트리의 기본 개념, 부모노드, 자식노드, 형제노드, 이진 트리, complete 이진 트리, full 이진 트리, skewed 이진 트리 등 용어에 대하여 살펴보았다.

⦿ 이진 트리를 컴퓨터에 저장하는 방법은 배열을 사용하는 방법과 연결리스트를 사용하는 방법이 있다. 배열은 간단한 반면, 기억공간 효율성, 확장성에 문제가 있다.

연결리스트 방식은 트리의 노드에 자식 노드를 가리키는 포인터를 사용하여 자식 노드와 연결을 시킨다.

대부분의 응용에서는 연결리스트를 사용한다.