COMP319 Algorithms 1
Lecture 7
Tree

Instructor: Gil-Jin Jang

Tree / Binary tree

Slide credits:

홍석원, 명지대학교, Discrete Mathematics, Spring 2013

김한준, 서울시립대학교, 자료구조 및 실습, Fall 2016

J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

기본용어

트리(tree)

이진 트리(binary tree)

이진 탐색 트리(binary search tree)

TREE

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

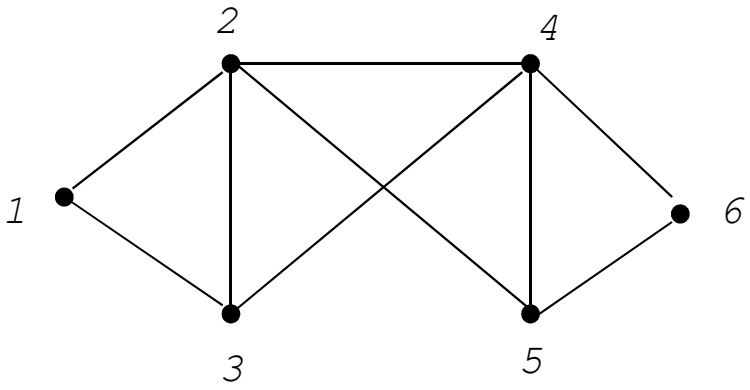
기본 용어

트리(tree)

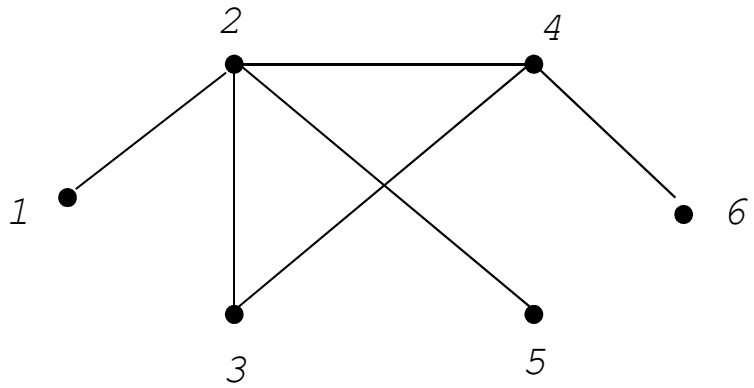
비방향 그래프 G={V, E}에서 모든 정점의 쌍 (u,v)가 연결되어 있고 순환(cycle)을 갖지 않을 때 G를 트리라고 한다.

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

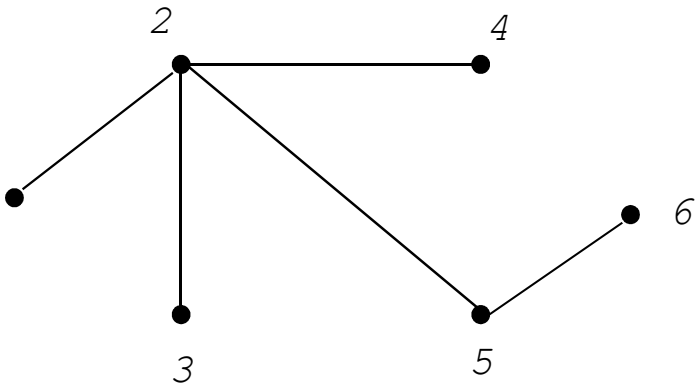
예: G_1과 G_2는 트리가 아니고 G_3와 G_4는 트리이다.



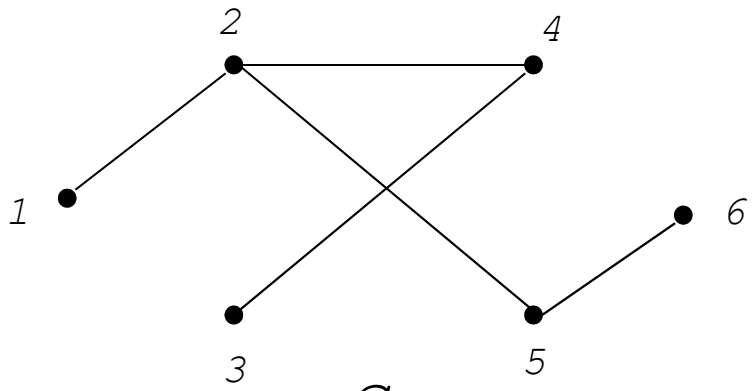
G_1



G_2



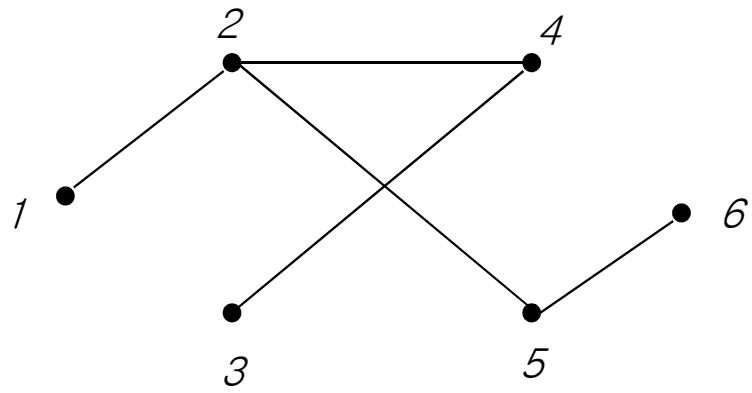
G_3



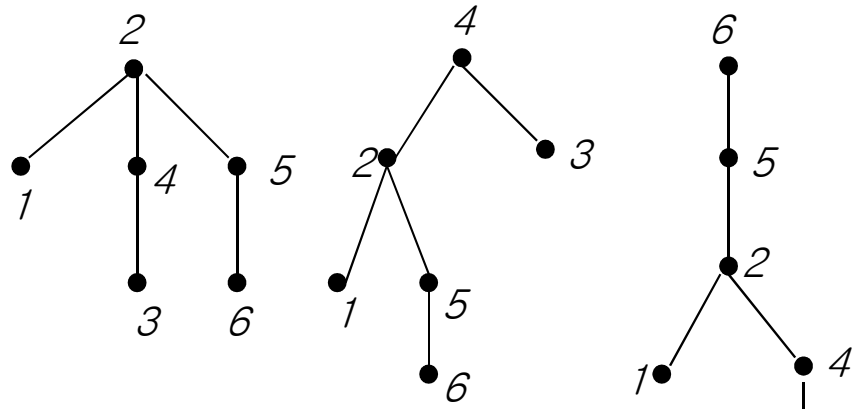
G_4

<정리>
비방향 그래프 G={V, E}의 모든 정점이 연결되어 있고 모든 정점의 쌍 (u, v) 사이에는 유일한 경로가 존재하면 그래프 G는 트리이다.

뿌리 트리(rooted tree)
트리의 한 정점에서부터 시작하여 연결선들이 방향을 갖는 트리를 뿌리 트리라고 하며 이 정점을 뿌리(root)라고 한다.



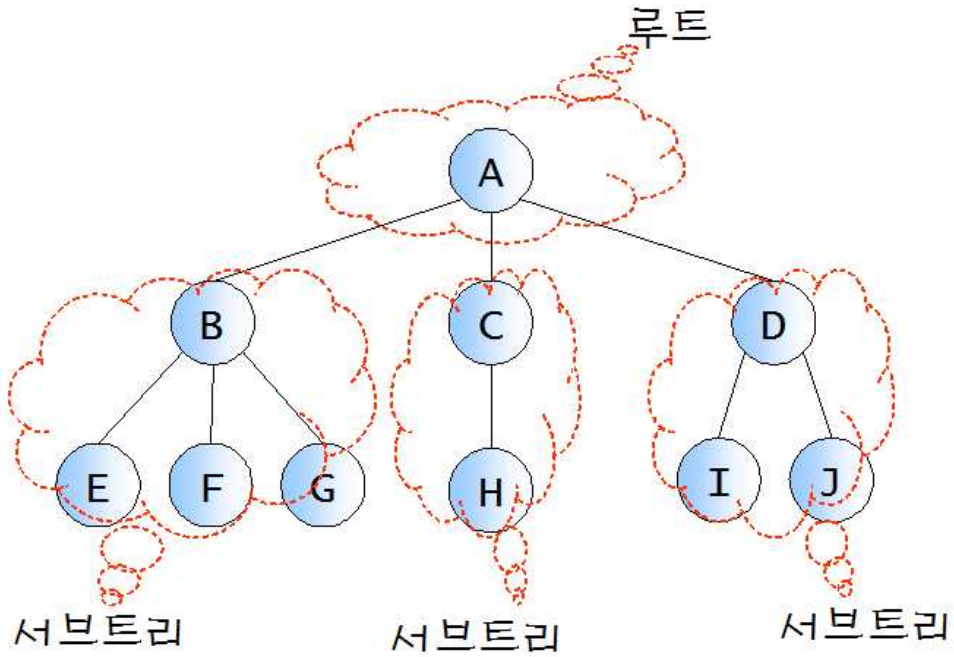
G



G의 뿌리 트리

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

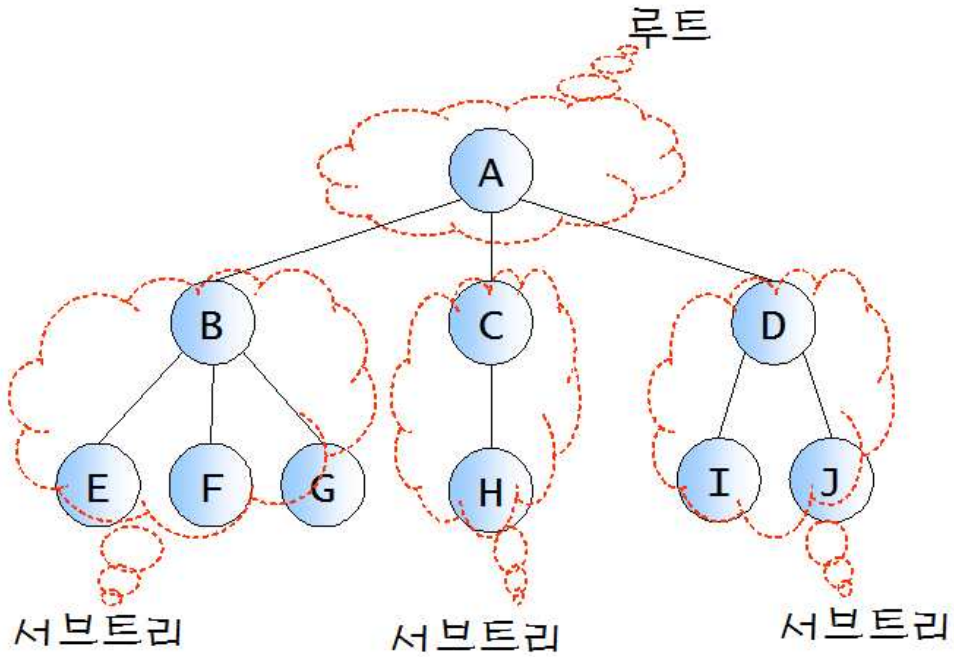
트리의 용어



- *노드(node): 트리의 구성요소*
- *루트(root): 부모가 없는 노드(A)*
- *서브트리(subtree): 하나의 노드와 그 노드들의 자손들로 이루어진 트리*

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

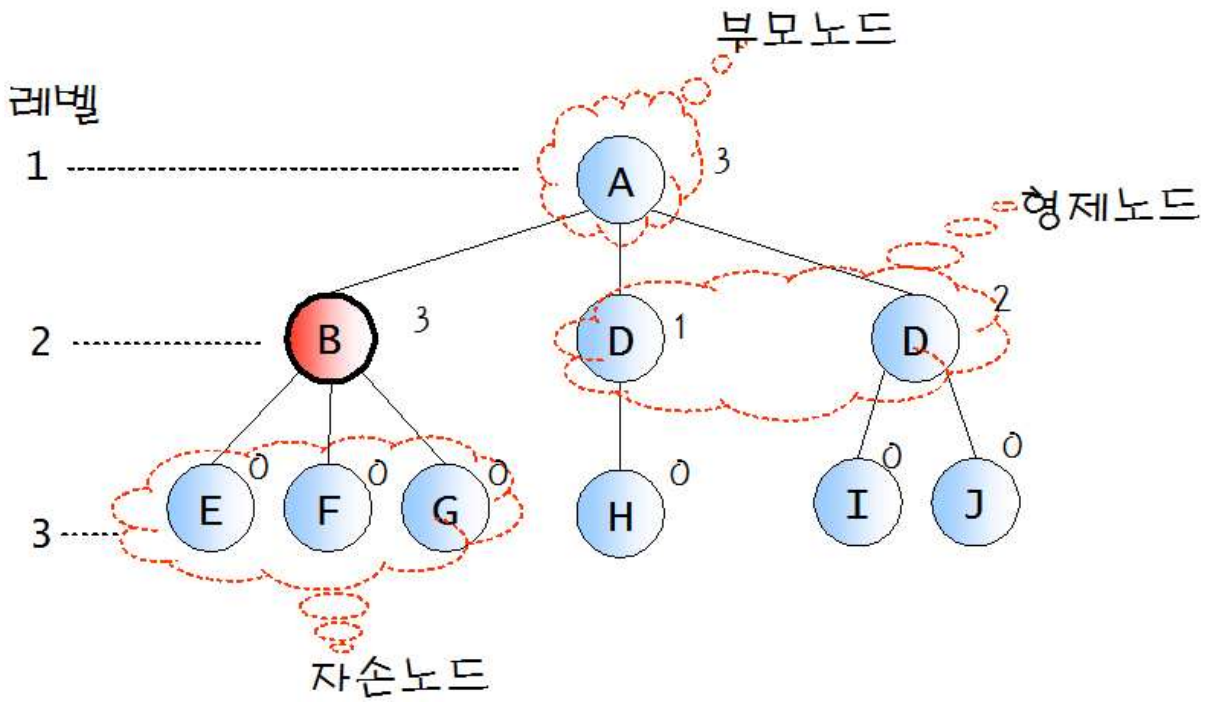
트리의 용어



- *단말노드(terminal node): 자식이 없는 노드(E,F,G,H,I,J)*
- *비단말노드(non-terminal node): 적어도 하나의 자식을 가지는 노드(A,B,C,D)*

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

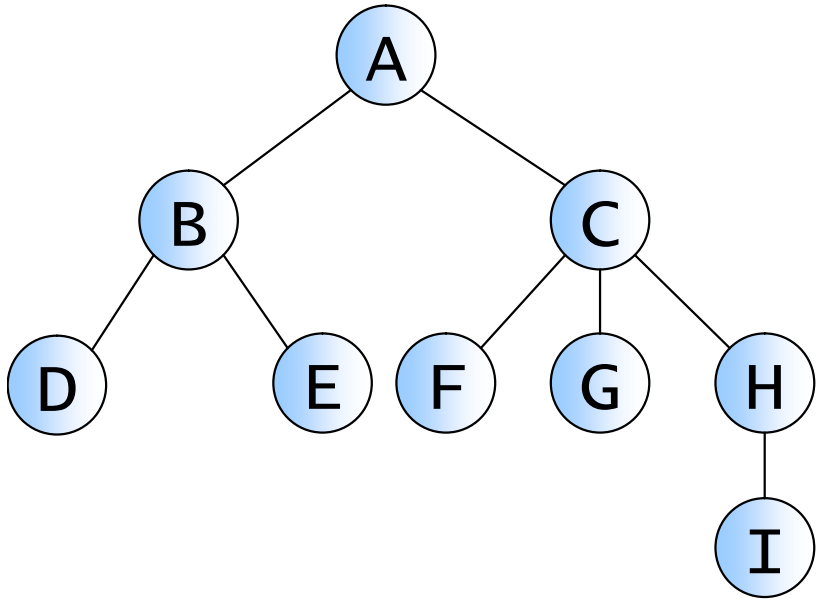
트리의 용어



- *노드(node):*
  - *자식(child), 부모(parent), 형제(sibling), 조상(ancestor), 자손(descendant)*
- *레벨(level): 트리의 각층의 번호*
- *높이(height): 트리의 최대 레벨(3)*
- *차수(degree): 노드가 가지고 있는 자식 노드의 개수*

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

예제



- A는 루트 노드이다.
- B는 D와 E의 부모노드이다.
- C는 B의 형제 노드이다.
- D와 E는 B의 자식노드이다.
- B의 차수는 2이다.
- 위의 트리의 높이는 4이다.

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

부모(parent), 자식(child), 형제(sibling)
서로 인접한 정점들 중에서 뿌리에 가까운 위치에 있는 정점을 부모, 먼 위치에 있는 정점을 자식이라고 한다.
그리고 동일한 부모를 갖는 정점들은 형제(sibling)라고 한다.

잎(leaf) : 자식이 없는 정점(terminal)

내부 정점(internal vertex) :자식을 갖는 모든 정점들

조상(ancestor): 루트로부터 그 정점에 이르는 경로 상에 존재하는 모든 정점들

자손(descendant): 그 정점을 조상으로 하는 모든 정점들

레벨(level) : 뿌리에서 시작하여 정점들이 연결되는 단계를

깊이(depth) 혹은 높이(height) :트리가 갖는 최대 레벨 수

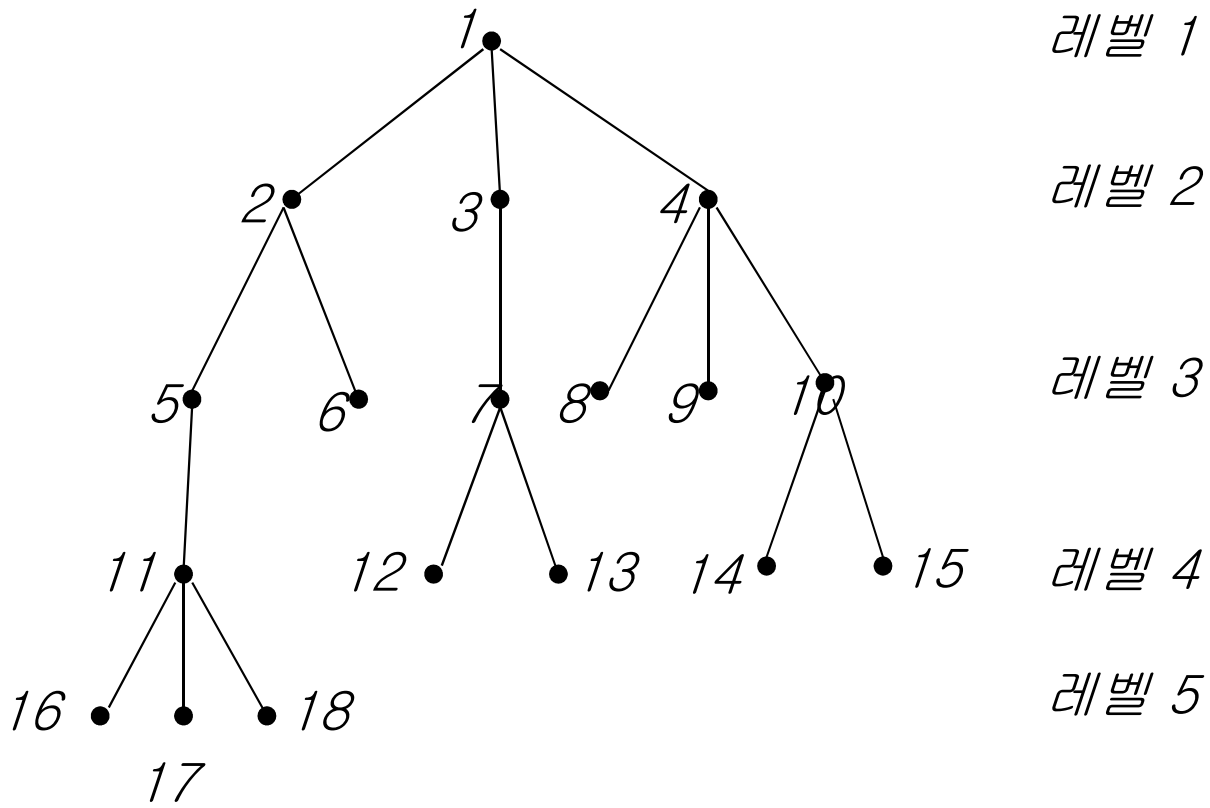
Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

예:

정점1과 (2, 3, 4), 정점 2와 (5,6), 5와 11, 11과 (16, 17, 18)은 부모와 자식의 관계

정점 8의 조상은 (1, 4) 이며, 정점 17의 조상은 (1, 2, 5, 11)

정점2의 자손은 (5, 6, 11, 16, 17, 18)



Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

순서 트리(ordered tree) : 각 정점의 자식들이 왼쪽에서 오른쪽으로 순서를 갖고 정렬이 되는 트리

n-트리(n-ary tree) : 자식이 최대 n개가 존재하는 트리

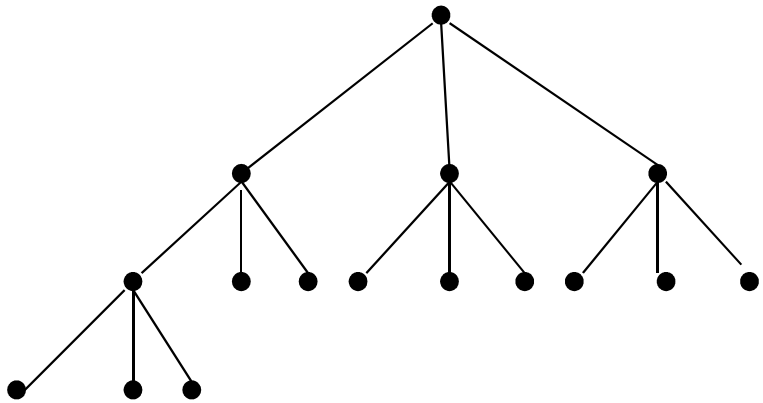
완전 n-트리(complete n-tree)
모든 부모의 자식들이 정확히 n개씩 존재하고 그 정점들이 중간에 비지 않고 순서대로 채워져 있는 트리

포화 n-트리(full n-tree)
마지막 레벨의 정점들이 모두 채워진 트리

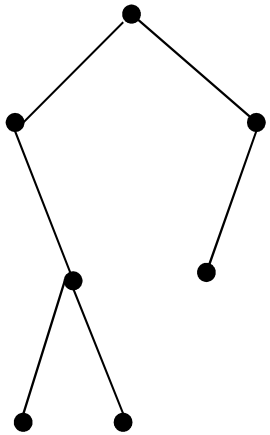
이진 트리(binary tree) : n=2인 트리

완전 이진 트리(complete binary tree)
모든 부모 정점이 왼쪽 자식과 오른쪽 자식을 갖는 이진 트리

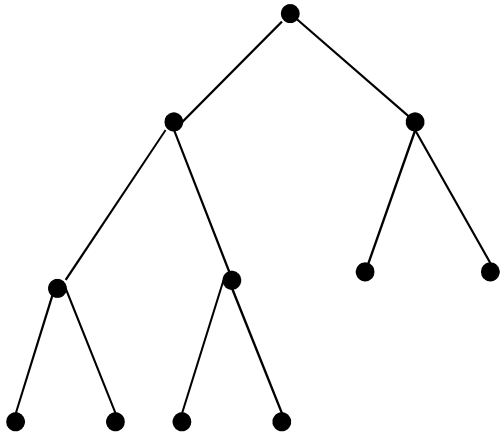
Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I



(a) 완전3-트리



(b) 이진 트리



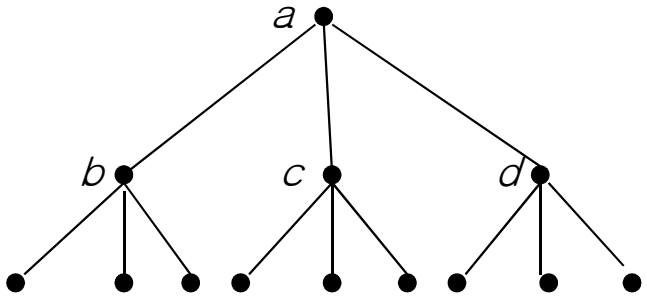
(c) 완전 이진 트리

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

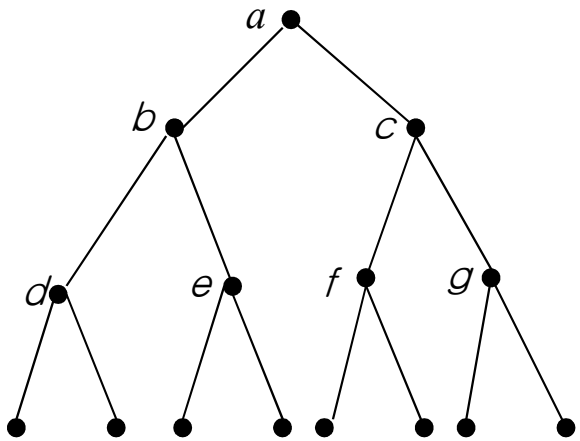
<정리>
n개의 정점을 갖는 트리의 총 연결선의 수는 n-1개이다.

트리는 그래프와 달리 뿌리를 제외하고는 모든 정점들이 뿌리를 향하여 하나의 연결선을 갖는다. 만약 그렇지 않다면 하나 이상의 경로가 존재하게 된다. 따라서 뿌리 노드를 제외한 모든 정점 n-1개는 n-1 연결선을 갖는다.

<정리>
포화 n-트리(full n-ary tree)가 k개의 내부 정점을 갖는다면 총 정점의 수는 nk+1 이다.



포화 3-트리



포화 이진 트리

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

기본용어

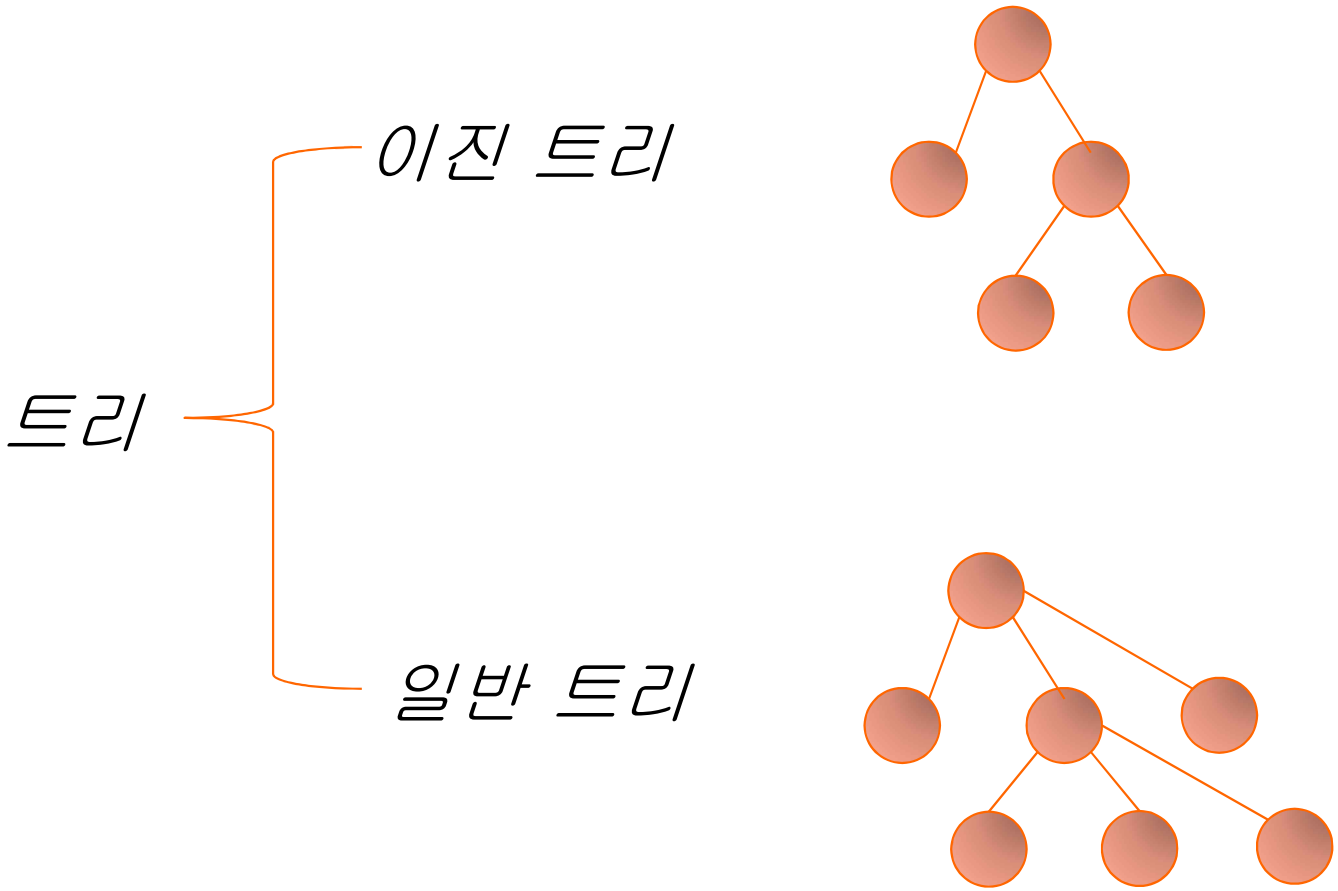
이진 트리(binary tree)

이진 탐색 트리(binary search tree)

BINARY TREE

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

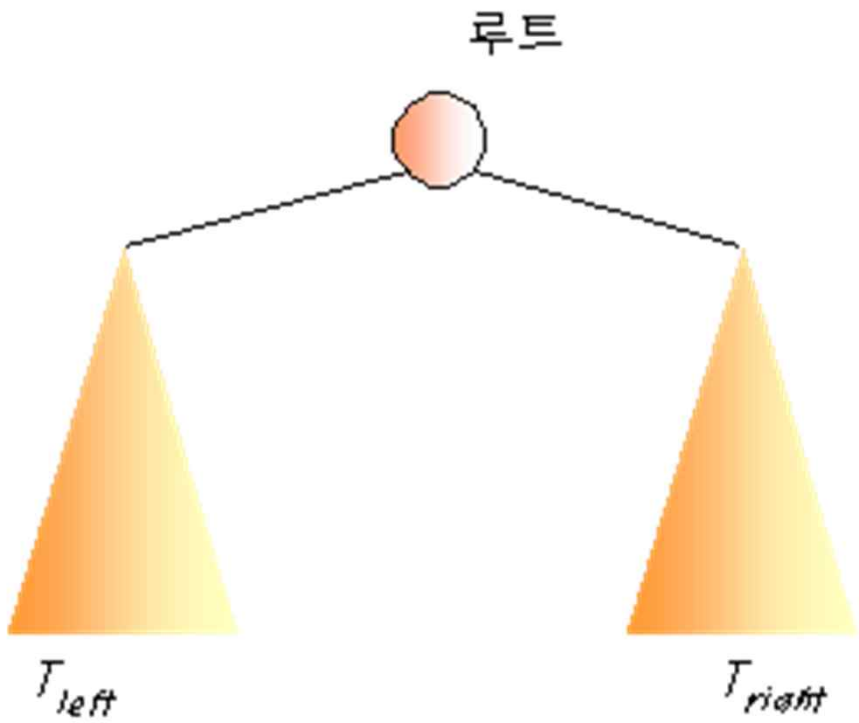
트리의 종류



Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

이진 트리 (binary tree)

- 이진 트리(binary tree) : 모든 노드가 2개의 서브 트리를 가지고 있는 트리
  - 서브트리는 공집합일수 있다.
- 이진트리의 노드에는 최대 2개까지의 자식 노드가 존재
- 모든 노드의 차수가 2 이하가 된다
  - 구현하기가 편리함
- 이진 트리에는 서브 트리간의 순서가 존재



Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

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.

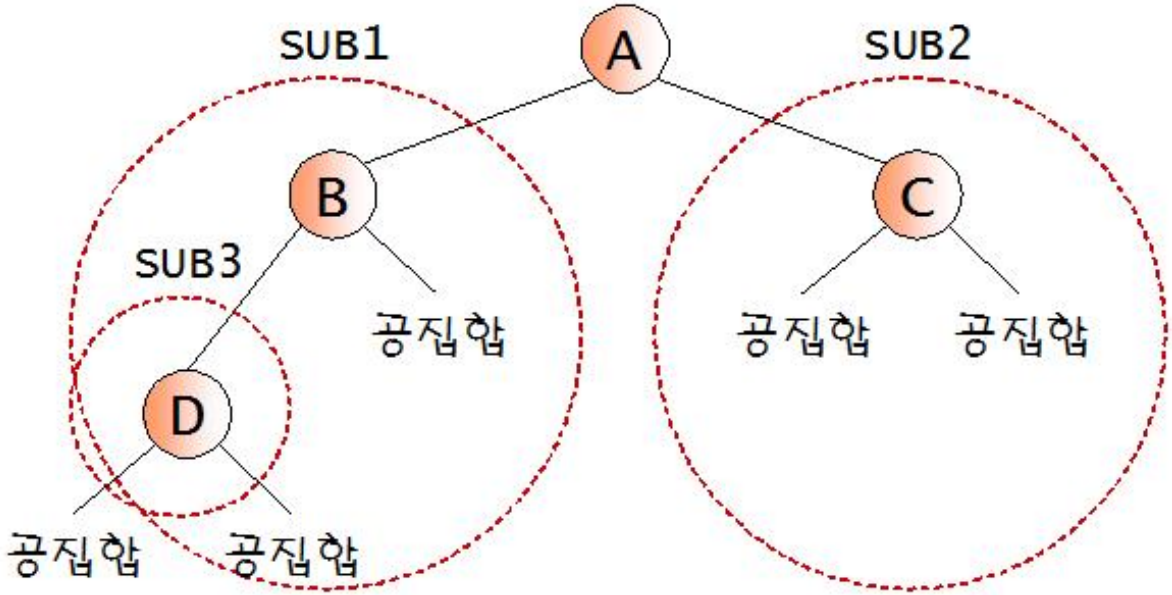


그림 7.8 이진트리 검증

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

이진 트리의 성질

- 노드의 개수가 *N* 개이면 간선의 개수는 *N*-1 개

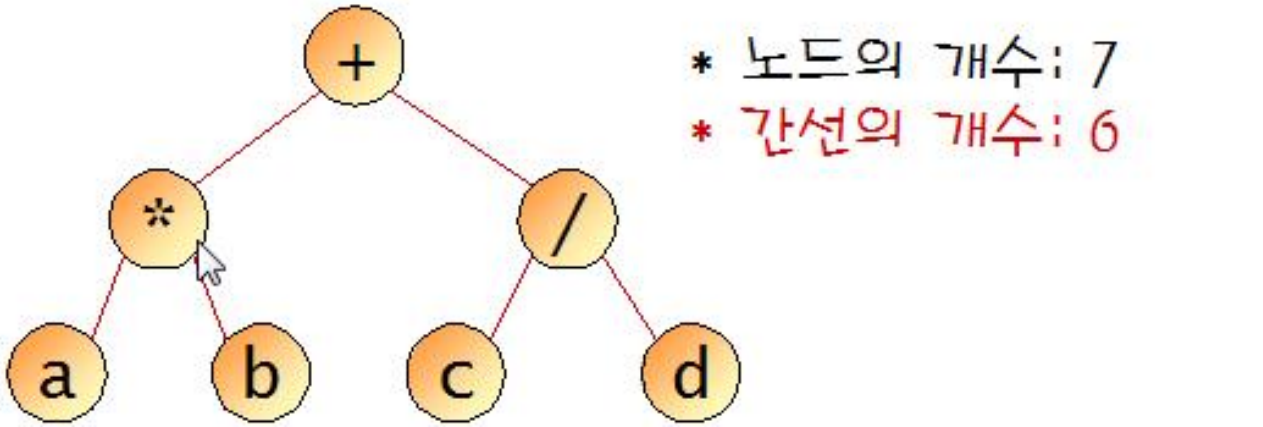
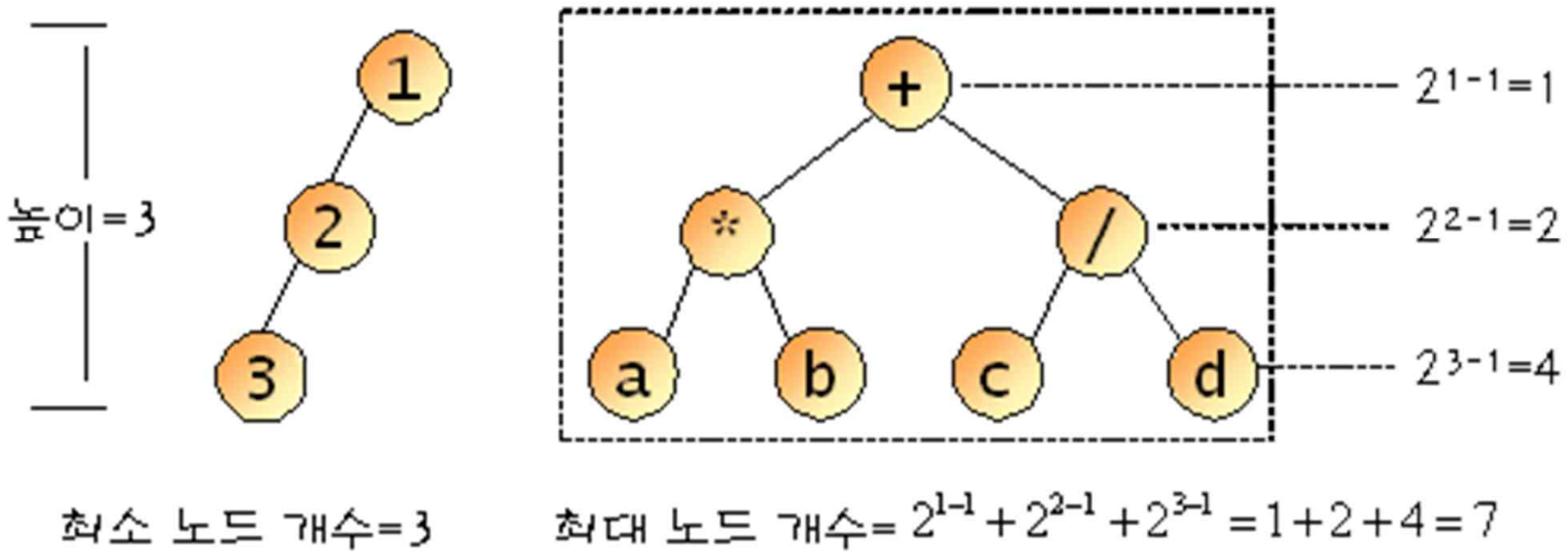


그림 7.10 노드의 개수와 간선의 개수와의 관계

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

이진트리의 성질

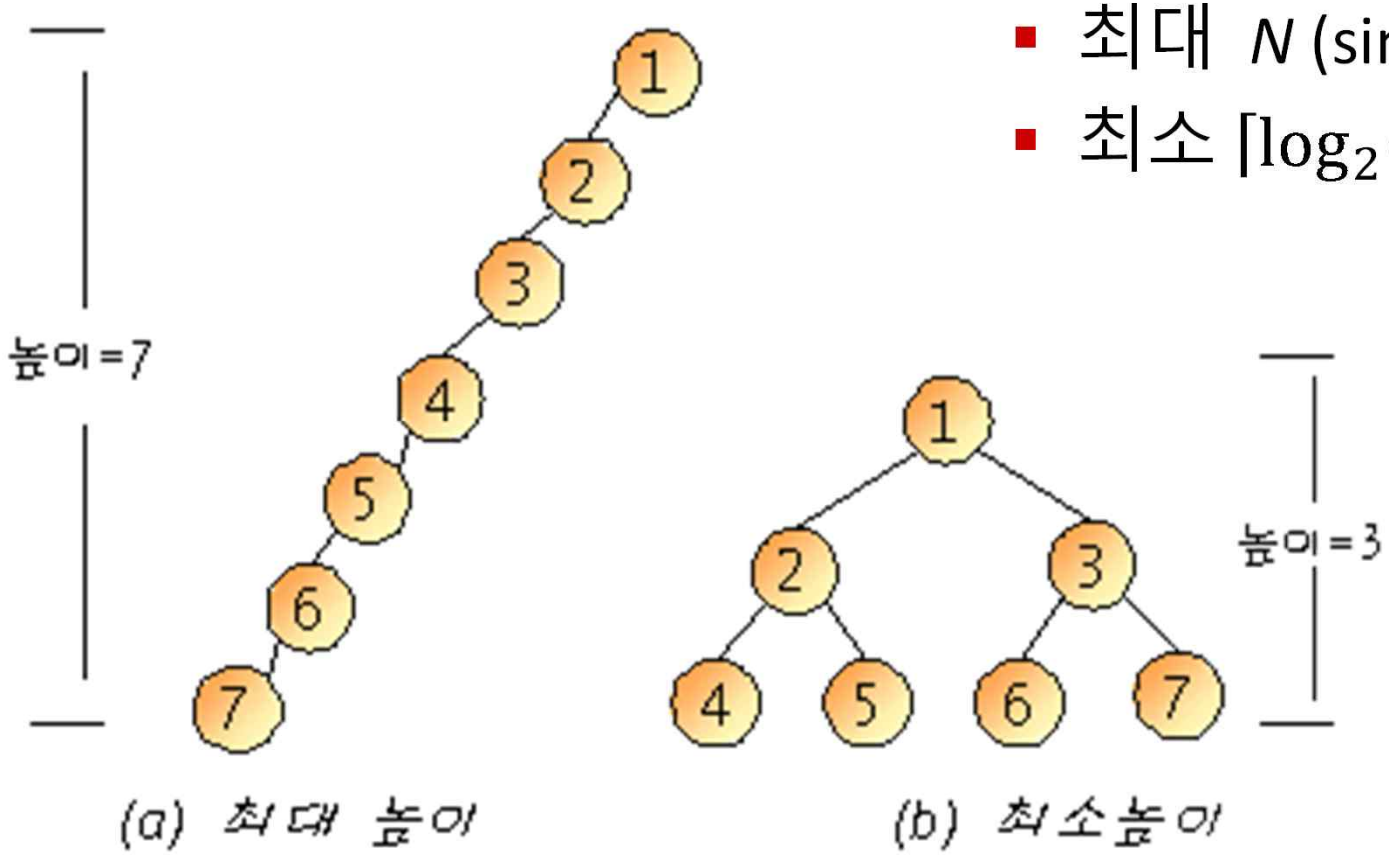
- 높이가 h인 이진트리의 경우
  - 최소 h개의 노드를 가지며(single child)
  - 최대 $2h$-1개의 노드를 가진다(full binary tree)



Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

이진 트리의 성질

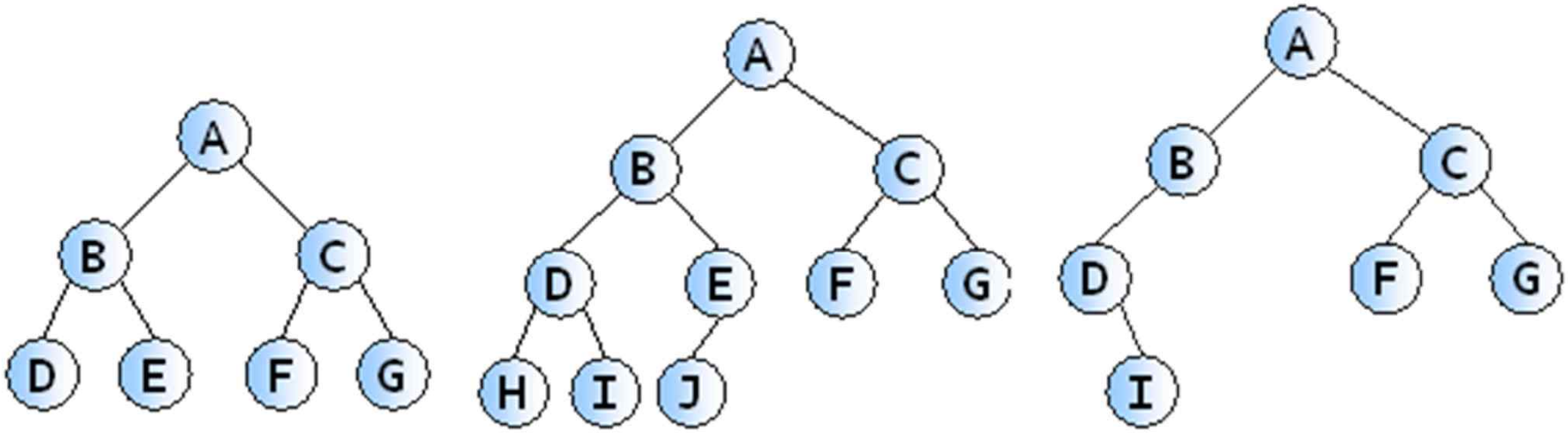
- N개의 노드를 가지는 이진트리의 높이
  - 최대 N (single child)
  - 최소 $\lceil \log_2(N+1) \rceil$ (full)



Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

이진 트리의 분류

- 포화 이진 트리(full binary tree)
- 완전 이진 트리(complete binary tree)
- 기타 이진 트리



(a) 포화이진트리 (b) 완전이진트리 (c) 기타이진트리

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

포화 이진 트리

- **(full binary tree)** 트리의 각 레벨에 노드가 꽉 차있는 이진트리를 의미한다.
- 포화 이진 트리에는 다음과 같이 각 노드에 번호를 붙일 수 있다.

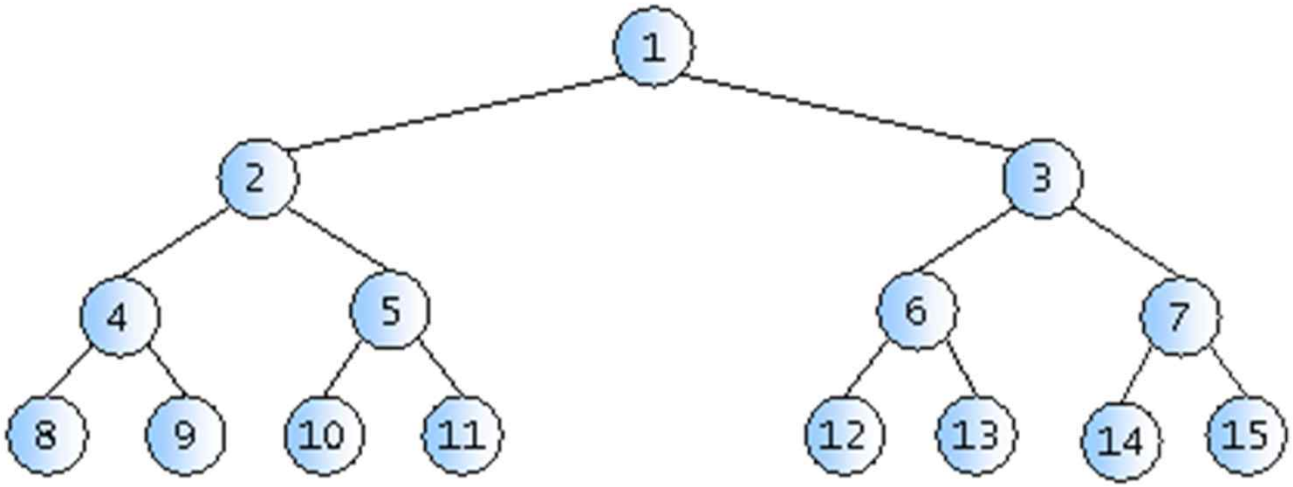


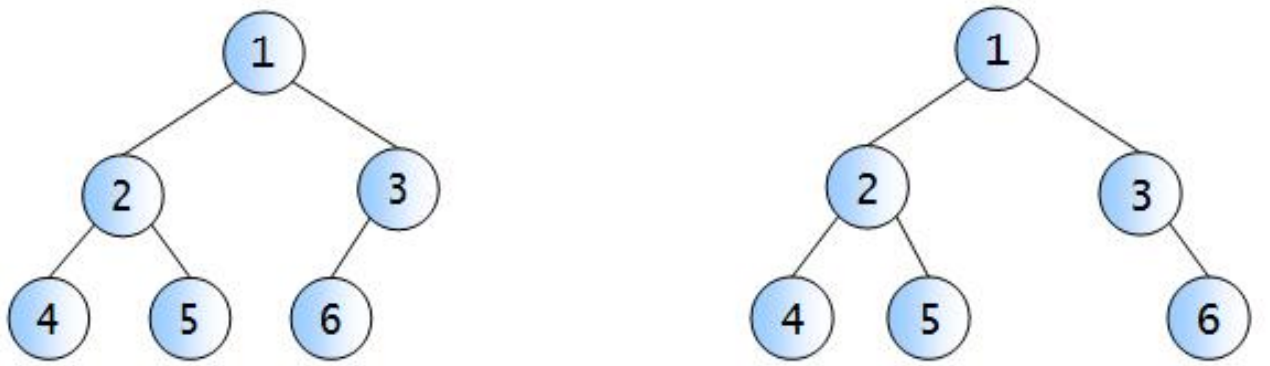
그림 7.15 포화이진트리에서의 노드의 번호

전체 노드 개수 : $2^{1-1}+2^{2-1}+2^{3-1}+\ldots+2^{k-1}=\sum_{i=0}^{k-1}2^i=2^k-1$

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

완전 이진 트리

- **(complete binary tree)** 레벨 1부터 h-1까지는 노드가 모두 채워져 있고 마지막 레벨 h에서는 왼쪽부터 오른쪽으로 노드가 순서대로 채워져 있는 이진트리
- 포화 이진 트리와 노드 번호가 일치



(a) 완전이진트리 (b) 완전이진트리가 아님

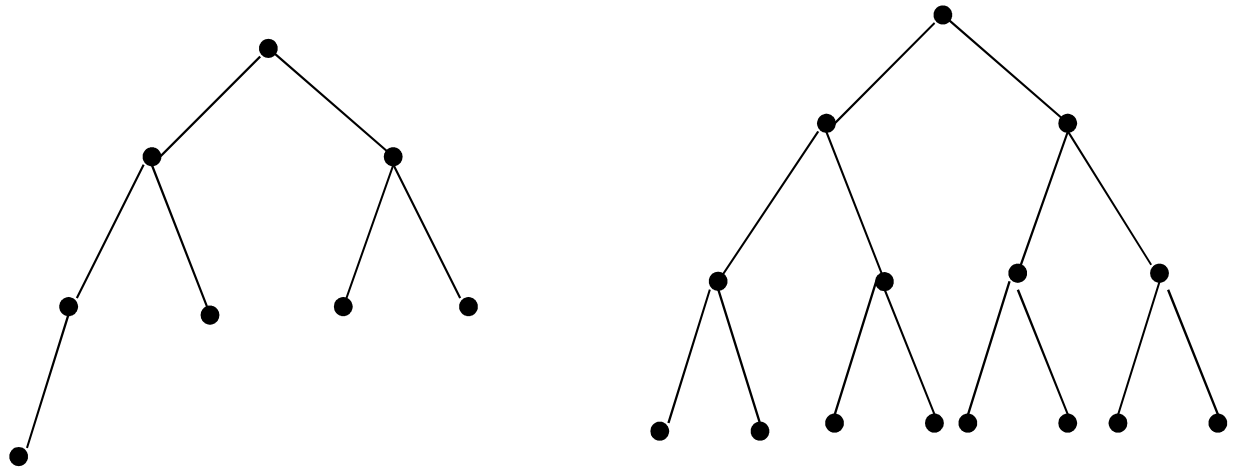
그림 7.16 완전 이진트리의 예

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

이진 트리의 특성

예:

다음의 트리들은 모두 높이 4를 갖는 완전 이진 트리이다.
(a)는 높이 4인 완전 이진 트리로서 가장 적은 수의 정점으로 구성된 예를 보여 주고 있다. 이때 총 정점의 수는 8이다.
(b)는 포화 완전 트리로서 총 정점의 수는 15개이다.



(a) (b)

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

완전 이진 트리의 정점의 수

<정리>
완전 이진 트리(complete binary tree)인 경우 트리의 높이가 h 라면 총 정점의 수 N 은

$$2^{h-1} \le N \le 2^h - 1$$

각 레벨에서 정점의 수는 $1, 2, 2^2, ..., 2^{h-1}$이므로 높이가 h 인 포화 이진 트리의 정점의 수는 다음과 같이 계산된다.

$$1+2+2^2+2^3+...+2^{h-1} = 2^h-1$$

높이가 h인 완전 이진 트리의 갯수는 가장 적을 경우 높이가 h-1인 포화 이진 트리의 수 보다 1이 많다. 즉

$$(1+2+2^2+2^3+...+2^{h-2})+1 = 2^{h-1}$$

따라서 높이가 h인 이진 트리의 정점의 수는 위와 같이 된다.

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

완전 이진 트리의 높이

<정리>
N개의 정점을 갖는 완전 이진 트리의 높이는 $\lfloor \log_2 N \rfloor + 1$이다.

앞의 정리의 식 양변을 log를 취하면,

$$h\text{-}1 \le \log_2 N \le \log(2^h\text{-}1)$$

이다. $\log(2^h\text{-}1) < \log(2^h) = h$ 이므로

$$h\text{-}1 \le \log_2 N < h$$
$$h \le \log_2 N + 1 < h+1$$

따라서 N개의 정점을 갖는 완전 이진 트리의 높이는 $\lfloor \log_2 N \rfloor + 1$ 이 된다.

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

배열(array)

단순 연결 리스트(linked list)

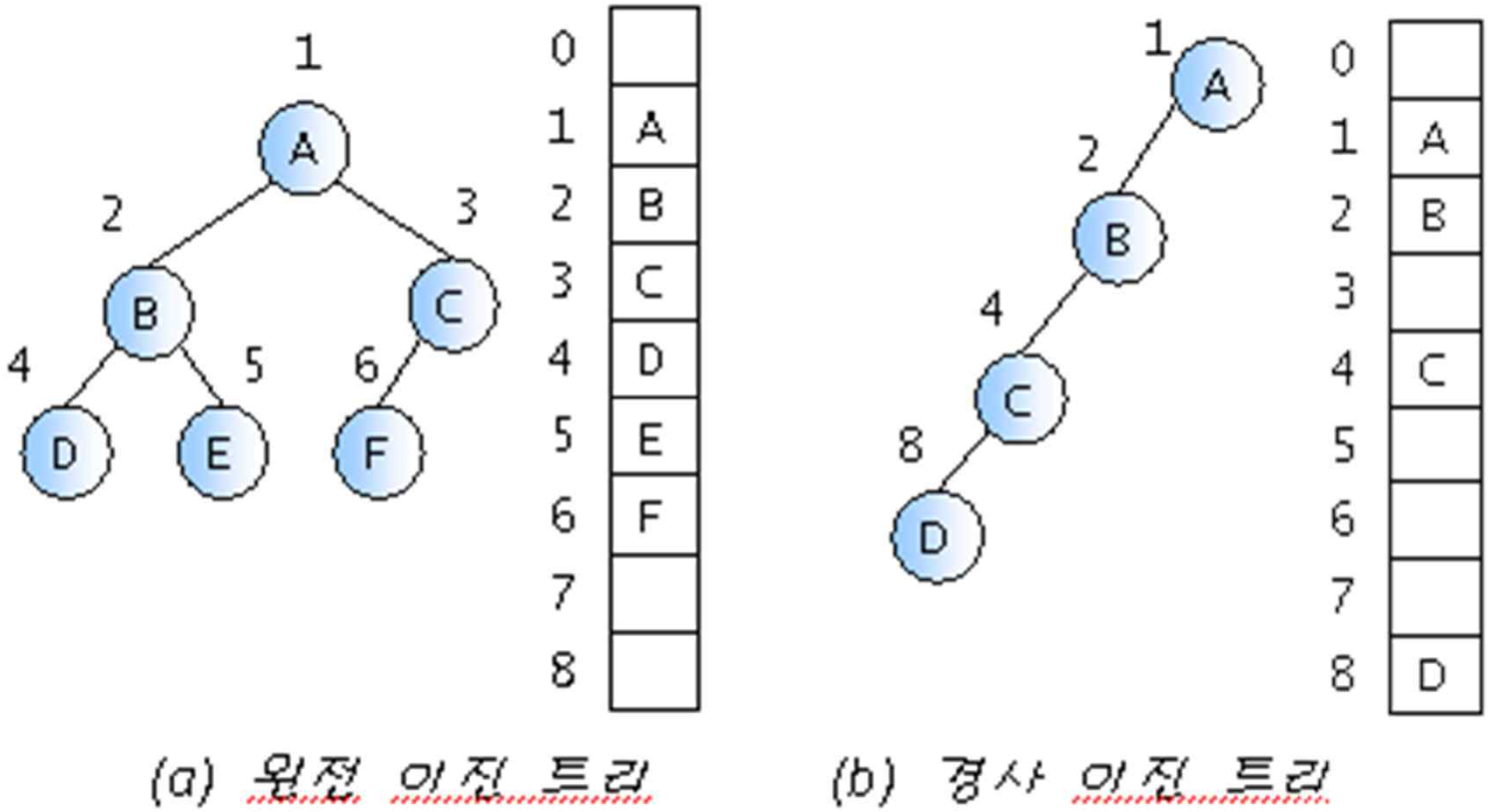
이중 연결 리스트(doubly linked list)

BINARY TREE IMPLEMENTATION

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

트리의 배열 표현

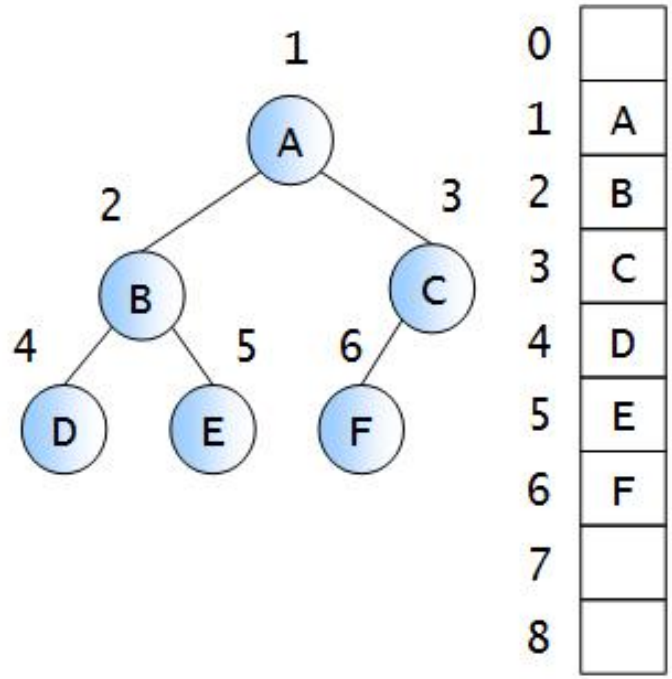
- 모든 이진 트리를 포화 이진 트리라고 가정하고 각 노드에 번호를 붙여서 그 번호를 배열의 인덱스로 삼아 노드의 데이터를 배열에 저장하는 방법



Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

부모와 자식 인덱스 관계

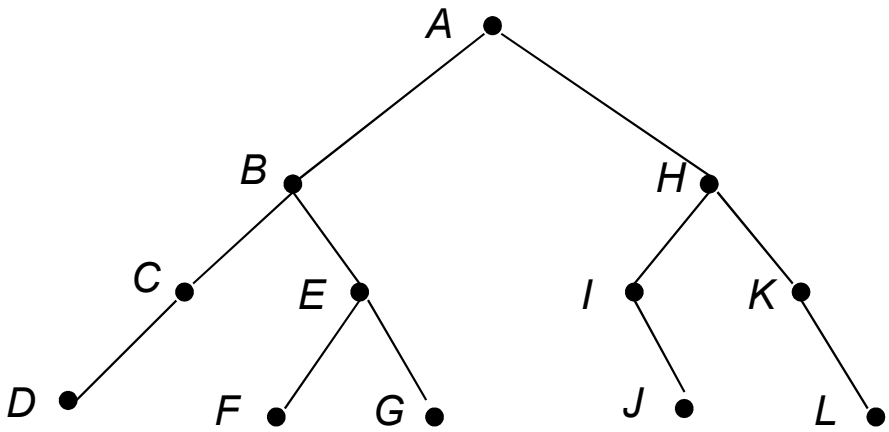
- 노드 *i*의 부모 노드 인덱스 = *i*/2
- 노드 *i*의 왼쪽 자식 노드 인덱스 = 2*i*
- 노드 *i*의 오른쪽 자식 노드 인덱스 = 2*i*+1

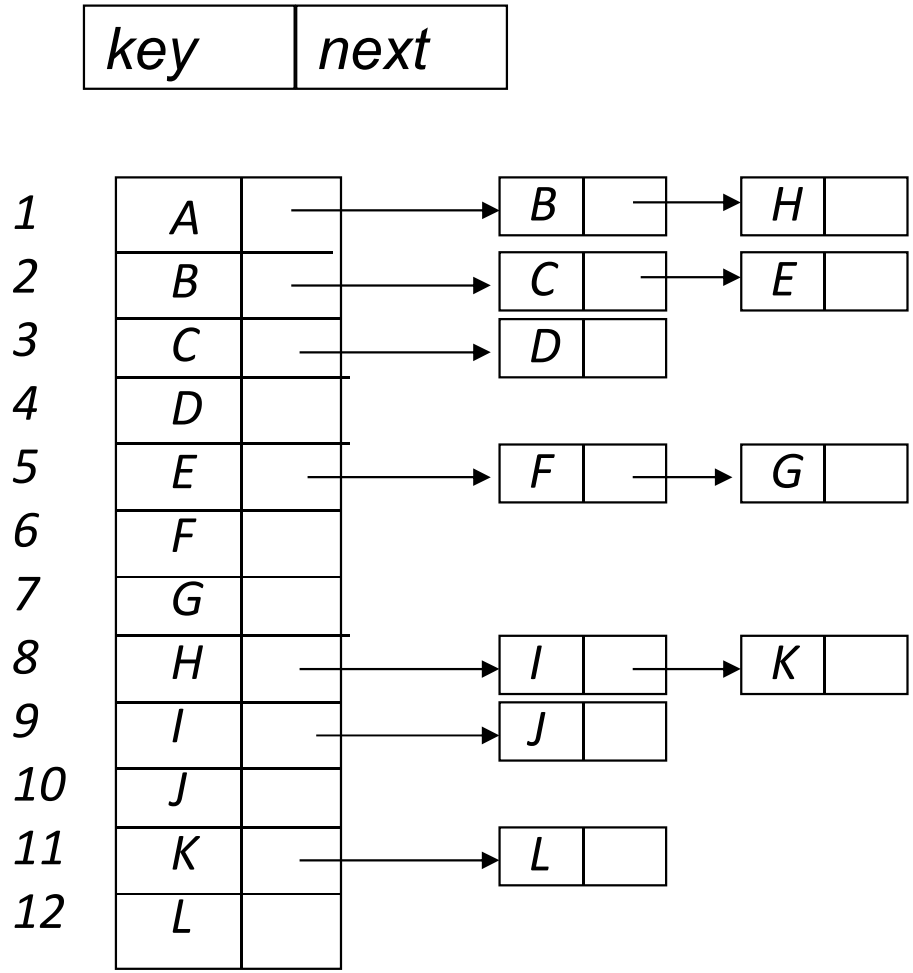


인덱싱의 편이성을 위해 맨 처음 저장공간을 사용하지 않는다.
인덱스 0를 root index로 간주한다.
만약 맨 처음 공간을 사용한다면 루트와 첫번째를 구분하기 위한 다른 방법이 필요하다.
Left(i) = 2i+1
Right(i) = 2(i+1)
Parent(i) = (i+1)/2

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

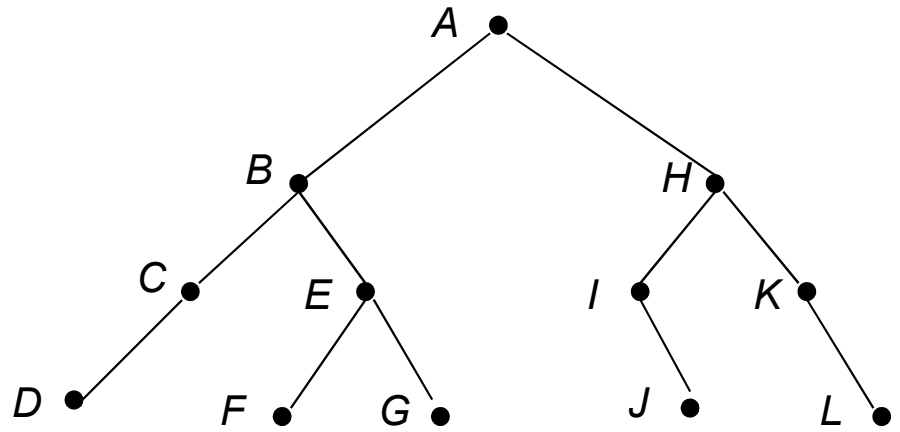
트리구현: 단순 연결 리스트

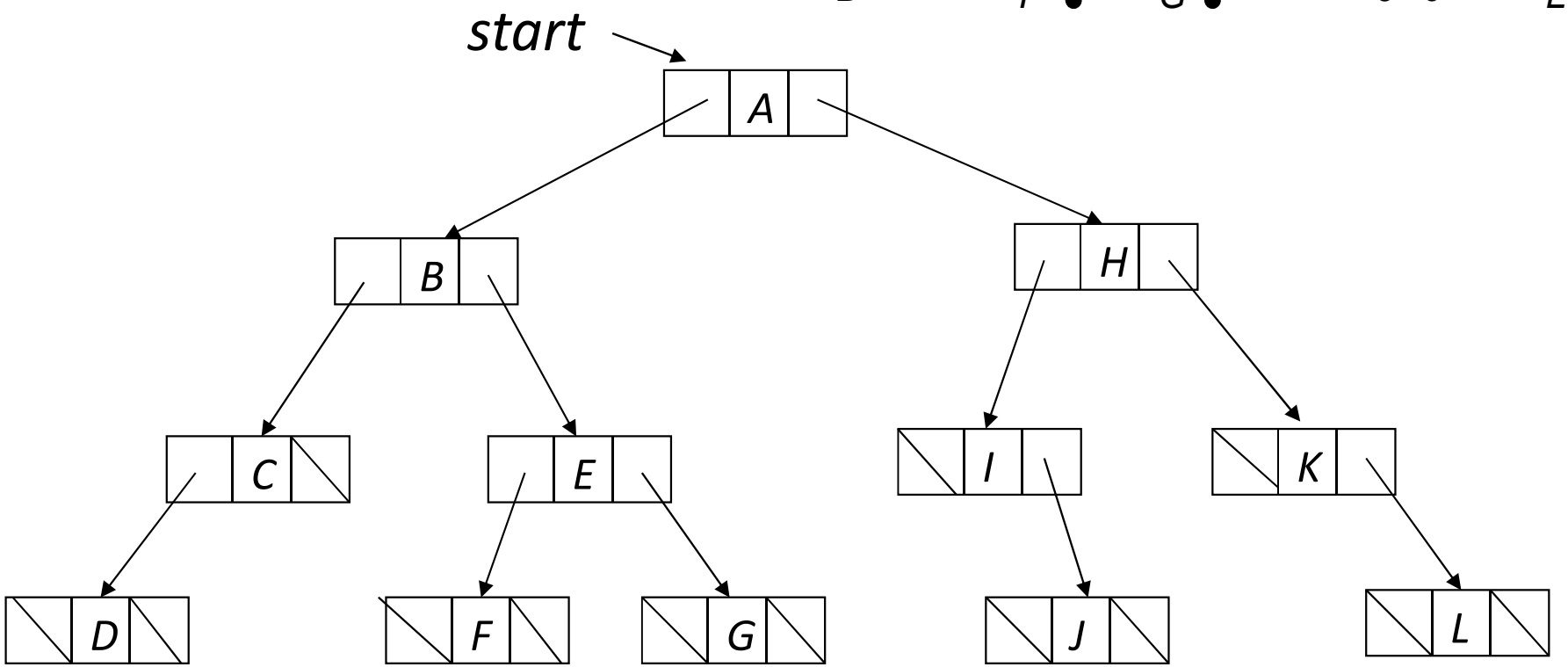




Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

트리구현: 이중 연결 리스트





Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Doubly linked list: C code example

```
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>

typedef struct TreeNode {
   int data;
   struct TreeNode *left,
   *right;
} TreeNode;

//       n1
//      /  |
//   n2    n3
```

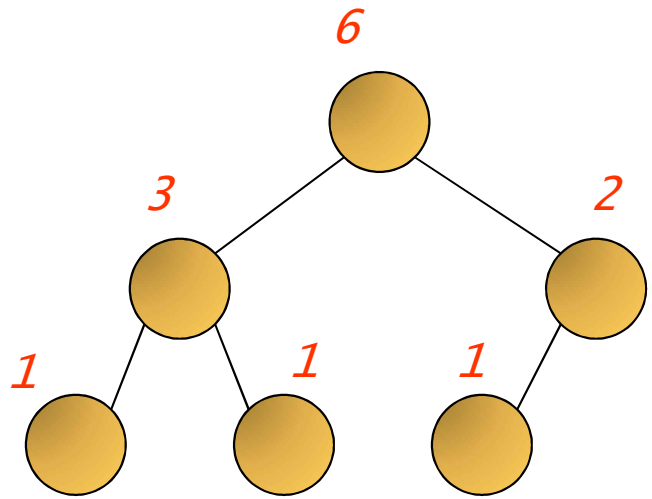
```
void  main()
{
   TreeNode *n1, *n2, *n3;

   n1= (TreeNode*)
    malloc(sizeof(TreeNode));
   n2= (TreeNode*)
    malloc(sizeof(TreeNode));
   n3= (TreeNode*)
    malloc(sizeof(TreeNode));

   // 첫번째 노드를 설정한다.
   n1->data = 10;
   n1->left = n2;
   n1->right = n3;

   // 두번째 노드를 설정한다.
   n2->data = 20;
   n2->left=n2->right=NULL;

   // 세번째 노드를 설정한다.
   n3->data = 30;
   n3->left=n3->right=NULL;
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

이진 트리 연산: 노드 개수

- 탐색 트리안의 노드의 개수를 계산
- 각각의 서브트리에 대하여 순환 호출한 다음,
  반환되는 값에 1을 더하여 반환

```
int get_node_count(TreeNode *node)
{
  if( node != NULL ) return
      get_node_count(node->left)
      + 1
      + get_node_count(node->right);
  else return 0;
}
```

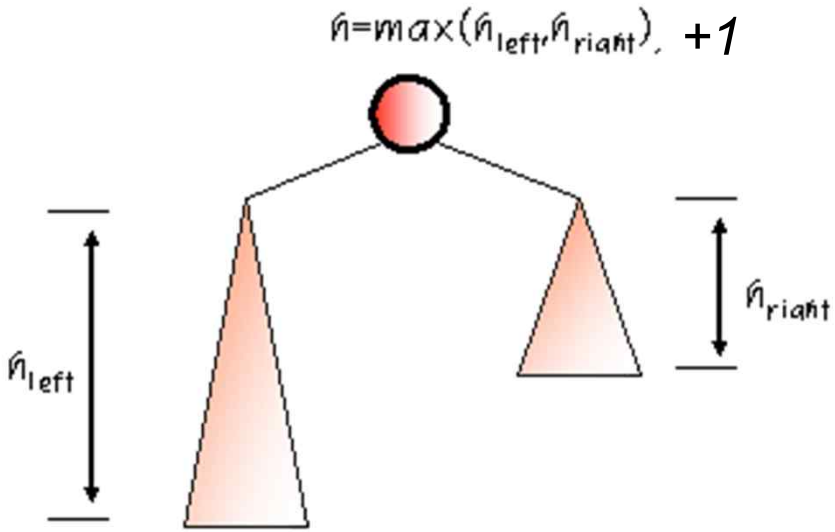


Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

이진 트리 연산: 높이

- 서브트리에 대하여 순환호출하고 서브트리들의 반환값 중에서 최대값을 구하여 반환

```
int get_height(TreeNode *node)
{
  if( node != NULL )
     return 1 +
       max(get_height(node->left),
           get_height(node->right));
  else return 0;
}
```



Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Next topics:

Heaps

Binary search tree (BST)

END OF LECTURE 7

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I