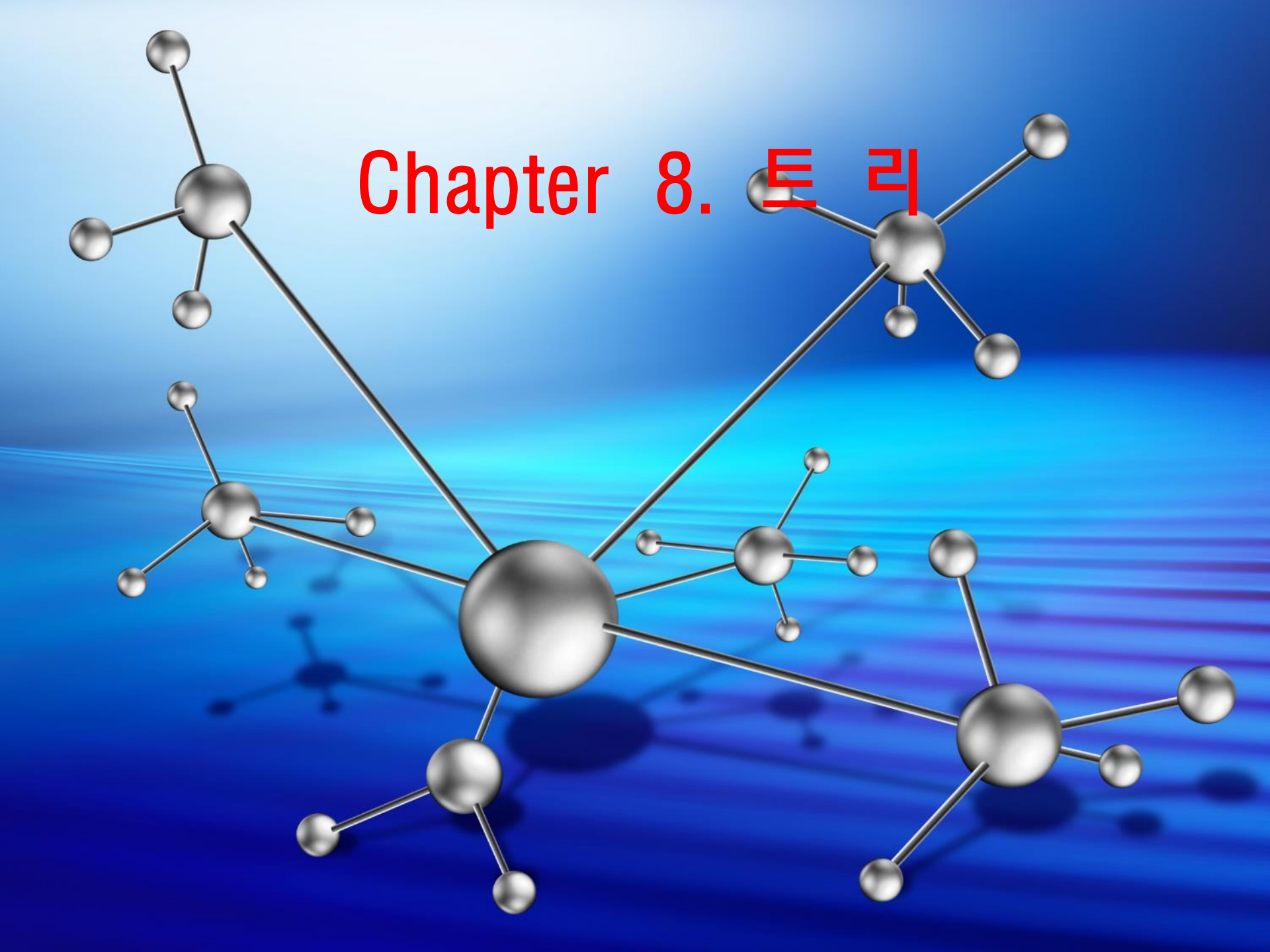

Chapter 8. 트 리

개요

- 트리의 개념, 용어, 방향 트리, 분자식과 같은 응용 분야들에 관해 알아봄
- 이진 트리의 종류, 배열 및 연결 리스트에 의한 이진 트리의 표현법과 더불어 중순위 탐방, 전순위 탐방 그리고 후순위 탐방에 대해서도 살펴봄
- 생성 트리와 최소 비용 생성 트리, 프림의 알고리즘과 크루스칼의 알고리즘을 알아봄
- 문법의 파싱, 허프만 코드, 결정 트리, 게임 등의 분야에서 트리의 응용을 고찰함

CONTENTS

트리(Tree)

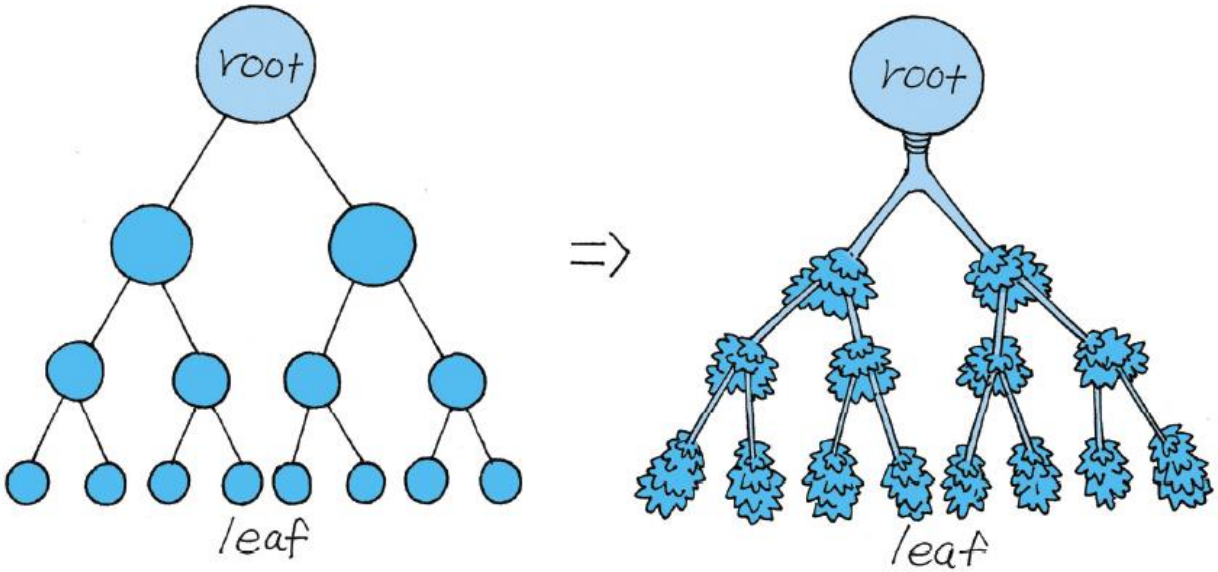
- 그래프 모양이 나무를 거꾸로 세워놓은 것처럼 생겼다고 해서 불리는 이름인데 트리(Tree) 또는 수형도(樹型圖)라고 함
- 그래프의 특별한 형태로서 컴퓨터를 통한 자료 처리와 응용에 있어서 매우 중요한 역할을 담당함
- 이진 트리의 경우에는 산술적 표현이나 자료 구조 등을 매우 간단하게 표현할 수 있는 장점이 있음
- 컴퓨터 기술의 발전과 더불어 수많은 응용 분야에 적용 가능함

8.1 트리의 기본 개념

정의 8-1 트리(tree)는 하나 이상의 노드(node)로 구성된 유한 집합으로서 다음의 2가지 조건을 만족한다.

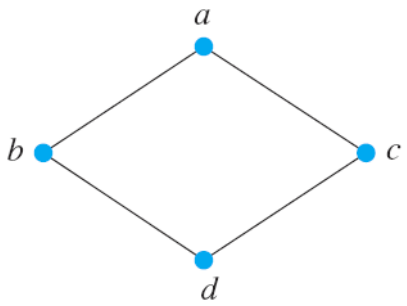
(1) 특별히 지정된 노드인 루트(root)가 있고,

(2) 나머지 노드들은 다시 각각 트리이면서 연결되지 않는(disjoint) $T_1, T_2, \cdots, T_n (N \geq 0)$으로 나누어진다. 이때 $T_1, T_2, \cdots, T_n$을 루트의 서브 트리(subtree)라고 한다.

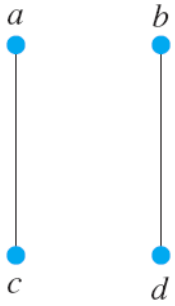


예제 8-1 다음의 여러 그래프들이 트리에 해당되는지를 판단해보자.

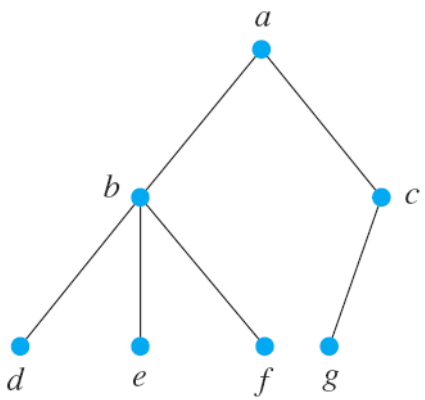
(1)



(2)



(3)



(4)

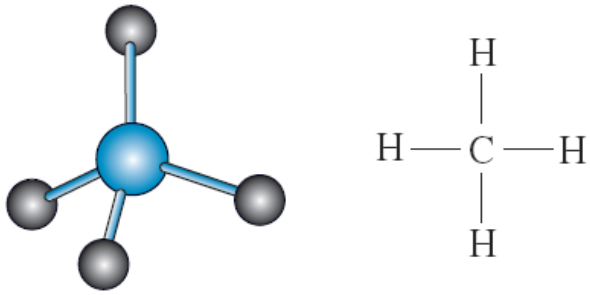
풀이 (1)과 (2)는 트리가 아니고 (3)과 (4)는 트리이다.

8.1 트리의 기본 개념

여기서 잠깐!!

트리는 1847년 전기회로의 법칙으로 유명한 독일의 물리학자 키르히호프(Kirchhoff)에 의해 회로 이론의 개발에 이용되었고, 영국의 수학자 케일리(Cayley)는 1857년 탄화수소 계열의 수많은 이성체(isomer)들을 보다 쉽게 구분하기 위하여 트리 개념을 도입하였으며, 처음으로 트리란 명칭을 사용하였다.

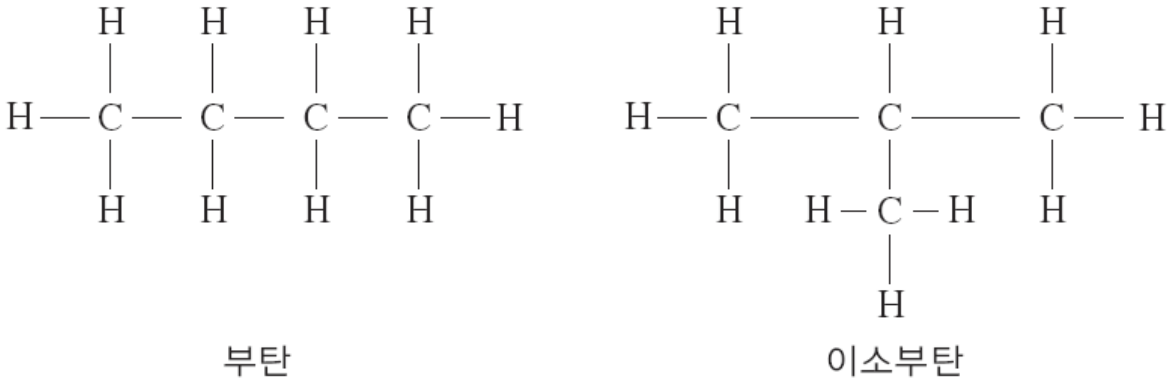
- 화합물인 포화 탄화수소는 $C_k H_{2k+2}$인 형태의 분자식을 가짐
- 탄소의 원자가는 4이고, 수소의 원자가는 1임을 고려하여 화합물 내에 결합되어 있는 원소들을 선으로 연결하여 구조를 표현함
- CH_4를 분자식으로 가지는 메탄의 구조는 트리의 형태로 나타냄



〈그림 8.1〉 메탄의 구조와 트리의 표현

8.1 트리의 기본 개념

- 부탄과 이소부탄은 화학식(C_4H_{10})으로는 같으나, 서로 다른 화학적 구조를 가지는 이성체임
- 트리 개념의 도입은 수많은 이성체들의 분자 구조를 규명하는데 결정적인 역할을 함
- 컴퓨터 기술의 발전과 확산에 힘입어 다양한 분야들에 적용됨



〈그림 8.2〉 이성체의 서로 다른 트리 표현

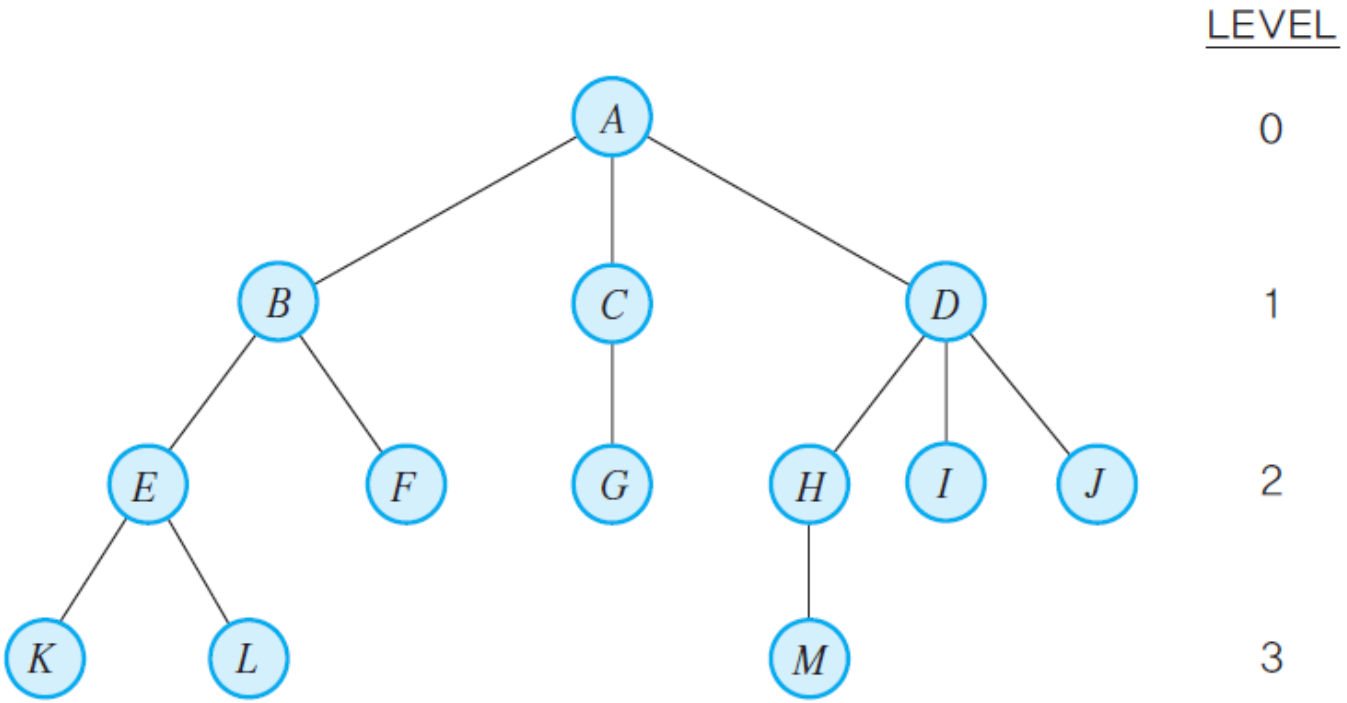
8.1 트리의 기본 개념

여기서 잠깐!!

트리의 응용 분야

- 최적화 문제의 해결
- 알고리즘(데이터 구조, 정렬)
- 분자구조식 설계와 화학결합의 표시, 유전학
- 언어들 간의 번역, 언어학
- 자료의 탐색, 정렬, 데이터베이스 구성
- 사회과학(조직의 분류에 있어서의 구조) 등

정의 8-2 트리에서 사용되는 용어들은 다음과 같다. 보다 이해를 쉽게 하기 위하여 〈그림 8.3〉을 예를 들어 설명한다.



〈그림 8.3〉 트리의 예

8.1 트리의 기본 개념

1) 루트(root) : 주어진 그래프의 시작 노드로서 통상 트리의 가장 높은 곳에 위치하며 노드 A가 루트 노드임

2) 차수(degree) : 어떤 노드의 차수는 그 노드의 서브 트리의 개수를 나타내고, 노드 A의 차수는 3, B의 차수는 2, F의 차수는 0임

3) 잎 노드(leaf node) : 차수가 0인 노드로서 K, L, F, G, M, I, J가 잎 노드에 해당됨

4) 자식 노드(children node) : 어떤 노드의 서브 트리의 루트 노드들을 말하는데 A의 자식 노드는 B, C, D임

5) 부모 노드(parent node) : 자식 노드의 반대 개념으로서, B의 부모 노드는 A이고 G의 부모 노드는 C임

6) 형제 노드(sibling node) : 동일한 부모를 가지는 노드인데, B, C, D는 모두 형제 노드들이고 K, L도 형제 노드들 임

7) 중간 노드(internal node) : 루트도 아니고 잎 노드도 아닌 노드를 말함

8) 조상(ancestor) : 루트로부터 그 노드에 이르는 경로에 나타난 모든 노드들을 말하는데, F의 조상은 B와 A이며, M의 조상은 H, D, A임

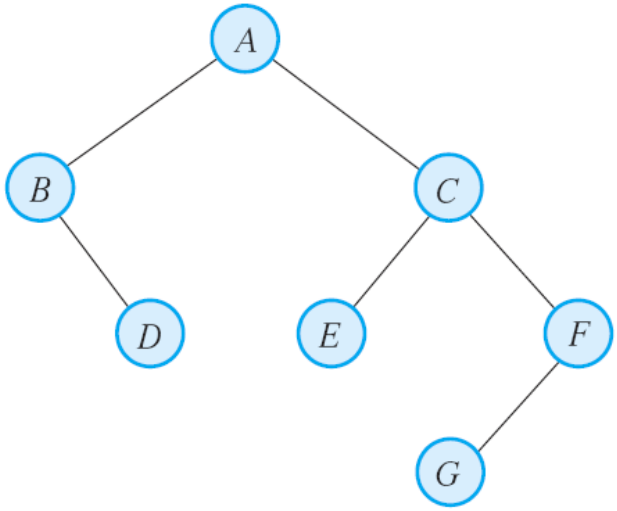
9) 자손(descendant) : 그 노드로 부터 잎 노드에 이르는 경로상에 나타난 모든 노드 들을 말하는데 B의 자손은 E, F, K, L이고 H의 자손은 M임

10) 레벨(level) : 루트의 레벨을 0으로 놓고 자손 노드로 내려가면서 하나씩 증가한다. 즉, 어떤 노드의 레벨이 p라면 그것의 자식 노드의 레벨은 p+1이 됨

11) 트리의 높이(height) : 트리에서 노드가 가질 수 있는 최대 레벨로서 트리의 깊이(depth)라고도 한다. 이 트리의 높이는 3이 됨

12) 숲(forest) : 서로 연결되지 않는 트리들의 집합으로서 트리에서 루트를 제거하면 숲을 얻을 수 있음

예제 8-2 다음 트리에서 주어진 물음에 답해보자.



(1) 잎 노드

(2) 트리의 높이

(3) C의 차수

(4) G의 조상 노드

풀이 (1) D, E, G (2) 3 (3) 2 (4) A, C, F

정리 8-1

n개의 노드를 가진 트리는 $n-1$개의 연결선을 가진다.

정리 8-2

그래프 $G=(V, E)$에서 $|V|=n$이고 $|E|=m$일 때 다음의 문장들은 모두 동치이다.

(1) G는 트리이다.

(2) G는 연결되어 있고 $m=n-1$이다.

(3) G는 연결되어 있고 어느 한 연결선만을 제거하더라도 G는 연결되지 않는다.

(4) G는 사이클을 가지지 않으며 $m=n-1$이다.

(5) G는 어느 한 연결선만 첨가하더라도 사이클을 형성하게 된다.

예제 8-3 15개의 정점을 가지는 트리는 몇 개의 연결선을 가지는지 알아보자.

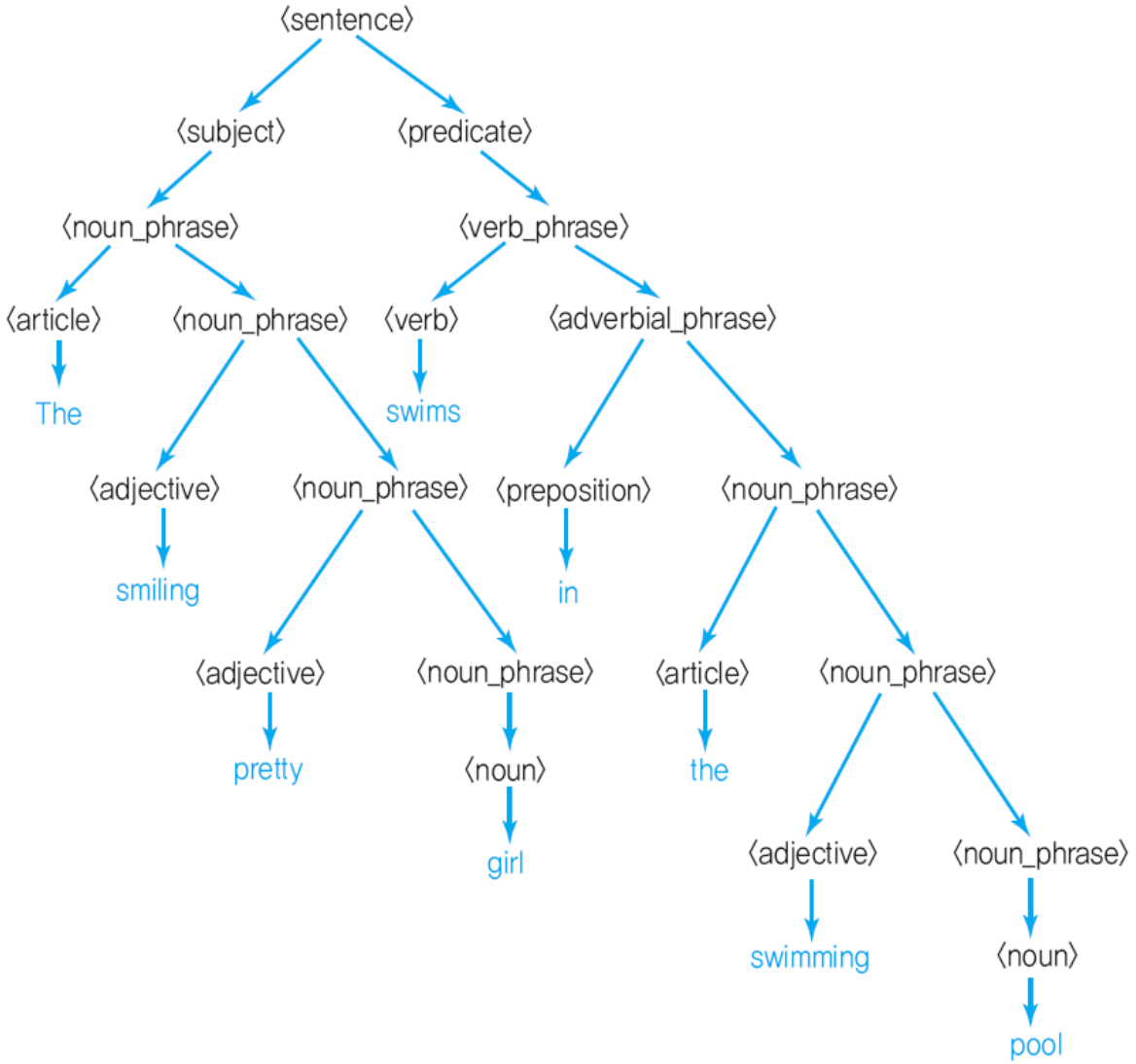
풀이 위의 정리에서 $m = n - 1$이므로 연결선의 개수는 $n = 14$이다.

정의 8-3 트리 T가 n-트리(n-ary tree)란 말은 모든 중간 노드들이 최대 n개의 자식 노드를 가질 때를 말하며, 특히 n이 2인 경우를 이진 트리(binary tree)라고 한다.

방향이 있고 순서화된 트리는 다음의 성질들을 만족하는 방향 그래프임

1) 선행자가 없는 루트라고 불리는 노드가 하나 있으나, 이 루트에서는 모든 노드로 갈 수 있는 경로가 있음
2) 루트를 제외한 모든 노드들은 오직 하나씩만의 선행자를 가짐
3) 각 노드의 후속자들은 통상 왼쪽으로부터 순서화됨

- 어떤 방향 트리를 그릴 때 그 트리의 루트는 가장 위에 있음
- 아크(연결선)들은 밑을 향하여 그려짐



⟨그림 8.4⟩ 순서화된 트리 구조의 예

8.3 이진 트리

정의 8-4 이진 트리(binary tree)는 노드들의 유한 집합으로서,
(1) 공집합이거나
(2) 루트와 왼쪽 서브 트리, 오른쪽 서브 트리로 이루어진다.

사향 이진 트리(skewed binary tree)

왼쪽 또는 오른쪽으로 편향된 트리의 구조를 가짐

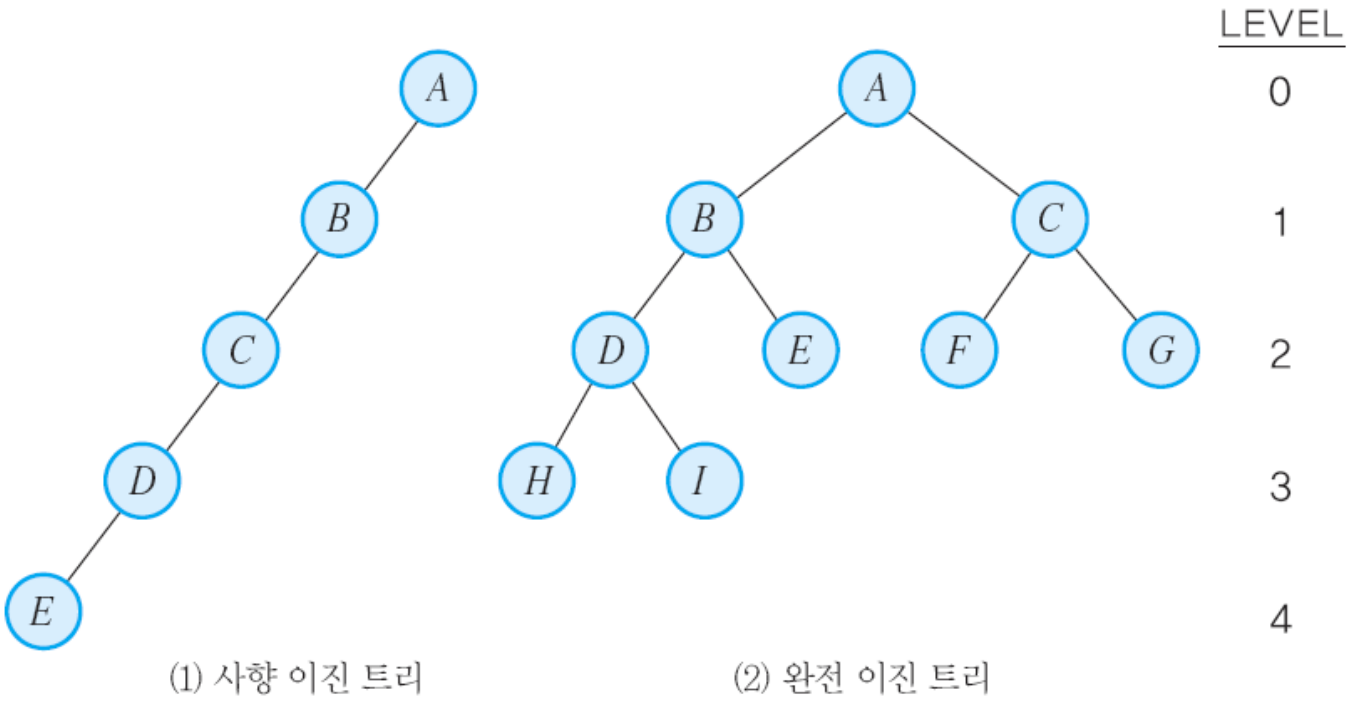
완전 이진 트리(complete binary tree)

높이가 k일 때 레벨 1부터 $k-1$까지는 모두 차 있고 레벨 k에서는 왼쪽 노드부터 차례로 차 있는 이진 트리임

포화 이진 트리(full binary tree)

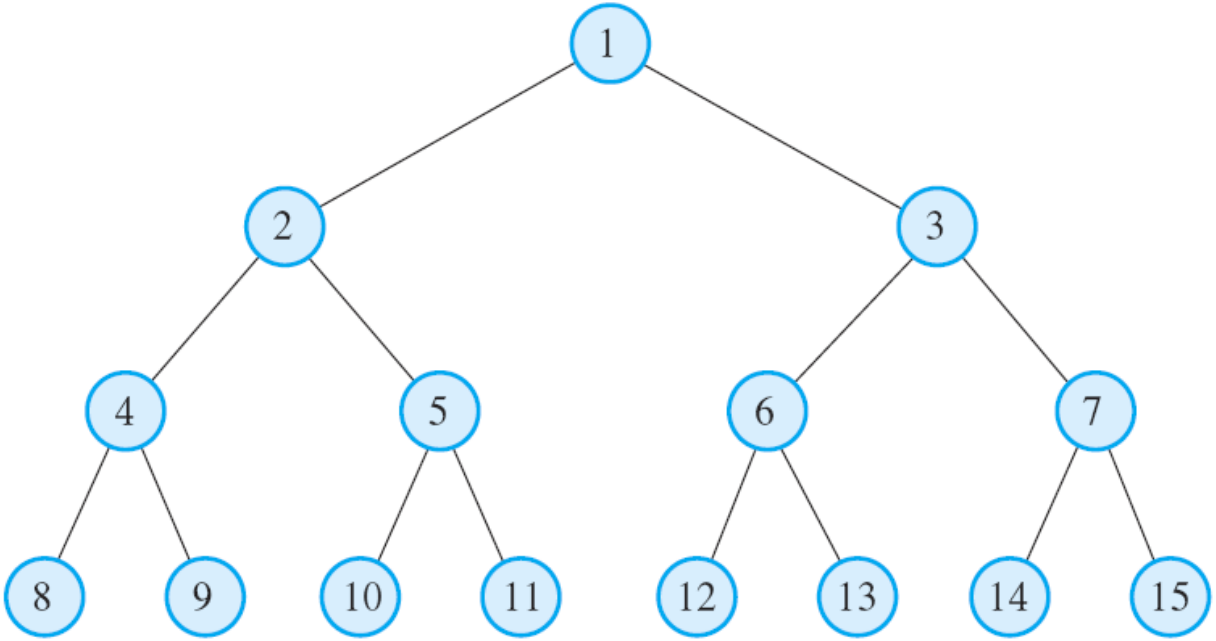
잎 노드가 아닌 것들은 모두 2개씩의 자식 노드를 가지며 트리의 높이가 일정한 경우를 말함. 즉 포화 이진 트리에서 전체가 꽉 차있는 경우임

8.3 이진 트리



(1) 사향 이진 트리 (2) 완전 이진 트리

〈그림 8.5〉 이진 트리의 예



〈그림 8.6〉 포화 이진 트리의 예

정리 8-3

이진 트리가 레벨 i에서 가질 수 있는 최대한의 노드 수는 2^i개이며, 높이가 k인 이진 트리가 가질 수 있는 최대한의 전체 노드 수는 $2^{k+1}-1$이다. 예를 들어, 레벨이 2인 경우에는 최대한 2^2개의 노드를 가질 수 있고, 전체적으로는 최대 $2^{k+1}-1$, 즉 7개를 가질 수 있다.

정리 8-4

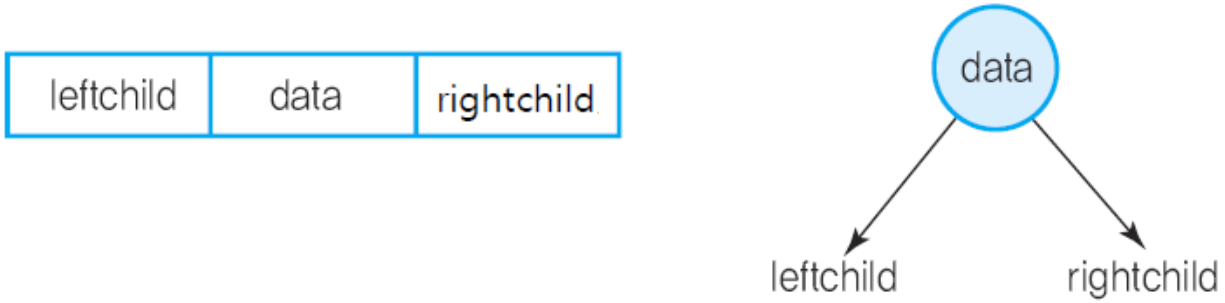
이진 트리에서 잎 노드의 개수를 n_0, 차수가 2인 노드의 개수를 n_2라고 할 때 $n_0 = n_2 + 1$의 식이 항상 성립한다.

이진 트리를 표현하는 방법

- 배열(array)에 의한 방법과 연결 리스트(linked list)에 의한 방법으로 나눌 수 있음
- 배열에 의한 방법은 트리의 중간에 새로운 노드를 삽입하거나 기존의 노드를 지울 경우 비효율적임
- 일반적으로 가장 많이 쓰이고 있는 연결 리스트에 의한 방법임

데이터(data)를 저장하는 중간 부분, 왼쪽 자식(left child), 오른쪽 자식(right child)을 각각 가리키는 포인터(pointer) 부분으로 구성되며, 다음 페이지에서와 같이 C언어로 나타낼 수 있음

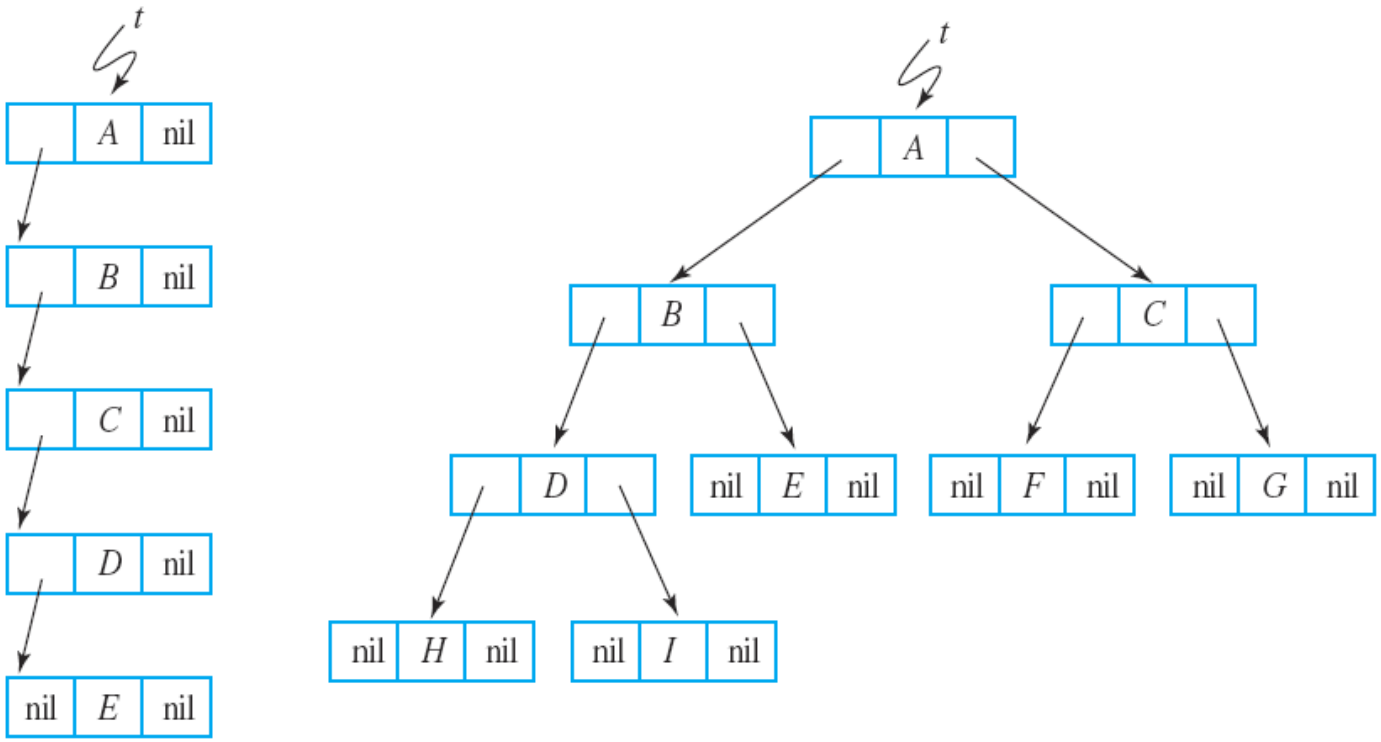
8.4 이진 트리의 표현



〈그림 8.7〉 노드의 구조

```
typedef struct treenode {
    struct treenode *leftchild;
    char data;
    struct treenode *rightchild;
} TREE;
```

이진 트리의 연결 리스트에 의한 표현



〈그림 8.8〉 이진 트리의 연결 리스트에 의한 표현

8.5 이진 트리의 탐방

- 트리는 자료의 저장과 검색에 있어서 매우 중요한 구조를 제공함
- 트리에서의 연산에는 여러 가지가 있으나 가장 빈번하게 하는 것은 각 노드를 꼭 한 번씩만 방문하는 탐방(traversal)일 것임
- 탐방의 결과 각 노드에 들어 있는 데이터를 차례로 나열하게 됨
- 이진 트리를 탐방하는 것은 여러 가지 응용에 널리 쓰임
- 각 노드와 그것의 서브 트리를 같은 방법으로 탐방할 수 있음

여기서 잠깐!!

트리 탐방의 세 가지 방법은 다음과 같다

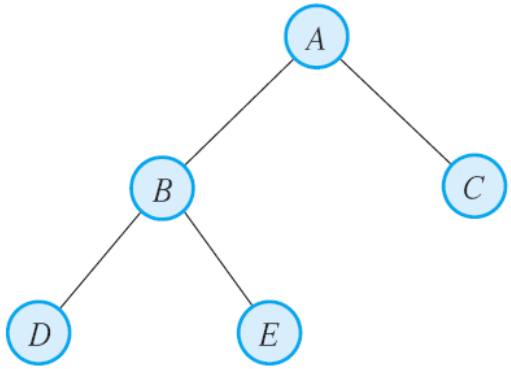
(1) 중순위 탐방

(2) 전순위 탐방

(3) 후순위 탐방

8.5 이진 트리의 탐방

- *L*, *D*, *R*을 각각 왼쪽(Left)으로의 이동, 데이터(Data) 프린트, 오른쪽(Right)으로의 이동을 나타낸다고 할 때 총 6가지의 나열 방법이 있음
- 왼쪽을 오른쪽보다 항상 먼저 방문한다고 가정하면 *LDR*, *DLR*, *LRD*의 3가지 경우가 있음
- 이것을 각각 중순위(inorder), 전순위(preorder), 후순위(postorder)탐방이라고 함
- 이 순서들은 수식 표현에서 중순위 표기(infix), 전순위 표기(prefix), 후순위 표기(postfix)와 각각 대응됨



〈그림 8.9〉 탐방에 쓰이는 이진 트리

8.5 이진 트리의 탐방

정의 8-5 중순위 탐방(inorder traversal)은 루트에서 시작하여 트리의 각 노드에 대하여 다음과 같은 재귀적(recursive) 방법으로 정의된다.

(1) 트리의 왼쪽 서브 트리를 탐방한다.

(2) 노드를 방문하고 데이터를 프린트한다.

(3) 트리의 오른쪽 서브 트리를 탐방한다.

```
void inorder(TREE *currentnode)
{
    if(currentnode != NULL)
    {
        inorder(currentnode->leftchild);
        printf("%c", currentnode->data);
        inorder(currentnode->rightchild);
    }
}
```

[프로그램 8.1] inorder

중순위 탐방의 재귀적 알고리즘(Recursive algorithm)

중순위 탐방은 루트로부터 시작하여

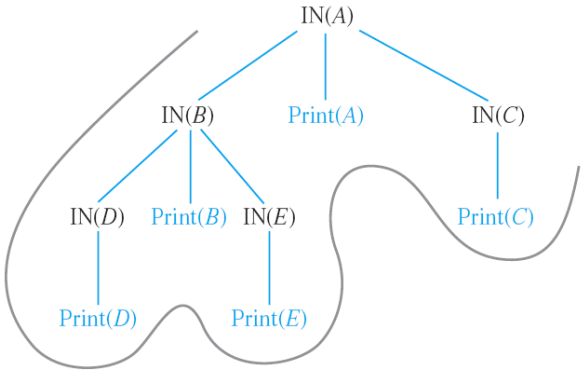
inorder(currentnode→leftchild)

printf(currentnode→data)

inorder(currentnode→rightchild)

의 트리 형태로 계속 전개시켜 나가서 프린트된 결과를 왼쪽부터 차례로 적어나감. [가장 쉽고도 정확한 방법임]

- 이 방법을 탐방에 쓰이는 이진 트리에 적용한 결과 *D*, *B*, *E*, *A*, *C*의 순서로 프린트함



〈그림 8.10〉 중순위 탐방

8.5 이진 트리의 탐방

정의 8-6 전순위 탐방(preorder traversal)은 루트에서 시작하여 트리의 각 노드에 대하여 다음과 같은 재귀적 방법으로 정의된다.

(1) 노드를 탐방하고 데이터를 프린트한다.

(2) 트리의 왼쪽 서브 트리를 탐방한다.

(3) 트리의 오른쪽 서브 트리를 탐방한다.

```
void preorder(TREE *currentnode)
{
    if(currentnode != NULL)
    {
        printf("%c", currentnode->data);
        preorder(currentnode->leftchild);
        preorder(currentnode->rightchild);
    }
}
```

[프로그램 8.2] preorder

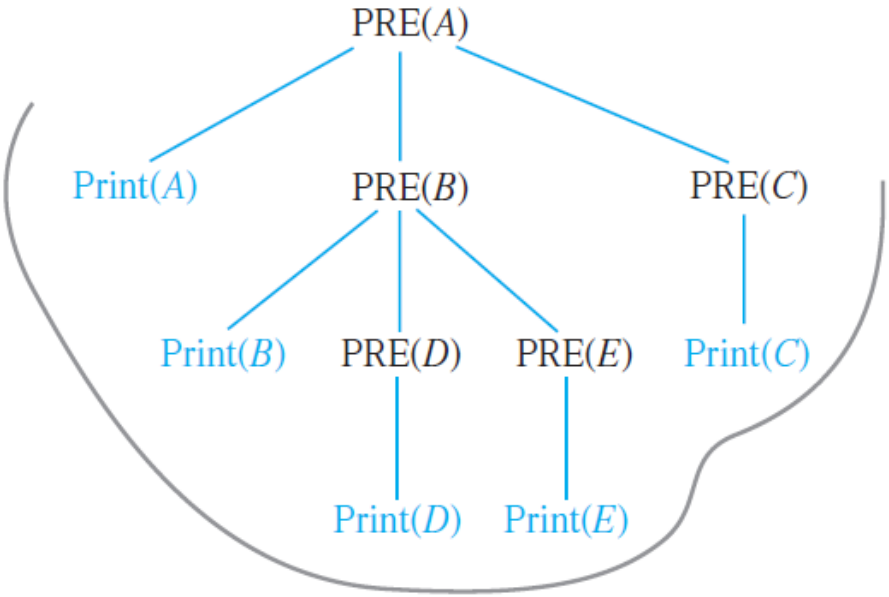
전순위 탐방(D, L, R)

printf(currentnode→data)

preorder(currentnode→leftchild)

preorder(currentnode→rightchild)

탐방에 쓰이는 이진 트리를 적용한 프로그램의 작동 결과는 A, B, D, E, C



〈그림 8.11〉 전순위 탐방

8.5 이진 트리의 탐방

정의 8-7 후순위 탐방(postorder traversal)은 루트에서 시작하여 트리의 각 노드에 대하여 다음과 같은 재귀적 방법으로 정의된다.

(1) 트리의 왼쪽 서브 트리를 탐방한다.

(2) 트리의 오른쪽 서브 트리를 탐방한다.

(3) 노드를 탐방하고 데이터를 프린트한다.

```
void postorder(TREE *currentnode)
{
    if(currentnode != NULL)
    {
        postorder(currentnode->leftchild);
        postorder(currentnode->rightchild);
        printf("%c", currentnode->data);
    }
}
```

[프로그램 8.3] postorder

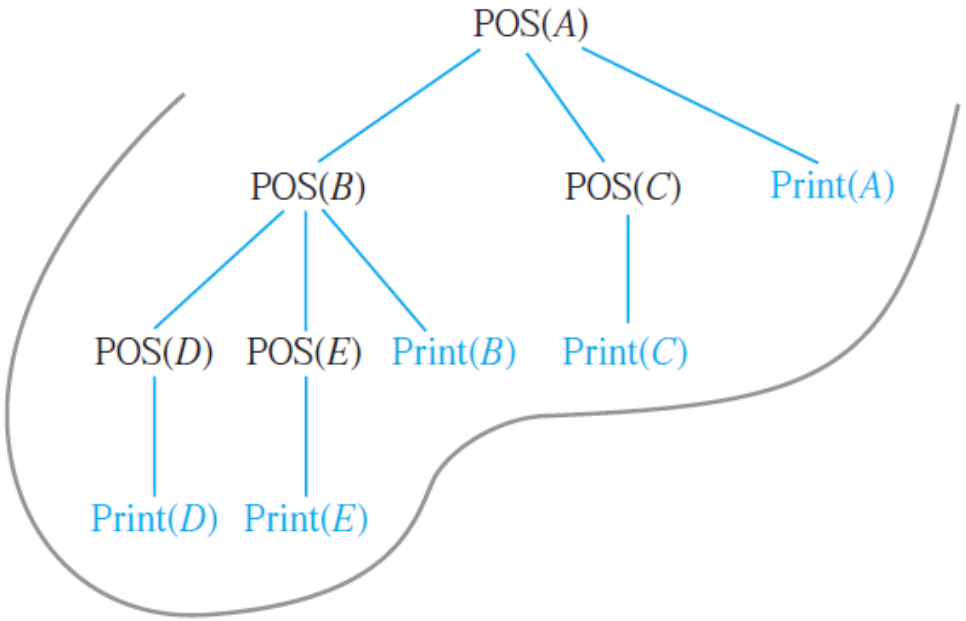
후순위 탐방(L, R, D)

postorder(currentnode→leftchild)

postorder(currentnode→rightchild)

printf(currentnode→data)

탐방에 쓰이는 이진 트리를 적용한 프로그램의 작동 결과는 D, E, B, C, A

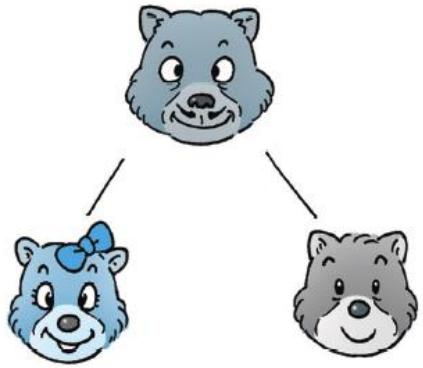


〈그림 8.12〉 후순위 탐방

8.5 이진 트리의 탐방

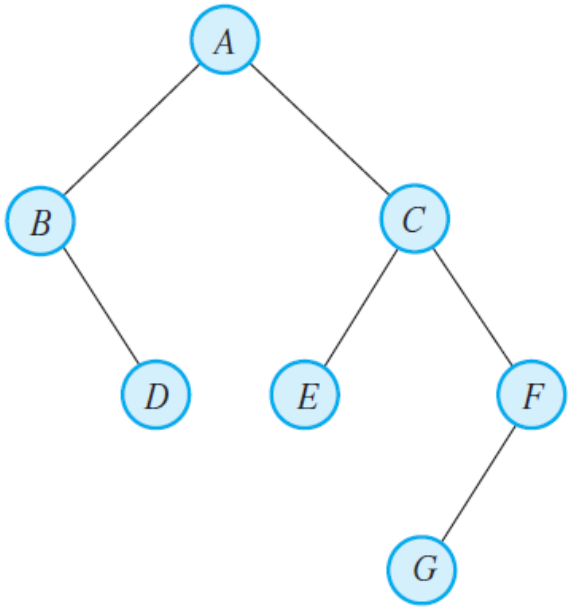
여기서 잠깐!!

중순위, 전순위, 후순위 탐방의 순서는 다음과 같다.

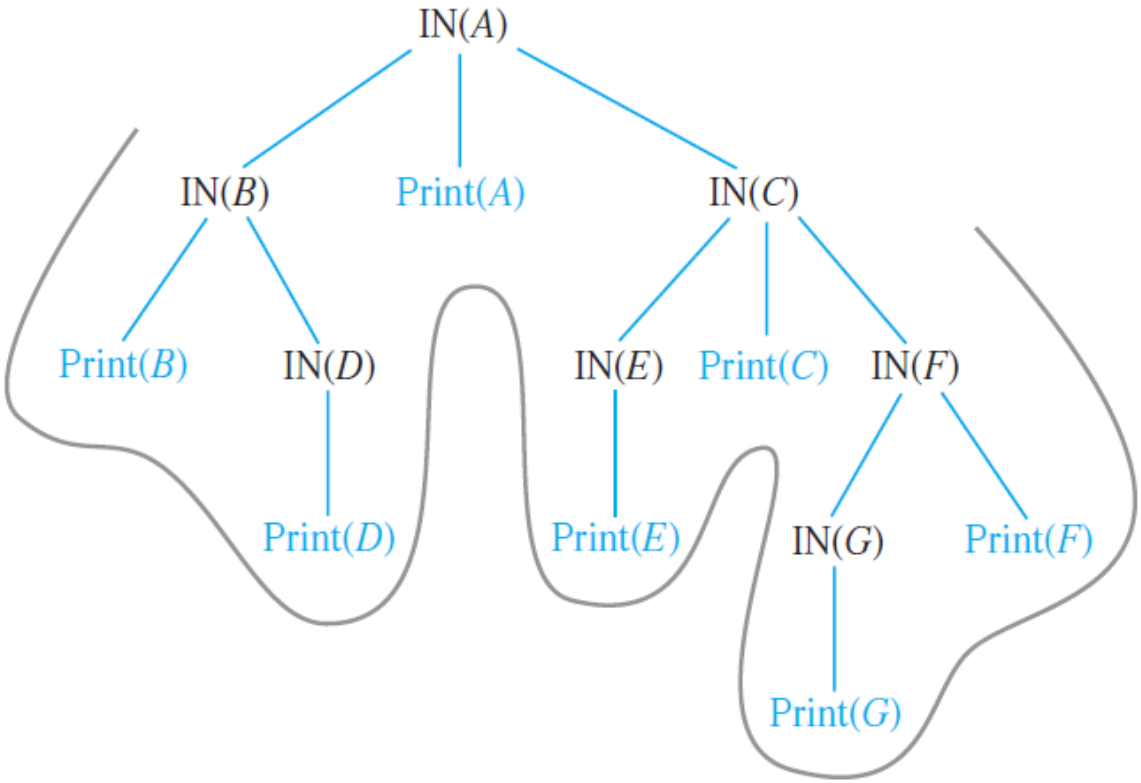


- 중순위 : 
- 전순위 : 
- 후순위 : 

예제 ⑧-4 다음의 이진 트리에서 중순위 탐방, 전순위 탐방, 후순위 탐방의 결과를 각각 구해보자.

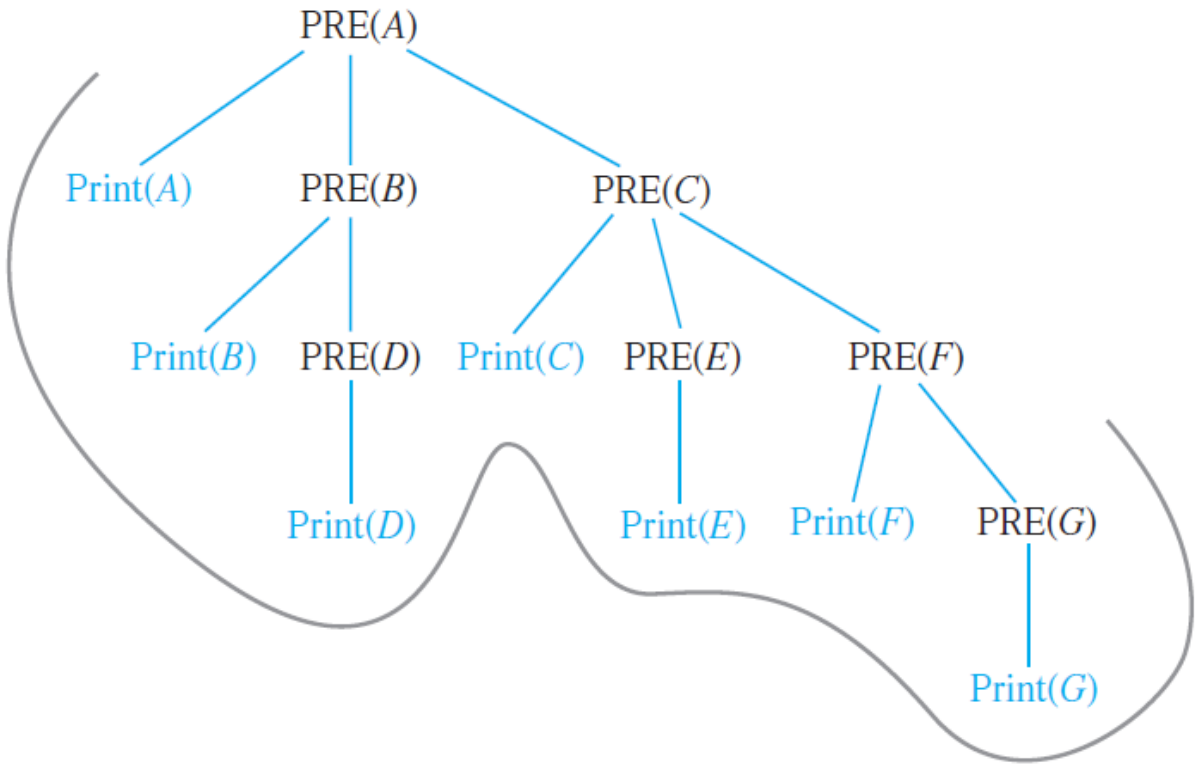


풀이 중순위 탐방의 결과는 위의 알고리즘에 따라 B, D, A, E, C, G, F이다.



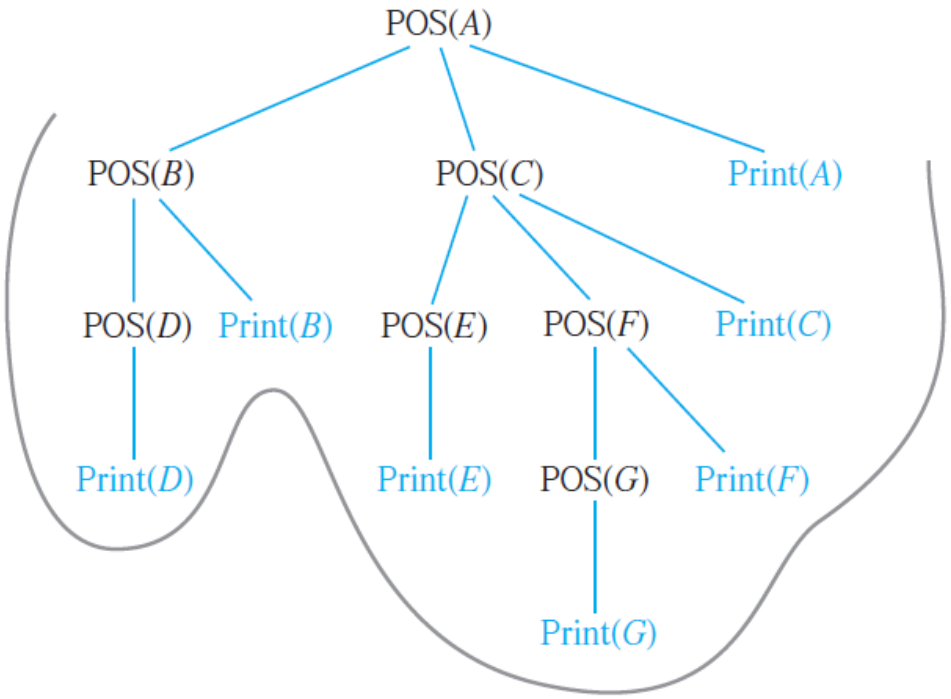
8.5 이진 트리의 탐방

전순위 탐방의 결과는 위의 알고리즘에 따라 A, B, D, C, E, F, G이다.



8.5 이진 트리의 탐방

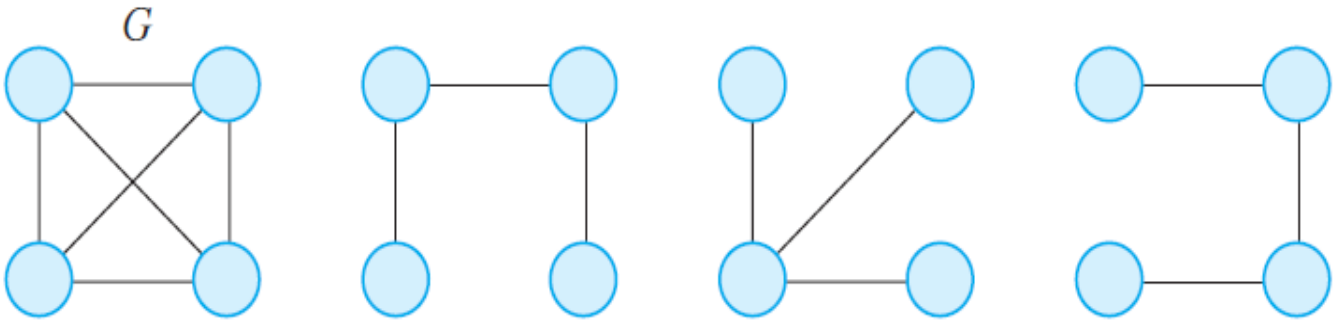
후순위 탐방의 결과는 위의 알고리즘에 따라 D, B, E, G, F, C, A이다.



8.6 생성 트리와 최소 비용 생성 트리

정의 8-8 어떤 그래프 G에서 모든 노드들을 포함하는 트리를 생성 트리(spanning tree)라고 한다.

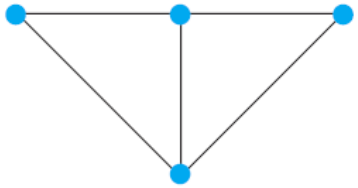
주어진 그래프 G와 그것의 3가지 생성 트리들



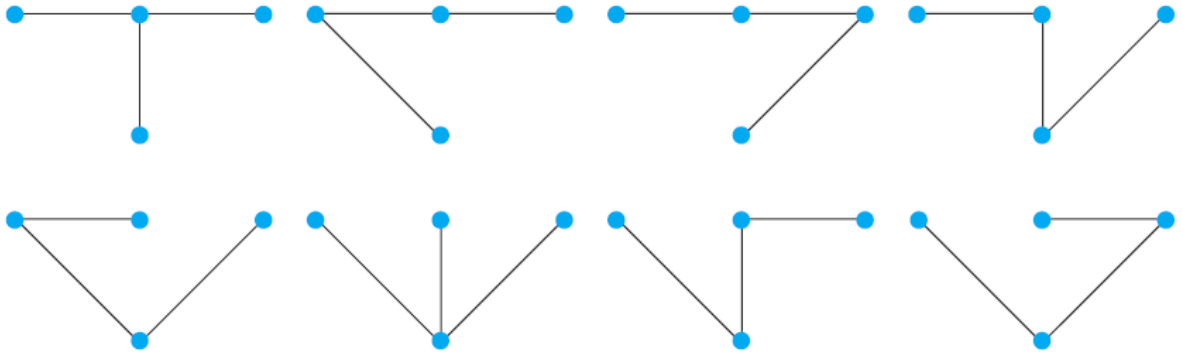
〈그림 8.13〉 주어진 그래프와 그것의 생성 트리들

8.6 생성 트리와 최소 비용 생성 트리

예제 8-5 다음과 같은 그래프 G의 생성 트리를 모두 구해보자.



풀이 G의 생성 트리는 모두 8가지인데 다음과 같다.



8.6 생성 트리와 최소 비용 생성 트리

정의 8-9 생성 트리의 비용(cost)은 트리 연결선의 값을 모두 합한 값이다.

정의 8-10 최소 비용 생성 트리(Minimum Spanning Tree : MST)란 최소한의 비용을 가지는 생성 트리를 말한다.

- 최소 비용 생성 트리의 대표적인 예는 통신 네트워크의 연결임
- 각 도시들을 연결하는 데 있어서 최소 비용으로 연결하는 방법을 찾는 것임
- 최소 비용 생성 트리의 대표적인 2가지 방법은 프림(Prim)의 알고리즘과 크루스칼(Kruskal)의 알고리즘임

8.6 생성 트리와 최소 비용 생성 트리

알고리즘 1 프림의 알고리즘(Prim's algorithm)

주어진 가중 그래프 $G = (V, E)$에서 $V = \{1, 2, \cdots, n\}$이라고 하자.

(1) 노드의 집합 U를 1로 시작한다.

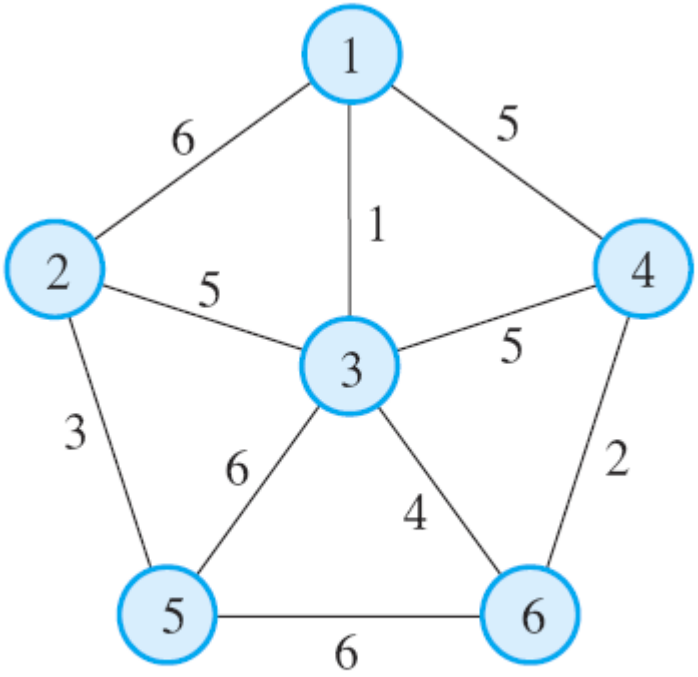
(2) $u \in U$, $v \in V - U$일 때 U와 $V - U$를 연결하는 가장 짧은 연결선인 (u, v)를 찾아서 v를 U에 포함시킨다. 이때 (u, v)는 사이클(cycle)을 형성하지 않는 것이라야 한다.

(3) (2)의 과정을 $U = V$ 때까지 반복한다.

```
void Prim(graph G: set_of_edges T)
    /* 프림의 알고리즘은 G에 대한 최소 비용 생성 트리를 만든다. */
{
    set_of_vertices U;
    vertex u, v;
    T = NULL
    U = {1};
    while( U != V )
    {
        let (u, v) be a lowest cost edge such that u is in U and v is in V - U;
        T = T ∪ {(u, v)};
        U = U ∪ {v};
    }
} /* Prim */
```

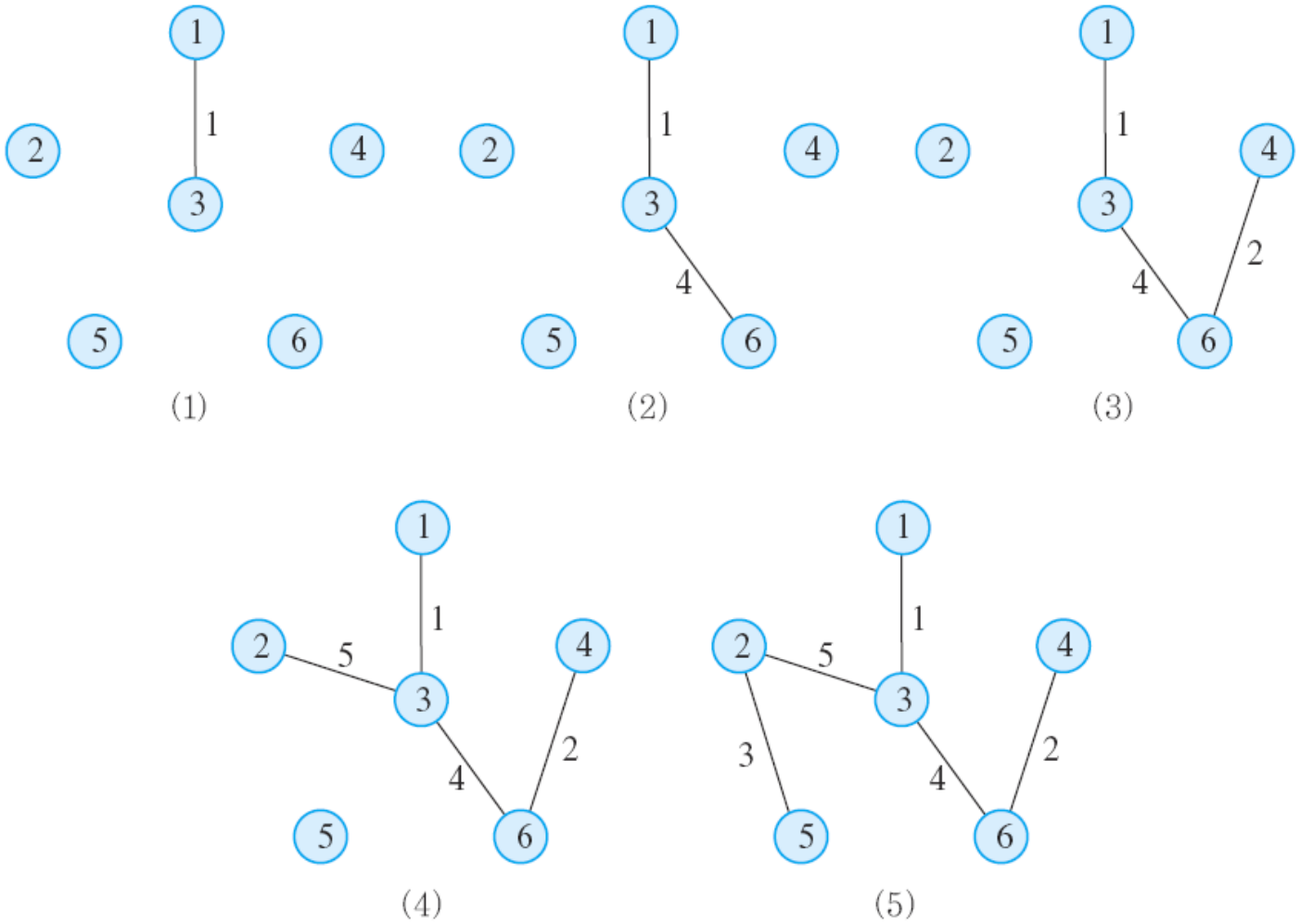
[프로그램 8.4] 프림의 알고리즘

8.6 생성 트리와 최소 비용 생성 트리



〈그림 8.14〉 가중 그래프

8.6 생성 트리와 최소 비용 생성 트리



〈그림 8.15〉 프림의 알고리즘에 의한 MST의 생성 과정

알고리즘 2 크루스칼의 알고리즘(Kruskal's algorithm)

주어진 가중 그래프 $G = (V, E)$에서 $V = \{1, 2, \cdots, n\}$이라고 하고 T를 연결선의 집합이라고 하자.

(1) T를 ϕ으로 놓는다.

(2) 연결선의 집합 E를 비용이 적은 순서로 정렬한다.

(3) 가장 최소값을 가진 연결선 (u, v)를 차례로 찾아서 (u, v)가 사이클을 이루지 않으면 (u, v)를 T에 포함시킨다.

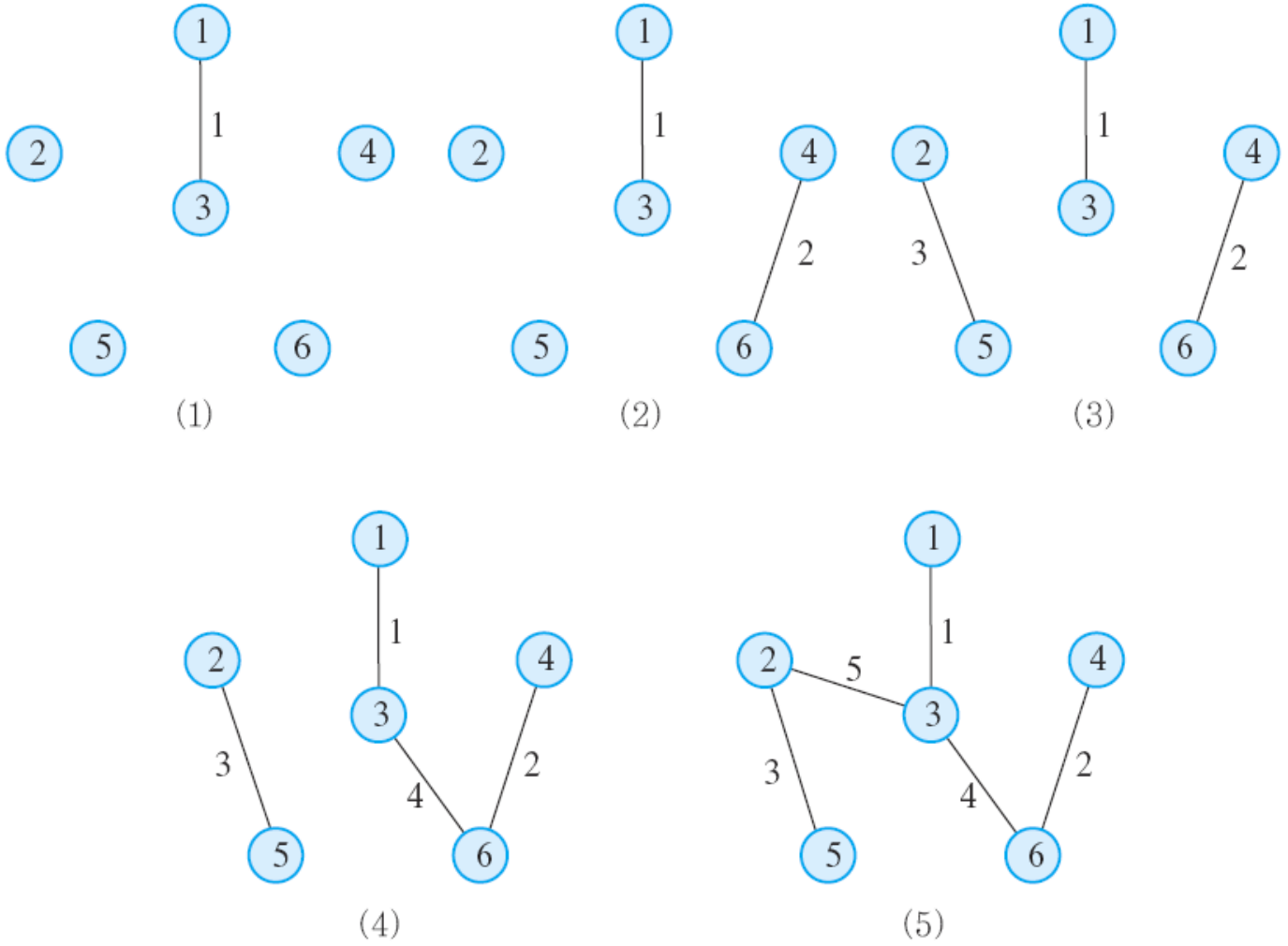
(4) (3)의 과정을 $|T| = |V| - 1$일 때까지 반복한다.

8.6 생성 트리와 최소 비용 생성 트리

```
T = NULL;
while(T contains less than n-1 edges and E not empty)
{
    choose an edge (v, w) from E of lowest cost;
    delete (v, w) from E;
    if ((v, w) does not create a cycle in T)
        add (v, w) to T;
    else discard (v, w);
}
if(T constains fewer than n-1 deges) printf("no spanning tree\n");
```

[프로그램 8.6] 크루스칼의 알고리즘

8.6 생성 트리와 최소 비용 생성 트리

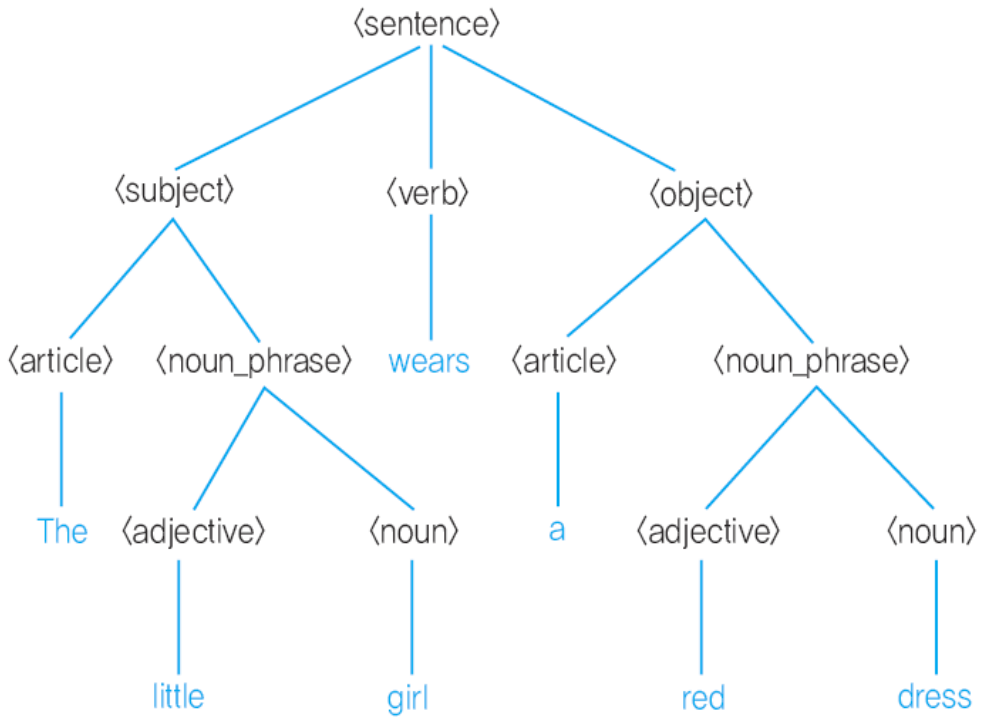


〈그림 8.16〉 크루스칼의 알고리즘에 의한 MST의 생성 과정

(1) 문법의 파싱(parsing)

- 트리는 문법의 파싱을 통하여 자연어 처리와 컴파일러(compiler) 등에 활용됨
- 문장 트리에서 문장은 주어, 동사, 목적어로 이루어짐
- 주어는 관사와 명사구로 나누어지고 목적어는 관사와 명사구로 나누어짐
- 명사구는 형용사와 명사로 다시 나누어지는 파싱 과정을 반복함

8.7 트리의 활용



⟨그림 8.17⟩ 문장 트리

(2) 허프만 코드(Huffman code)

알파벳의 열(sequence)로서 이루어진 메시지가 있고, 각 메시지의 영문자가 각각 독립적이고 위치에 관계없이 어떤 정해진 확률로 나타남

예를 들자면, 5개의 영문자 a, b, c, d, e가 나타날 확률이 각각 0.12, 0.4, 0.15, 0.08, 0.25라고 할 때, 이 영문자를 각각 0과 1의 열로서 코드화하고자 하면, 어느 영문자의 코드도 다른 어떤 영문자의 코드의 접두어로 표현되어서는 안됨

a가 01이고 b가 010일 때 a는 b의 접두어가 되므로 적합하지 않음

8.7 트리의 활용

- 허프만 코드 특성을 이용하여 최소 개의 코드로써 정확하게 송신할 수 있으며 수신된 코드를 정확하게 코드화가 가능함
- 접두어 성질을 만족하는 최적의 코드화 방법을 허프만 알고리즘이라고 함
- 그 결과 나타날 확률이 높은 문자는 코드 길이가 짧고, z와 같이 나타나는 확률이 작은 문자는 코드 길이가 커지는 원리임

영문자	확률	코드 1	코드 2
a	.12	000	000
b	.40	001	11
c	.15	010	01
d	.08	011	001
e	.25	100	10

〈그림 8.18〉 문자가 나타날 확률과 이진 코드의 예

알고리즘 3 허프만 알고리즘(Huffman algorithm)

(1) 가장 낮은 확률을 가진 두 문자 a, b를 선택하여 가상의 다른 문자 x로 대치한다. x가 나타날 확률은 a와 b의 확률을 합한 것이다.

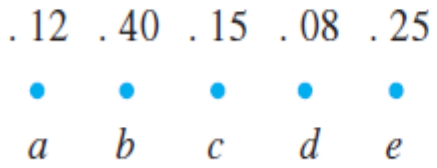
(2) 위의 과정을 반복하여 최종 확률이 1이 될 때까지 계속한다.

(3) 최종적인 트리가 완성되면 각 서브 트리의 왼쪽에는 0을 부여하고 오른쪽에서 1을 각각 부여하여 각 문자의 코드를 결정한다.

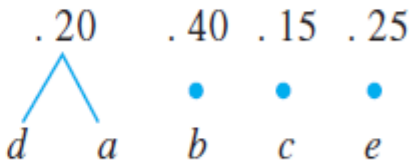
예제 ⑧-6 〈그림 8.18〉의 a, b, c, d, e에 대해 허프만 알고리즘을 적용하여 허프만 코드를 생성하는 과정을 살펴보자.

풀이 처음 시작 단계에서는 다음과 같은 그림의 (2)와 같이 확률이 가장 낮은 d와 a를 골라 통합하고 그 확률의 합이 0.20이므로 다시 c의 0.15와 통합하여 0.35를 만든다. 이어서 0.35, 0.40, 0.25 중 가장 작은 0.35와 0.25를 통합하여 0.60을 만들고 최종적으로 b와 통합하여 1.00을 만들면 완료된다.

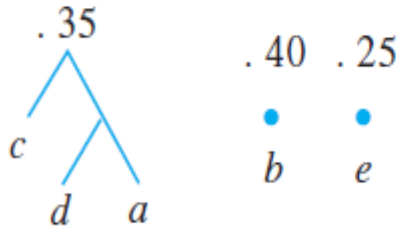
여기서 각 서브 트리의 왼쪽 부분에는 0, 오른쪽 부분에는 1을 부여하면 그림에서 (5)와 같은 결과를 얻는다. 이 경우 a는 1111, b는 0, c는 110, d는 1110, e는 10을 각각 코드로 가진다. 코드의 평균 길이는 4×0.12, 1×0.40, 3×0.15, 4×0.08, 2×0.25를 모두 합하면 2.15로서 최적의 접두어 성질을 가진 허프만 코드를 얻는다.



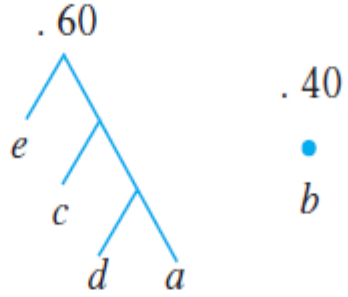
(1) 시작 단계



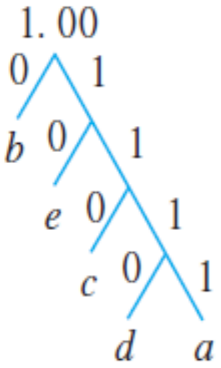
(2) a와 d를 통합



(3) a, d와 c를 통합



(4) a, d, c와 e를 통합



(5) 최종 트리

(3) 결정 트리(decision tree)

- 트리를 이용할 때 매우 유용하게 쓰이는 것은 결정 트리임
- 가능성 있는 경우의 수가 너무나 많기 때문에 모든 면에서 입증하기가 매우 어려운 문제를 만날 수가 있음
- 결정 트리를 활용하면 주어진 문제를 일목요연하게 입증할 수 있음

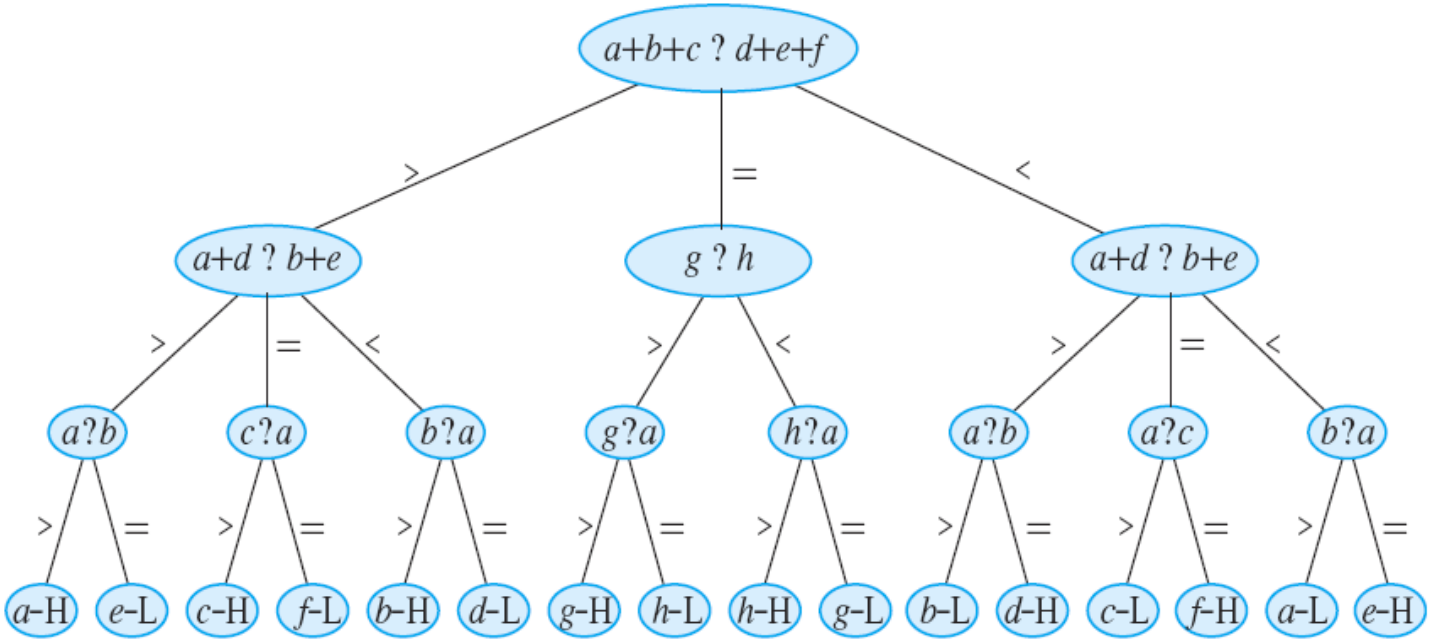
8개의 동전 문제(eight-coins problem)

- 결정 트리의 예로 잘 알려져 있음
- 크기와 색깔이 똑같은 8개의 동전 a, b, c, d, e, f, g, h 중에서 한 개는 불량품이어서 다른 동전들과는 다른 무게를 가짐
- 무게가 같거나 크고 작은 것만을 판단할 수 있는 하나의 천칭 저울을 사용하여 단 세 번만의 계량으로 어느 동전이 불량품이고, 다른 동전보다 무겁거나 가벼운지를 동시에 판단하고자 함

8.7 트리의 활용

- a+b+c < d+e+f인 경우 불량품이 6개 가운데 있으며 이때의 g와 h는 정상품이라는 것을 알 수 있음
- 다음 단계에서 d와 b를 양 천칭 저울에서 바꾸어 계량한 결과 a+d < b+e로 부등호에 변화가 없다면 다음의 2가지 가능성을 말해줌
- a가 불량품이거나 e가 불량품인 경우임
- 이런 경우는 정상품과 비교하여 H나 L을 판단할 수 있음

- 만약 a+d = b+e인 경우에는 c나 f가 불량품임
- 또한 b나 d가 불량품인 경우에는 다른 정상품과 비교하여 b 또는 d가 H나 L이 되게 결정됨



〈그림 8.19〉 8개의 동전 문제 결정 트리

8.7 트리의 활용

- 최종 판단을 위한 프로시저인 comp의 프로그램임
- 프로시저 eight-coins는 앞의 결정 트리의 판단과 같은 기능을 수행함

```
void comp(int x, int y, int z)
   /* x를 정상품인 z와 비교한다 */
{
   if (x > z) printf("%d heavy \n", x);
   else printf("%d light \n", y);
}  /* comp */
```

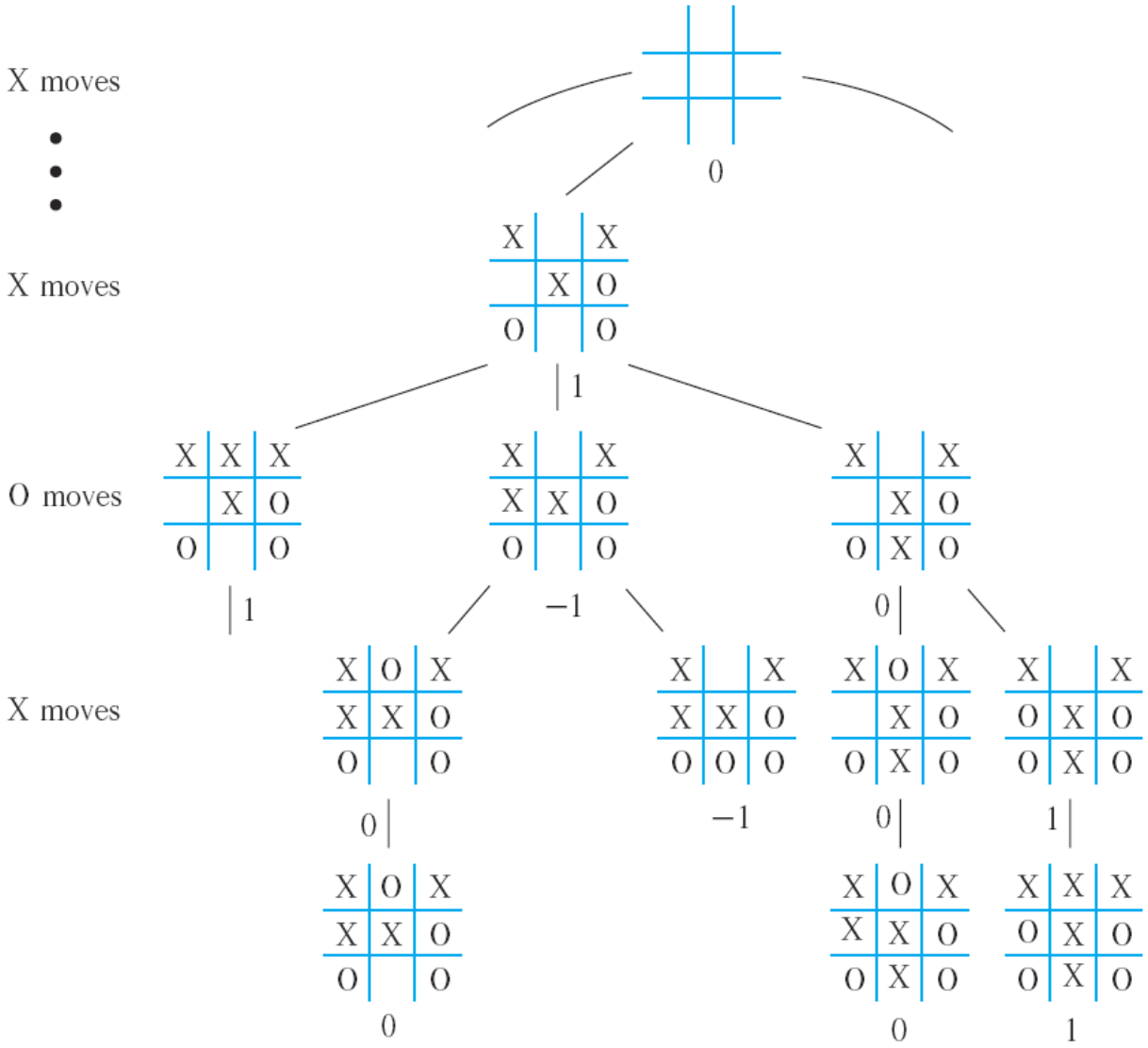
[프로그램 8.7] comp

```
void eight-coins()
   /* 8개의 동전 무게가 입력되면 단 세 번의 비교로써 비정상적인 동전을 발견할 수 있다. */
{
   int a, b, c, d, e, f, g, h;
   scanf("%d %d %d %d %d %d %d %d", &a, &b, &c, &d, &e, &f, &g, &h);
   switch(compare(a+b+c, d+e+f)) {
      case '=' : if(g > h) comp(g, h, a);
                 else comp(h, g, a);
                 break;
      case '>' : switch(compare(a+d, b+e)) {
                 case '=' : comp(c, f, a);
                 case '>' : comp(a, e, b);
                 case '<' : comp(b, d, a);
                }
                break;
      case '<' : switch(compare(a+d, b+e)) {
                 case '=' : comp(f, c, a);
                 case '>' : comp(d, b, a);
                 case '<' : comp(e, a, b);
                }
   }  /* switch */
}  /* eightcoins */
```

[프로그램 8.8] eight-coins

(4) 게임(game)

- 트리는 체스, 틱텍토(tic-tac-toe), 장기, 바둑 등 게임에 있어서의 진행과 전략을 구사할 수 있는 게임 트리로도 활용됨
- 가로, 세로, 대각선으로 연속된 세 개를 놓으면 이기는 게임인 tic-tac-toe 게임의 일부를 나타냄



〈그림 8.20〉 tic-tac-toe 게임 트리

- 게임 트리를 이용하는 또 다른 응용으로는 8-puzzle 게임이 있음
- 이 게임은 3×3 크기의 박스 안에서 초기 상태에서 인접한 숫자를 빈 곳으로 계속적으로 움직여서 목표 상태(goal)로 만드는 서양식 게임임

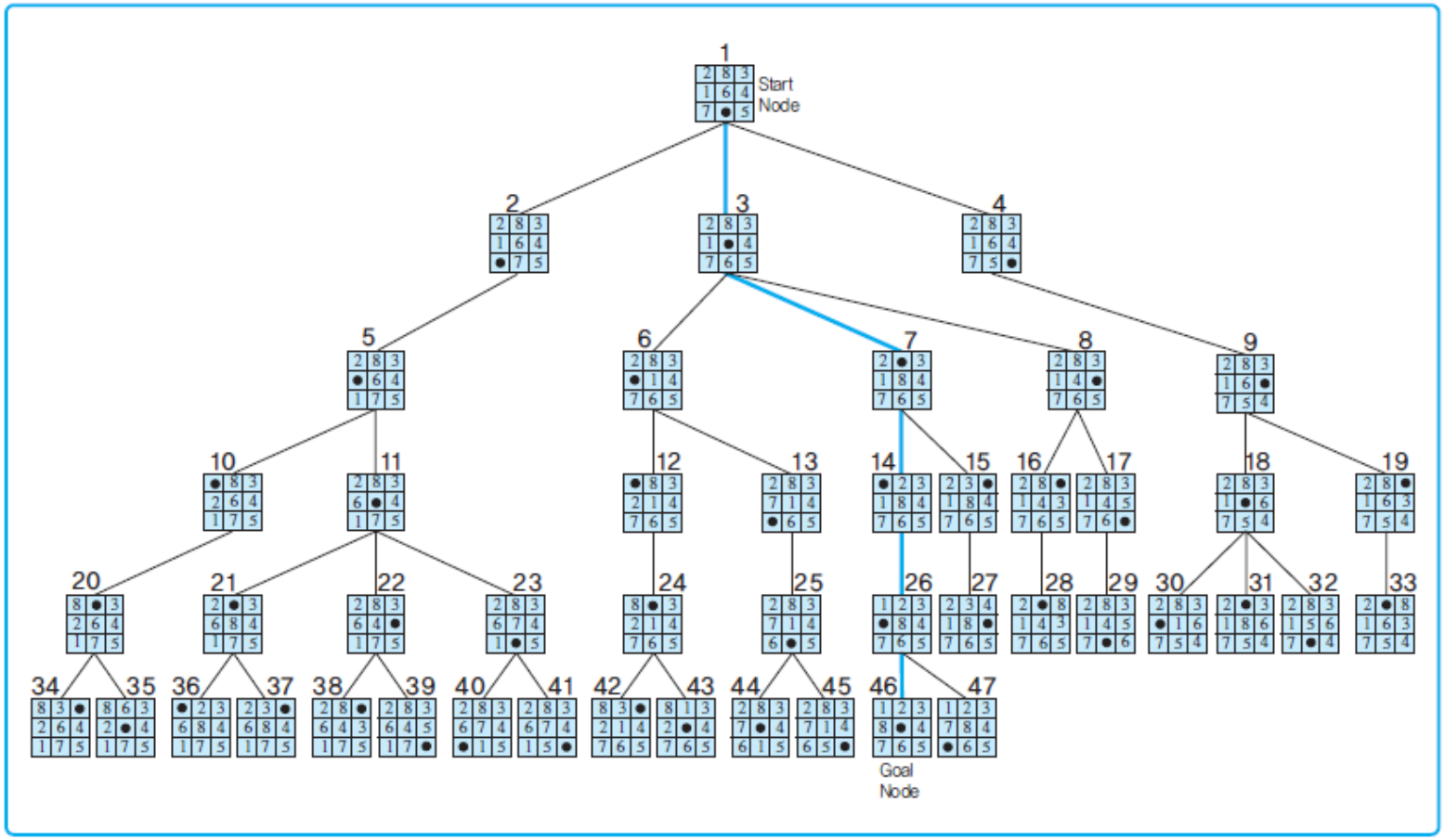
Start

2	8	3
1	6	4
7	•	5

Goal

1	2	3
8	•	4
7	6	5

〈그림 8.21〉 시작 상태와 목표 상태



〈그림 8.22〉 8-puzzle 트리

우선순위 큐 priority queue: Heap

- 우선순위 큐?
  - Stack: first in – last out (FILO / LIFO)
  - Queue: first in – first out (FIFO)
  - 우선순위 큐
    - 우선순위가 높은 것이 먼저 서비스 된다
    - 계급 사회

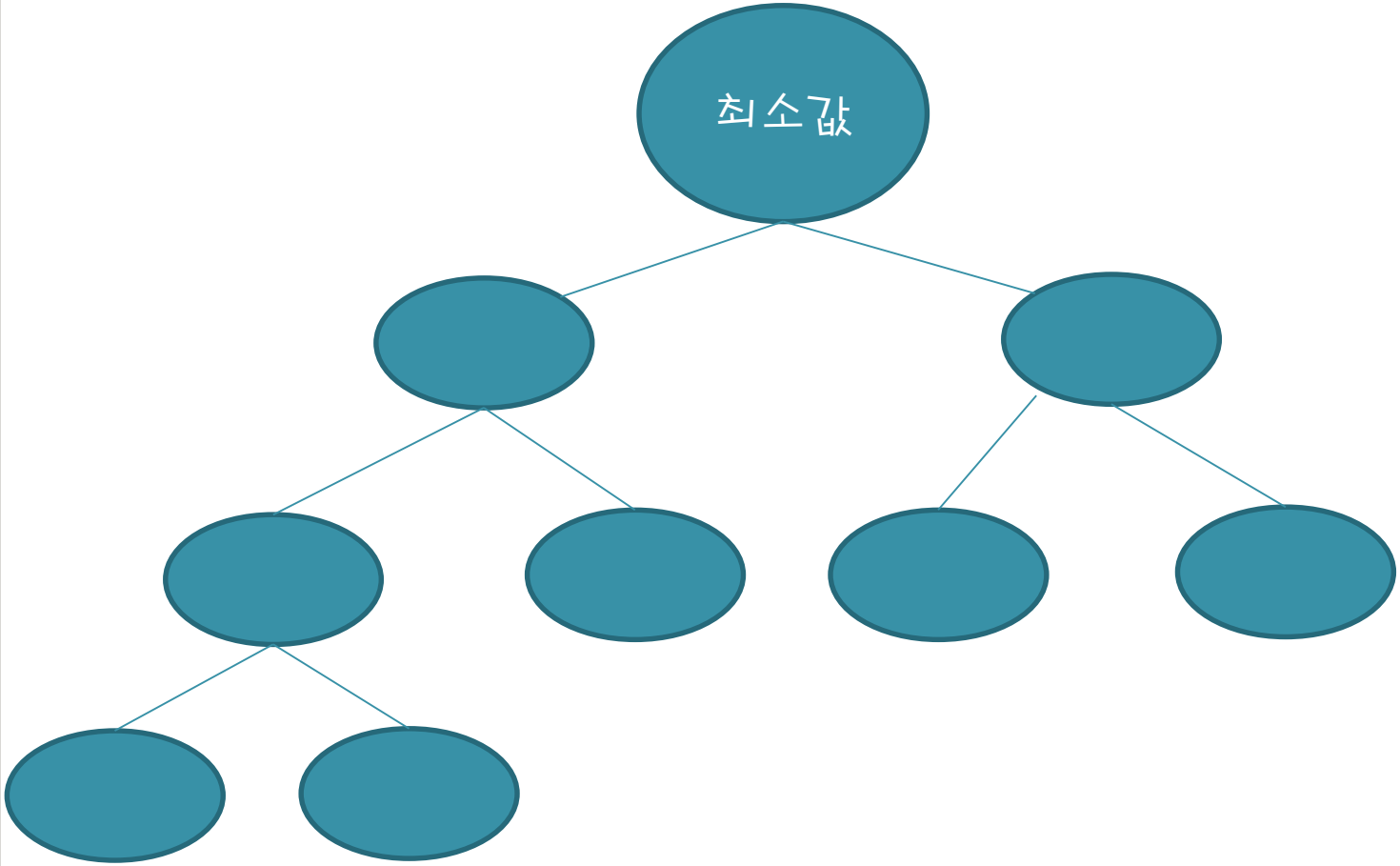
Min heap / max heap

- Min heap
  - 가장 작은 값을 가진 것이 먼저 나감
- Max heap
  - 가장 큰 값을 가진 것이 먼저 나감

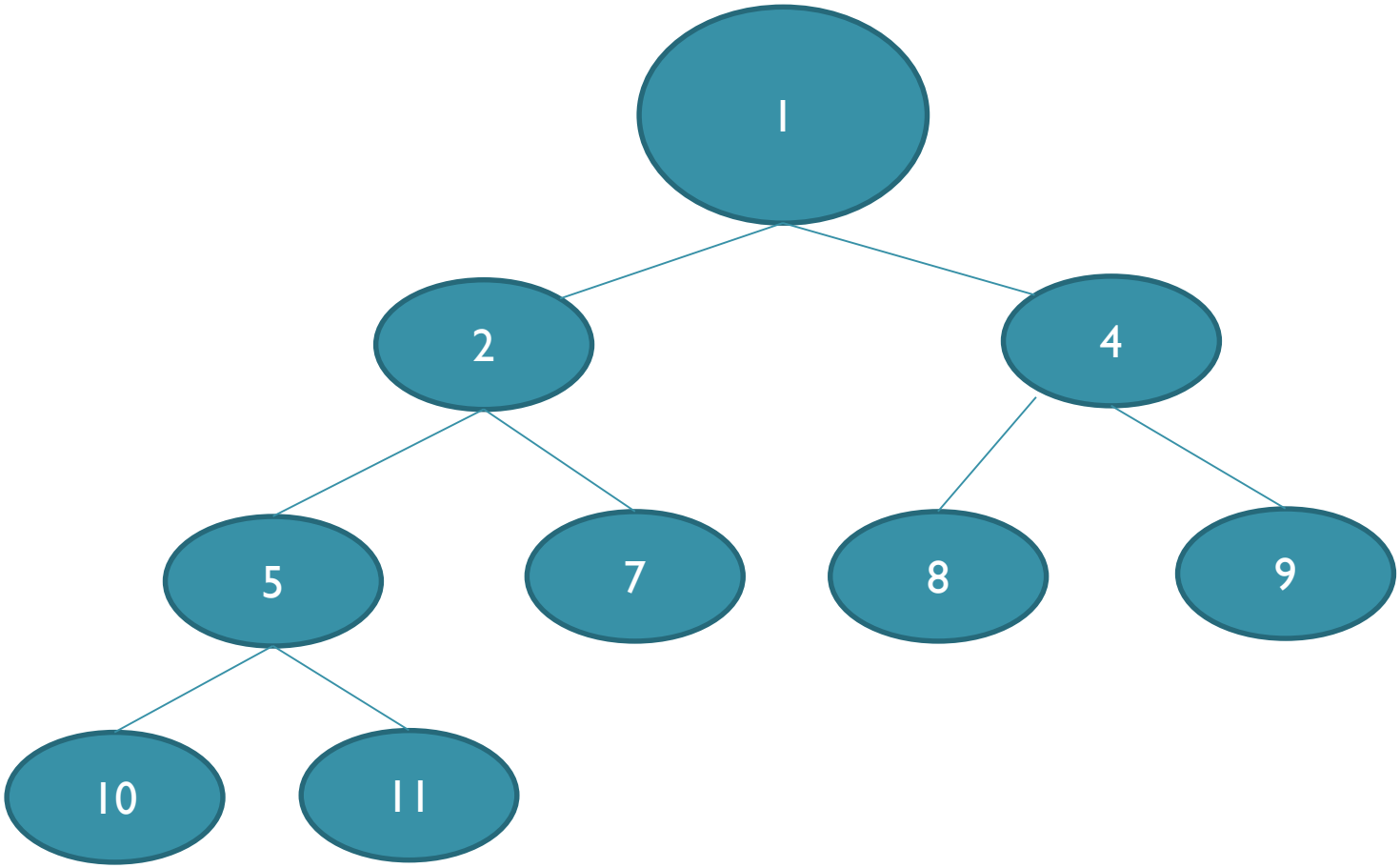
- 트리로 구현할 수 있음
  - 여기서는 min heap만 고려해 보자
    - Max heap도 동일하게 구현 가능

Min heap

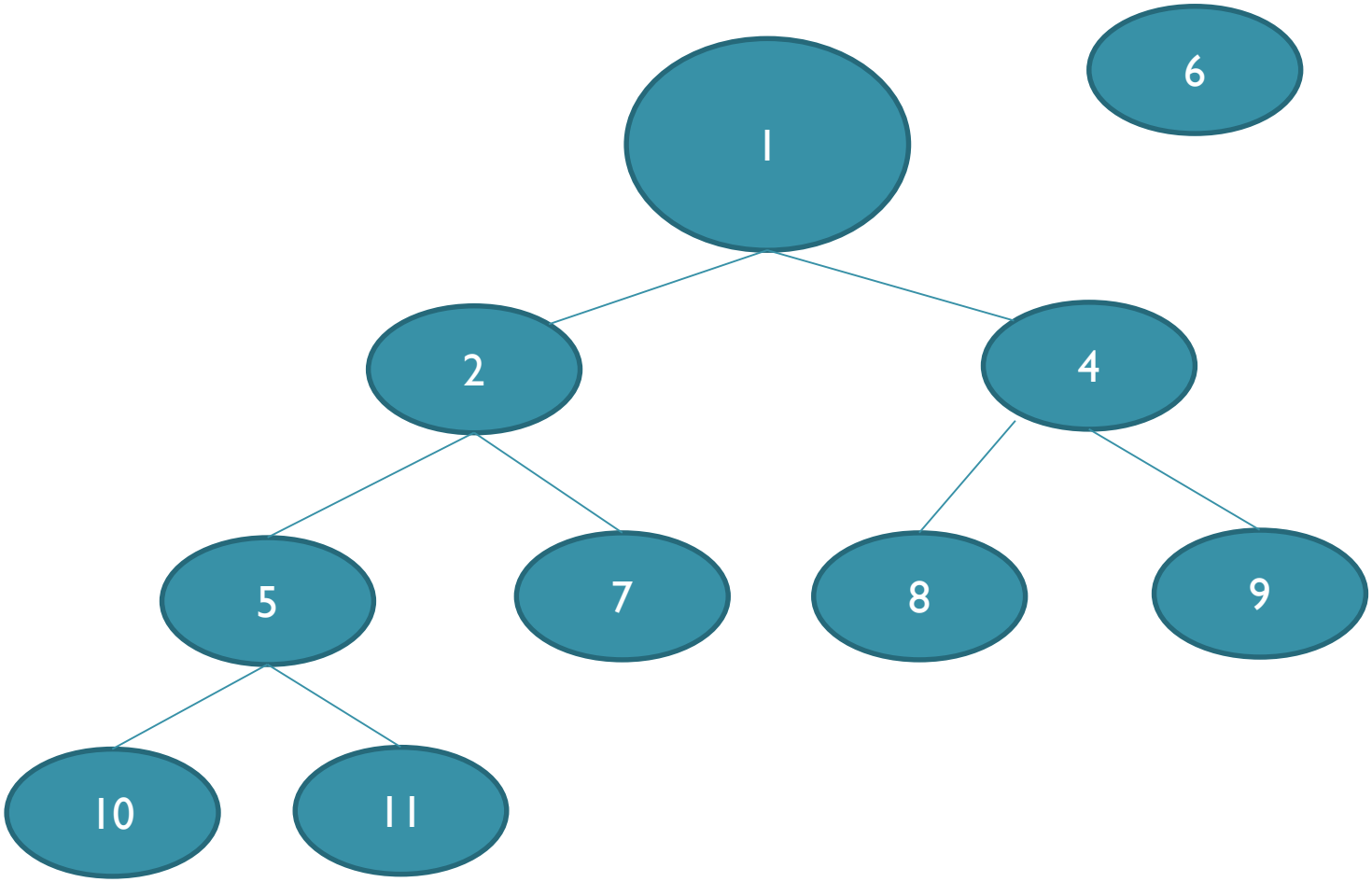
- 가장 작은 값을 가진 노드가 루트



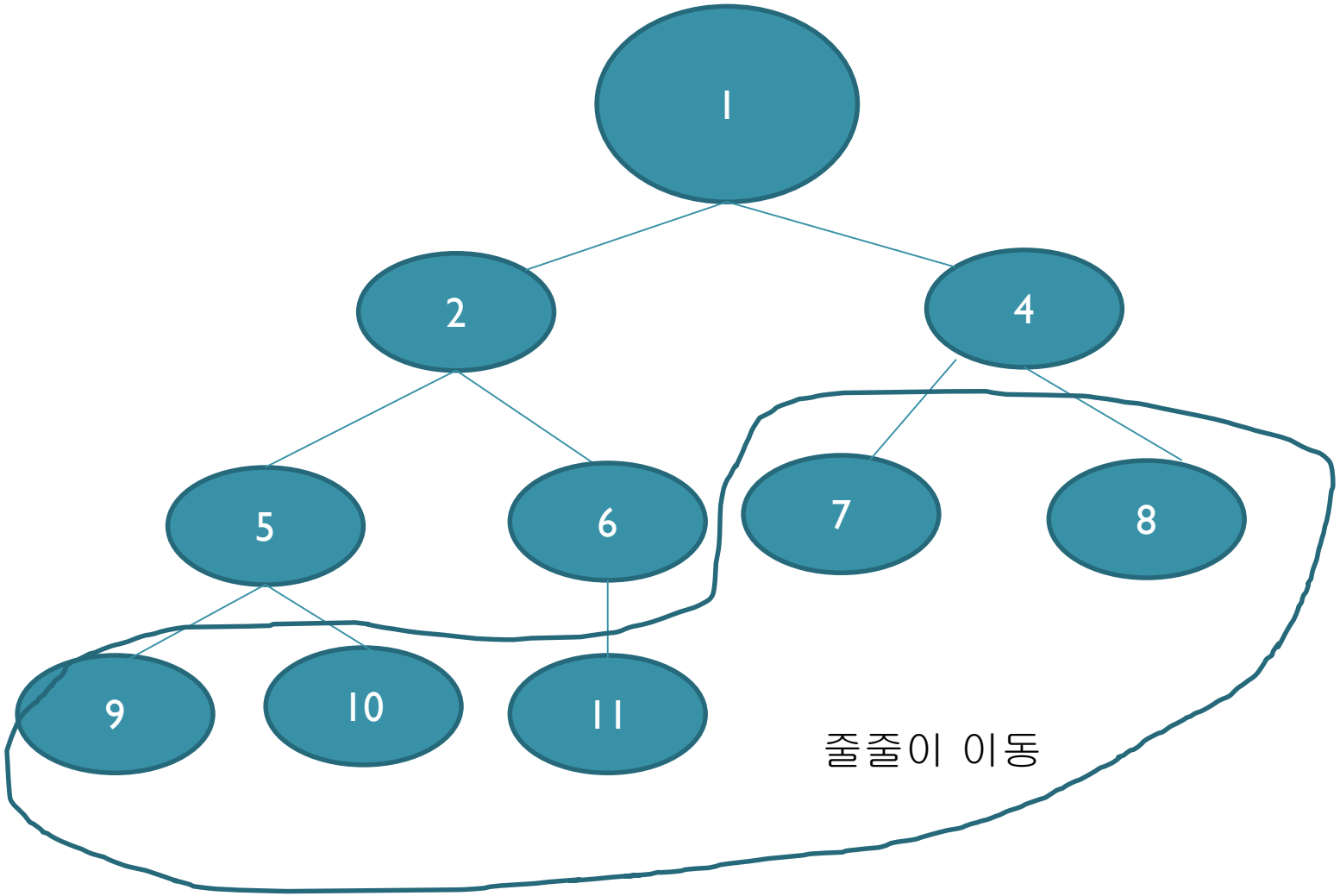
정렬을 해 놓으면?



새로운 데이터가 들어왔다

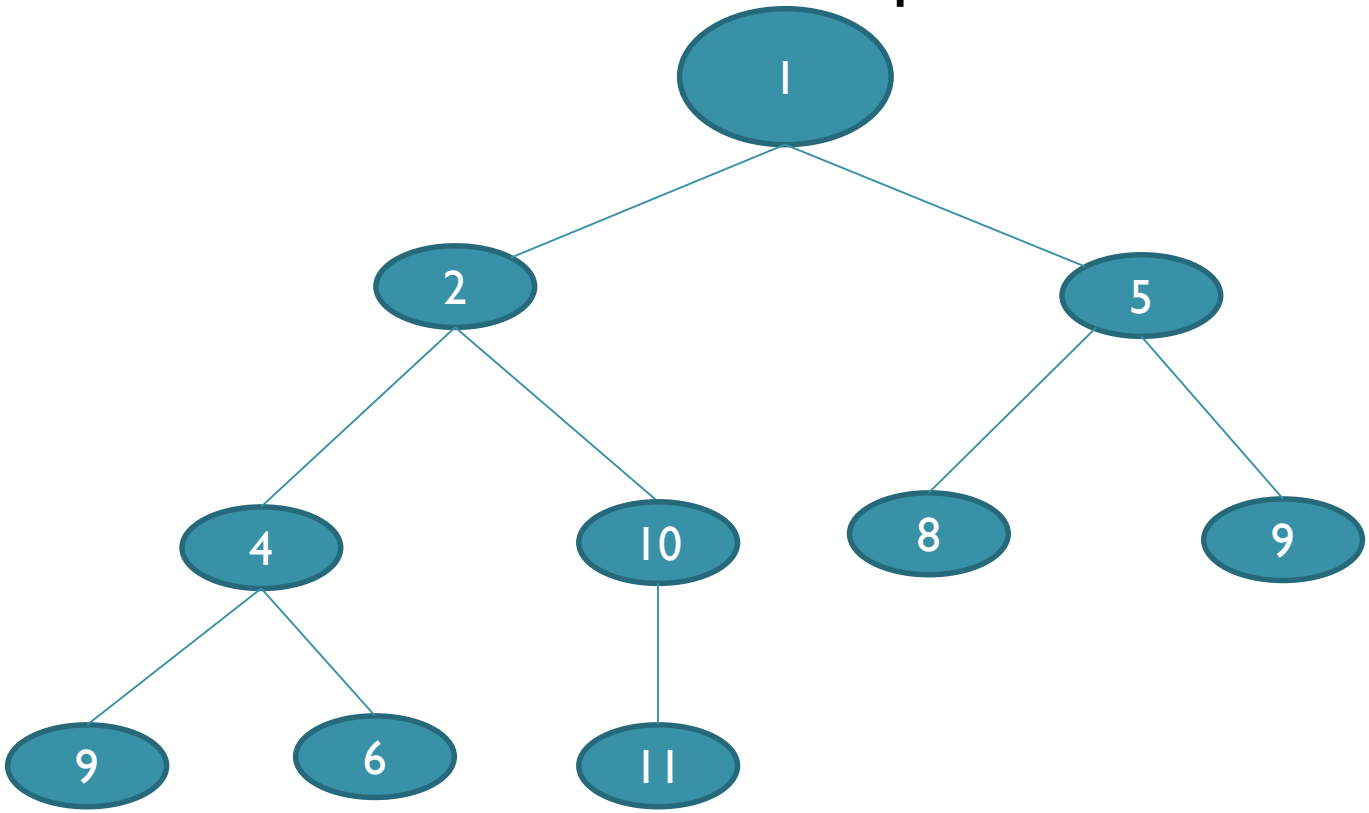


새로운 데이터가 들어왔다



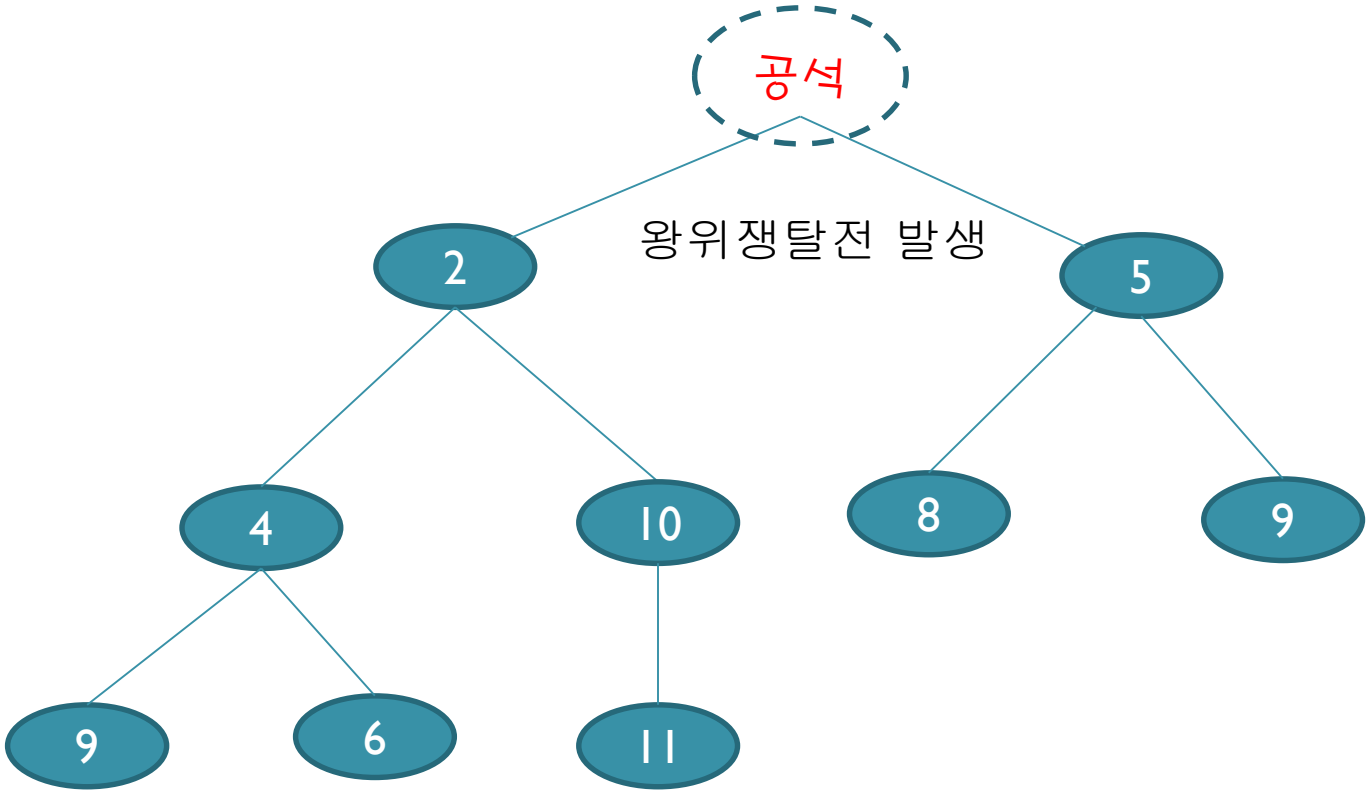
느슨한 조건

- 상사는 부하보다 뛰어나야 한다
  - 이 조건만 만족하면 heap



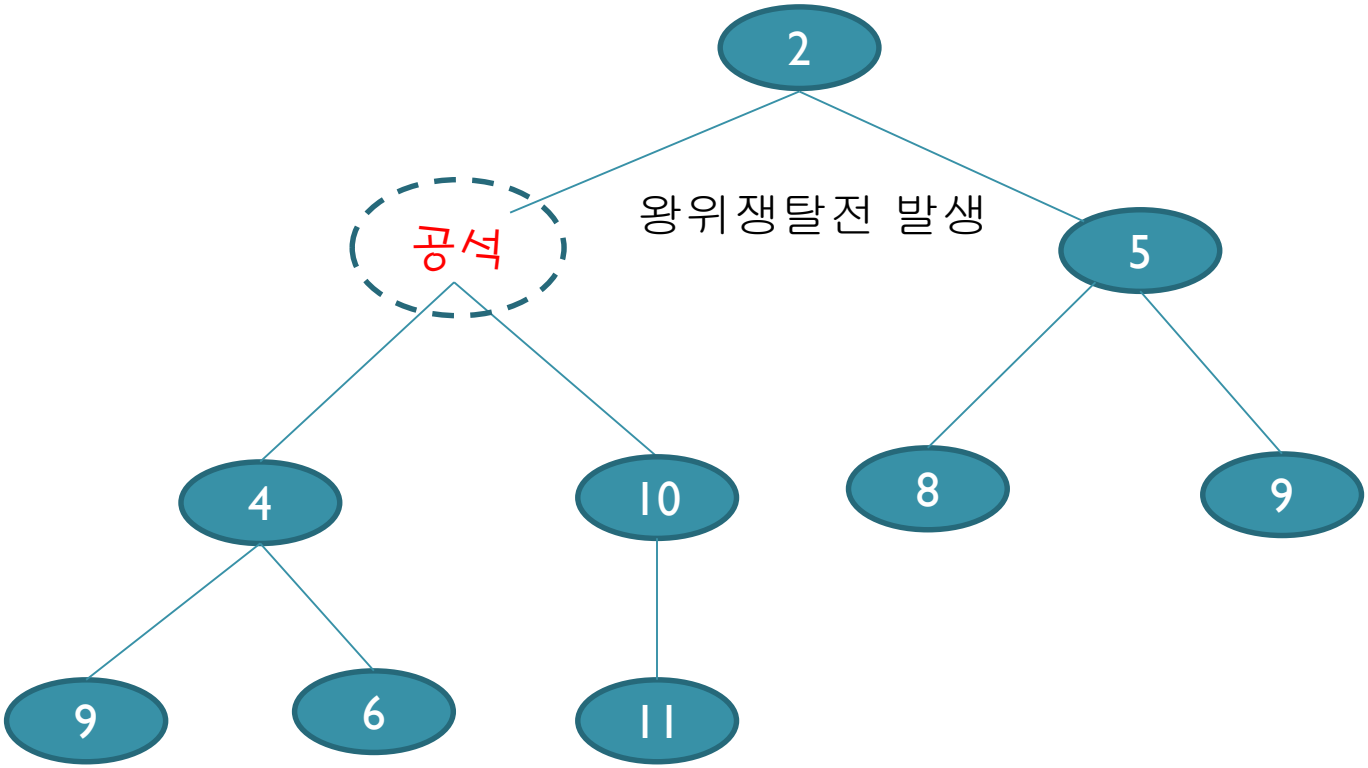
힙에서 데이터를 가지고 오는 법

- 루트를 떼서 주면 최솟값!



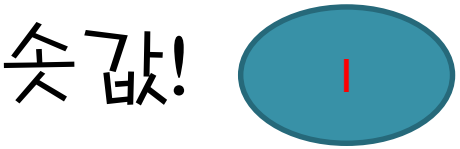
힙에서 데이터를 가지고 오는 법

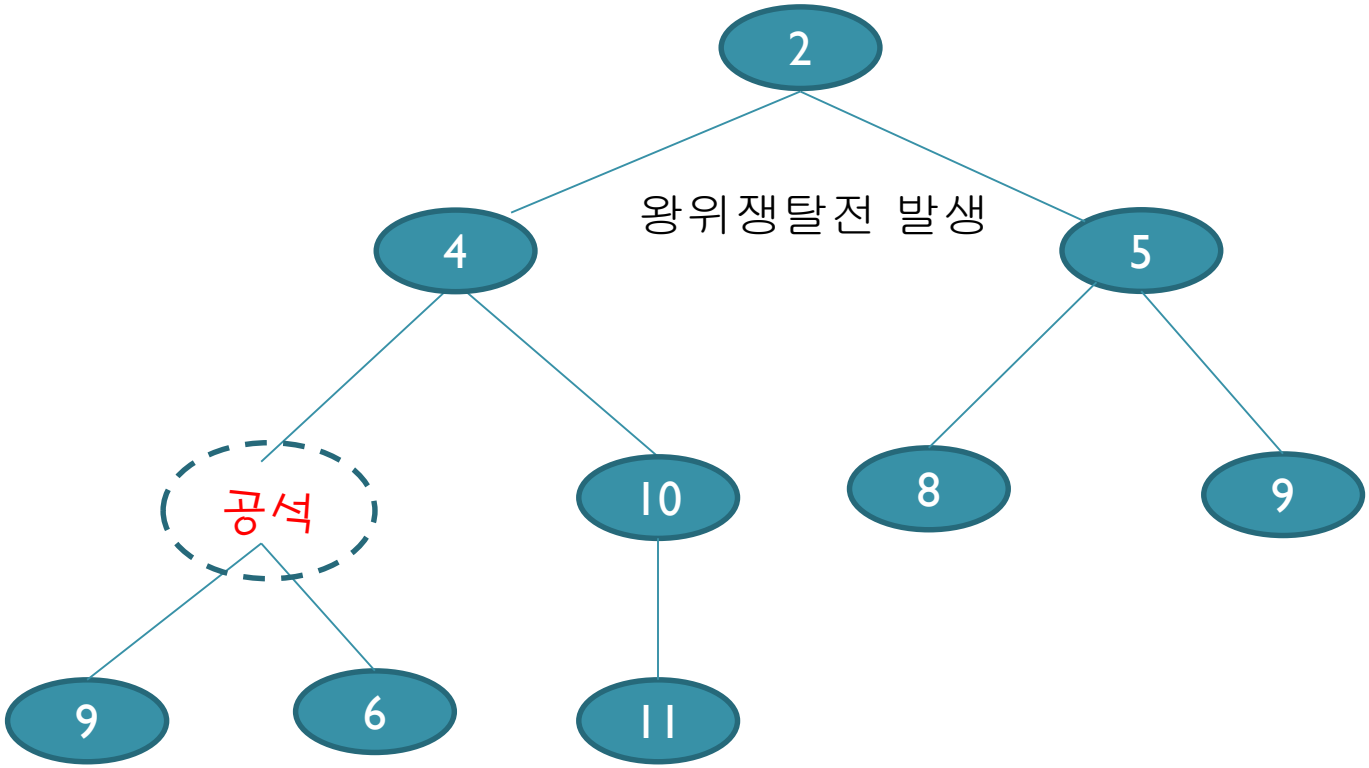
- 루트를 떼서 주면 최솟값!



힙에서 데이터를 가지고 오는 법

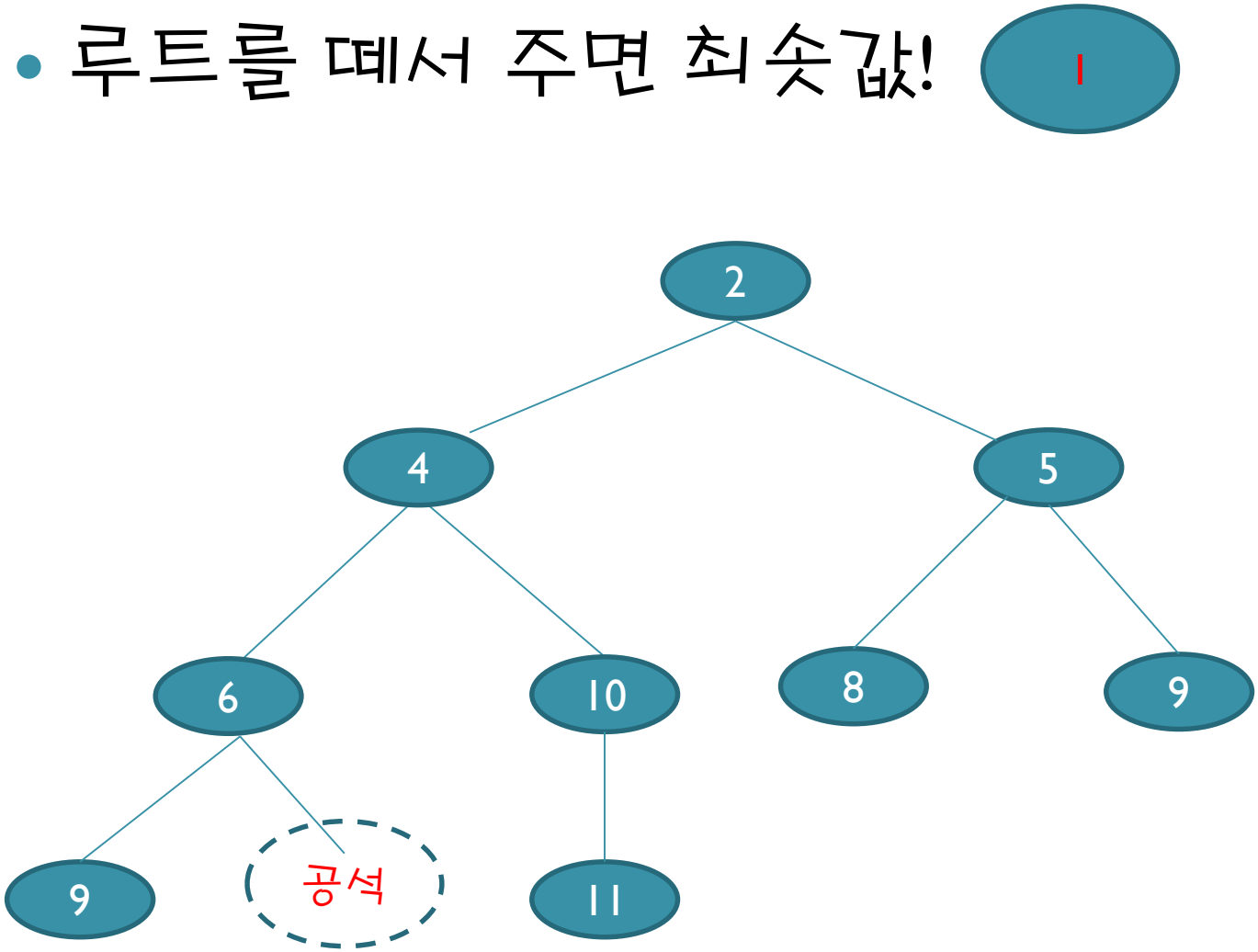
- 루트를 떼서 주면 최솟값!





힙에서 데이터를 가지고 오는 법

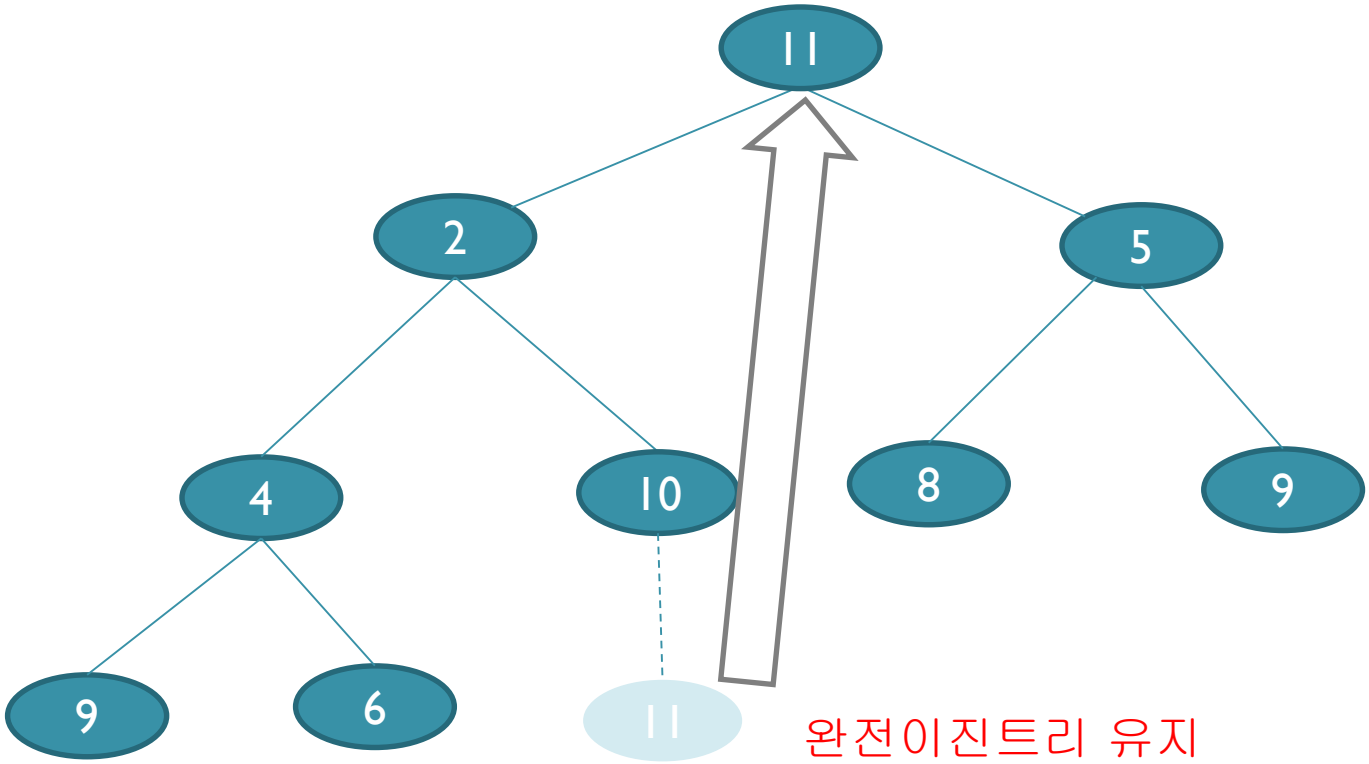
- 루트를 떼서 주면 최솟값!



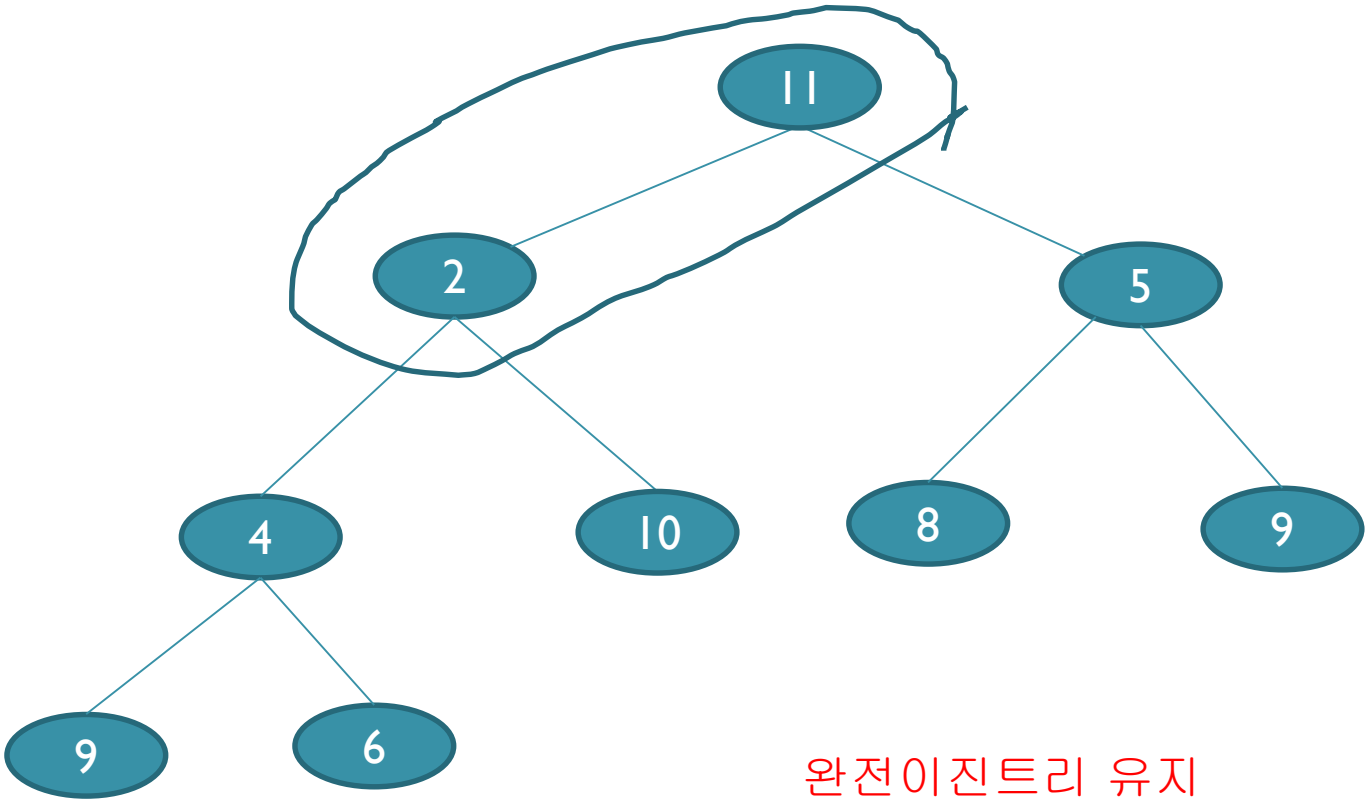
조직 재건???? → 완전이진트리 아님

제대로 된 왕위계승

- 바지 임금을 세운다: 맨 마지막 요소
  - 그리고, 자기 자리로 보낸다

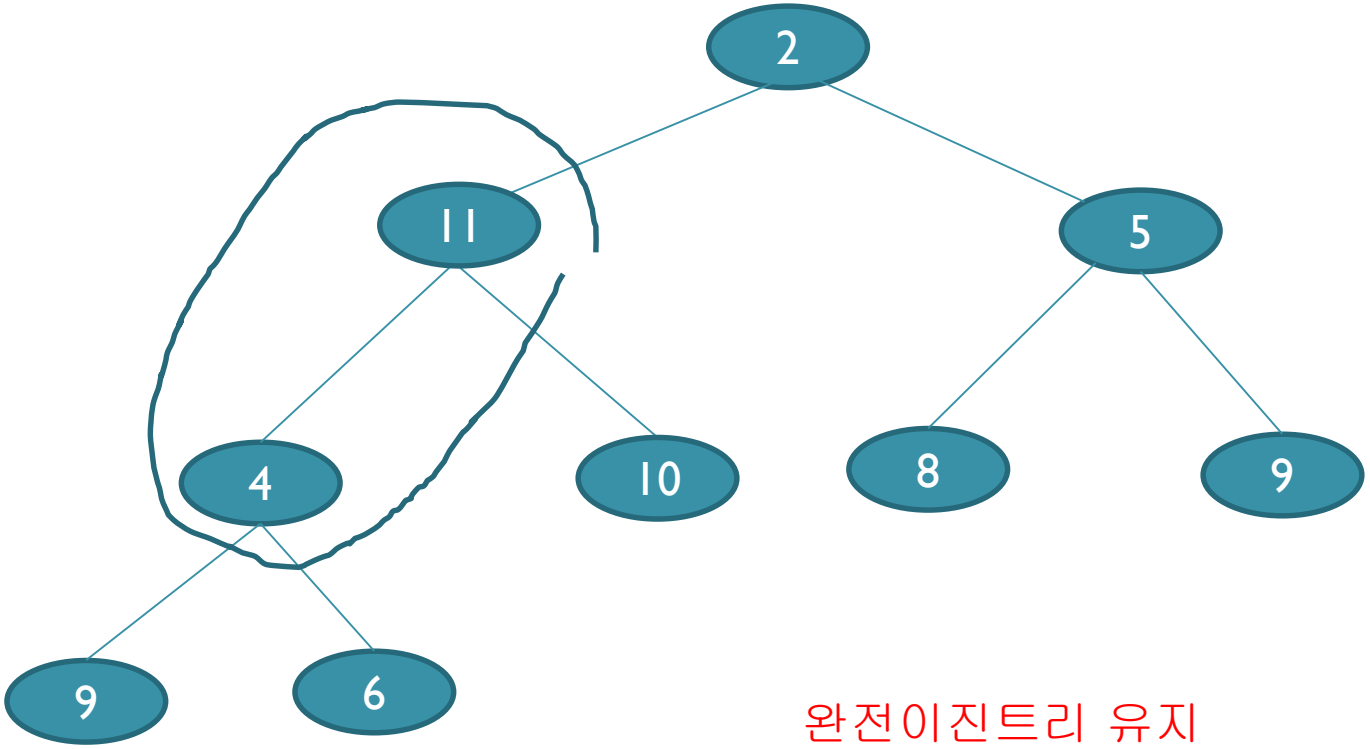


자신의 분수에 맞게 양위



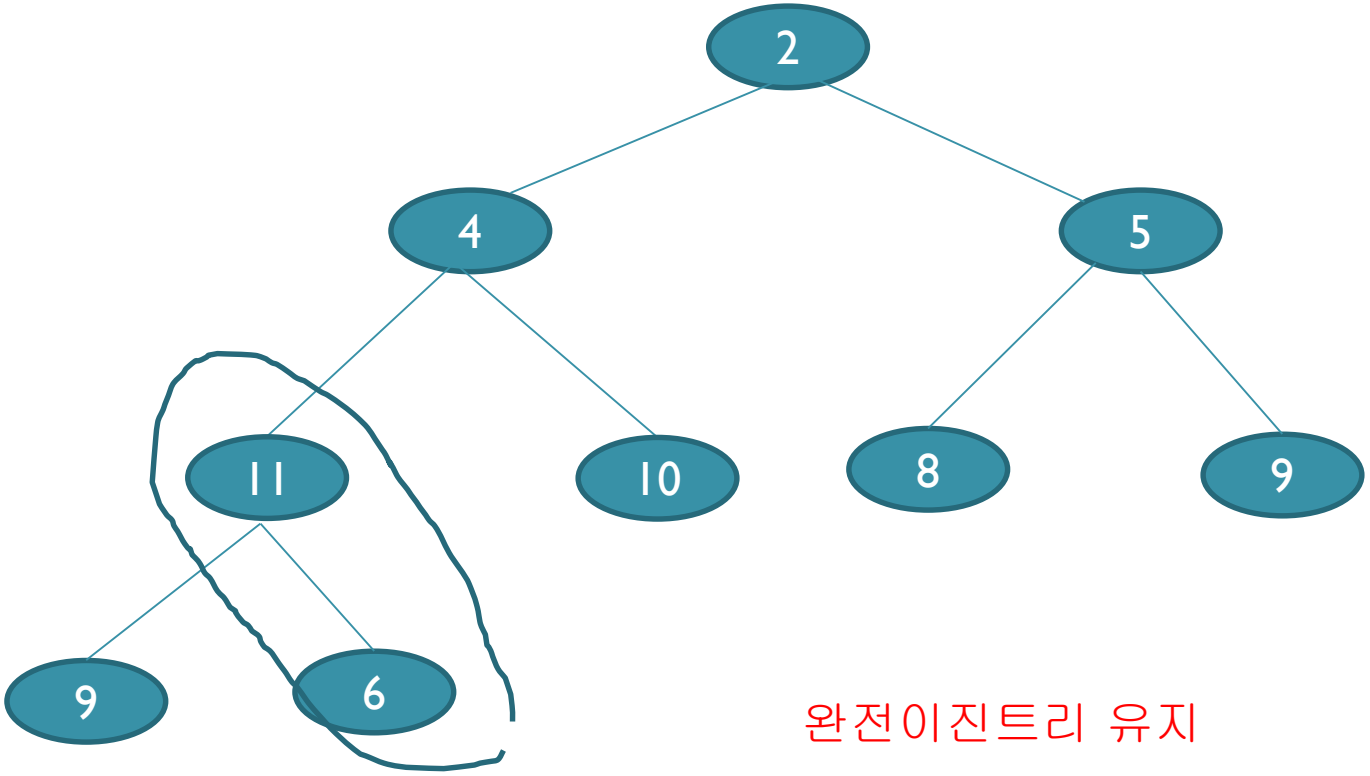
완전이진트리 유지

자신의 분수에 맞게 양위



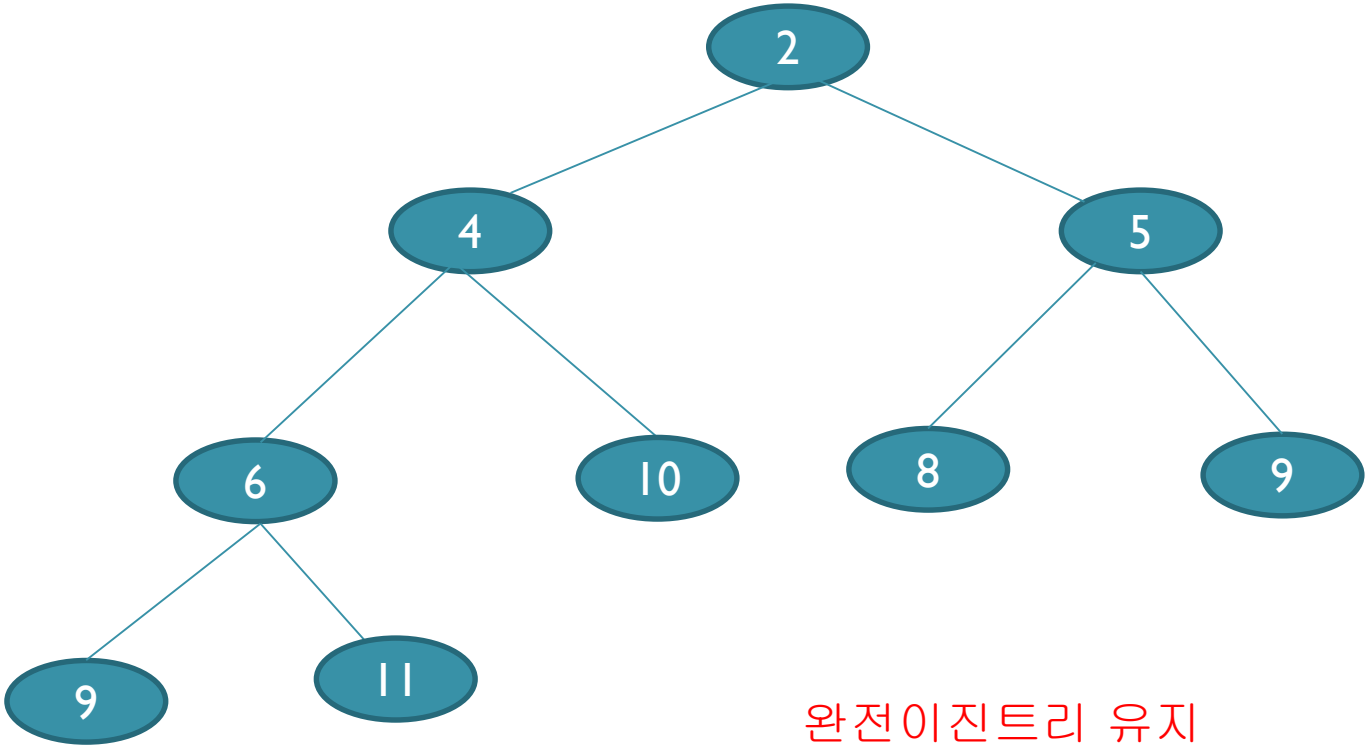
완전이진트리 유지

자신의 분수에 맞게 양위



완전이진트리 유지

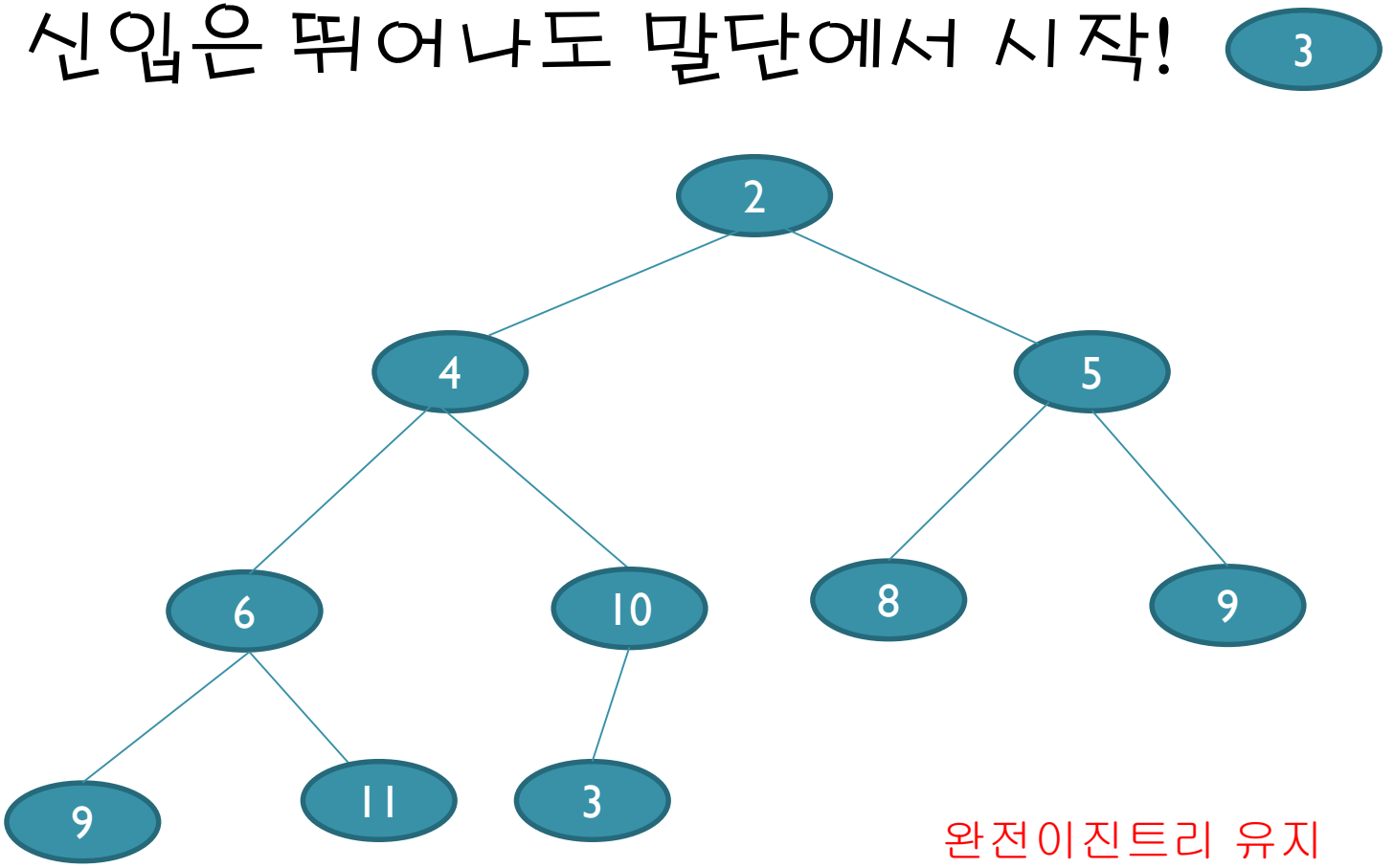
자신의 분수에 맞게 양위



완전이진트리 유지

데이터가 추가되면?

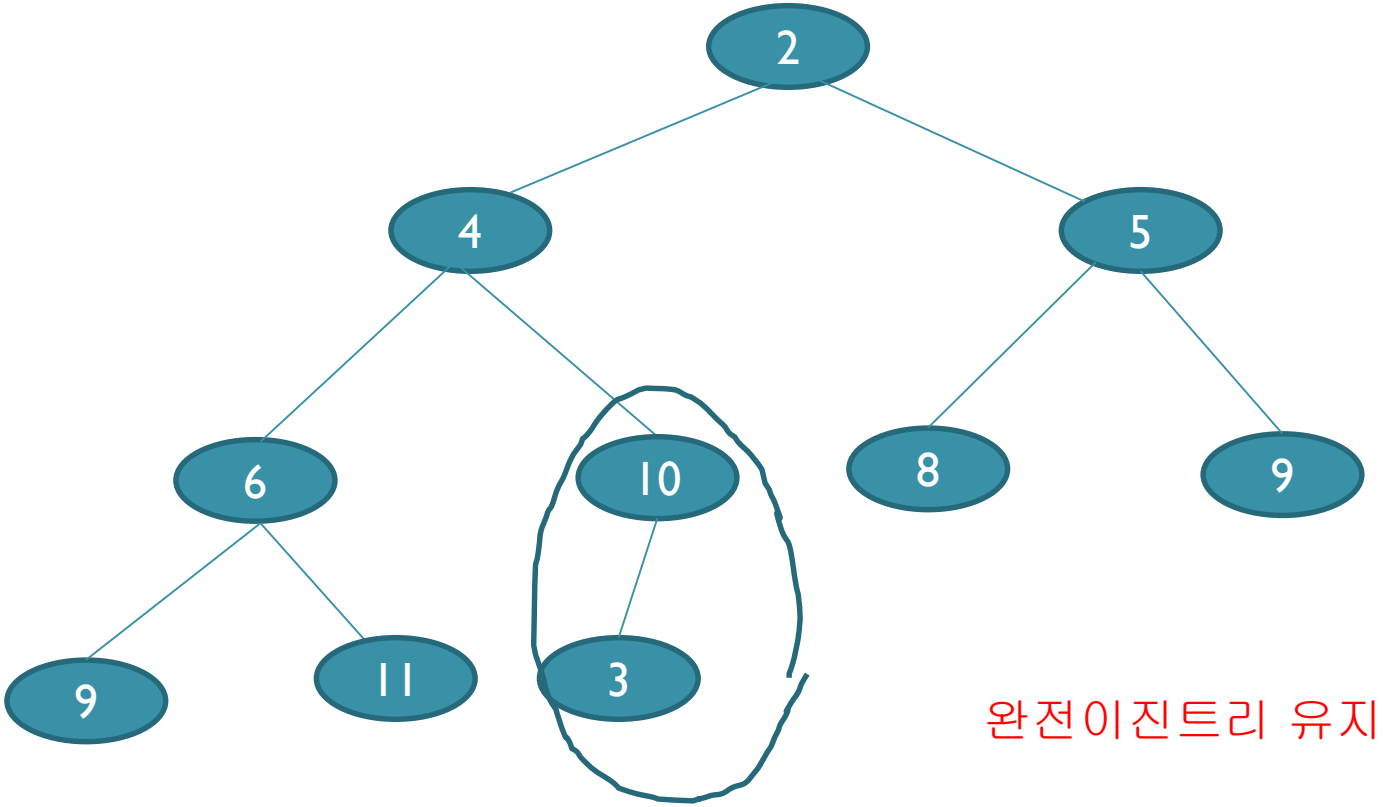
- 신입은 뛰어나도 말단에서 시작!



완전이진트리 유지

데이터가 추가되면?

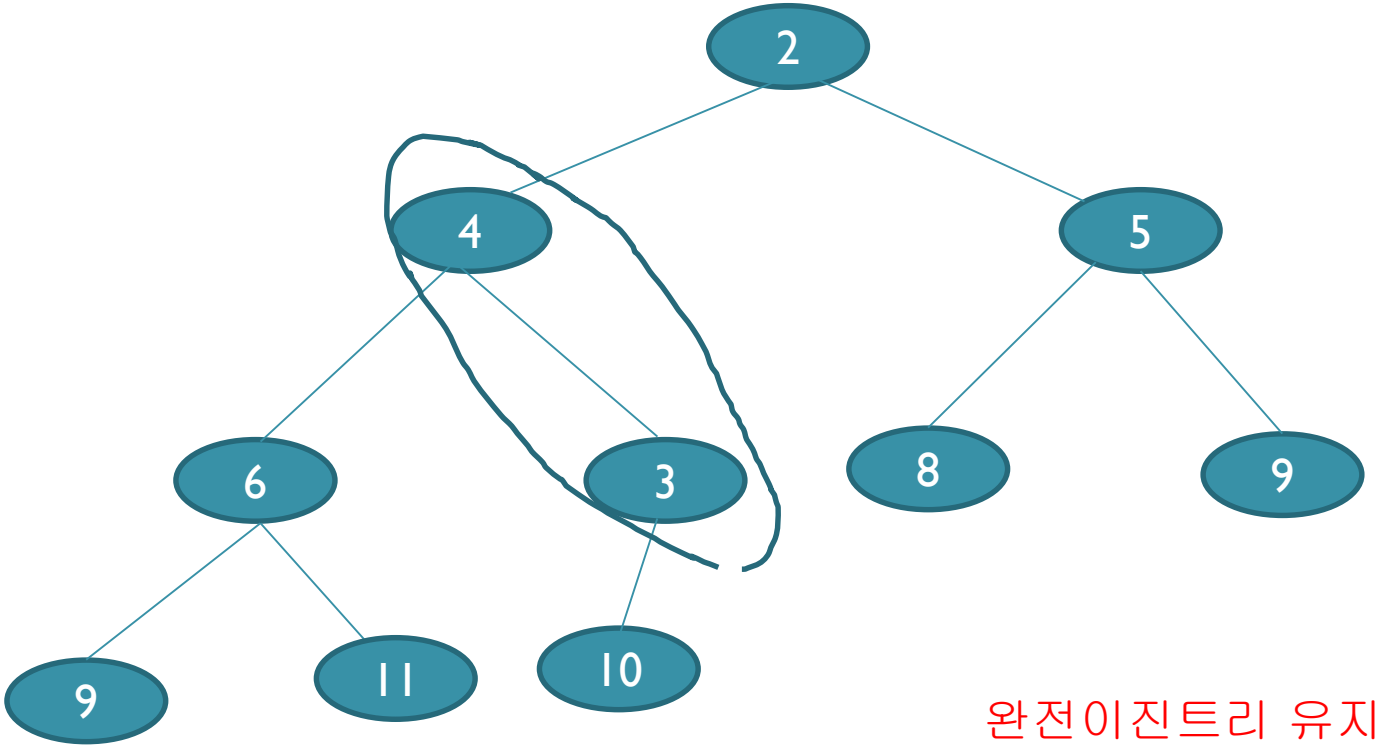
- 실력에 따라 상사를 이기고 올라감



완전이진트리 유지

데이터가 추가되면?

- 실력에 따라 상사를 이기고 올라감

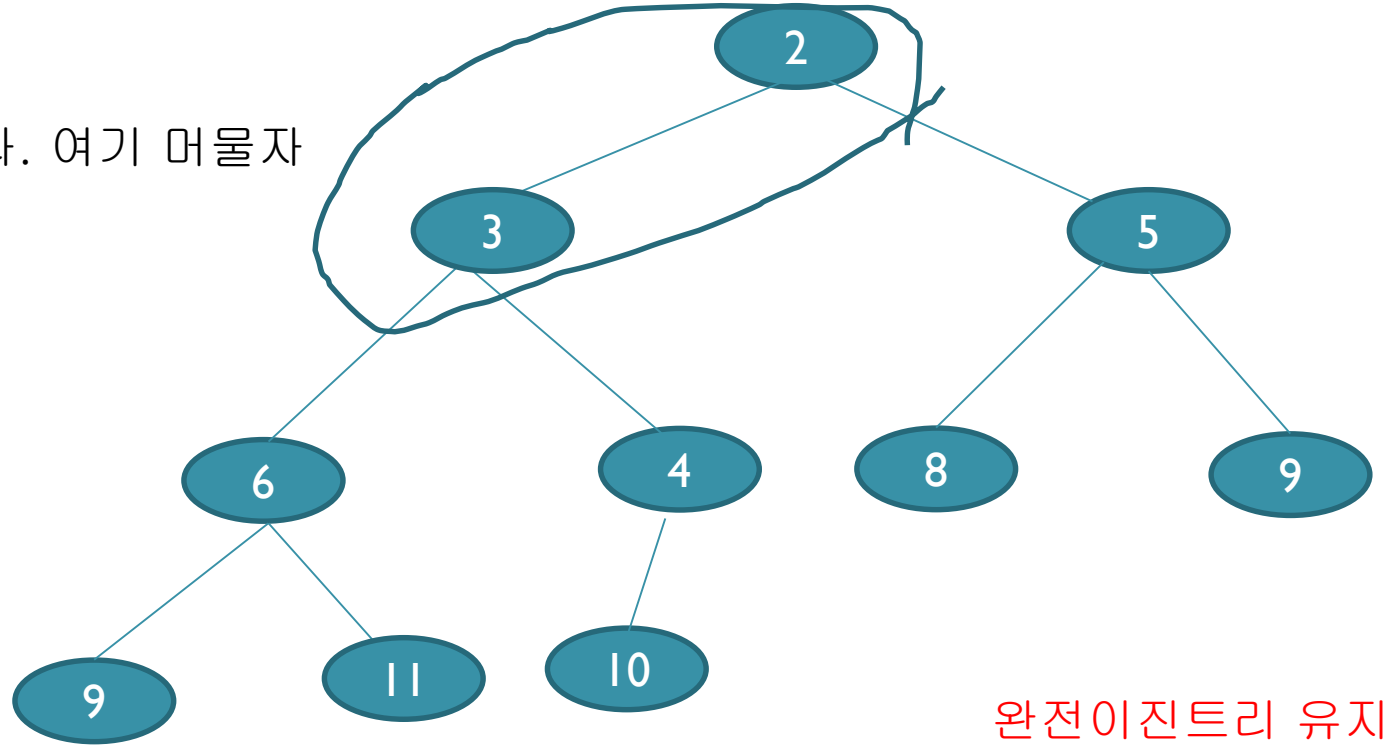


완전이진트리 유지

데이터가 추가되면?

- 실력에 따라 상사를 이기고 올라감

졌다. 여기 머물자



완전이진트리 유지

요약

- 트리는 특별히 지정된 노드인 루트가 있고, 나머지 노드들은 자신이 트리인 서로 겹치지 않는 $T_1, T_2, \cdots, T_n$과 같은 서브 트리들로 이루어진다.
- 트리에서 사용되는 용어들은 루트, 차수, 잎 노드, 자식 노드, 부모 노드, 형제 노드, 중간 노드, 레벨, 깊이, 조상, 자손, 숲 등이다.
- 트리는 연결되어 있고 사이클이 없는 그래프로서 어느 한 연결선만 첨가하더라도 사이클을 형성하게 된다.
- 트리 T가 n-트리란 말은 모든 중간 노드들이 최대 n개의 자식 노드를 가질 때를 말하며, 특히 n이 2인 경우를 이진 트리라고 한다. 이진 트리는 루트와 왼쪽 서브 트리 그리고 오른쪽 서브 트리로 이루어진다.
- 이진 트리에는 사향 이진 트리, 완전 이진 트리, 포화 이진 트리 등이 있으며, 이진 트리가 레벨 i에서 가질 수 있는 최대한의 노드 수는 2^i개이고, 높이가 k인 이진 트리가 가질 수 있는 최대한의 전체 노드 수는 $2^{k+1} - 1$이다.

요약

- 이진 트리를 표현하는 방법으로는 배열에 의한 방법과 연결 리스트에 의한 방법이 있다.
- 트리 탐방의 세 가지 방법에는 중순위 탐방, 전순위 탐방, 후순위 탐방이 있다.
- 중순위 탐방은 루트에서 시작하여 (1) 트리의 왼쪽 서브 트리를 탐방하고, (2) 노드를 방문하고 데이터를 프린트하며, (3) 트리의 오른쪽 서브 트리를 탐방한다.
- 전순위 탐방은 루트에서 시작하여 (1) 노드를 탐방하고 데이터를 프린트하고, (2) 트리의 왼쪽 서브 트리를 탐방하며, (3) 트리의 오른쪽 서브 트리를 탐방한다.
- 후순위 탐방은 루트에서 시작하여 (1) 트리의 왼쪽 서브 트리를 탐방하고, (2) 트리의 오른쪽 서브 트리를 탐방하며, (3) 노드를 탐방하고 데이터를 프린트한다.

요약

- 어떤 그래프 G에서 모든 노드들을 포함하는 트리를 생성 트리라고 하며, 생성 트리의 비용은 트리 연결선의 값을 합한 값이다.
- 최소 비용 생성 트리란 최소한의 비용을 가지는 생성 트리를 말하는데, 최소 비용 생성 트리를 구하는 2가지 방법은 프림의 알고리즘과 크루스칼의 알고리즘이다.

응용

- 트리의 활용
  - 문법의 파싱
  - 허프만 코드
  - 결정 트리
  - 게임 등

응용

- 트리의 응용
  - 통신 네트워크
  - 최적화 문제의 해결
  - 자료의 탐색
  - 정렬 데이터베이스 구성
  - 전기회로망 설계와 응용
  - 분자구조식 설계
  - 유전학
  - 언어들 간의 번역
  - 언어학
  - 사회과학 등