

08

CHAPTER

트리

Tree Theory & Its Applications



단원의 주요 목표

트리의 의미를 이해하고 그래프와 관련된 다양한 문제들을 고찰한다.

- 트리의 개념, 용어, 응용 분야들에 관해 알아본다.
- 방향 트리, 이진 트리, 이진 트리의 표현법을 파악한다.
- 중순위 탐색을 비롯한 3가지 주요 탐색 방법들을 학습한다.
- 최소비용 생성 트리와 프림과 크루스칼의 알고리즘을 알아본다.
- 문법의 파싱과 결정 트리 등 트리의 활용을 이해한다.
- 트리의 응용 분야와 4차 산업혁명과의 관계를 살펴본다.

08

CHAPTER

트리

Tree Theory & Its Applications



CONTENTS

- 8.1 트리의 기본 개념
- 8.2 방향 트리
- 8.3 이진 트리
- 8.4 이진 트리의 표현
- 8.5 이진 트리의 탐색
- 8.6 생성 트리와 최소비용 생성 트리
- 8.7 트리의 활용
- 8.8 트리의 응용과 4차 산업혁명과의 관계
- 요약 및 생활 속의 응용
- 연습문제

트리(Tree)

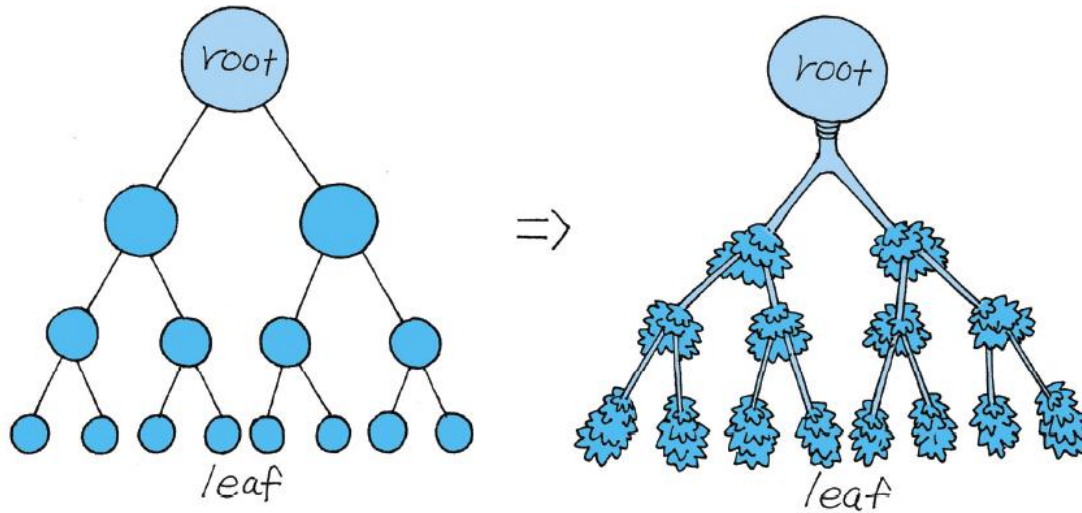
- 그래프 모양이 나무를 거꾸로 세워놓은 것처럼 생겼다고 해서 불리는 이름인데 **트리(Tree)** 또는 **수형도(樹型圖)**라고 함
- 그래프의 특별한 형태로서 컴퓨터를 통한 자료 처리와 응용에 있어서 매우 중요한 역할을 담당함
- 이진 트리의 경우에는 산술적 표현이나 자료 구조 등을 매우 간단하게 표현할 수 있는 장점이 있음
- 컴퓨터 기술의 발전과 더불어 수많은 응용 분야에 적용 가능함

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) $T_1, T_2, \dots, T_n (n \geq 0)$ 으로 나누어진다. 이때 T_1, T_2, \dots, T_n 을 루트의 서브 트리(subtree)라고 한다.

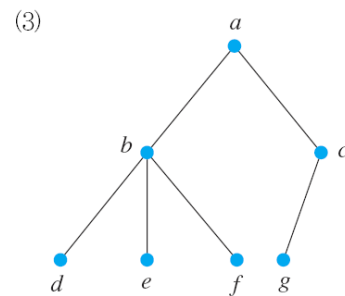
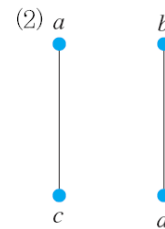
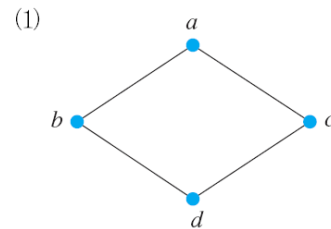


8.1 트리의 기본 개념



예제 8-1

다음의 여러 그래프들이 트리에 해당되는지를 판단해보자.



풀이 (1)과 (2)는 트리가 아니고 (3)과 (4)는 트리이다.

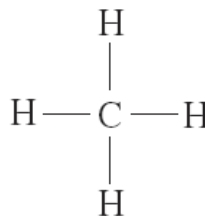
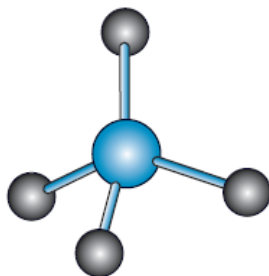
8.1 트리의 기본 개념



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_4 를 분자식으로 가지는 메탄의 구조는 트리의 형태로 나타냄

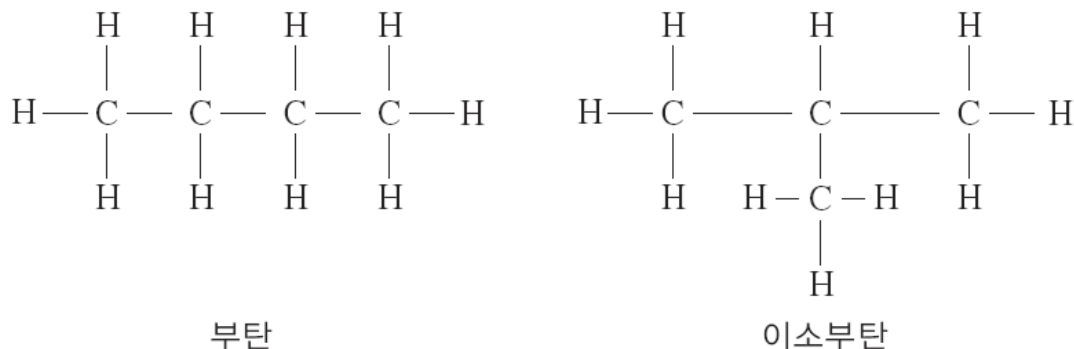


〈그림 8.1〉 메탄의 구조와 트리의 표현

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

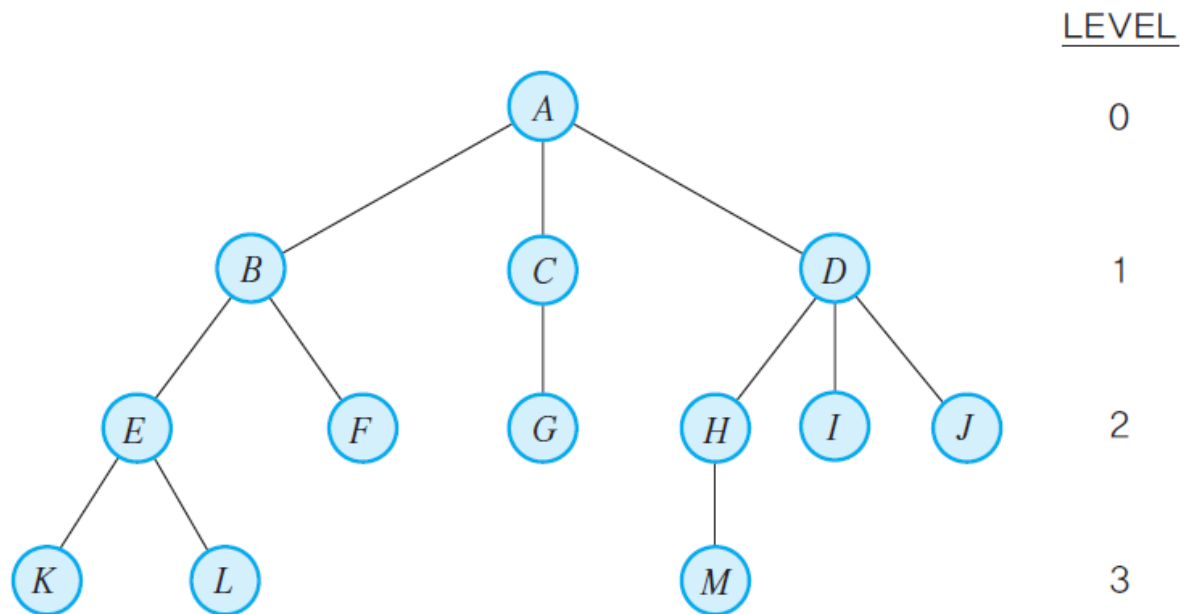
- 최적화 문제의 해결
- 알고리즘(데이터 구조, 정렬)
- 분자구조식 설계와 화학결합의 표시, 유전학
- 언어들 간의 번역, 언어학
- 자료의 탐색, 정렬, 데이터베이스 구성
- 사회과학(조직의 분류에 있어서의 구조) 등



정의 8-2

트리에서 사용되는 용어들은 다음과 같다. 보다 이해를 쉽게 하기 위하여 <그림 8.3>을 예로 들어 설명한다.

8.1 트리의 기본 개념



〈그림 8.3〉 트리의 예

8.1 트리의 기본 개념



- 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

8.1 트리의 기본 개념



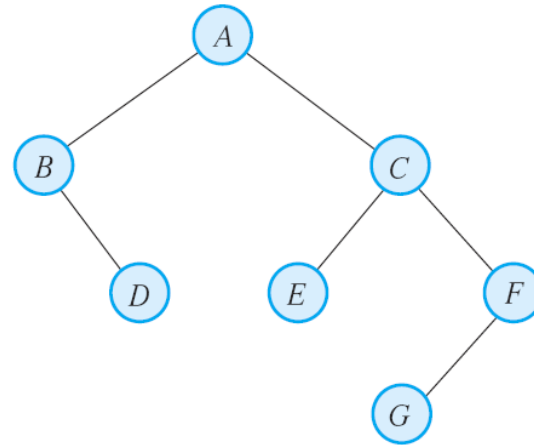
- 9) **자손(descendant)** : 그 노드로 부터 잎 노드에 이르는 경로상에 나타난 모든 노드 들을 말하는데 B의 자손은 E, F, K, L이고 H의 자손은 M임
- 10) **레벨(level)** : 루트의 레벨을 0으로 놓고 자손 노드로 내려가면서 하나씩 증가한다. 즉, 어떤 노드의 레벨이 p라면 그것의 자식 노드의 레벨은 p+1이 됨
- 11) **트리의 높이(height)** : 트리에서 노드가 가질 수 있는 최대 레벨로서 트리의 깊이(depth)라고도 한다. 이 트리의 높이는 3이 됨
- 12) **숲(forest)** : 서로 연결되지 않는 트리들의 집합으로서 트리에서 루트를 제거하면 숲을 얻을 수 있음

8.1 트리의 기본 개념



예제 8-2

다음 트리에서 주어진 물음에 답해보자.



- (1) 잎 노드
- (2) 트리의 높이
- (3) C 의 차수
- (4) G 의 조상 노드

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

8.1 트리의 기본 개념



정리 8-1

n 개의 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) G 는 트리이다.
- (2) G 는 연결되어 있고 $m = n - 1$ 이다.
- (3) G 는 연결되어 있고 어느 한 연결선만을 제거하더라도 G 는 연결되지 않는다.
- (4) G 는 사이클을 가지지 않으며 $m = n - 1$ 이다.
- (5) G 는 어느 한 연결선만 첨가하더라도 사이클을 형성하게 된다.

8.1 트리의 기본 개념



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 T 가 n -트리(n -ary tree)란 말은 모든 중간 노드들이 최대 n 개의 자식 노드를 가질 때를 말하며, 특히 n 이 2인 경우를 이진 트리(binary tree)라고 한다.

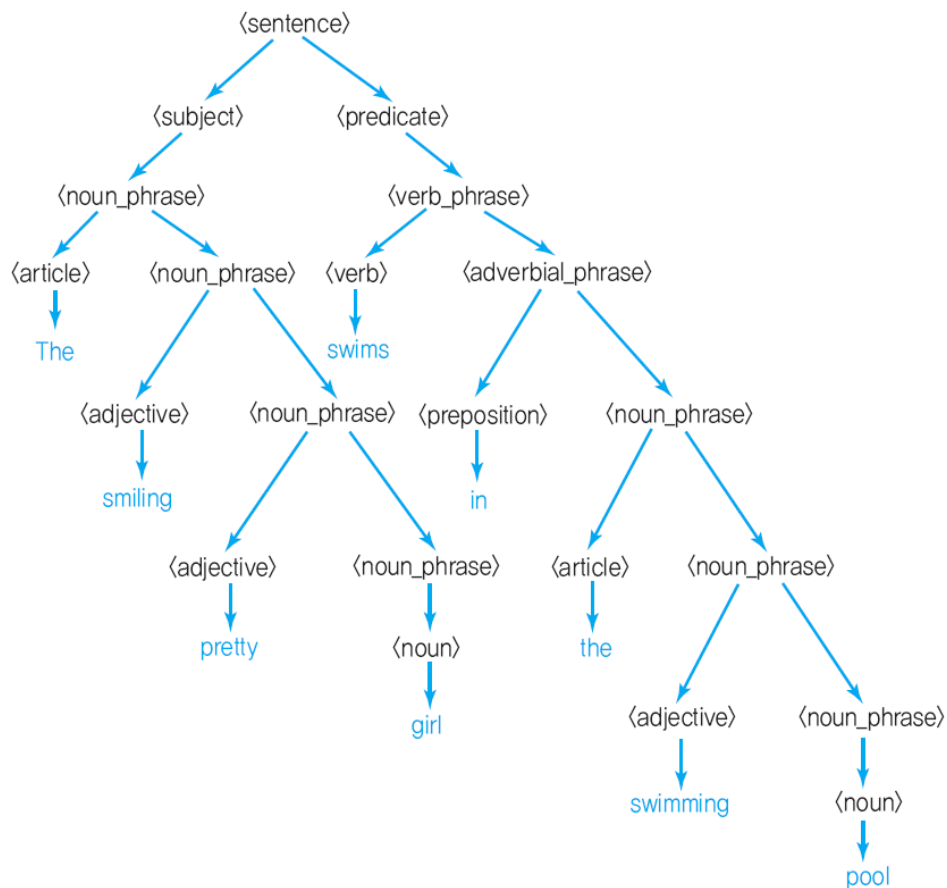
8.2 방향 트리



방향이 있고 순서화된 트리는 다음의 성질들을 만족하는 방향 그래프임

- 1) 선행자가 없는 루트라고 불리는 노드가 하나 있으나, 이 루트에 서는 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

8.2 방향 트리



〈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 k 일 때 레벨 1부터 $k-1$ 까지는 모두 차 있고 레벨 k 에서는 왼쪽 노드부터 차례로 차 있는 이진 트리임

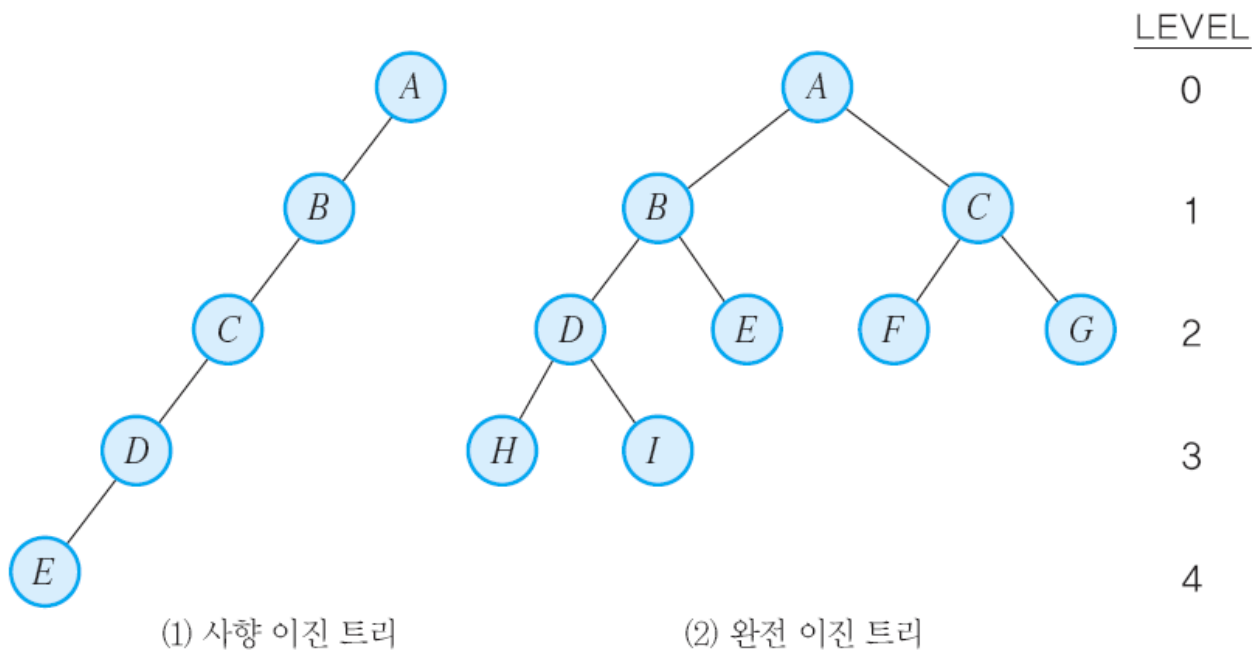
포화 이진 트리(full binary tree)

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

8.3 이진 트리



여러 가지 이진 트리

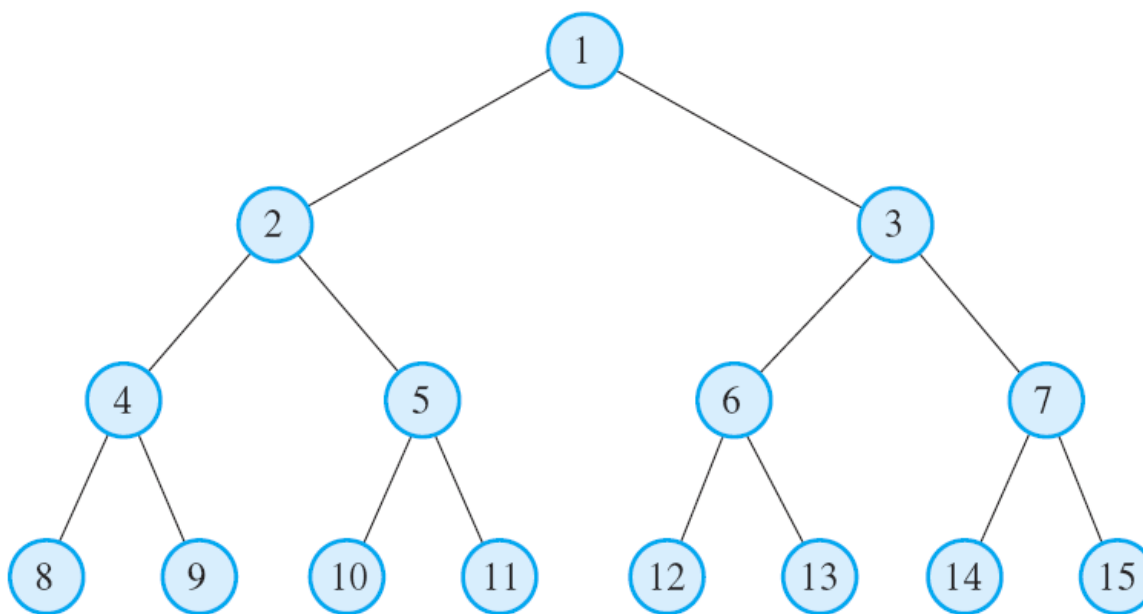


〈그림 8.5〉 이진 트리의 예

8.3 이진 트리



포화 이진 트리



〈그림 8.6〉 포화 이진 트리의 예

8.3 이진 트리



정리 8-3

이진 트리가 레벨 i 에서 가질 수 있는 최대한의 노드 수는 2^i 개이며, 높이가 k 인 이진 트리가 가질 수 있는 최대한의 전체 노드 수는 $2^{k+1}-1$ 이다. 예를 들어, 레벨이 2인 경우에는 최대한 2^2 개의 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_2 라고 할 때 $n_0 = n_2 + 1$ 의 식이 항상 성립한다.

8.4 이진 트리의 표현

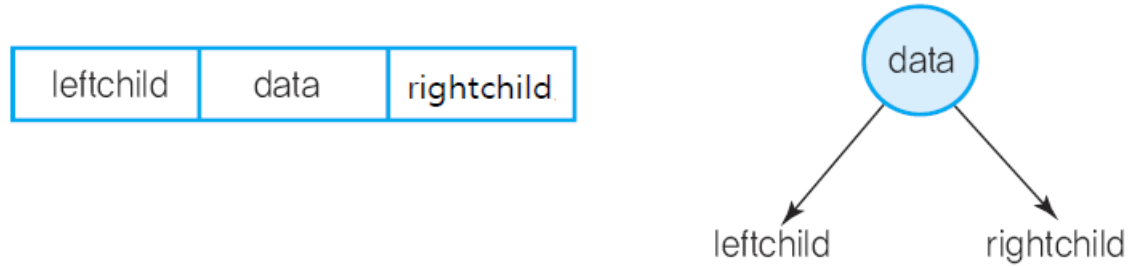


이진 트리를 표현하는 방법

- 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



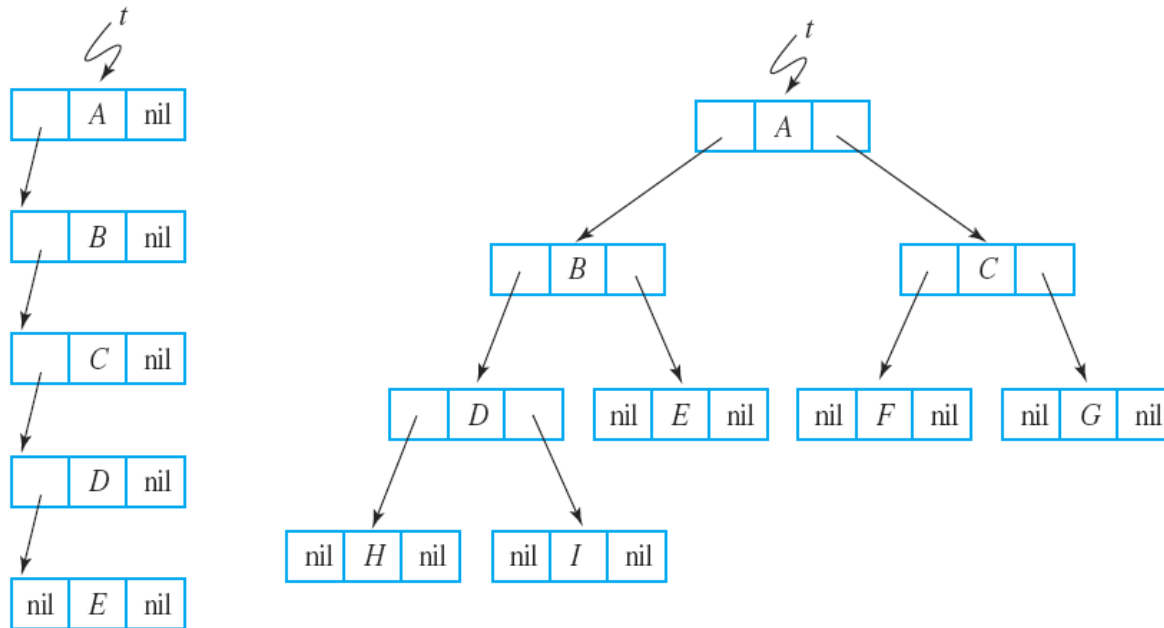
〈그림 8.7〉 노드의 구조

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

8.4 이진 트리의 표현



이진 트리의 연결 리스트에 의한 표현



〈그림 8.8〉 이진 트리의 연결 리스트에 의한 표현



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!!

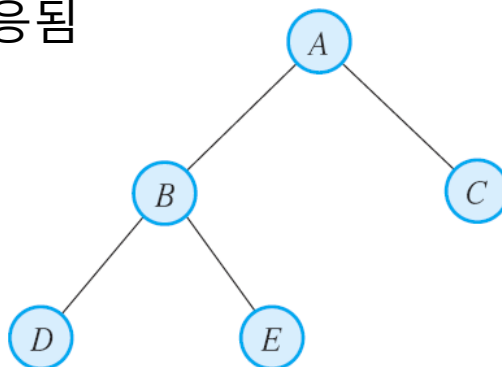
트리 탐방의 세 가지 방법은 다음과 같다

- (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)** **탐방**이라고 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

8.5 이진 트리의 탐방



중순위 탐방의 재귀적 알고리즘(Recursive algorithm)

중순위 탐방은 루트로부터 시작하여

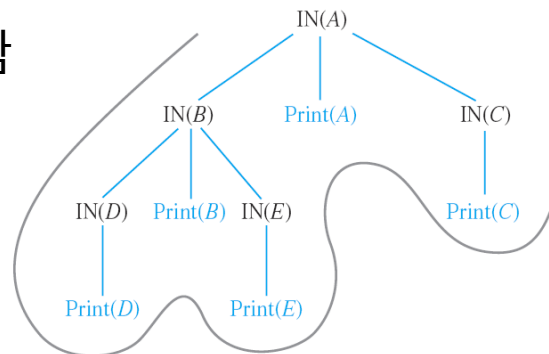
inorder(currentnode→leftchild)

printf(currentnode→data)

inorder(currentnode→rightchild)

의 트리 형태로 계속 전개시켜 나가서 프린트된 결과를 왼쪽부터 차례로 적어 나감. [가장 쉽고도 정확한 방법임 – 직접 해보아야 함]

- 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);
    }
}
```

8.5 이진 트리의 탐방



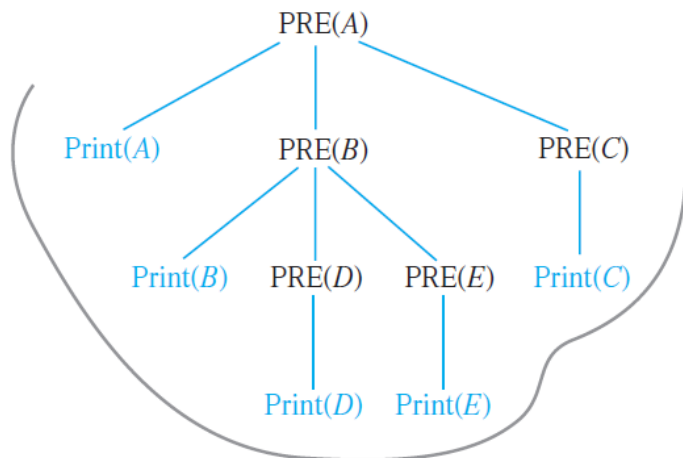
전순위 탐방(D, L, R)

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

8.5 이진 트리의 탐방



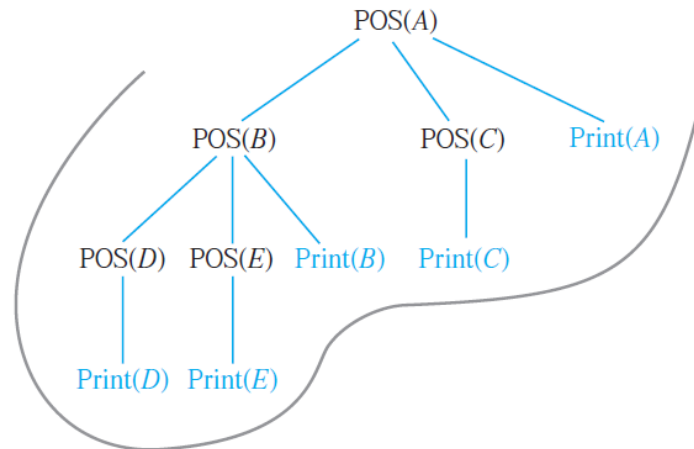
후순위 탐방(L, R, D)

postorder(currentnode→leftchild)

postorder(currentnode→rightchild)

printf(currentnode→data)

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



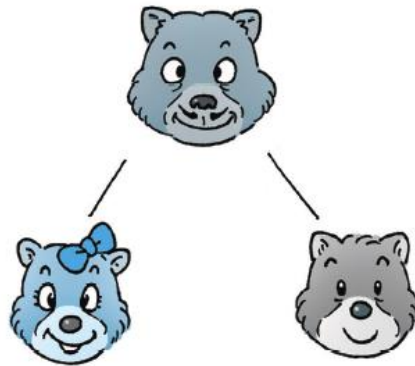
〈그림 8.12〉 후순위 탐방

8.5 이진 트리의 탐방



여기서 잠깐!!

중순위, 전순위, 후순위 탐방의 순서는 다음과 같다.



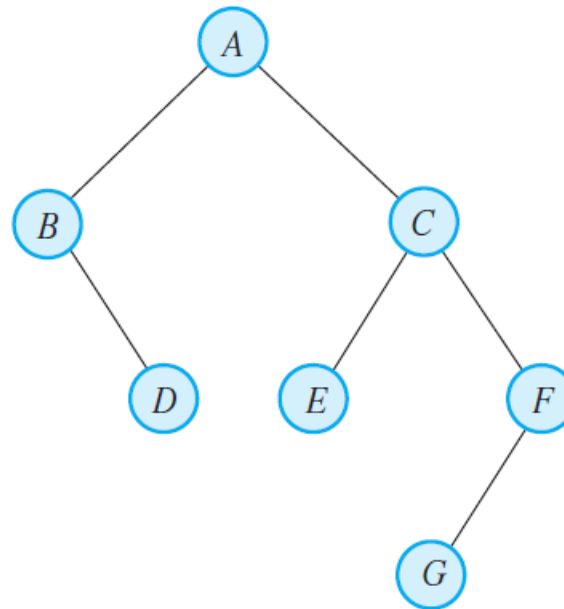
- 중순위 : - -
- 전순위 : - -
- 후순위 : - -

8.5 이진 트리의 탐방



예제 8-4

다음의 이진 트리에서 중순위 탐방, 전순위 탐방, 후순위 탐방의 결과를 각각 구해보자.

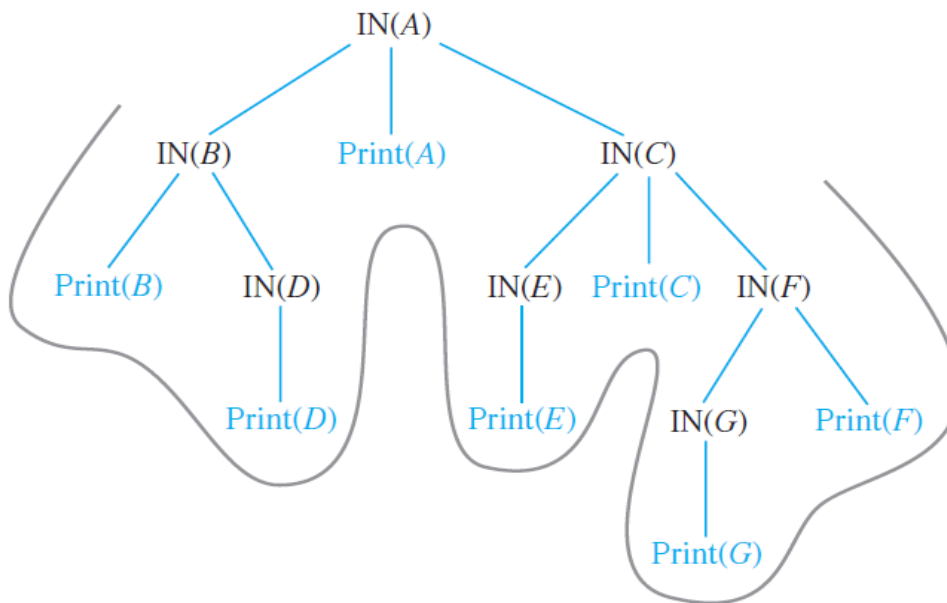
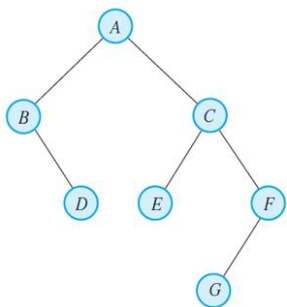


8.5 이진 트리의 탐방



중순위 탐방 과정

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

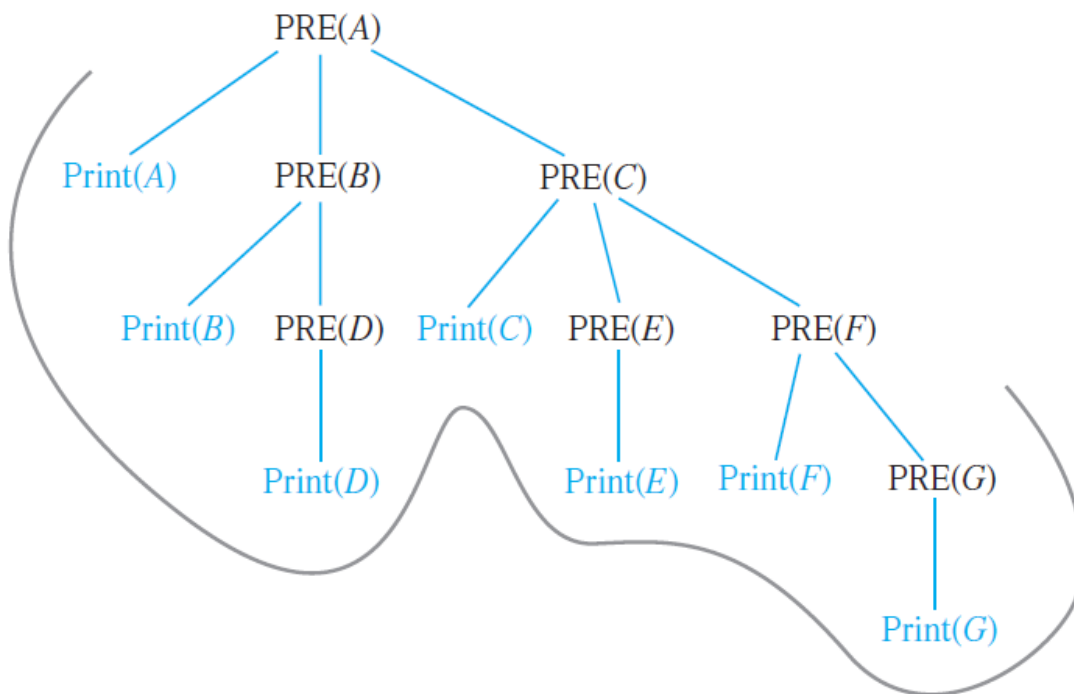
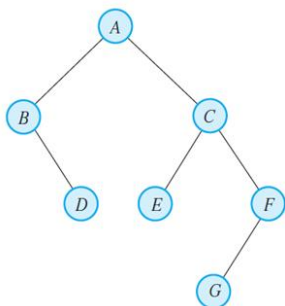


8.5 이진 트리의 탐방



전순위 탐방 과정

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

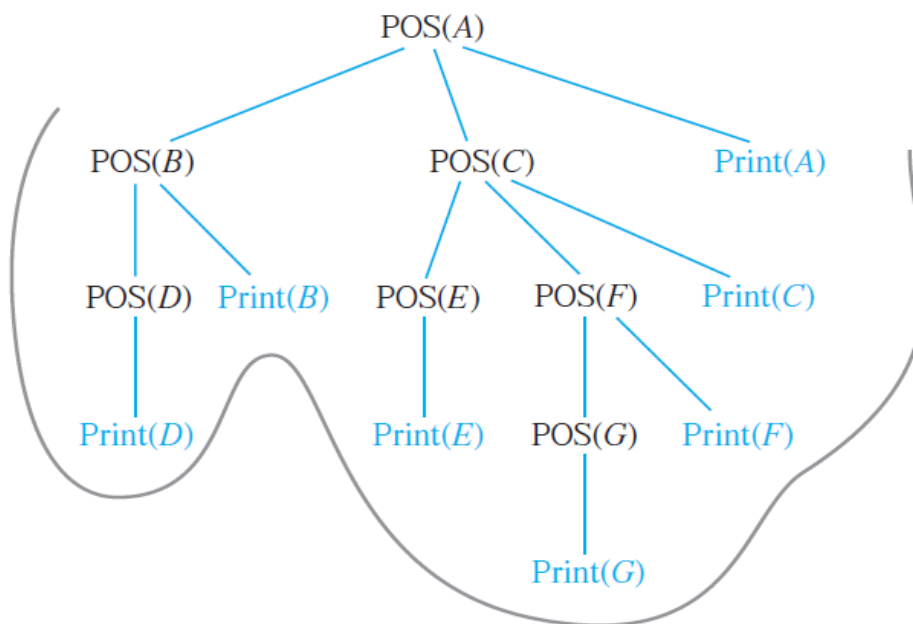
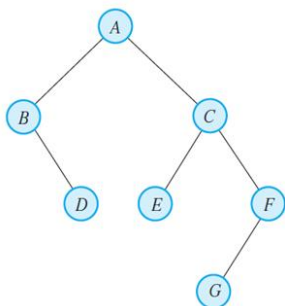


8.5 이진 트리의 탐방



후순위 탐방 과정

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

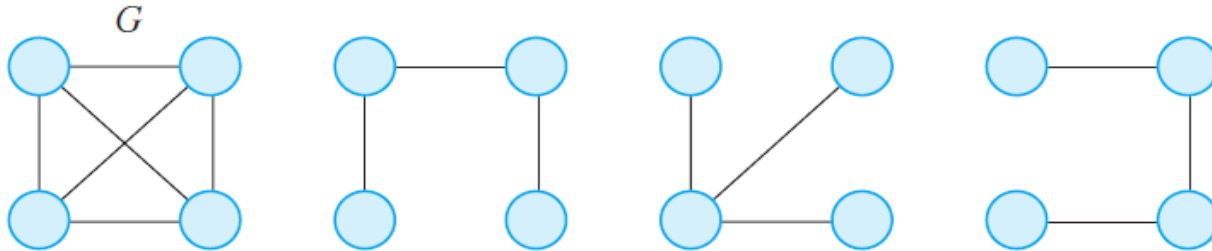


8.6 생성 트리와 최소비용 생성 트리



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

주어진 그래프 G 와 그것의 3가지 생성 트리들



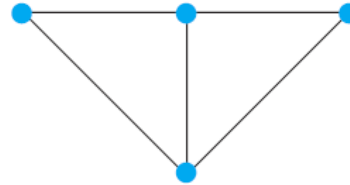
〈그림 8.13〉 주어진 그래프와 그것의 생성 트리들

8.6 생성 트리와 최소비용 생성 트리

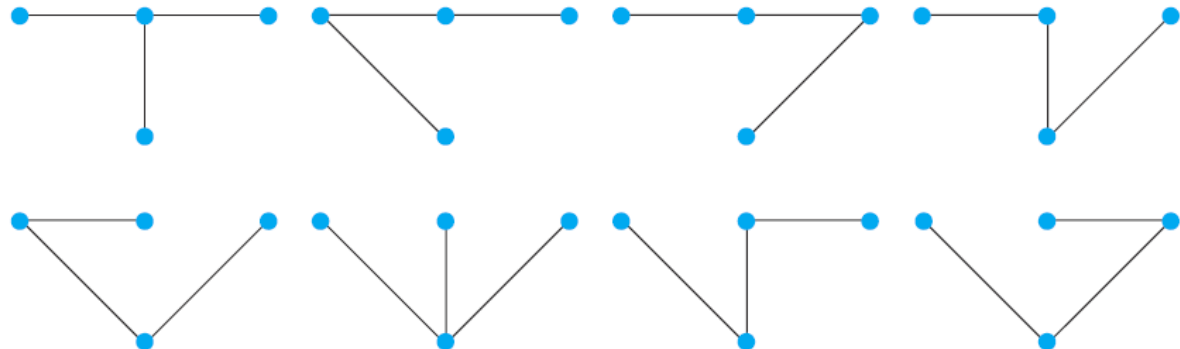


예제 8-5

다음과 같은 그래프 G 의 생성 트리를 모두 구해보자.



풀이 G 의 생성 트리는 모두 8가지인데 다음과 같다.



8.6 생성 트리와 최소비용 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 트리**의 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 알고리즘**과 **크루스칼(Kruskal)의 알고리즘**임

8.6 생성 트리와 최소비용 생성 트리



알고리즘 1

프림의 알고리즘(Prim's algorithm)

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

- (1) 노드의 집합 U 를 1로 시작한다.
- (2) $u \in U, v \in V - U$ 일 때 U 와 $V - U$ 를 연결하는 가장 짧은 연결선인 (u, v) 를 찾아서 v 를 U 에 포함시킨다. 이때 (u, v) 는 사이클(cycle)을 형성하지 않는 것이라야 한다.
- (3) (2)의 과정을 $U = V$ 때까지 반복한다.

8.6 생성 트리와 최소비용 생성 트리



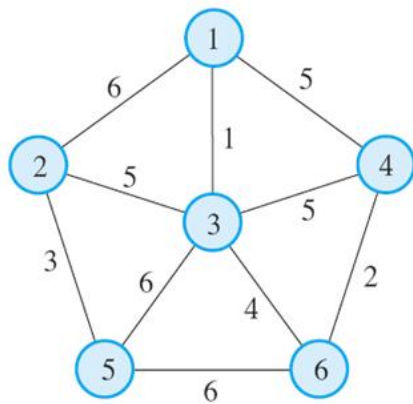
```
void Prim(graph G: set_of_edges T)
    /* 프림의 알고리즘은 G에 대한 최소 비용 생성 트리를 만든다. */
    {
        set_of_vertices U;
        vertex u, v;
        T = NULL
        U = {1};
        while( U != V )
        {
            let (u, v) be a lowest cost edge such that u is in U and v is in V - U;
           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

[프로그램 8.4] 프림의 알고리즘

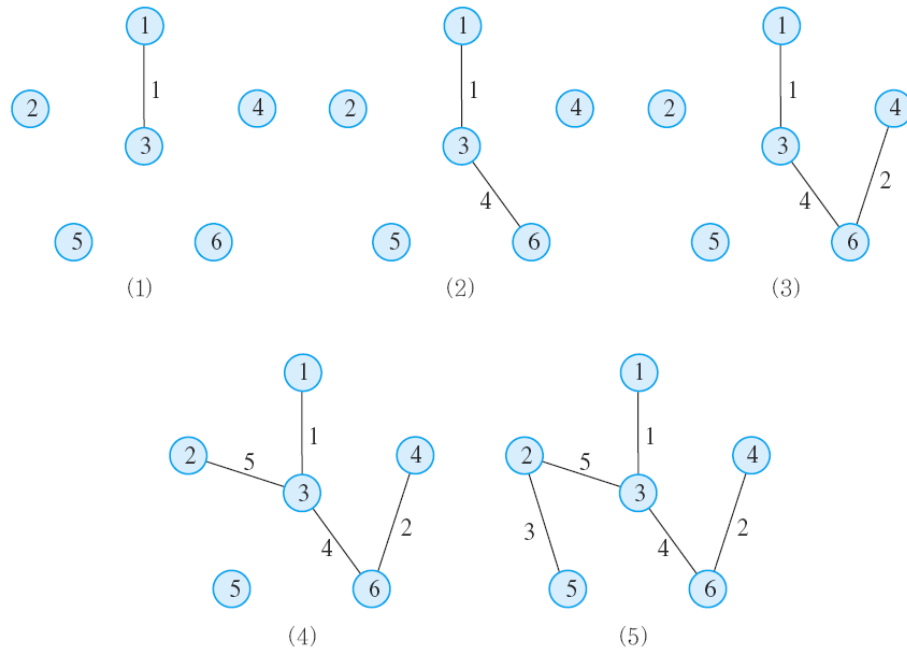
8.6 생성 트리와 최소비용 생성 트리



프림의 알고리즘으로 MST 구하는 과정



〈그림 8.14〉 가중 그래프



〈그림 8.15〉 프림의 알고리즘에 의한 MST의 생성 과정

8.6 생성 트리와 최소비용 생성 트리



알고리즘 2

크루스칼의 알고리즘(Kruskal's algorithm)

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

- (1) T 를 ϕ 로 놓는다.
- (2) 연결선의 집합 E 를 비용이 적은 순서로 정렬한다.
- (3) 가장 최소값을 가진 연결선 (u, v) 를 차례로 찾아서 (u, v) 가 사이클을 이루지 않으면 (u, v) 를 T 에 포함시킨다.
- (4) (3)의 과정을 $|T| = |V| - 1$ 일 때까지 반복한다.

8.6 생성 트리와 최소비용 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 contains fewer than n-1 edges) printf("no spanning tree\n");
```

[프로그램 8.6] 크루스칼의 알고리즘

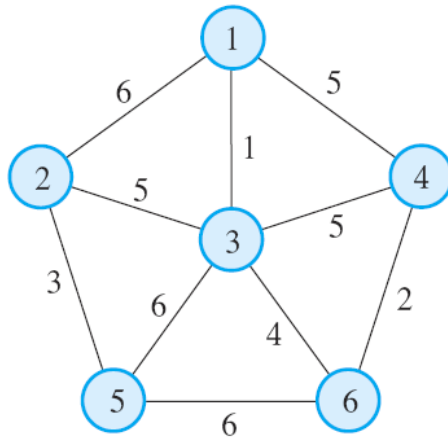
8.6 생성 트리와 최소비용 생성 트리



여기서 잠깐!!

크루스칼 알고리즘의 쉬운 해설

먼저 연결선들을 비용이 적은 순서대로 나열한다. 그 후 연결선을 제거한 그래프에서 연결선의 비용이 적은 연결선의 순서로 사이클이 생기지 않도록 연결해 나가면 된다. 만약 비용이 같은 경우에는 어떤 순서로 적용해도 무방하다.



〈그림 8.14〉 가중 그래프

먼저 비용이 적은 순서로 나열

1, 2, 3, 4, 5, 5, 5, 6, 6, 6

8.6 생성 트리와 최소비용 생성 트리

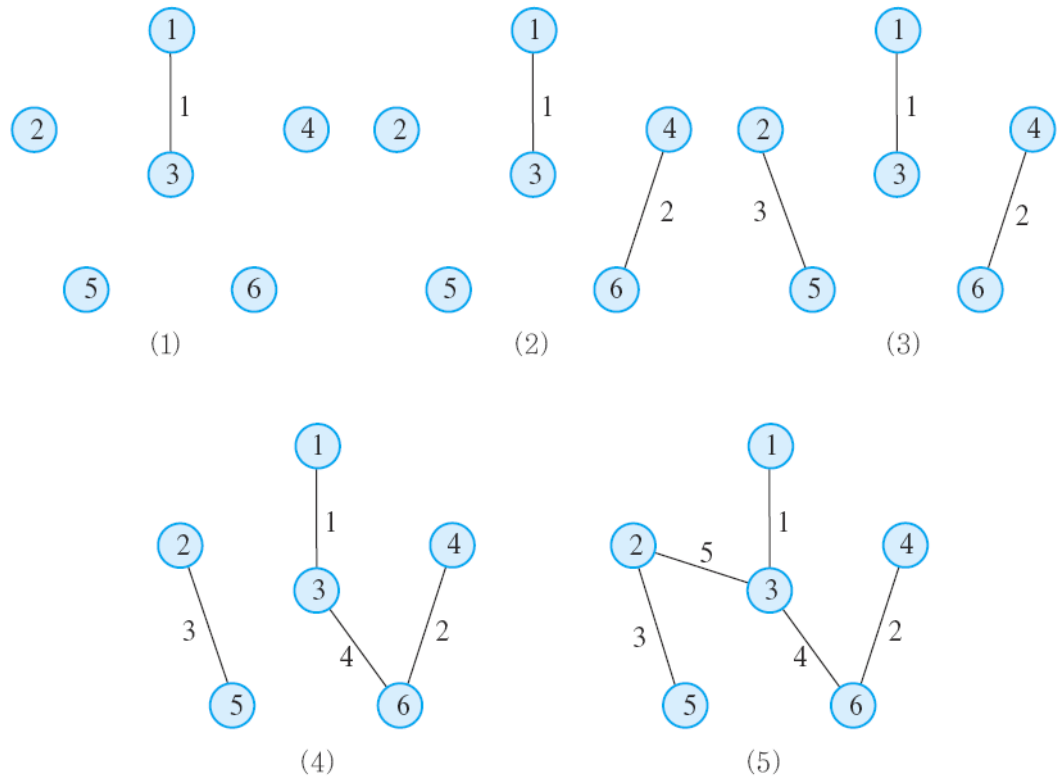


크루스칼의 알고리즘으로 MST 구하는 과정

비용이 적은 순서대로 적용

1, 2, 3, 4, 5, 5, 5, 6, 6, 6

단, 같은 값일 때는 어느 것을 선택하여
연결해도 관계없음



〈그림 8.16〉 크루스칼의 알고리즘에 의한 MST의 생성 과정

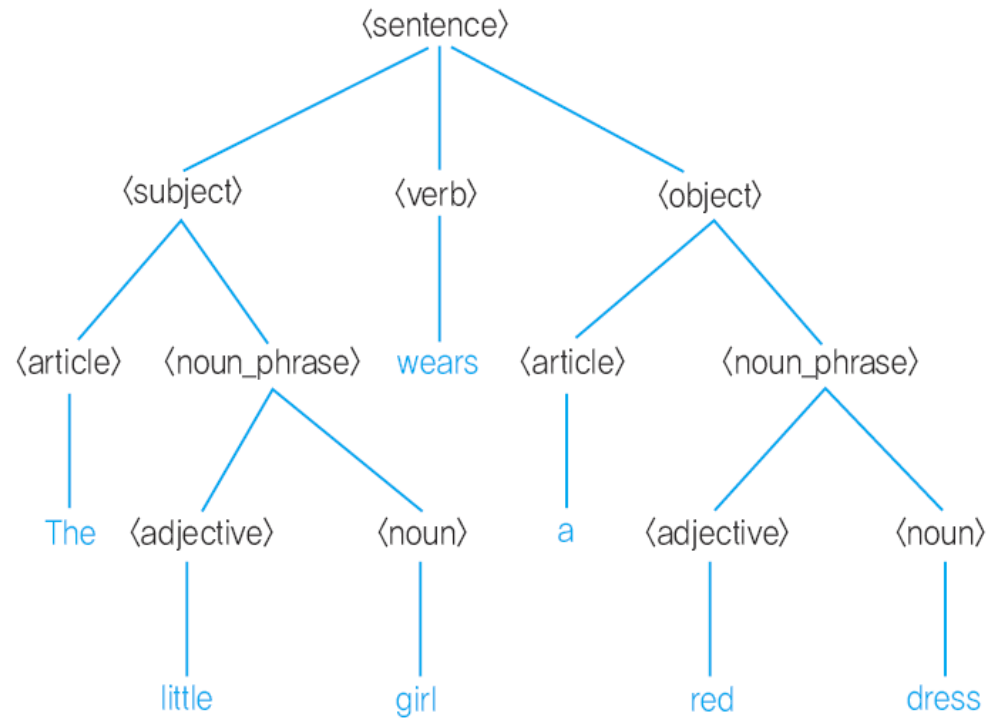
8.7 트리의 활용



(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) 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

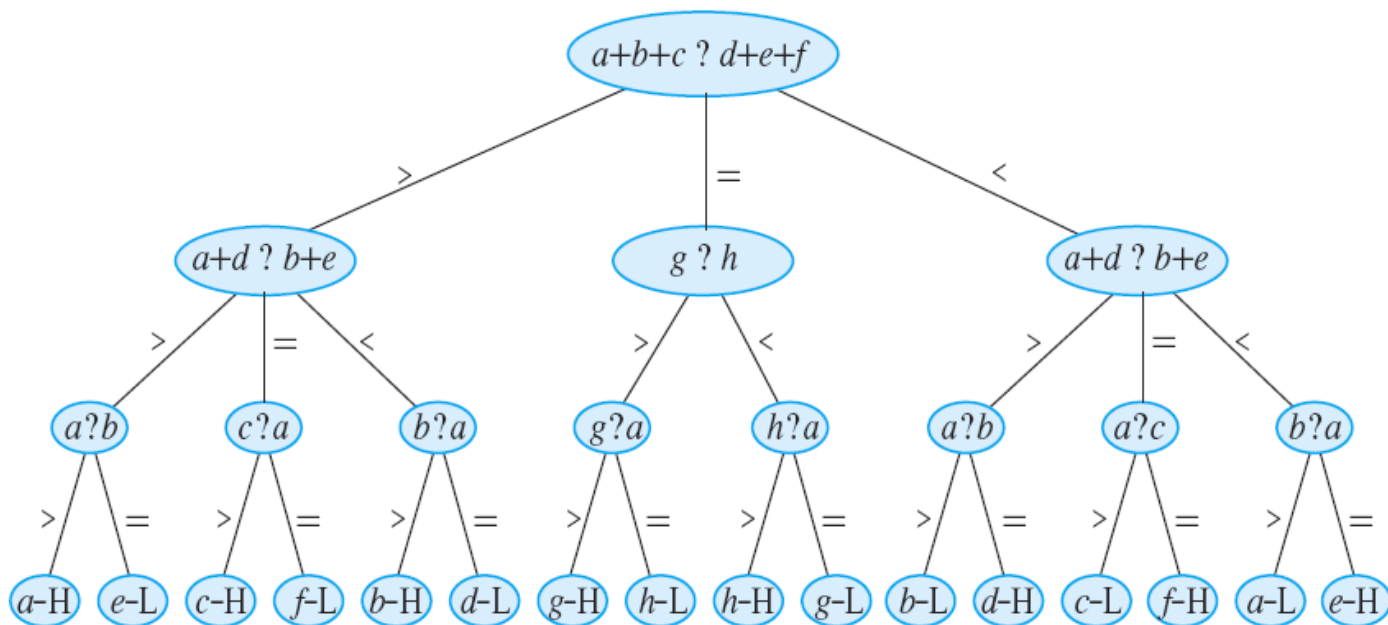


- $a+b+c < d+e+f$ 인 경우 불량품이 6개 가운데 있으며 이때의 g 와 h 는 정상품이라는 것을 알 수 있음
- 다음 단계에서 d 와 b 를 양 천칭 저울에서 바꾸어 계량한 결과 $a+d < b+e$ 로 부등호에 변화가 없다면 다음의 2가지 가능성을 말해줌
- a 가 불량품이거나 e 가 불량품인 경우임
- 이런 경우는 정상품과 비교하여 H 나 L 을 판단할 수 있음
- 만약 $a+d = b+e$ 인 경우에는 c 나 f 가 불량품임
- 또한 b 나 d 가 불량품인 경우에는 다른 정상품과 비교하여 b 또는 d 가 H 나 L 이 되게 결정됨

8.7 트리의 활용



8개의 동전 문제 결정 트리



〈그림 8.19〉 8개의 동전 문제 결정 트리

8.7 트리의 활용



- 최종 판단을 위한 프로시저인 **comp 프로그램**

```
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

- 프로시저 **eight-coins**는 앞의 결정 트리의 판단과 같은 기능을 수행함

8.7 트리의 활용



```
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 */
```

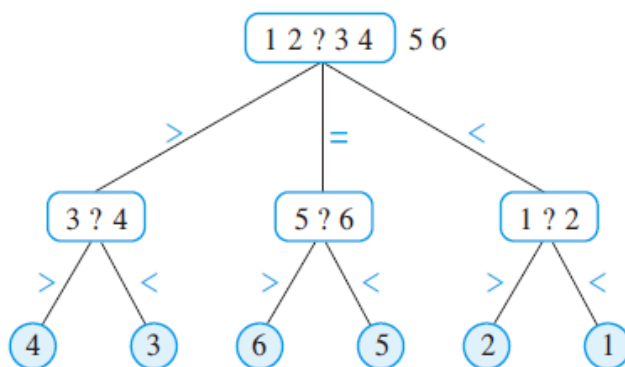
8.7 트리의 활용



예제 8-6

크기와 모양이 똑같은 6개의 공이 있는데 이 중 5개의 무게는 같고 한 개는 약간 더 가볍다고 한다. 천칭 저울을 이용하여 최소 몇 번의 측정으로 1개의 가벼운 공을 찾아낼 수 있을까? 게임 트리를 만들어 찾아내시오.

풀이 두 번 만에 가능하다. 먼저 6개의 공 가운데 4개를 골라 3개씩 양쪽 저울에 올린다. 만약 평형을 이루면 나머지 2개를 측정하여 가벼운 것을 고르면 된다. 만약 가벼운 공이 좌우 2개씩으로 나뉜 그룹 내에 있다면 가벼운 그룹에 속한 2개 공끼리 천칭 저울에 올려 가벼운 것을 고르면 된다. 따라서 두 번 만에 가능하다. 이것을 결정 트리로 나타내면 <그림 8.19>와 같다.



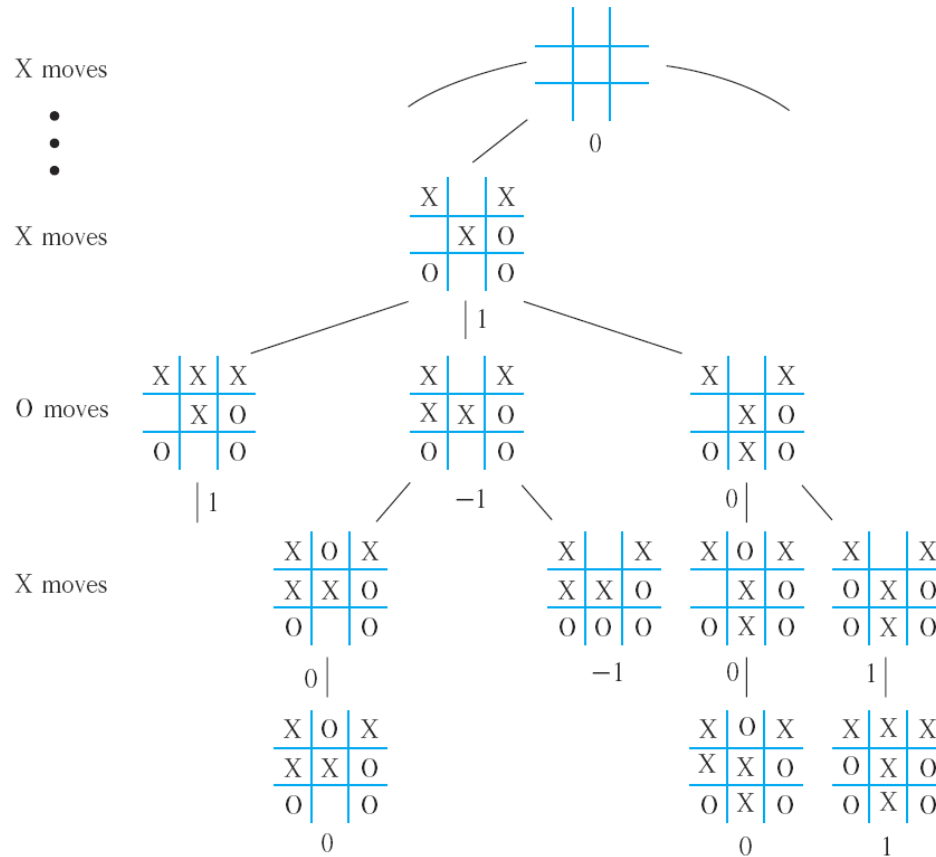
<그림 8.19> 가벼운 공을 찾아내는 결정 트리



(3) 게임 트리(game tree)

- 트리는 체스, 틱택토(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

8.7 트리의 활용



〈그림 8.20〉 tic-tac-toe 게임 트리



(4) 게임 트리의 인공지능 문제에서의 응용

8-puzzle 게임 – 미국 초등학생들의 두뇌개발 게임 중 하나

게임 트리를 이용하는 또 다른 응용으로는 인공지능 문제를 해결하는 8-puzzle 게임이 있다. 이 게임은 <그림 8.21>과 같이 3×3 크기의 박스 안에서 인접한 숫자를 빈 곳으로 계속적으로 움직여서 목표 상태(goal)로 만드는 서양식 게임이다.

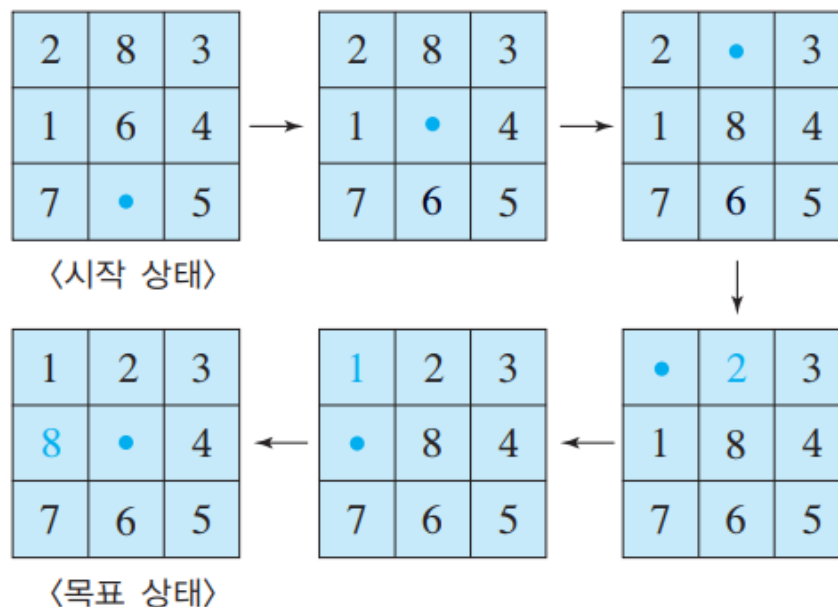
Start			Goal		
2	8	3	1	2	3
1	6	4	8	•	4
7	•	5	7	6	5

<그림 8.21> 시작 상태와 목표 상태

8.7 트리의 활용



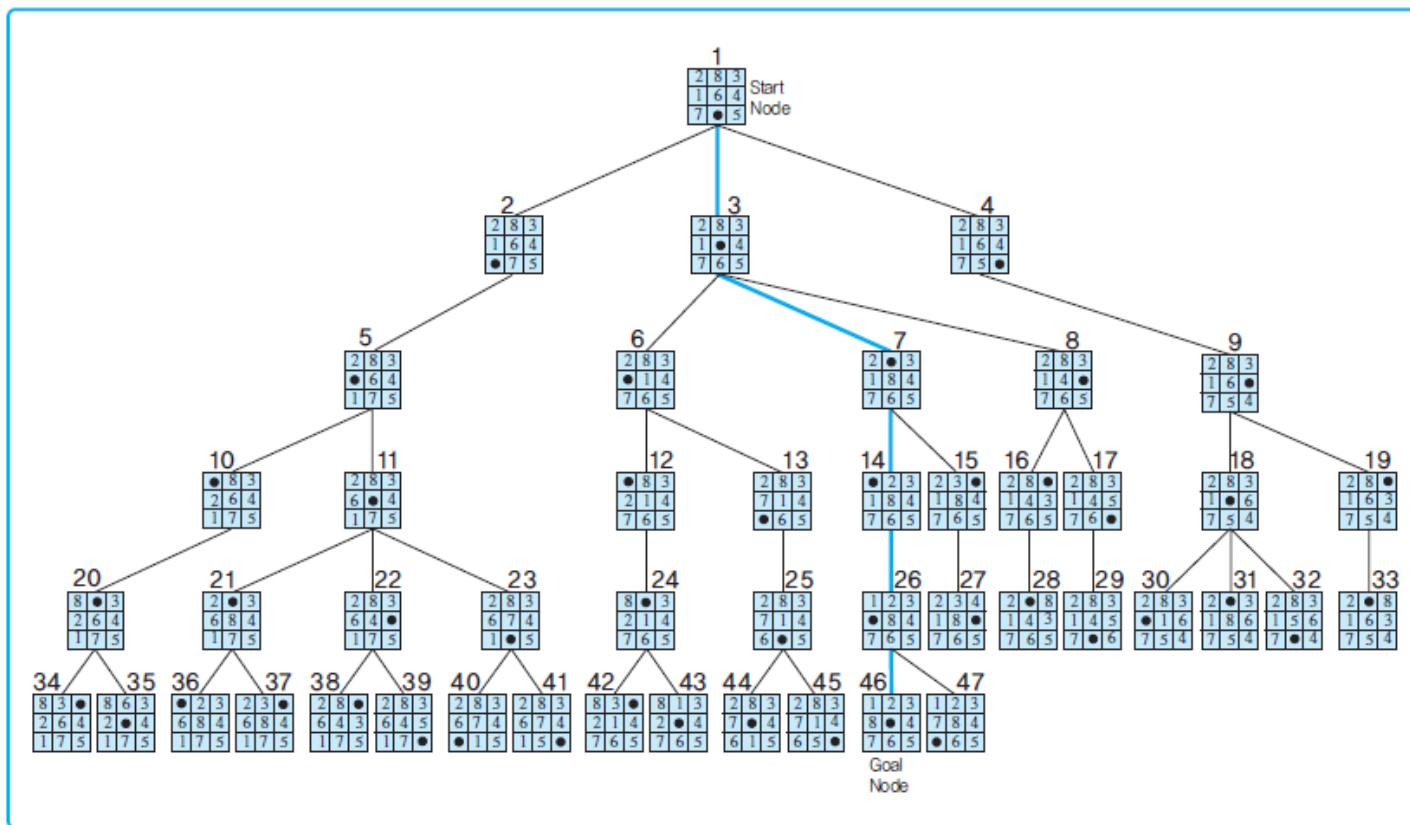
이것을 시작 상태에서 목표 상태까지의 5개의 중간 상태 이동을 나타내면 <그림 8.22>와 같다.



8.7 트리의 활용



8-puzzle 트리의 결과



〈그림 8.22〉 8-puzzle 트리

8.8 트리의 응용과 4차 산업혁명과의 관계



(1) 트리의 응용 분야

① 복잡한 문제 분석과 다양한 학문 분야에서의 응용

트리 구조의 가장 큰 역할은 트리를 통해 복잡한 경우의 수를 명확히 분류할 수 있다는 점이다. 따라서 트리는 전기회로망 설계와 응용, DNA 구성과 관련된 유전학, 화학 결합에서의 분자구조식 설계, 복잡한 사회 현상 규명을 위한 사회 과학 등에 폭넓게 응용되고 있다.

② 컴퓨터 관련 문제 해결에의 응용

트리는 컴퓨터 관련 논제 중 데이터베이스의 구성, 최적화 문제의 해결, 통신 네트워크, 문법의 파싱, 언어들 간의 번역, 자료의 탐색, 결정 트리, 게임 등에 매우 유용하게 응용되고 있다.

8.8 트리의 응용과 4차 산업혁명과의 관계



(2) 트리와 4차 산업혁명과의 관계 : 인공지능

인공지능은 컴퓨터를 사용하여 <그림 8.24>와 같이 인간의 지능을 모델링하는 기술로서 4차 산업혁명의 핵심 분야 중 하나이다. 즉 인간의 지능으로 할 수 있는 사고, 학습, 추론, 음성인식, 자연어 이해 등을 컴퓨터가 할 수 있도록 하는 방법을 연구하는 분야이다.



<그림 8.24> 인간의 지능을 모델링하는 인공지능

4차 산업혁명의 시작과 더불어 인공지능에 대한 관심이 더욱 커지고 있다. 최근 인공지능은 로봇 저널리스트, 요리사, 인공지능 로봇 등에도 활용되기 시작했다. 트리는 인공지능에 있어서의 효율적인 탐색 기법을 제공하는데, 게임에 있어서의 인공지능적인 탐색이나 최단 경로 찾기 등에 있어서 매우 중요한 역할을 담당한다.

요약 및 생활 속의 응용



- 트리는 특별히 지정된 노드인 루트가 있고, 나머지 노드들은 자신이 트리인 서로 겹치지 않는 T_1, T_2, \dots, 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.
- 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.

[illegible]

- 64

08

CHAPTER

트리

Tree Theory & Its Applications

CHAPTER 8 FINISH

