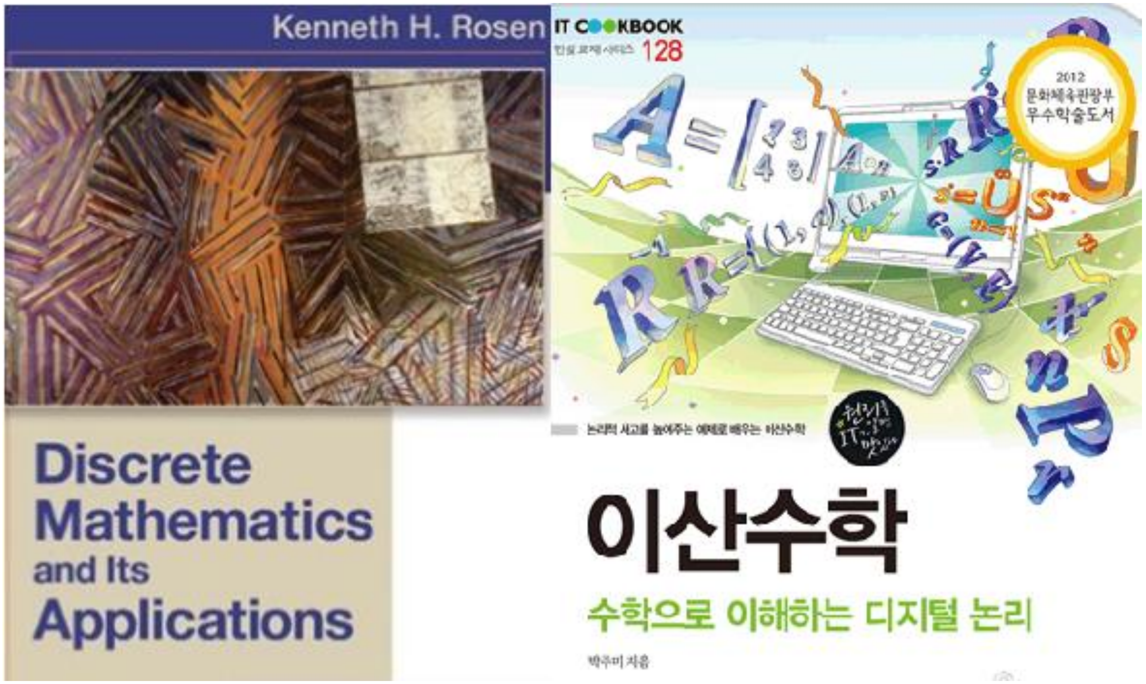
이산수학
Discrete Mathematics



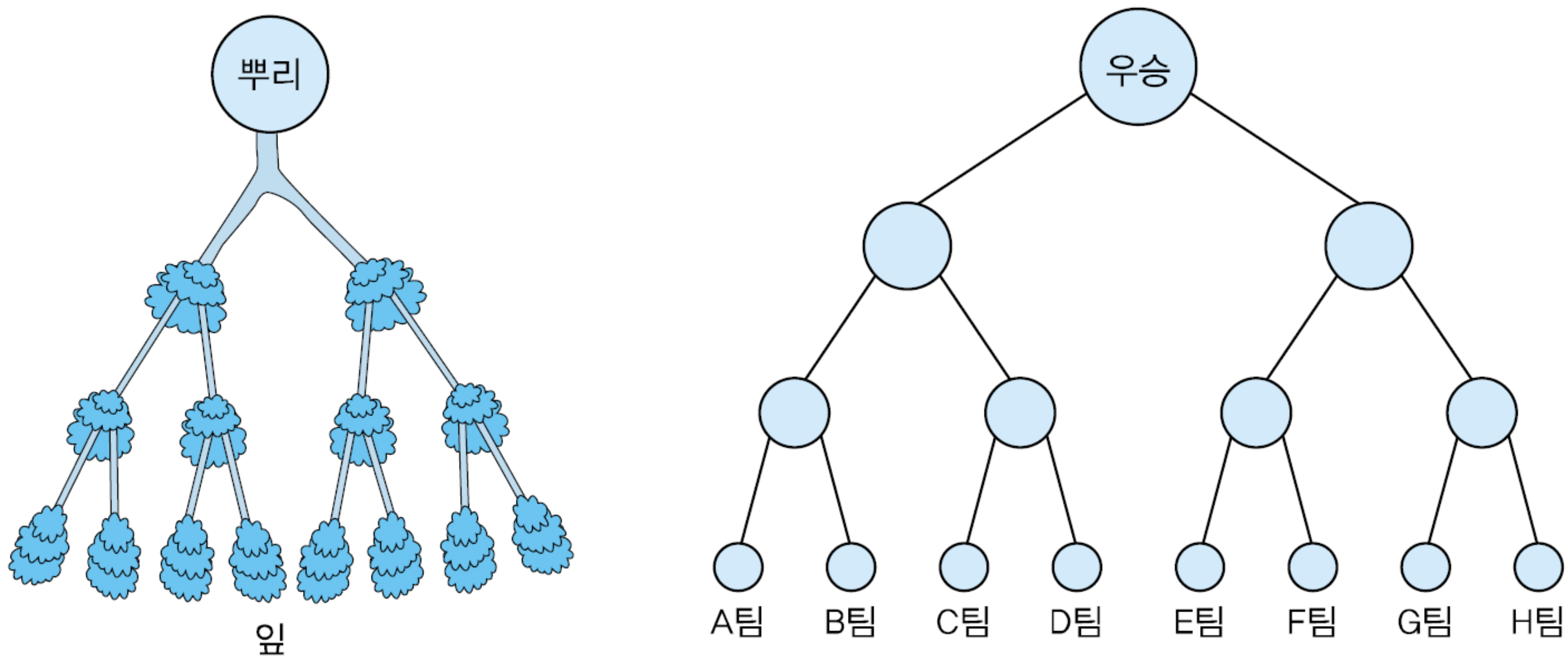
Chapter 09:
트리

Soongsil University : Kim Chang Wook

트리

트리(Tree)

- 트리 T는 비순환, 연결 그래프로 다음과 같은 특징이 있음
  - 특별한 노드인 루트(root)는 반드시 하나 있음
  - 트리 T를 구성하는 정점 v, w간에 v 에서 w 로 가는 단순 경로가 있음



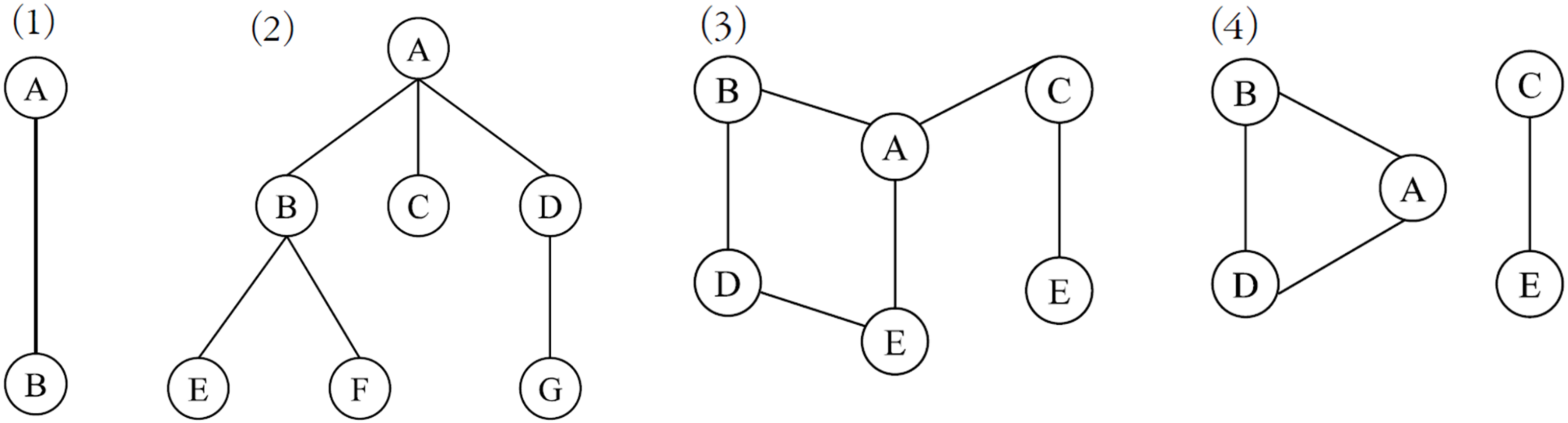
[그림 9-1] 나무 뿌리 모양의 토너먼트 표

트리에는 루트라고 불리는 노드가 반드시 있어야 한다.
루트를 중심으로 하나 이상의 정점들이 순환 경로 없이 연결되어 있는 형태

트리

예제 9-1

다음 중 트리인 것을 골라라.



풀이

(1) A와 B 중 루트가 무엇인지는 확실하지 않으나 둘 중 하나는 루트의 역할을 하며, A와 B 간에 단순 경로가 존재하므로 트리다.

(2) A가 루트며 그래프를 구성하는 꼭짓점 A부터 G 사이에는 단순 경로만 존재하므로 트리다.

(3) A-B-D-E 간에 순환 경로가 존재하므로 트리가 아니다.

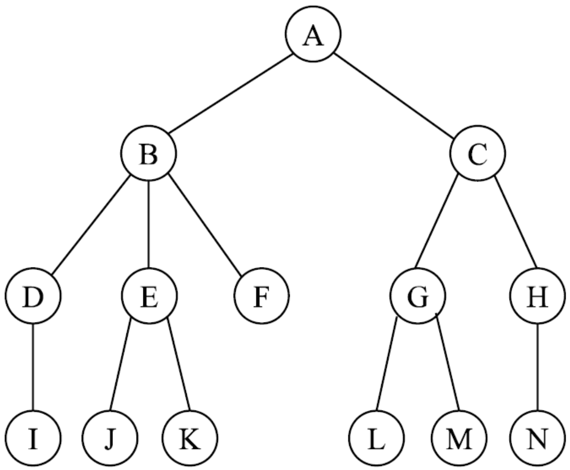
(4) A-B-D 간에 순환 경로가 존재하고, A-B-D와 C-E가 연결되어 있지 않으므로 비연결 그래프다. 그러므로 트리가 아니다.

❖서브 트리 (Sub Tree)

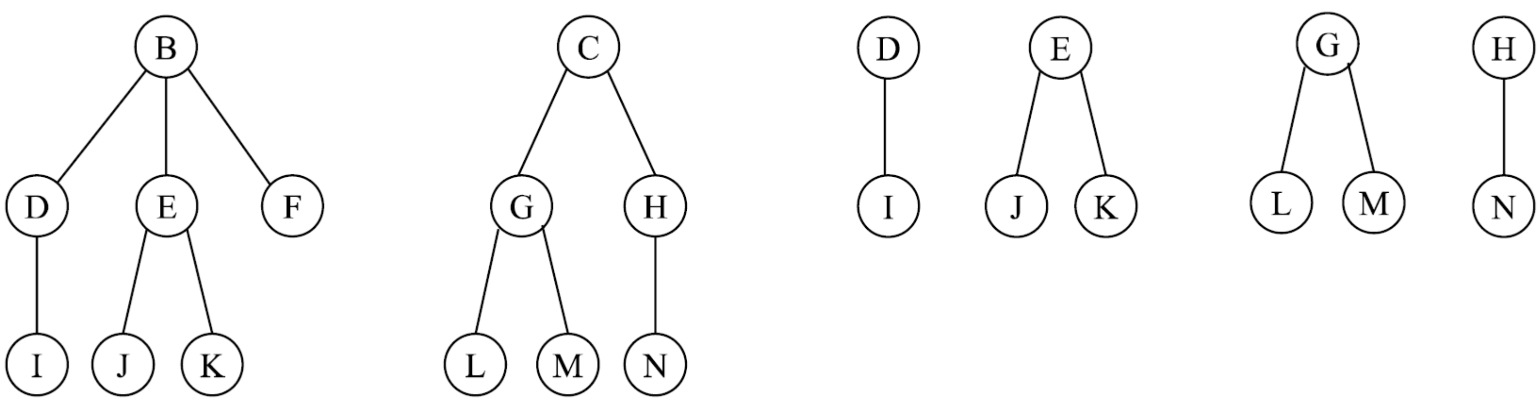
- T를 구성하는 정점 v를 루트로 하는 트리

예제 9-2

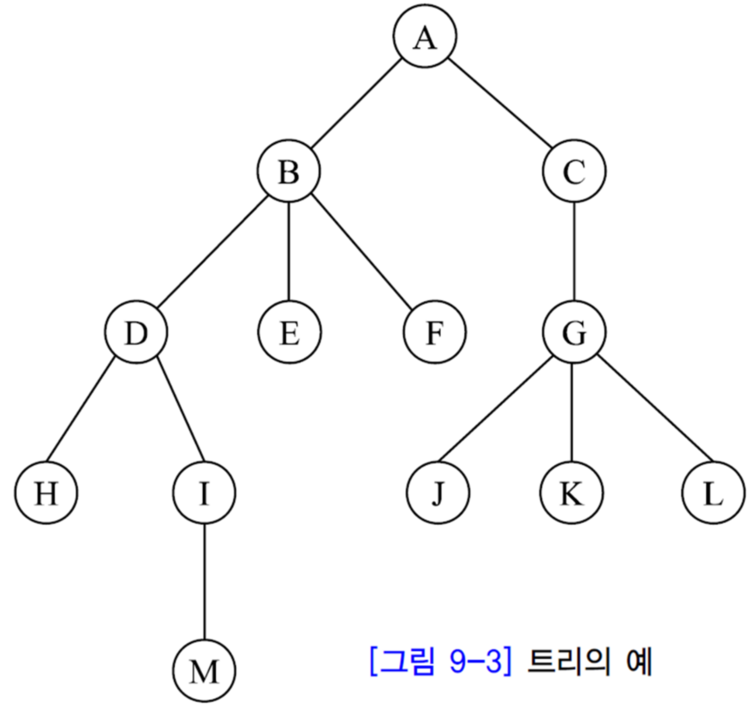
다음 트리의 서브 트리를 모두 구하라.



총 6가지의 서브 트리



트리 관련 용어



[그림 9-3] 트리의 예

❖노드(Node)

- 그래프 T를 구성하는 정점
  - [그림 9-3]의 A, B, C, D, E, F, G, H, I, J, K, L, M

❖루트(Root)

- 그래프 T의 시작 노드, 그래프 T의 가장 높은 곳에 위치
  - [그림 9-3]의 A

❖부모 노드(Parent Node)

- 어떤 노드의 한 단계 상위 노드
  - [그림 9-3]의 A : B와 C의 부모 노드
    B : D, E, F의 부모 노드
    C : G의 부모 노드
    D : H, I의 부모 노드
    G : J, K, L의 부모 노드
    I : M의 부모 노드

❖자식 노드(Child Node)

- 어떤 노드의 한 단계 하위 노드
  - [그림 9-3]의 B, C : A의 자식 노드
    D, E, F : B의 자식 노드
    G : C의 자식 노드
    H, I : D의 자식 노드
    J, K, L : G의 자식 노드
    M : I의 자식 노드

트리

❖형제 노드(Sibling Node)

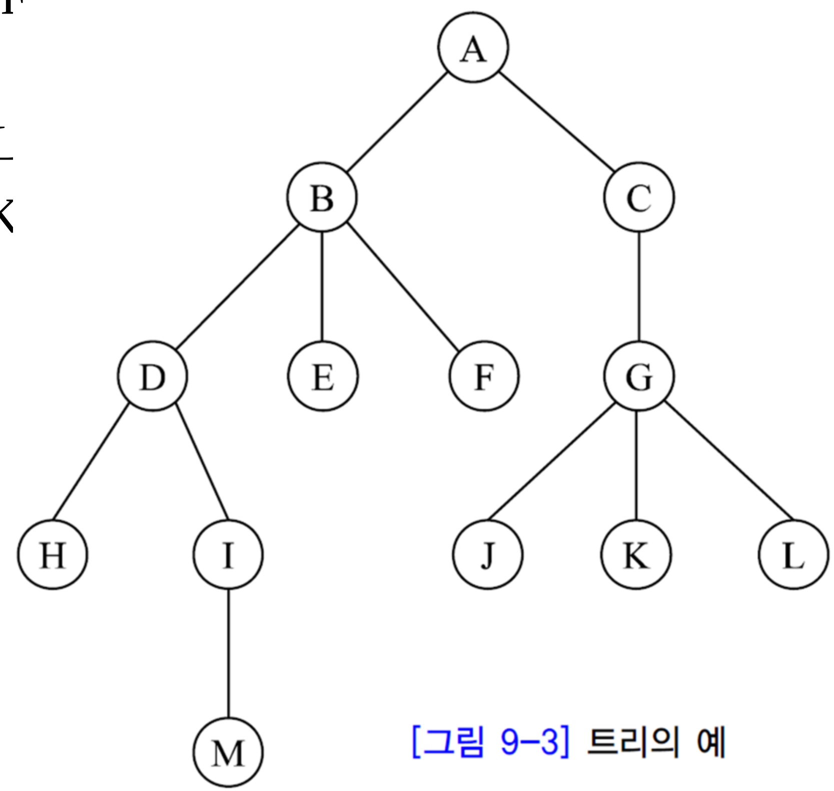
- 같은 단계에 있으면서 부모가 같은 노드들
  - [그림 9-3]의 B의 형제 노드 : C
    C의 형제 노드 : B
    D의 형제 노드 : E, F
    E의 형제 노드 : D, F
    F의 형제 노드 : D, E
    H의 형제 노드 : I
    I의 형제 노드 : H
    J의 형제 노드 : K, L
    K의 형제 노드 : J, L
    L의 형제 노드 : J, K

❖잎 노드(Leaf Node)

- 자식 노드가 없는 노드
  - [그림 9-3]의 H, M, E, F, J, K, L

❖중간 노드(Internal Node)

- 루트 노드나 잎 노드가 아닌 노드
  - [그림 9-3]의 B, C, D, G, I



[그림 9-3] 트리의 예

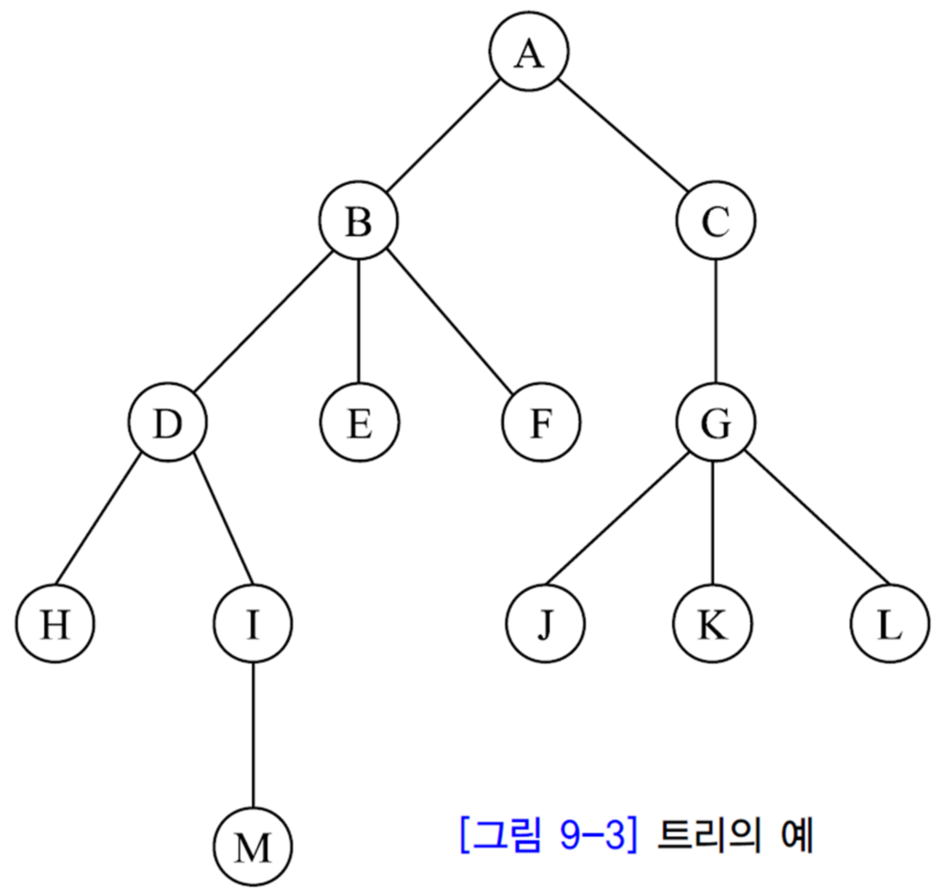
트리

❖조상 노드(Ancestor Node)

- 루트 노드에서 어떤 노드에 이르는 경로에 포함된 모든 노드
  - [그림 9-3]의
    M의 조상 노드 : I, D, B, A
    G의 조상 노드 : C, A
    F의 조상 노드 : B, A
    L의 조상 노드 : G, C, A

❖자손 노드(Descendant Node)

- 어떤 노드에서 잎 노드에 이르는 경로에 포함된 모든 노드
  - [그림 9-3]의
    A의 자식 노드
    : B, C, D, E, F, G, H, I, J, K, L, M
    B의 자식 노드 : D, E, F, H, I, M
    C의 자식 노드 : G, J, K, L
    D의 자식 노드 : H, I, M
    I의 자식 노드 : M



[그림 9-3] 트리의 예

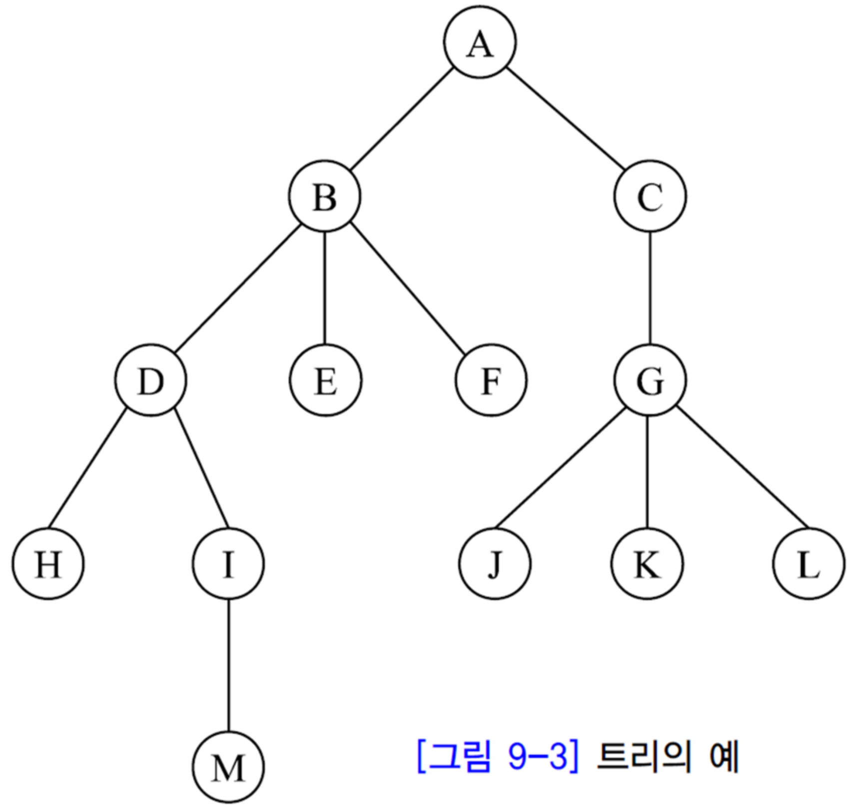
트리

❖차수(Degree)

– 어떤 노드에 포함된 자식 노드의 수 [그림] A의 차수: 2　B의 차수: 3 … M의 차수: 0

❖레벨(Level)

- 루트 노드를 0으로 시작하여, 자식 노드로 내려갈 때마다 하나씩 증가하는 단계
  - [그림 9-3]의 레벨 0에 속하는 노드 : A
    레벨 1에 속하는 노드 : B, C
    레벨 2에 속하는 노드 : D, E, F, G
    레벨 3에 속하는 노드 : H, I, J, K, L
    레벨 4에 속하는 노드 : M



[그림 9-3] 트리의 예

❖트리의 높이(Height) / 트리의 깊이(Depth)

- 트리가 가지는 최대 레벨
  - [그림 9-3]의 트리의 높이(깊이)는 4

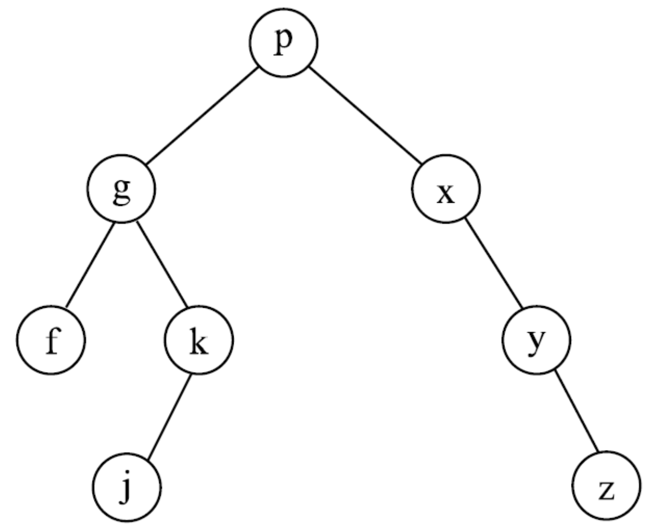
❖숲(Forest)

- 루트 노드를 제거하여 얻는 서브 트리의 집합
  - [그림 9-3]의 B, C를 루트로 하는 서브 트리

예제 9-3

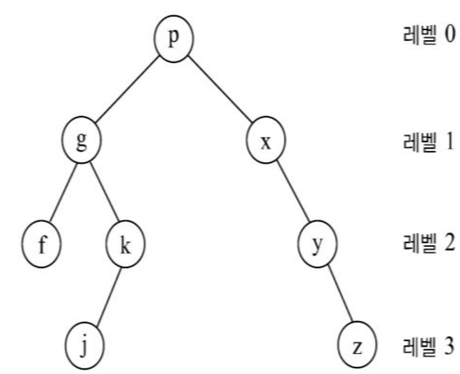
주어진 트리를 보고 다음을 구하라.

(1) 트리를 구성하는 모든 노드
(2) 트리의 루트 노드
(3) 트리의 높이
(4) 노드의 레벨
(5) 트리의 잎 노드
(6) 트리의 중간 노드
(7) 트리의 숲



풀이

(1) p, g, f, k, j, x, y, z가 이 트리를 구성하고 있다.
(2) 트리의 루트는 p다.
(3) 트리의 높이는 최대 레벨을 나타내므로 3이다.
(4) 레벨 0은 p, 레벨 1은 g, x, 레벨 2는 f, k, y, 레벨 3은 j, z다.
(5) 서브 트리를 갖지 않는 노드가 잎 노드이므로 f, j, z다.
(6) 트리에서 루트 노드와 잎 노드를 제외한 노드는 g, x, k, y다.
(7) 이 트리는 g와 x를 루트로 갖는 서브 트리로 구성



예제 9-3

주어진 트리를 보고 다음을 구하라.

(8) 트리에서 차수가 가장 높은 노드

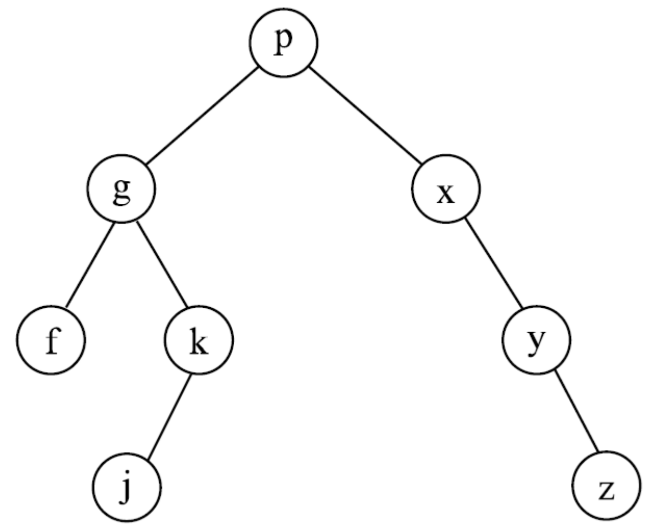
(9) 노드 x의 자손 노드

(10) 노드 j의 조상 노드

(11) 노드 f의 형제 노드

(12) 노드 g의 자식 노드

(13) 노드 x의 부모 노드



풀이

(8) 각 노드가 갖는 자식 노드의 수가 차수며, 각 노드의 차수는 다음과 같다.

노드 p : 2, 노드 g : 2, 노드 x : 1, 노드 f : 0, 노드 k : 1, 노드 y : 1, 노드 j : 0 노드 z : 0

그러므로 차수가 가장 높은 노드는 p와 g다.

(9) x로에서 잎 노드까지의 노드가 자손 노드이므로 y, z다.

(10) j로에서 루트 노드까지의 노드가 조상 노드이므로 k, g, p다.

(11) f와 같은 부모 밑에서 같은 단계에 있는 노드는 k다.

(12) g의 한 단계 아래에 있는 자식 노드는 f, k다.

(13) x의 한 단계 위에 있는 부모 노드는 p다.

트리의 성질

[정리 9-1] 노드와 변에 대한 정리

트리에서 노드의 개수를 v, 변의 개수를 e라고 하면, $e = v - 1$.

증명) 수학적 귀납법으로 증명하면, $v = 1$일 때, $e = 0$이고, $e = v - 1 = 1 - 1$이 성립

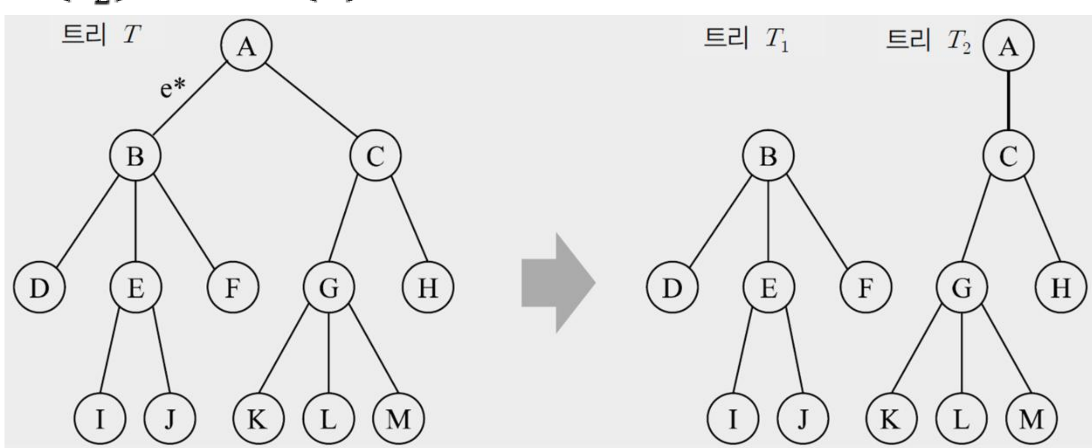
$v < k$ 일 때, $e = k - 1$ 성립한다고 가정, $v = k$ 일 때 위의 정리가 성립하는지 증명.

k 개 노드를 갖는 트리 T에서 임의 변 e^*를 제거하면 두 개의 트리 T_1 과 T_2 로 분리가능.
각 트리의 노드 수를 $v(T), v(T_1), v(T_2)$, 변의 수를 $e(T), e(T_1), e(T_2)$ 라고 할때,
$v(T_1)$과 $v(T_2)$는 각각 k보다 작으므로 다음과 같이 정의할 수 있다.

$e(T_1) = v(T_1) - 1$ $\qquad\qquad$ $e(T_2) = v(T_2) - 1$

이때 $v(T) = v(T_1) + v(T_2)$고 T_1과 T_2는 T 에서 변 하나를 제거하여 구한
트리이므로 변의 수는 다음과 같다.

$$\begin{aligned} e(T) &= e(T_1) + e(T_2) + 1 = \{v(T_1) - 1\} + \{v(T_2) - 1\} + 1 \\ &= v(T_1) + v(T_2) - 1 = v(T) - 1 \end{aligned}$$



트리의 성질

예제 9-4

다음 질문에 답하라.

(1) 노드의 수가 17개인 트리에 존재하는 변의 수

(2) 변의 수가 22개인 트리에 존재하는 노드의 수

풀이

(1) $e = v - 1$ 이므로 $e = 17 - 1 = 16$ ∴ 변 16개로 구성된 트리

(2) $e = v - 1$ 이므로 $22 = v - 1$, $v = 23$ ∴ 노드 23개로 구성된 트리

[정리 9-2] 트리에 대한 정리

n개의 정점을 갖는 연결 그래프 T에 대해 다음은 동치다.

(1) T 는 트리다.

(2) T 의 변의 수는 $n - 1$ 개다.

(3) T 에서 변 하나를 제거하면 연결 그래프가 아니다.

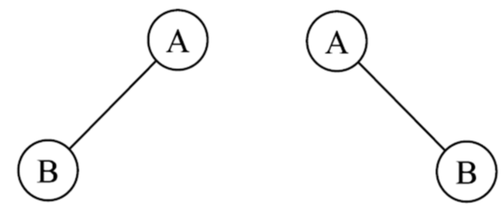
(4) T의 서로 다른 정점 w, v 에 대해 w에서 v로 가는 유일 경로 존재.

이진트리

❖ n항 트리(n-ary tree) : 트리의 최대 차수가 n 인 트리

❖ 이진 트리(Binary Tree)

- 트리 T를 구성하는 부모 노드가 갖는 자식 노드의 수가 최대 2개인 트리
- 자식 노드 수를 제한하면, 트리로 표현하는 연산을 단순하고 명확하게 표현 가능.
- 최대 차수를 2로 제한하므로 왼쪽 서브 트리와 오른쪽 서브 트리를 구분



(a) 왼쪽 서브 트리 (b) 오른쪽 서브 트리

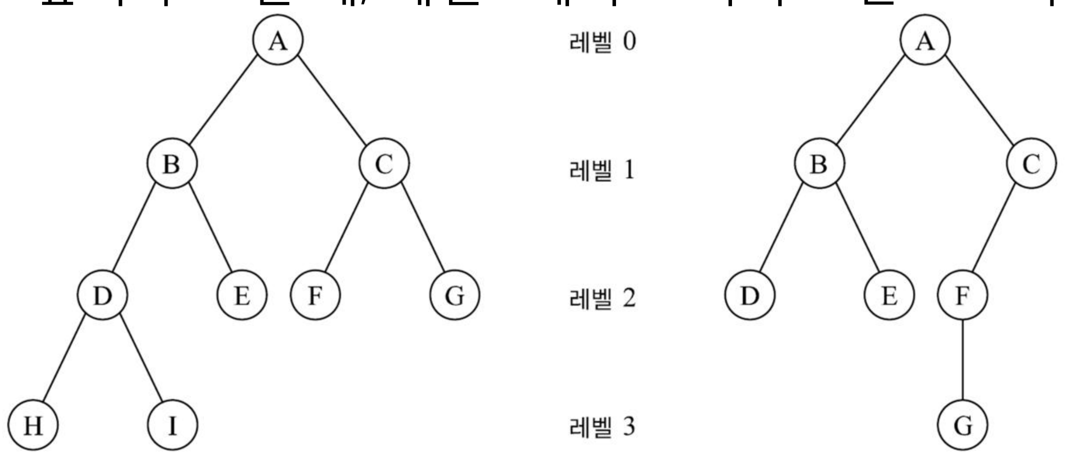
[그림 9-4] 이진 서브 트리

❖ 완전 이진 트리(Complete Binary Tree)

- 높이가 h일 때 레벨 1부터 h-1까지 모든 노드가 두 개씩 채워져 있고 (h-2까지 자식 노드가 두 개다), 레벨 h는 왼쪽부터 노드가 채워져 있는 트리

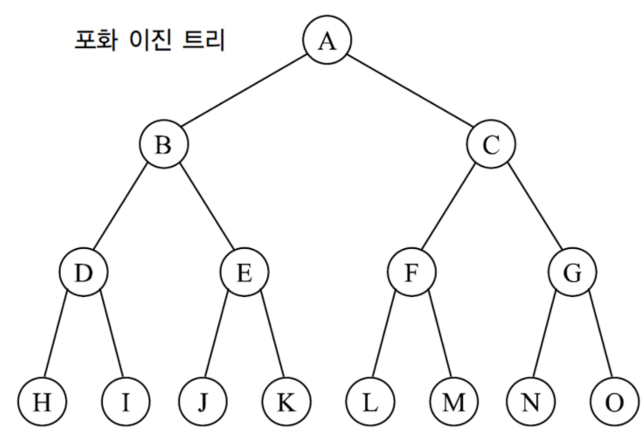
❖ 포화 이진 트리(Full Binary Tree)

- 높이가 h 일 때, 레벨 1에서 h 까지 모든 노드가 두 개씩 채워져 있는 트리



(a) 완전 이진 트리

(b) 완전 이진 트리가 아닌 트리



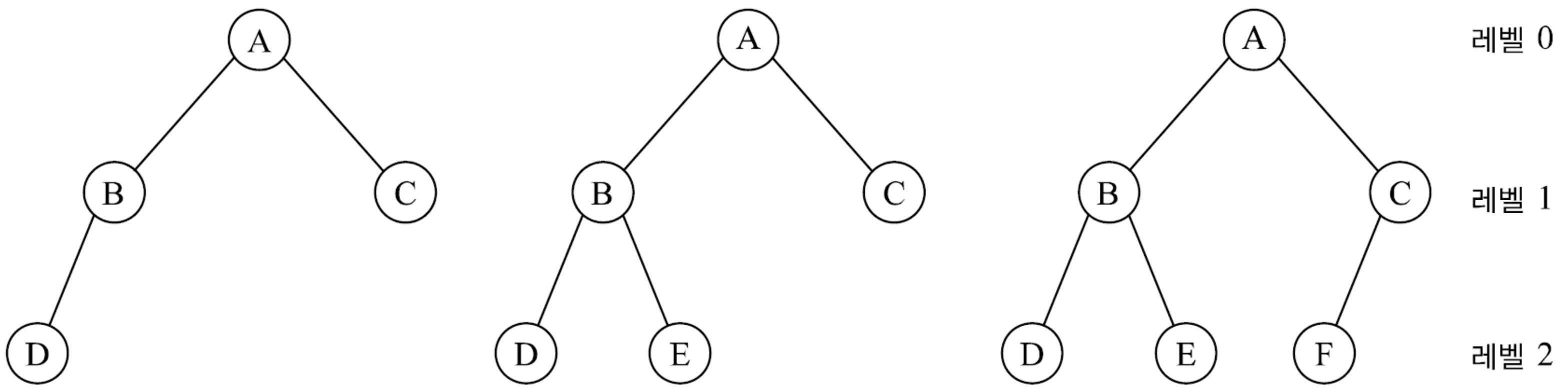
이진 트리

예제 9-5

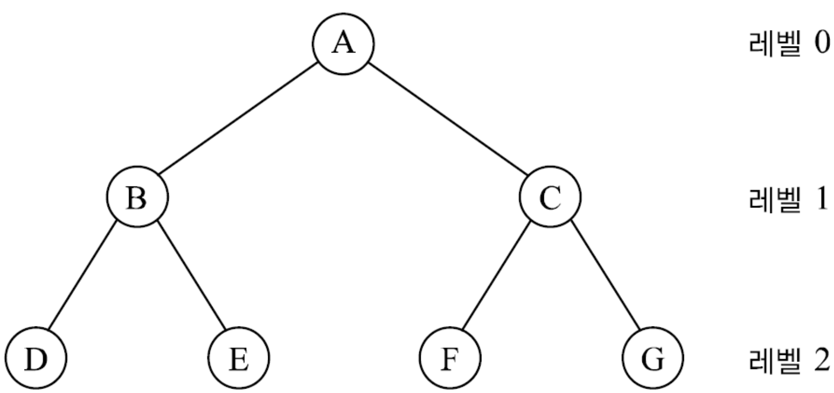
높이가 2인 완전 이진 트리와 포화 이진 트리를 그려라.

풀이

- 완전 이진 트리



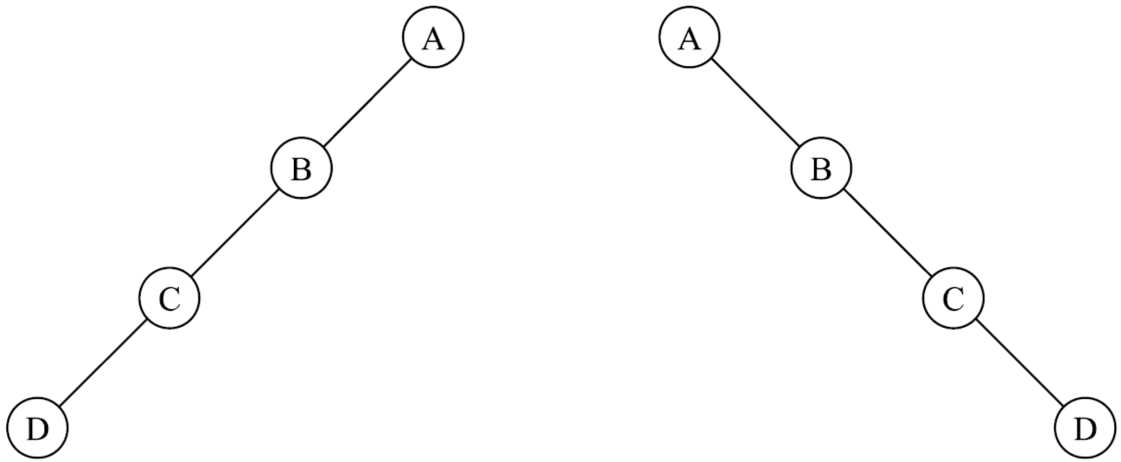
- 포화 이진 트리



이진트리

❖편향 이진 트리(Skewed Binary Tree)

- 왼쪽이나 오른쪽 서브 트리만 가지는 트리

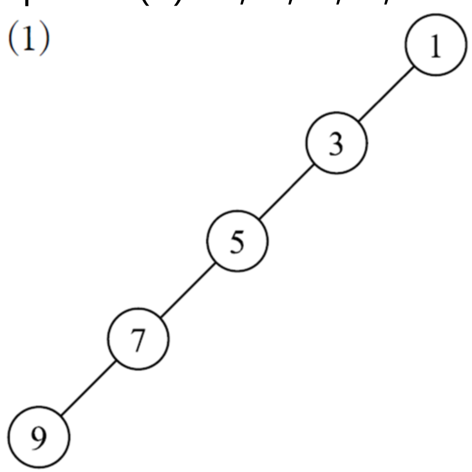


예제 9-6

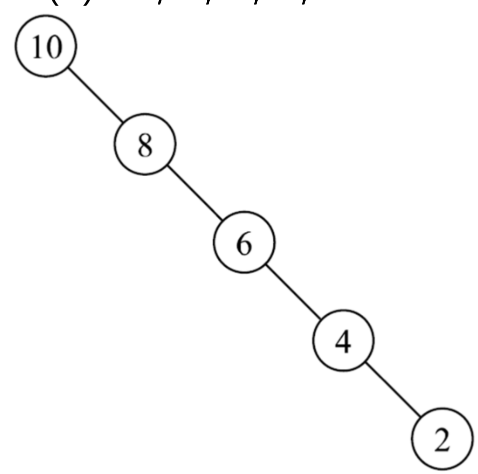
이전 값보다 큰 값은 왼쪽 노드에, 작은 값은 오른쪽 노드에 구성하려고 한다. 다음과 같은 입력을 트리로 구성하라. (1) 1, 3, 5, 7, 9 (2) 10, 8, 6, 4, 2

풀이

(1)



(2)



이진 트리

[정리 9-3] 이진 트리의 최대 노드 수

(1) 레벨 k 에서 가질 수 있는 최대 노드 수 : 2^k 개

증명)

(1) 수학적 귀납법을 이용해 증명한다. $k = 0$일 때, $2^0 = 1$로 레벨 0에는 노드 한 개 있으므로 식은 성립한다. $k = n$ 일 때, 최대 2^n개의 노드가 있다고 가정하고, $k = n + 1$일 때, 2^{n+1} 개의 노드가 있는지 확인한다.
이진 트리는 한 노드가 가지는 최대 자식 노드의 수가 2개다.
그러므로 레벨이 n 이면 최대 노드 수는 2^n 개가 되고,
레벨이 $n + 1$ 이면 $2 \cdot 2^n$이므로 최대 노드 수는 2^{n+1} 이 된다.
$\therefore$ 레벨 k 에서 가질 수 있는 최대 노드 수는 2^k이다.

이진 트리

[정리 9-3] 이진 트리의 최대 노드 수

(2) 높이가 m 인 트리가 가질 수 있는 최대 노드 수 : $2^{m+1}-1$ 개

증명)

(1) 수학적 귀납법을 이용해 증명한다. $m=0$일 때,$2^{0+1}-1=1$로 레벨 0에서 노드 한 개가 있으므로 식은 성립한다. $m=n$ 일 때, 최대 $2^{n+1}-1$개의 노드가 있다고 가정하고, $m=n+1$일 때, $2^{(n+1)+1}-1$ 개의 노드가 있는지 확인한다. 레벨 k 에서 가질 수 있는 최대 노드 수는 2^k 라는 정리를 이용하면 $m=n+1$일 때의 최대노드 수는 $2^{n+1}-1$개($m=n$ 일 때의 최대의 노드 수)에 2^{n+1} 개를 더한 것과 같다. 그러므로 $2^{n+1}-1+2^{n+1}=2\cdot 2^{n+1}-1=2^{n+2}-1=2^{(n+1)+1}-1$ 이 된다.

$\therefore$ 높이가 m 인 트리가 가질 수 있는 최대 노드 수는 $2^{m+1}-1$ 개 이다.

이진 트리

[정리 9-3] 이진 트리의 최대 노드 수

(3) 높이가 m 인 이진 트리가 가질 수 있는 최소 노드 수 : $m+1$개

증명)

(3) 수학적 귀납법을 이용해 증명한다. $m=0$ 일 때, 루트 노드 하나가 존재하는 경우므로 $0+1=1$이 성립한다. $m=n$일 때, 최소 $n+1$ 개 노드가 있다고 가정하고, $m=n+1$일 때, 최소 $(n+1)+1=n+2$ 노드가 있는지 확인한다. 레벨 n 에서 하나 더 깊어져 레벨 $n+1$ 이 되었다는 것은 최소 하나의 노드가 레벨 $n+1$에 있다는 것을 의미. 그러므로 레벨 n 일 때, 최소 $n+1$개의 노드가 있다고 가정했으므로 레벨 $n+1$ 일 때의 최소 노드의 수는 $(n+1)+1=n+2$ 가 된다.

$\therefore$ 높이가 m 인 이진 트리가 가질 수 있는 최소 노드 수는 $m+1$ 이다.

예제 9-7

다음 질문에 답하라.

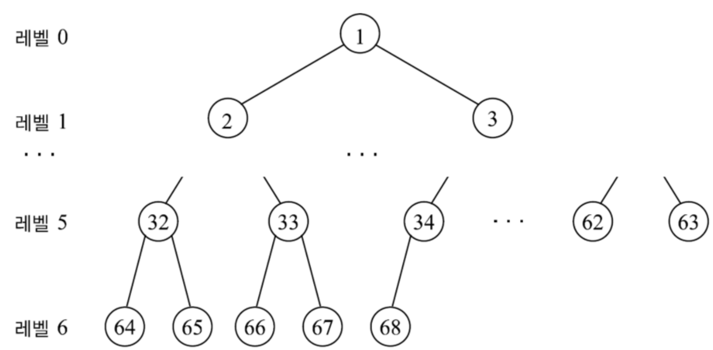
(1) 전체 노드 수가 68개인 완전 이진 트리의 높이를 구하라.

(2) 높이가 5인 완전 이진 트리의 레벨 3이 갖는 노드 수는 몇 개인가?

(3) 높이가 4인 포화 완전 이진 트리의 총 노드 수는 몇 개인가?

풀이

(1) 높이가 m일 때 이진 트리가 가질 수 있는 최대 노드 수는 $2^{m+1}-1$이다.
$2^{m+1}-1=63$ 일 때 높이가 5인 포화 완전 이진 트리가 된다.
$2^{m+1}=64$ 이므로 $m+1=6$. $\therefore m=5$
그런데 노드 수가 68이므로 높이가 5인 포화 완전 이진 트리보다 노드가 5개가 더 있으므로 높이가 6인 이진 트리다. 높이가 6인 이진 트리의 최대 노드 수는 $127(=2^{6+1}-1=2^7-1)$이다. 따라서 레벨 6의 노드 수는 5개고, 왼쪽부터 노드가 채워진 완전 이진 트리가 된다.



(2) 높이가 5인 완전 이진 트리는 레벨 4까지 모든 노드가 있고 레벨 5는 왼쪽부터 $2^5=32$보다 적은 개수의 노드들로 채워져 있는 이진 트리를 말한다.
그러므로 레벨 3에는 모든 노드가 있다. $\therefore$ 레벨 3의 총 노드 수는 $2^3=8$ 개다.

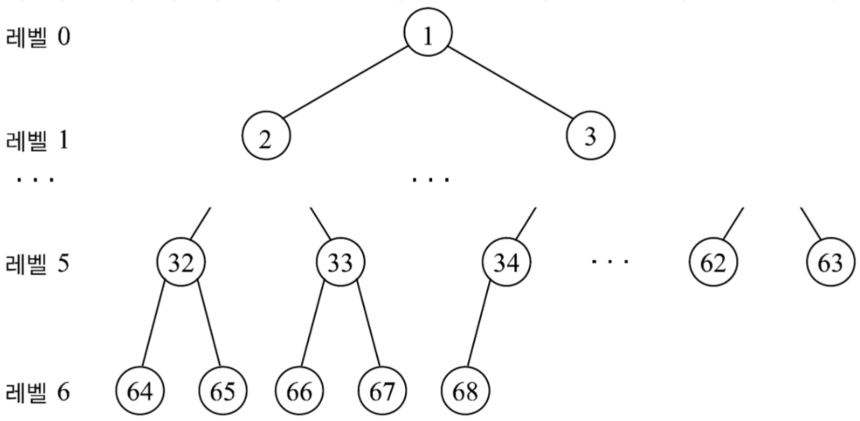
(3) 높이가 4인 포화 완전이진트리는 레벨 4까지 모든노드가 채워져 있는 이진트리
$\therefore$ 높이가 4인 포화 완전 이진 트리의 총 노드 수는 $2^{4+1}-1=2^5-1=32-1=31$ 개.

이진 트리

예제 9-7

풀이

(1) 높이가 m일 때 이진 트리가 가질 수 있는 최대 노드 수는 $2^{m+1}-1$이다.
$2^{m+1}-1 = 63$ 일 때 높이가 5인 포화 완전 이진 트리가 된다.
$2^{m+1} = 64$ 므로 $m+1 = 6$이 된다.
$\therefore m = 5$
그런데 노드 수가 68이므로 높이가 5인 포화 완전 이진 트리보다 노드가 5개가 더 있으므로 높이가 6인 이진 트리다. 높이가 6인 이진 트리의 최대 노드 수는 $127(= 2^{6+1}-1 = 2^7-1)$이다. 따라서 레벨 6의 노드 수는 5개고, 왼쪽부터 노드가 채워진 완전 이진



이진트리 탐방(순회)

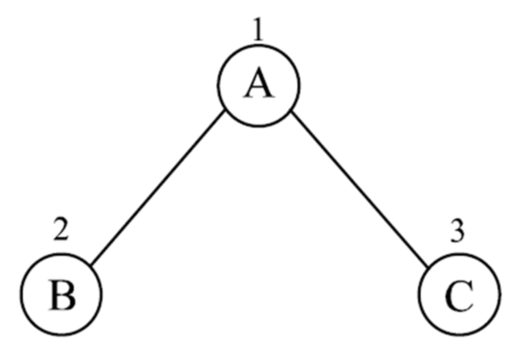
❖탐방(Traversal) : 모든 노드의 데이터를 처리할 수 있도록 한 번씩 방문하는 것.
트리 탐방 방법 : 전위 (Preorder)탐방, 중위(Inorder) 탐방, 후위(Postorder)탐방.

❖탐방의 규칙

- 항상 루트에서 시작한다. 즉 트리의 레벨 0에서 시작한다.
- 서브 트리에 대한 탐방의 순서는 항상 왼쪽에서 오른쪽으로 이루어진다.
- 데이터를 읽기 전에 왼쪽, 혹은 오른쪽 노드가 있는지 확인하는 작업을 한다.

❖전위탐방(Preorder Traversal)

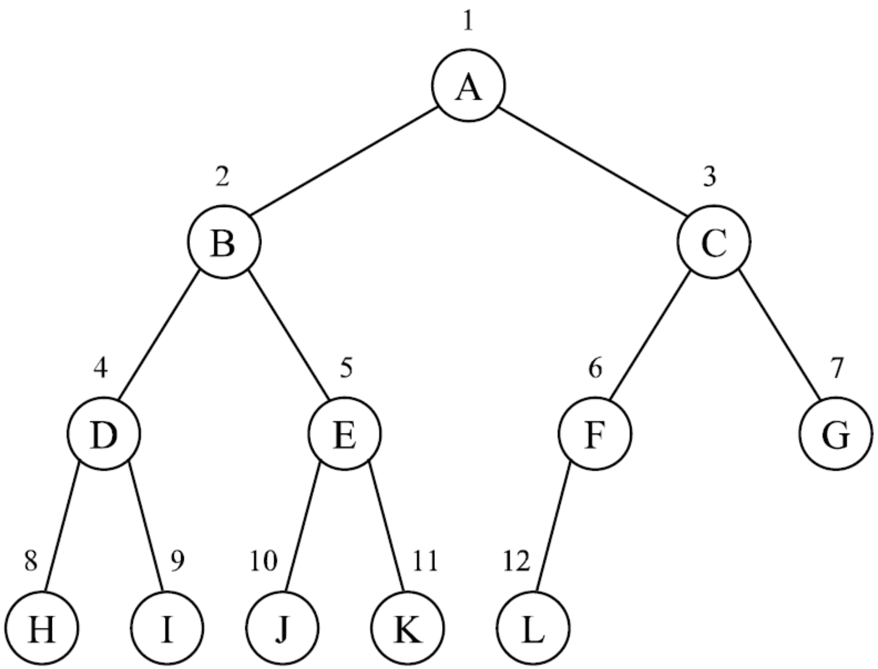
- 루트 노드 - 왼쪽 노드 - 오른쪽 노드 순으로 방문하는 탐방 방식
  - 노드 1에 방문 : 데이터 A를 읽고(P) 왼쪽 노드가 있는지 확인
  - 노드 2에 방문 : 데이터 B를 읽고(L) 왼쪽 노드가 있는지 확인
  - 왼쪽 노드가 없으므로 오른쪽 노드가 있는지 확인
  - 오른쪽 노드도 없으므로, 다시 노드 1에 방문 : 오른쪽노드가 있는지 확인
  - 노드 3에 방문 : 데이터 C를 읽고(R) 왼쪽노드가 있는지 확인
  - 왼쪽 노드가 없으므로 오른쪽 노드가 있는지 확인
  - 오른쪽 노드도 없으므로 다시 노드 1에 방문 : 탐방 종료

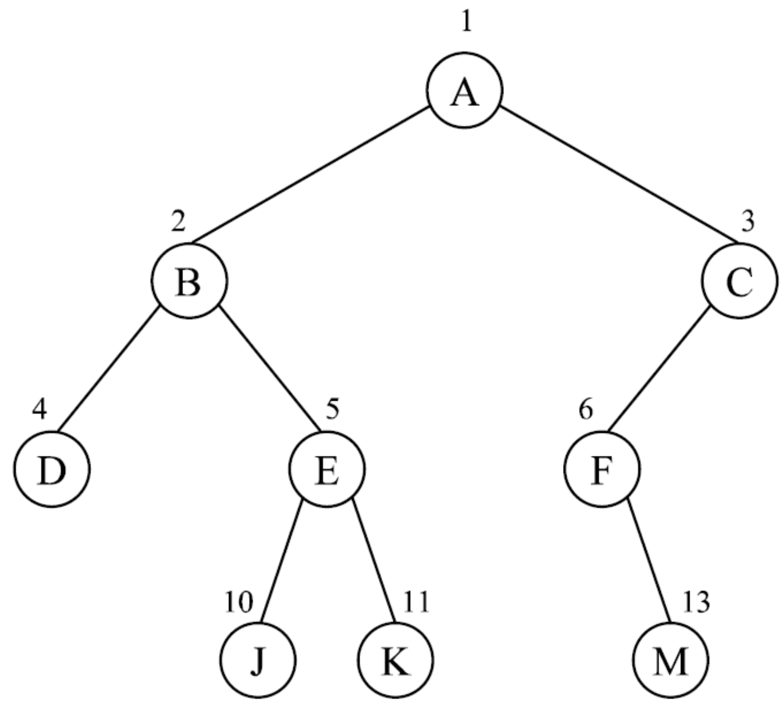


이진 트리 전위탐방 (Preorder Traversal)

예제 9-10

다음 이진 트리를 전위탐방 (Preorder Traversal) 으로 탐방하라.





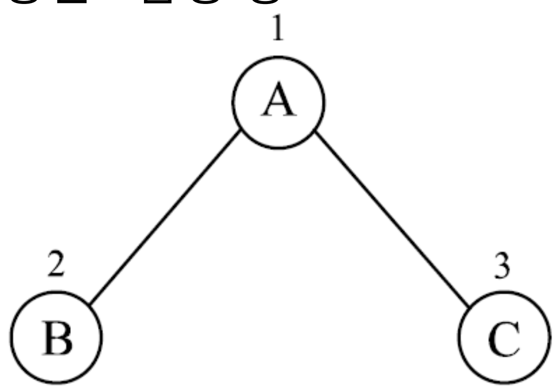
$\therefore A - B - D - H - I - E - J - K - C - F - L - G$

$\therefore A - B - D - E - J - K - C - F - M$

이진 트리 탐방

❖중위탐방(Inorder Traversal)

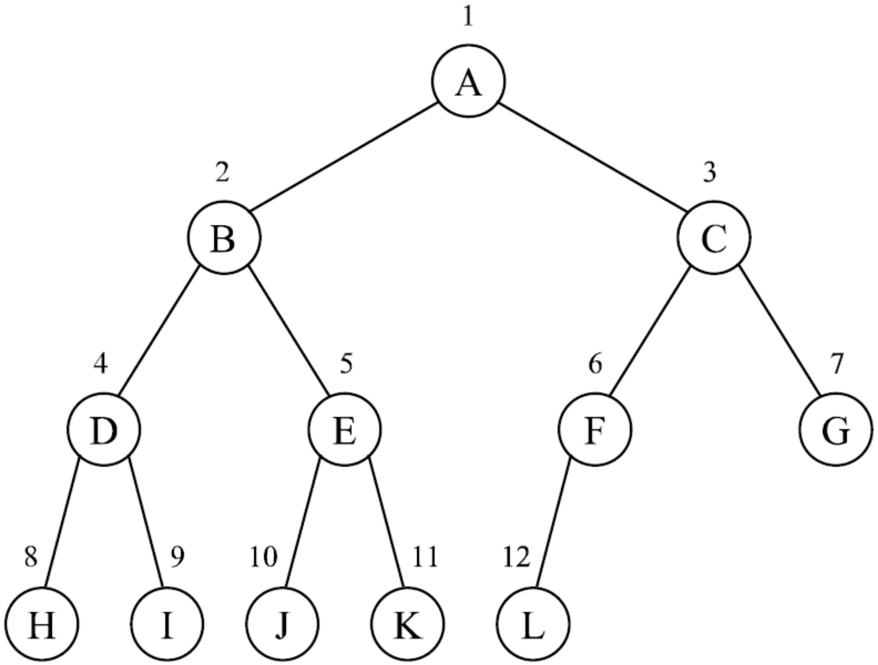
- 왼쪽 노드 - 루트 노드 - 오른쪽 노드 순으로 방문하는 탐방 방식
  - 노드 1에 방문 : 왼쪽 노드가 있는지 확인
  - 노드 2에 방문 : 왼쪽 노드가 있는지 확인
  - 왼쪽 노드가 없으므로, 노드 2의 데이터 B를 읽는다(L).
  - 노드 2의 오른쪽 노드가 있는지 확인 : 오른쪽 노드도 없다.
  - 다시 노드 1에 방문 : 데이터 A를 읽는다(P).
  - 노드 1의 오른쪽 노드가 있는지 확인
  - 노드 3에 방문 : 왼쪽 노드가 있는지 확인
  - 왼쪽 노드가 없으므로, 노드 3의 데이터 C를 읽는다(R)
  - 노드 3의 오른쪽 노드가 있는지 확인 : 오른쪽 노드도 없다.
  - 다시 노드 1에 방문 : 탐방 종료

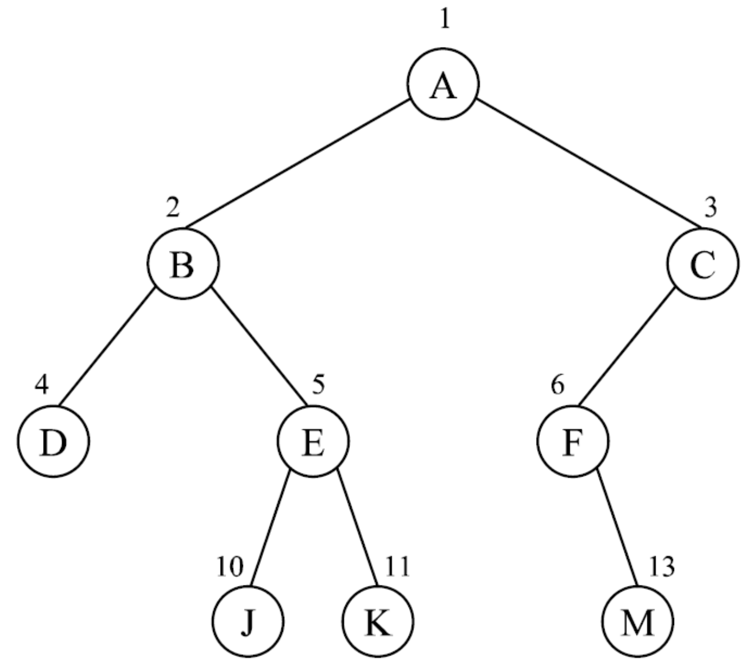


이진 트리 중위탐방 (Inorder Traversal)

예제 9-10

다음 이진 트리를 중위탐방 (Inorder Traversal) 으로 탐방하라.





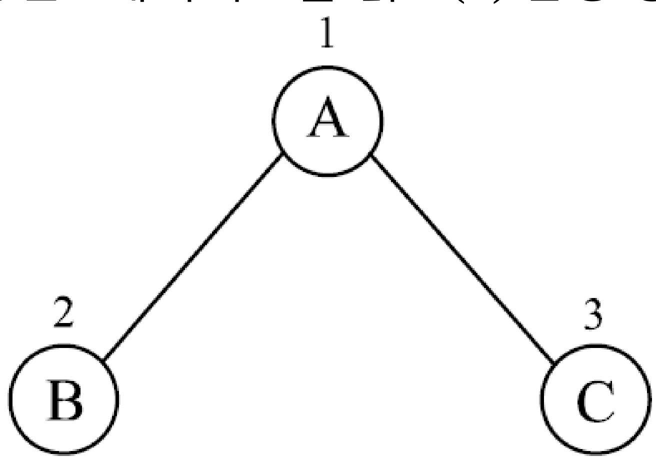
∴중위 $H - D - I - B - J - E - K - A - L - F - C - G$
(전위) $A - B - D - H - I - E - J - K - C - F - L - G$

∴중위 $D - B - J - E - K - A - F - M - C$
(전위) $A - B - D – E - J - K - C - F - M$

이진 트리 후위탐방(Postorder Traversal)

❖후위탐방(Postorder Traversal)

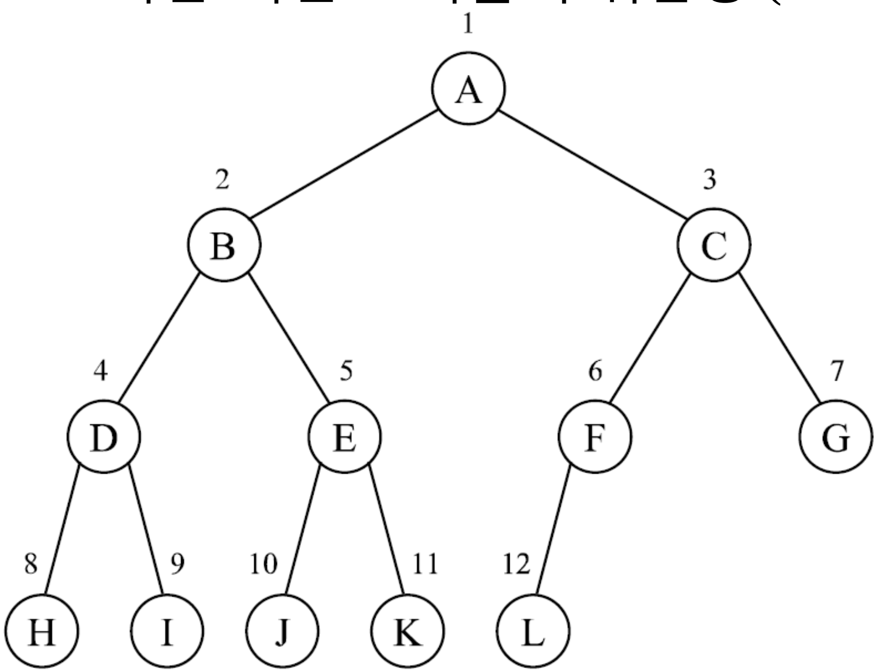
- 왼쪽 노드 - 오른쪽 노드 - 루트 노드 순으로 방문하는 탐방 방식
  - 노드 1에 방문 : 왼쪽 노드가 있는지 확인
  - 노드 2에 방문 : 왼쪽 노드가 있는지 확인
  - 왼쪽 노드가 없으므로, 노드 2의 오른쪽 노드가 있는지 확인
  - 오른쪽 노드도 없으므로, 노드 2의 데이터 B를 읽는다(L).
  - 다시 노드 1에 방문 : 노드 1의 오른쪽 노드가 있는지 확인
  - 노드 3에 방문 : 왼쪽 노드가 있는지 확인
  - 왼쪽 노드가 없으므로, 노드 3의 오른쪽 노드가 있는지 확인
  - 오른쪽 노드도 없으므로, 노드 3의 데이터 C를 읽는다(R).
  - 다시 노드 1에 방문 : 데이터 A를 읽고(P) 탐방 종료

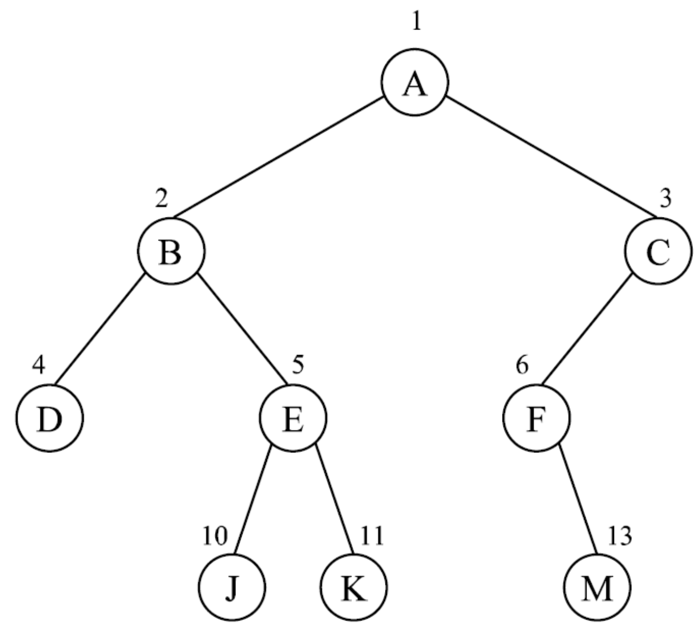


이진 트리 후위탐방(Postorder Traversal)

예제 9-10

다음 이진 트리를 후위탐방 (Postorder Traversal) 으로 탐방하라.



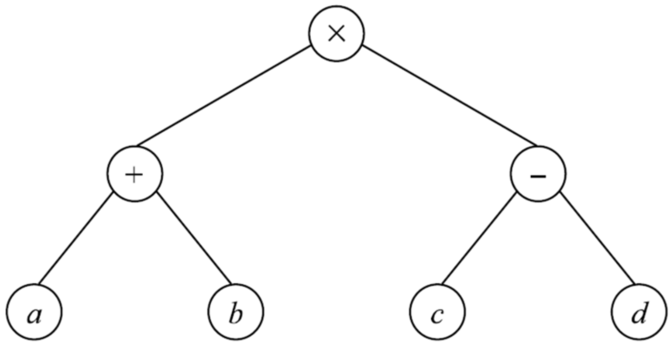


∴후위 $H - I - D - J - K - E - B - L - F - G - C - A$
(중위) $H - D - I - B - J - E - K - A - L - F - C - G$
(전위) $A - B - D - H - I - E - J - K - C - F - L - G$

∴후위 $D - J - K - E - B - M - F - C - A$
(중위) $D - B - J - E - K - A - F - M - C$
(전위) $A - B - D – E - J - K - C - F - M$

이진 트리 수식 표현

❖이진 트리는 수식을 표현하는 방법에 사용 가능
예) (a+b)×(c-d)를 표현하는 트리



[그림 9-15] $(a+b)\times(c-d)$

- (a+b)×(c-d)의 풀이 순서

① a, b, c, d 입력
② (a + b) 계산
③ (c - d) 계산
④ (a + b) × (c - d) 계산

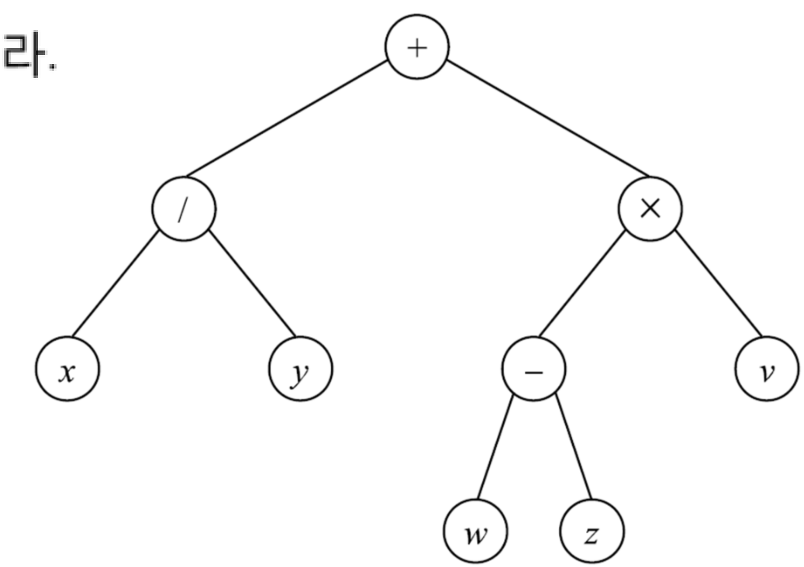
- 가장 나중에 연산되는 × 가 루트에 위치. 입력값들은 항상 잎(leaf) 노드에 위치.
  먼저 연산될수록 서브 트리에 속함 (+, –).

예제 9-13

식 $(x/y)+(w-z)\times v$를 이진 트리로 표현하라.

풀이 $(x/y)+(w-z)\times v$ 의 풀이 순서

① v, w, x, y, z 입력
② x/y 계산
③ $w-z$ 계산
④ $(w-z)\times v$ 계산
⑤ $(x/y)+(w-z)\times v$ 계산



이진 트리 탐방 표기법

❖전위표기

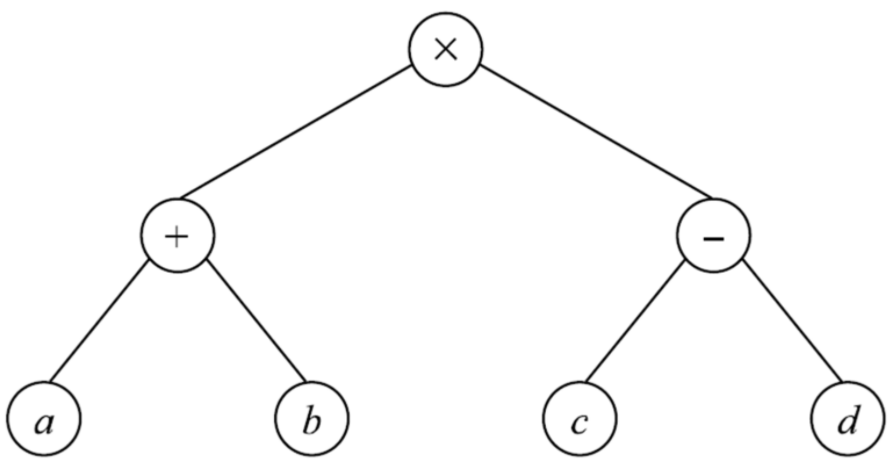
- 수식에서의 연산자가 피연산자보다 앞에 작성되는 표기법
- 연산자 - 피연산자1 - 피연산자2 [그림9-15] **× + a b – c d**

❖중위표기

- 수식에서의 연산자가 피연산자들의 중간에 작성되는 표기법
- 피연산자1 - 연산자 - 피연산자2 [그림9-15] **a + b × c – d**

❖후위표기

- 수식에서의 연산자가 피연산자들의 뒤에 작성되는 표기법
- 피연산자1 - 피연산자2 – 연산자 [그림9-15] **a b + c d – ×**



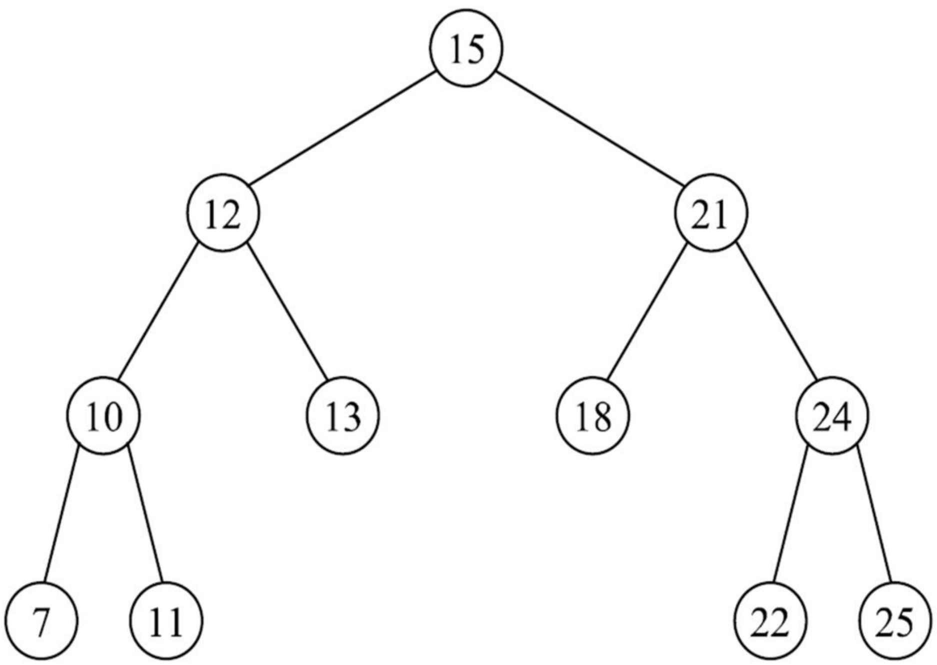
[그림 9-15] $(a+b)\times(c-d)$

이진 탐색 트리

❖이진 탐색 트리(Binary Search Tree)

- 노드가 가지는 데이터의 크기에 따라 노드의 위치를 탐색할 수 있는 트리
  - 트리에서 탐색되는 모든 원소는 서로 다른 유일키를 갖는다.
  - 좌측 서브 트리에 있는 원소의 키들은 그 루트의 키보다 작다.
  - 우측 서브 트리에 있는 원소들의 키들은 그 루트의 키보다 크다.

예) 이진 탐색 트리 예



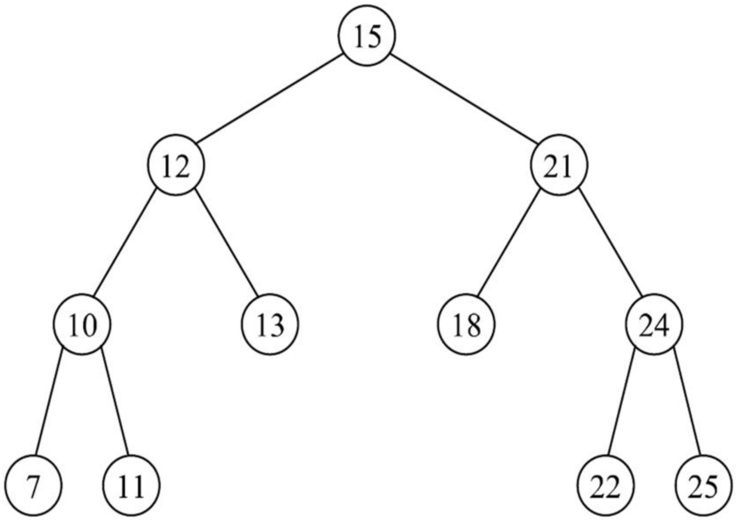
- 루트 15를 기준으로 좌측 서브 트리는 15보다 작은값, 우측은 큰값들로 구성.
  15는 탐색되는 모든 원소들의 유일키
- 루트 좌측 서브 트리: 12보다 작은값은 좌측, 큰값은 우측 서브트리 구성.
  15보다 작은값들은 12를 유일키로 탐색
- 루트 우측 서브 트리:21보다 작은값은 좌측, 큰값은 우측 서브트리 구성.
  15보다 작은값들은 21을 유일키로 탐색.

❖이진탐색트리에서 탐색되는 모든 원소는 유일키를 하나씩 가지며, 유일키를 기준으로 좌측 서브트리에는 유일키보다 작은값이, 우측 서브트리에는 유일키보다 큰값이 구성

이진 탐색 트리

예제 9-15

이진 탐색 트리를 이용해 11의 탐색 과정에 대해 설명하라.



풀이

① 11은 루트 15보다 작은 값이므로 15가 루트인 **좌측** 서브 트리에서 탐색된다.

② 11은 **좌측** 서브 트리의 루트 12보다 작은 값이므로 12가 루트인 **좌측** 서브 트리에서 탐색된다.

③ 11은 **좌측** 서브 트리의 루트 10보다 큰 값이므로 10이 루트인 **우측** 서브 트리에서 탐색된다.

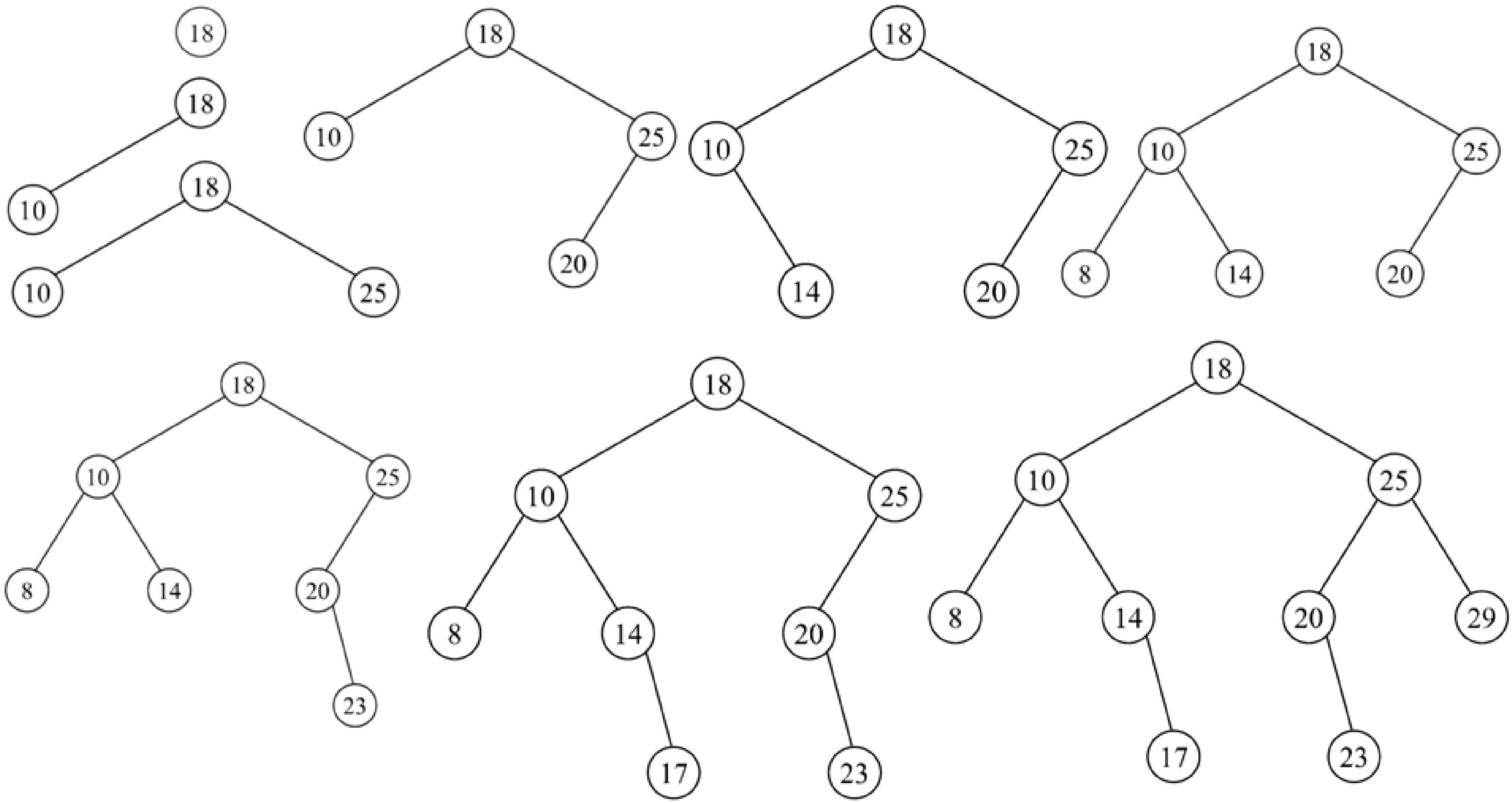
④ 11은 **우측** 서브 트리의 루트면서 잎 노드. **탐색 종료.**

∴ 11은 10보다 크고 15보다 작은 값이다.

이진 탐색 트리

예제 9-16

다음과 같은 차례로 입력될 때, 이진 탐색 트리를 만들어라.
18, 10, 25, 20, 14, 8, 23, 17, 29

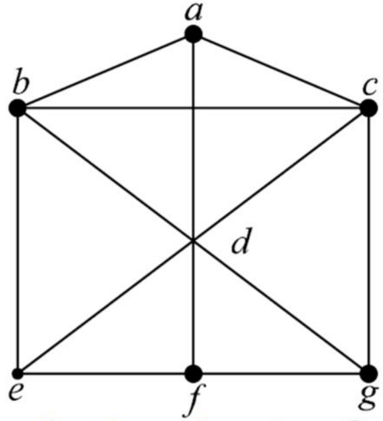


트리의 활용

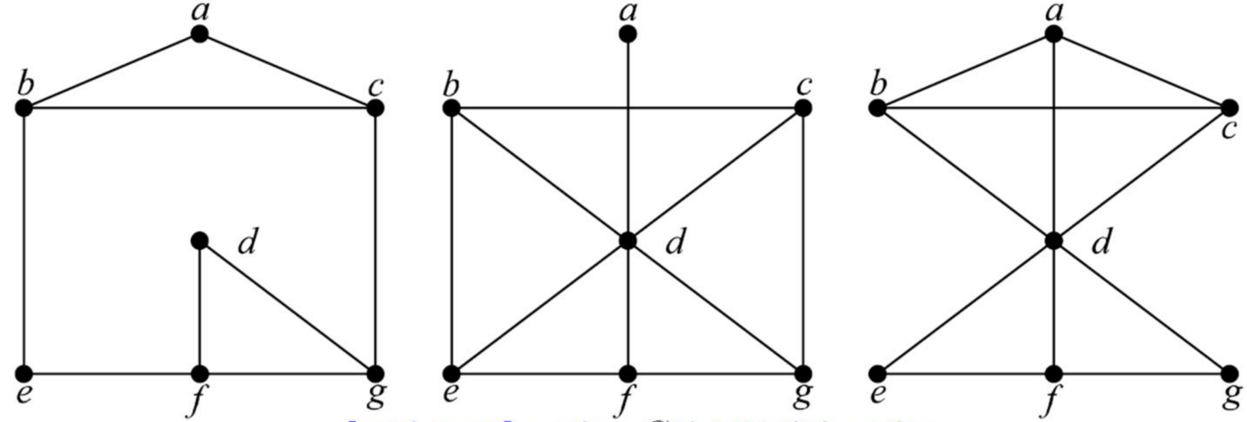
❖정점과 변으로 구성된, 순환이 발생하지 않는 그래프가 트리.
하나의 그래프가 주어졌을 때 다양한 트리를 유도 가능.

❖부분 생성(신장) 그래프

- 부분 생성(신장) 그래프는 그래프 G의 정점 집합 V의 모든 원소를 포함하면서, 변의 집합 E의 일부 원소만 포함하는 그래프.

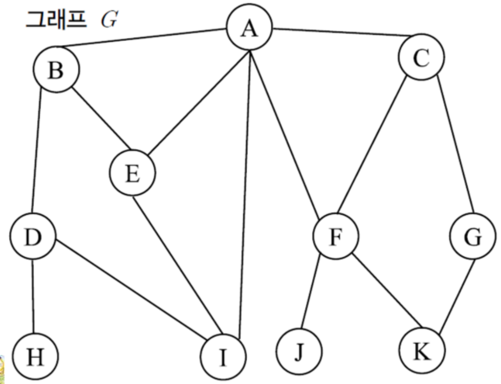


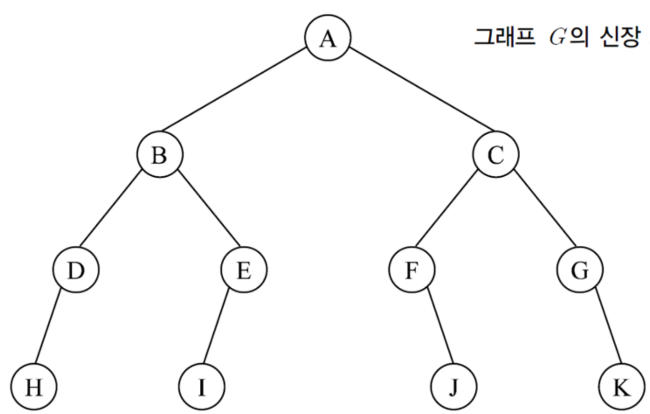
[그림 9–17] 그래프 G

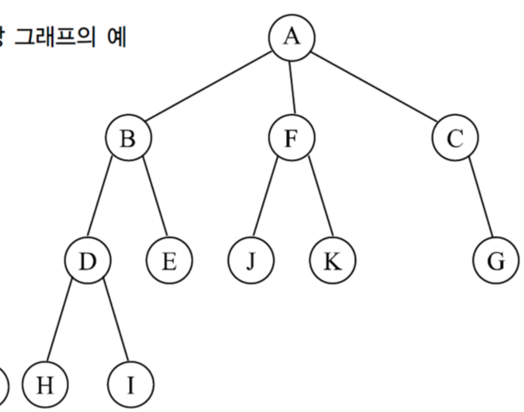


[그림 9–18] 그래프 G의 부분신장 그래프

❖Spanning Tree(생성,신장 트리) :그래프 G의 정점을 모두 노드로 포함하는 트리 T.







트리의 활용

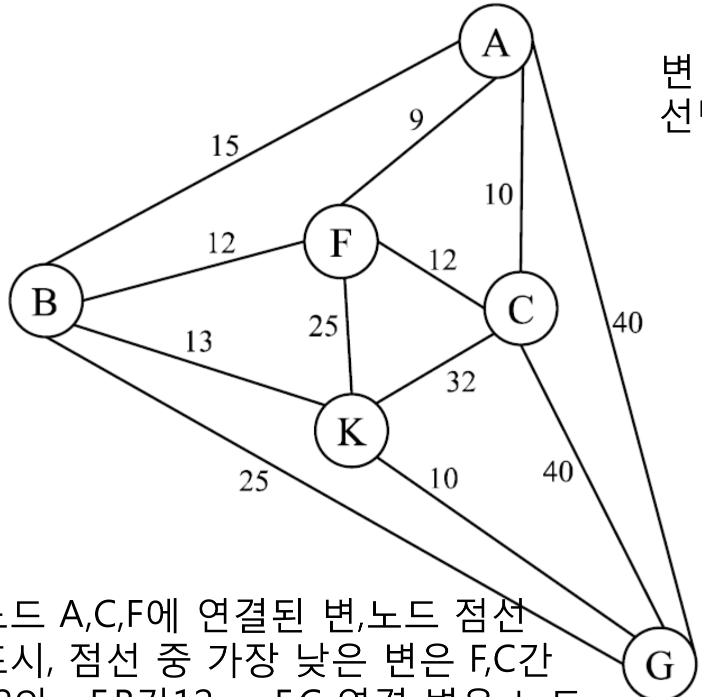
❖최소 생성(신장) 트리(Minimal Spanning Tree)

- 그래프 G의 정점을 모두 노드로 포함하면서 비용을 최소로 하는 트리 T
- 최소 신장 트리를 구하기 위해서는 노드와 노드를 연결하는 변에 가중치가 부여된 그래프 및 알고리즘이 필요.

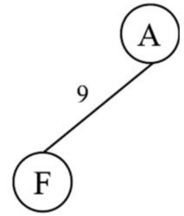
❖비용을 최소로 하는 알고리즘 : 비용이 가장 낮은 변들로 트리를 구성하는 알고리즘

- 프림 알고리즘(Prim Algorithm)
  - 가중치가 가장 작은 변을 선택
  - 연결된 정점들과 연결된 모든 변들 중 가중치가 가장 작은 변을 선택
  - 가중치가 같은 변은 임의로 선택
  - 선택된 변에 의해 순환이 형성되는 경우는 선택하지 않음
  - n개의 정점에 대하여 n-1개의 변이 연결되면 종료
- 크루스칼 알고리즘(Kruskal Algorithm)
  - 가중치가 가장 작은 변을 차례로 선택하여 노드들을 연결
  - 가중치가 같은 변은 임의로 선택
  - 선택된 변에 의해 회로가 형성되는 경우는 선택하지 않음
  - n개의 정점에 대하여 n-1개의 변이 연결되면 종료

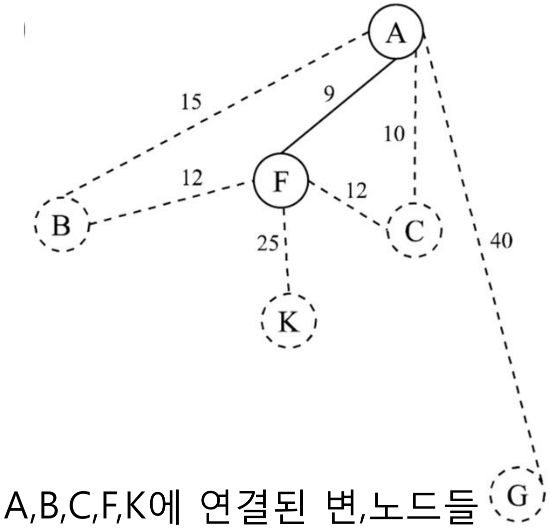
트리의 활용 : 프림 알고리즘



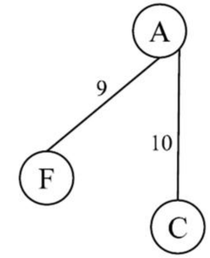
변 비용 중 가장 작은 9를 선택, 노드 A, F가 연결



노드 A, F에 연결된 변,노드들은점선 표시

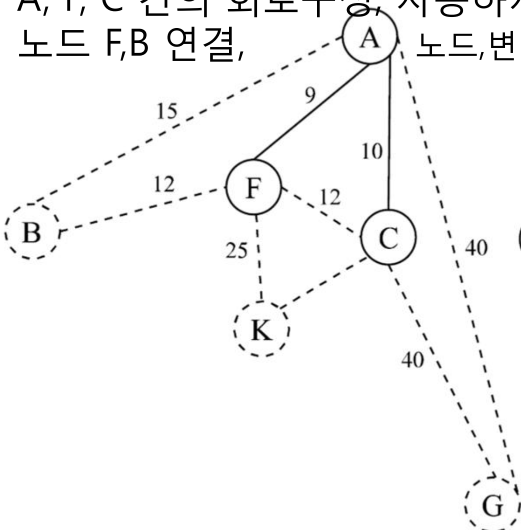


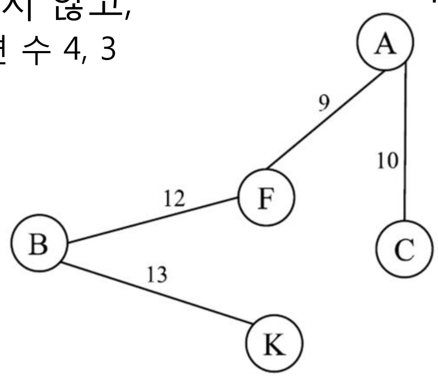
점선 중 비용이 가장 낮은 10을 선택. 노드 A와 C가 연결



노드 A,C,F에 연결된 변,노드 점선 표시, 점선 중 가장 낮은 변은 F,C간 12와 F,B간12. F,C 연결 변은 노드 A, F, C 간의 회로구성, 사용하지 않고, 노드 F,B 연결,

노드,변 수 4, 3

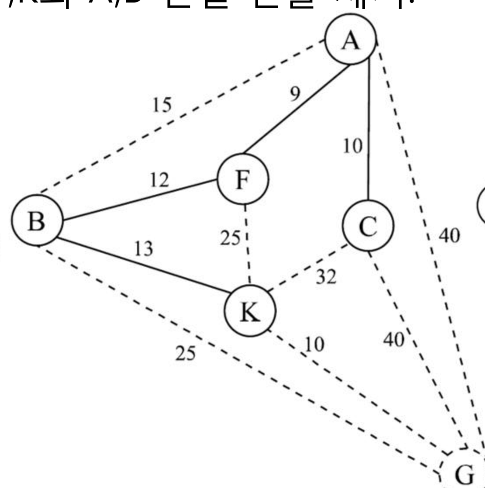




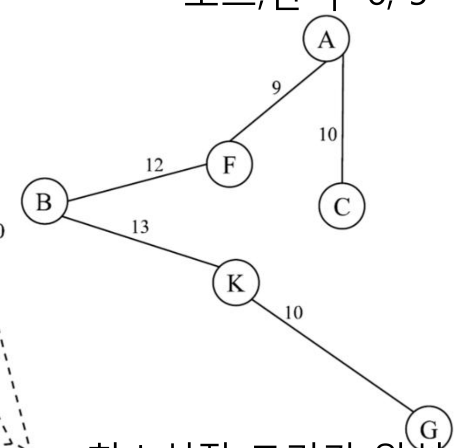
노드 A,B,C,F 에 연결된 최소 변: B,K 연결,

노드,변 수 5, 4

노드 A,B,C,F,K에 연결된 변,노드들 점선표시. 회로가 생성 되는 F,C와 F,K와 A,B 연결 변을 제거.



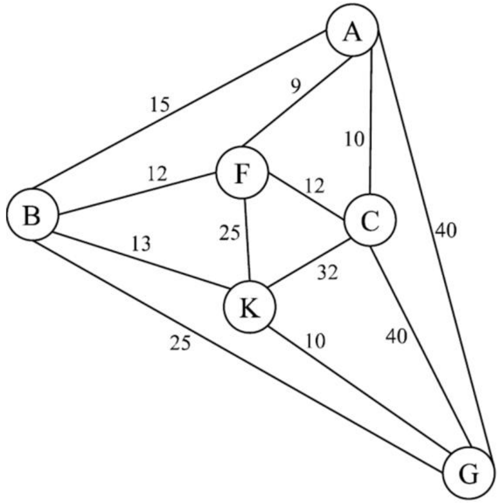
점선 중 최소 비용 변: K,G 연결된 10 노드,변 수 6, 5



최소신장 트리가 완성.
최소신장 트리비용:
9+10+12+13+10= 54

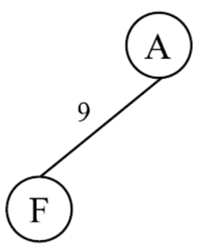
트리의 활용 : 트루스칼 알고리즘

❖모든 노드에 연결된 가지의 비용을 정리하면 다음 표와 같다.

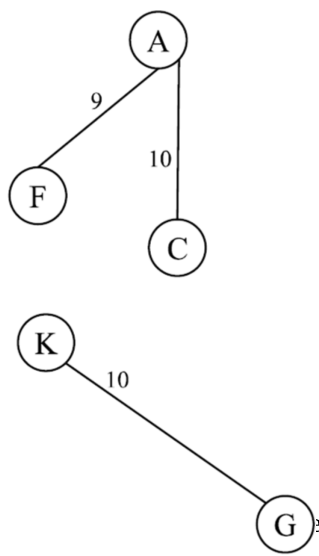


노드	연결 비용	노드	연결 비용
A-B	15	A-C	10
A-F	9	A-G	40
B-F	12	B-K	13
B-G	25	F-K	25
F-C	12	C-K	32
C-G	40	K-G	10

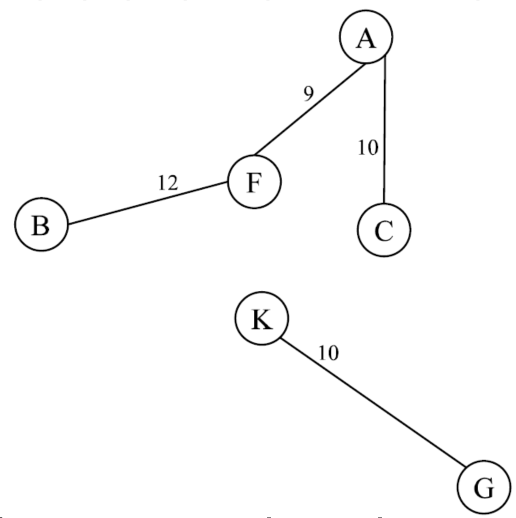
표에서 비용이 가장 낮은 노드의 연결은 A-F.



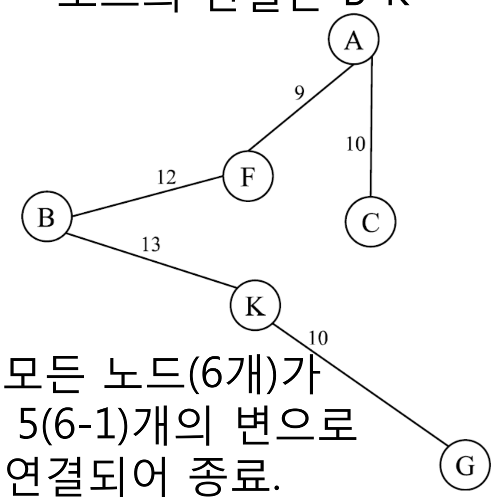
두 번째로 낮은 노드의 연결은 A-C와 K-G



세 번째로 낮은노드의 연결은 B-F와 F-C. F-C는 A-F-C 사이에 회로가 생성, 불사용.



네 번째로 낮은 노드의 연결은 B-K

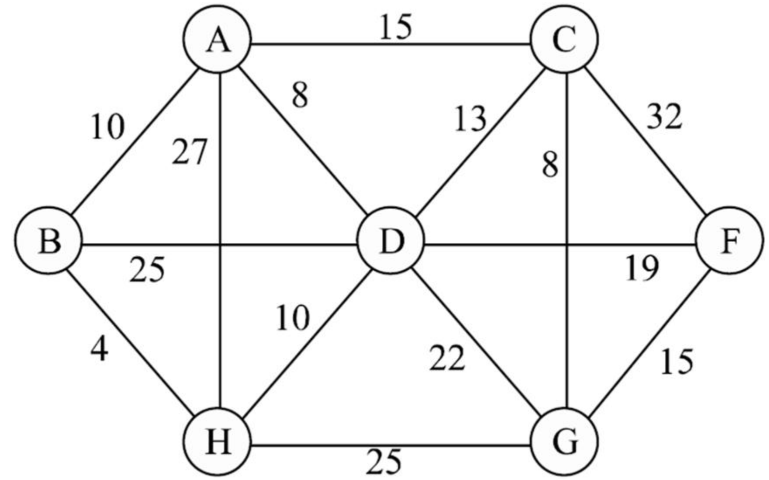


모든 노드(6개)가 5(6-1)개의 변으로 연결되어 종료.
이때 최소신장 트리 비용:
9+10+12+13+10=54

트리의 활용 : 프림알고리즘

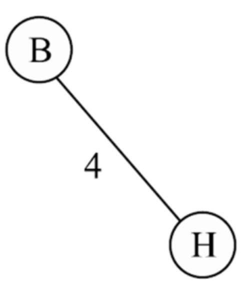
예제 **9-17**

다음 그래프 에서
(1) 프림 알고리즘을 이용하여
최소 생성 트리를 작성하고, 비용을 구하라.
(2) 크루스칼 알고리즘을 이용하여
최소 생성 트리를 작성하고, 비용을 구하라.

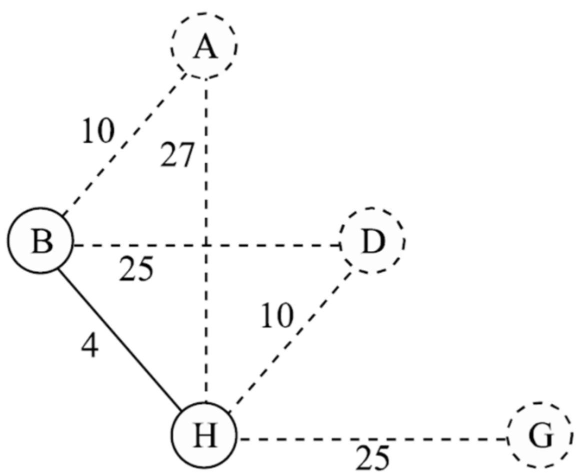


풀이
(1)

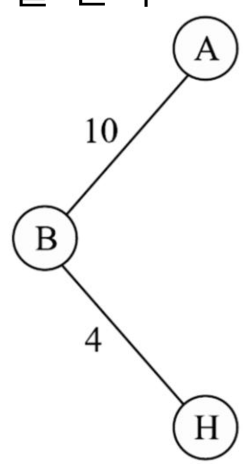
① 변에 부여된 비용 중 가장 낮은 4를 선택하면 B와 H가 연결.



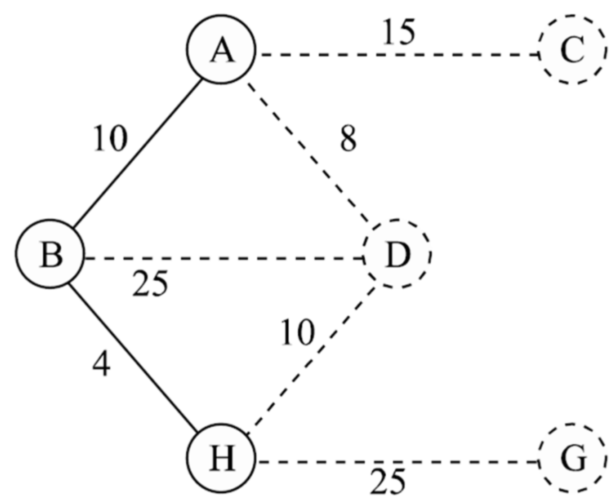
② 노드 B와 H에 연결된 변들과 노드들은 다음 점선으로 표시.

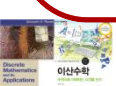


③ ②의 점선 중 비용이 가장 낮은 변은 B와 A, H와 D가 연결된 10. 이 중 B와 A를 선택.



④ 노드 A, B, H에 연결된 변, 노드들은 점선으로 표시.
A와 H간 변은 A,B,H간 회로 생성, 제외하였다.

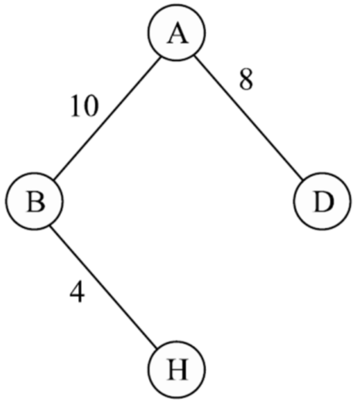




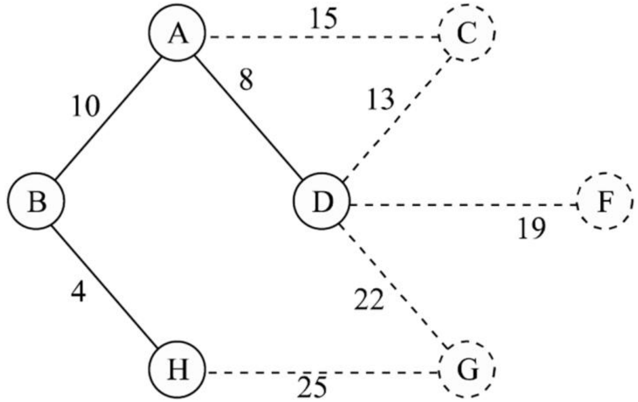
트리의 활용 : 프림알고리즘

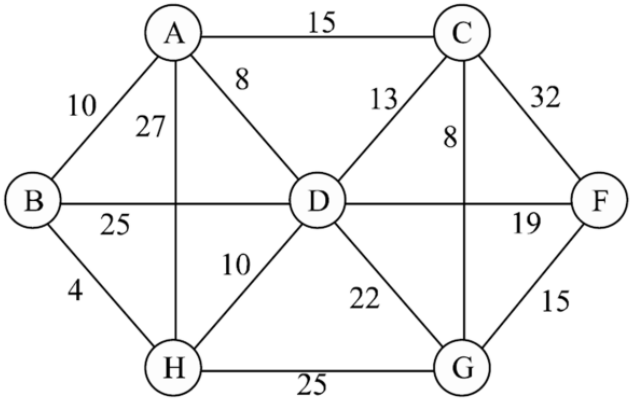
예제 9-17

⑤ ④의 점선 중 최소 비용 변은 A와 D가 연결된 8.



⑥ 노드 A, B, D, H에 연결된 변들과 노드들은 다음 점선으로 표시. A와 H, B와 D, H와 D 간의 변은 A,B, D,H 간 회로를 생성, 제외

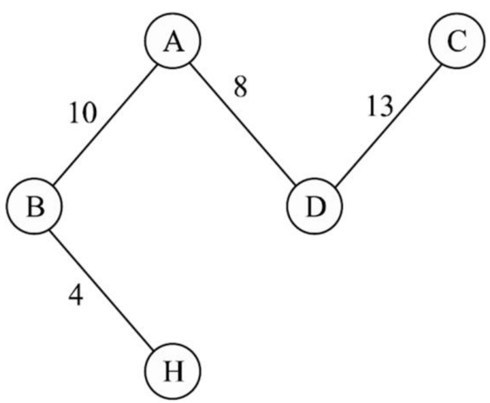


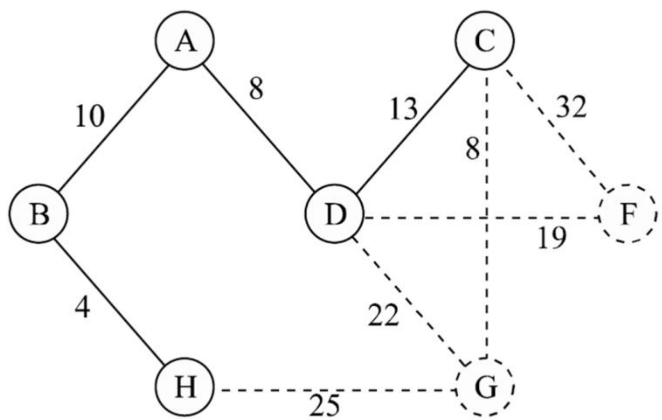


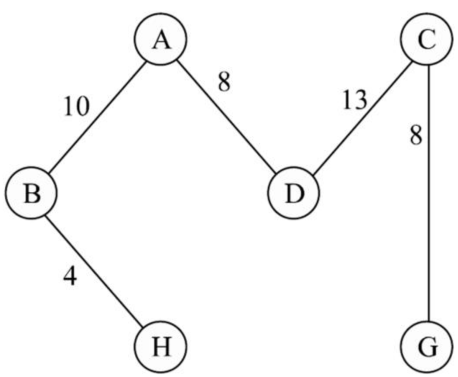
⑦ ⑥의 점선 중 최소 비용 변은 D와 C가 연결된 13.

⑧ 노드 A,B,C, D,H에 연결된 변,노드들은 점선으로 표시. A-C 변은 A,C,D간 회로 생성, 제외.

⑨ ⑧의 점선 중 최소 비용 변은 C와 G가 연결된 8.

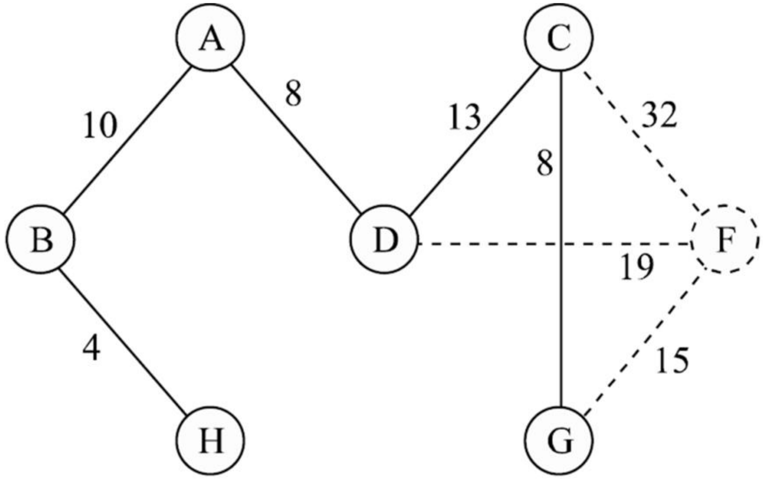


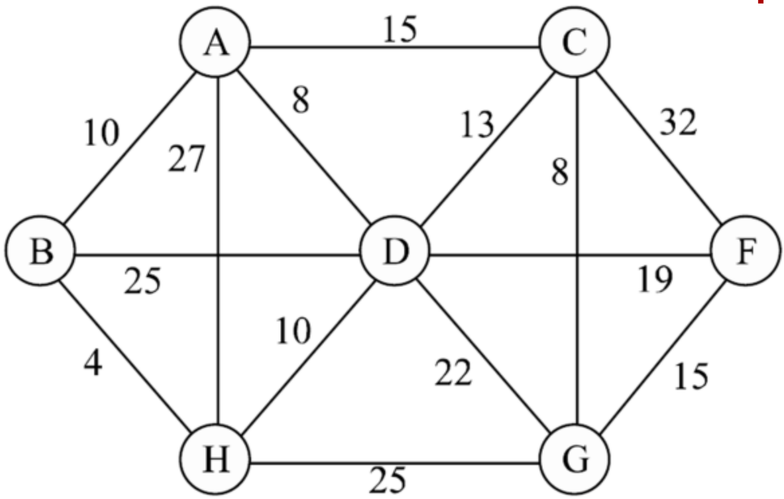




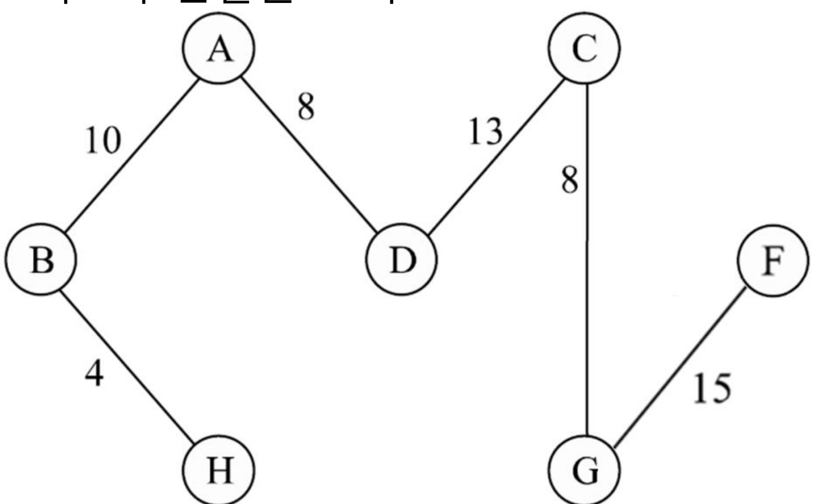
예제 9-17

⑩ 노드 A, B, C, D, G, H에 연결된 변들과 노드들은 점선으로 표시. D와 G 간의 변은 C, D, G 사이에 회로를 생성하므로 최소 비용 대상에서 제외.





⑪ ⑩의 점선 중 비용이 최소인 변은 G와 F가 연결된 15다

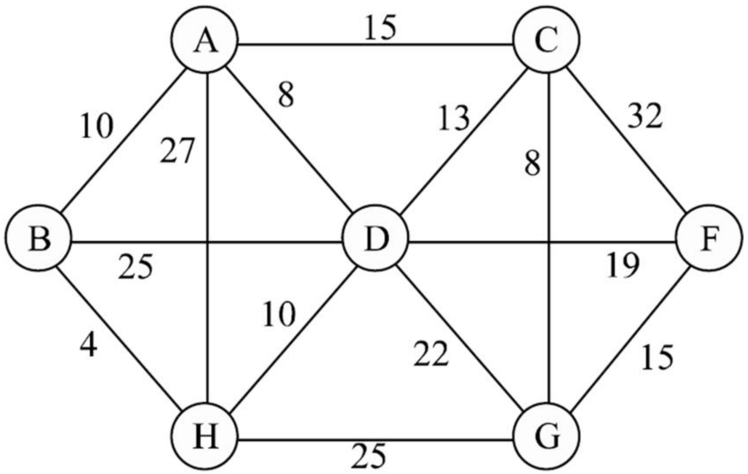


⑫ 모든 노드 7개가 6(7-1)개의 가지와 연결되었으므로 종료한다.
이때 최소신장 트리의 비용은
4 + 10 + 8 + 13 + 8 + 15 = 58 이다.

트리의 활용 : 크루스칼 알고리즘

예제 9-17

(2) 크루스칼 알고리즘 이용

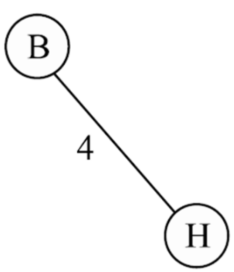


① 모든 노드에 연결된 변의 비용을 정리한 표.

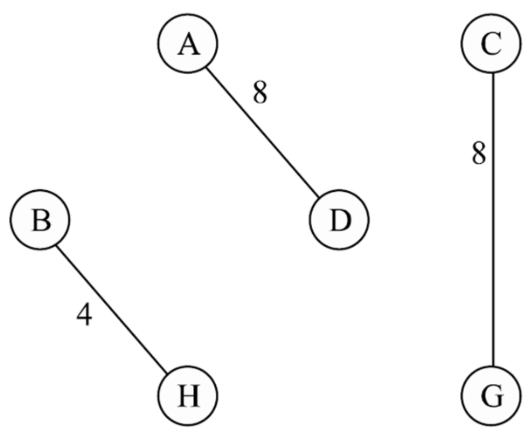
노드연결	비용	노드연결	비용
A-B	10	A-C	15
A-H	27	A-D	8
B-D	25	B-H	4
C-D	13	C-G	8
C-F	32	D-H	10
D-G	22	D-F	19
F-G	15	H-G	25

①의 표에서

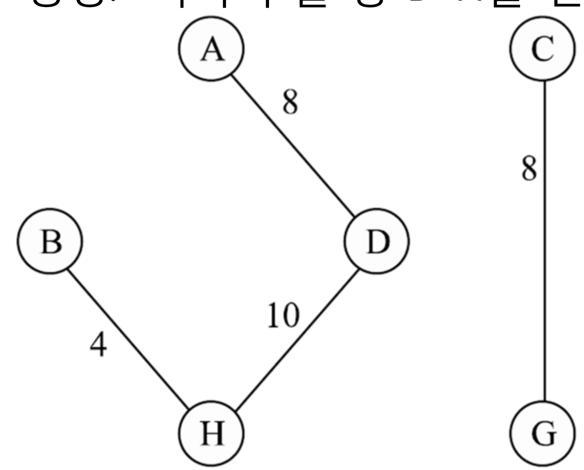
② 가장 비용이 낮은 노드 연결은 B-H.



③ 두 번째로 비용이 낮은 노드 연결은 A-D와 C-G



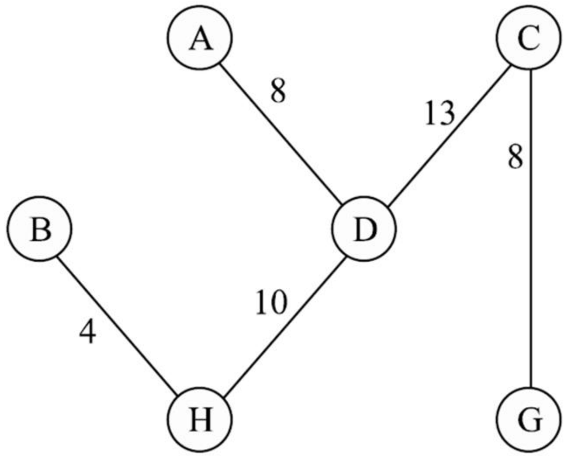
④ 세 번째로 비용이 낮은 노드 연결은 A-B와 D-H. 둘 다 선택하면 A, B, D, H 사이에 회로가 생성. 여기서 둘 중 D-H를 선택



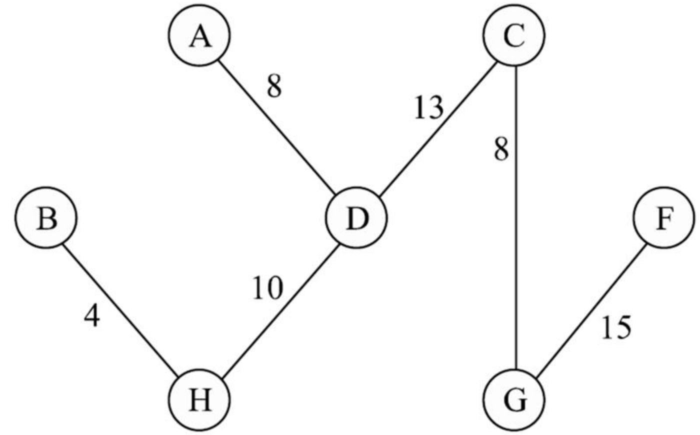
트리의 활용 : 트루스칼 알고리즘

예제 9-17

⑤ 네 번째로 비용이 낮은 노드 연결은 C-D



⑥ 다섯 번째로 비용이 낮은 노드 연결은 A-C와 F-G. 그러나 A-C는 A, C, D 사이에 회로가 발생하므로 사용하지 않는다.



⑦ 모든 노드가 6(7-1)개의 가지와 연결되었으므로 종료한다.
이때 최소신장 트리의 비용은
4 + 10 + 8 + 13 + 8 + 15 = 58이다.