탐색트리

- Comparable 인터페이스

Public int compareTo(Object object)

c<0 이면, this < object

c=0 이면, this = object

c>0 이면, this > object

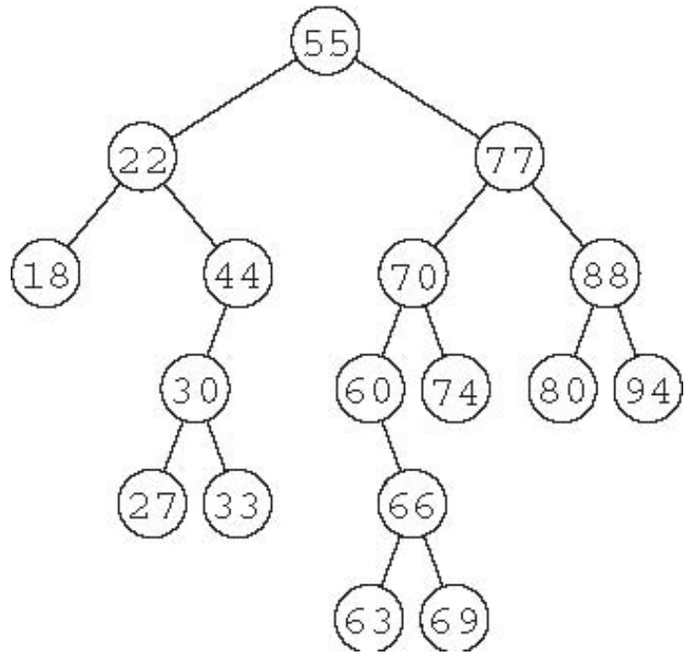
탐색 트리가 만족해야 하는 조건

– 탐색 트리 구조 내에 있는 **키는 유일함**
– 각각의 키는 그것이 표현하는 데이터의 주소를 가지고 있음

– 키의 타입은 java.lang.Comparable인터페이스를 구현함
– 주소의 타입은 Address인터페이스를 구현함

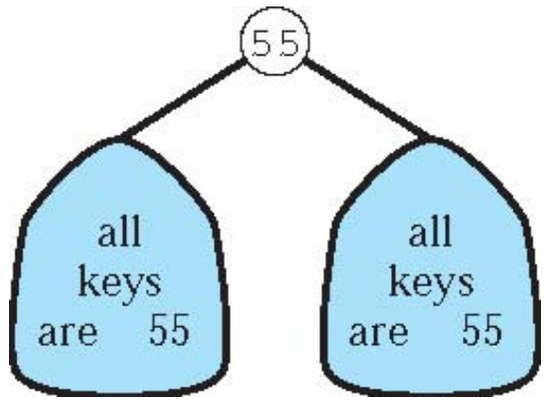
이진 탐색 트리 (BST: binary search tree)

: 각각의 노드가 BST 특성을 만족하는 키-주소 쌍을 가지고 있는 이진트리

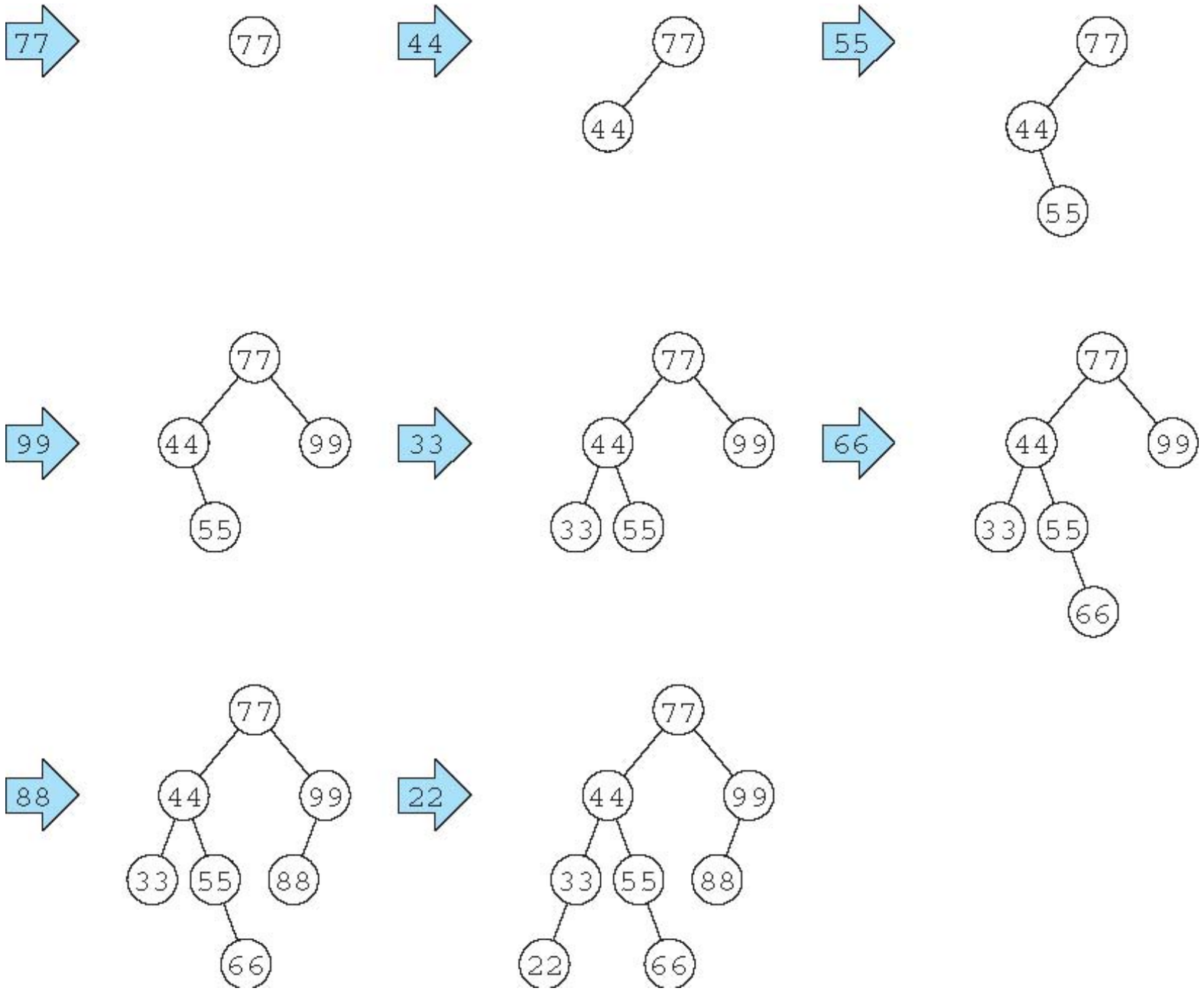


- 특징

트리에 있는 각각의 키에 대해, 왼쪽 서브트리에 있는 모든 키는 이것보다 작고, 오른쪽 서브트리에 있는 키는 이것보다 큼

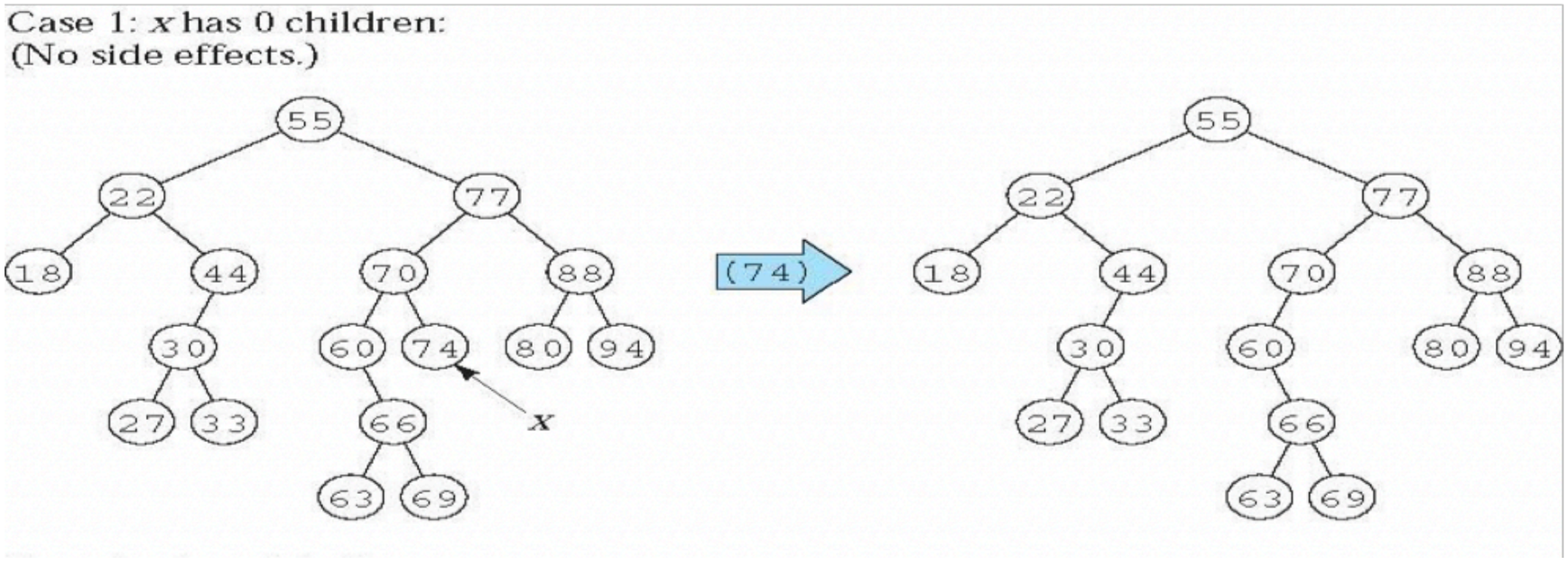


- BST 삽입 예


77
77
44
77
44
55
77
44
55
99
77
44
99
55
33
77
44
99
33
55
66
77
44
99
33
55
66
88
77
44
99
33
55
88
66
22
77
44
99
33
55
88
22
66

- BST 삭제

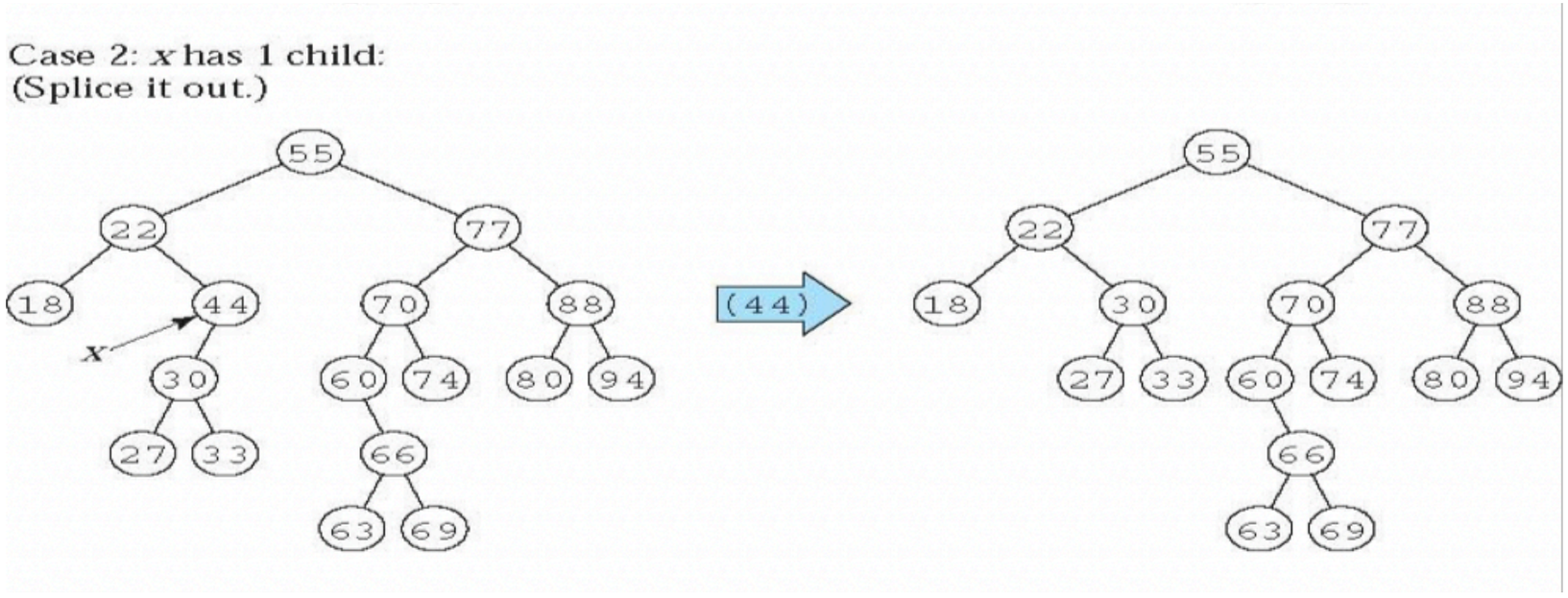
1. 삭제하려는 노드가 단일 노드인 경우 -> 해당 노드만 삭제



- BST 삭제

2. 삭제하려는 노드가 하나의 서브트리만 가지고 있는 경우

-> 자식 노드를 부모 노드에 붙여줌



- BST 삭제

3. 삭제하려는 노드가 두개의 서브트리 모두 가지고 있는 경우

-> 중위 후속자(y)를 해당 위치에 위치시켜 준다.

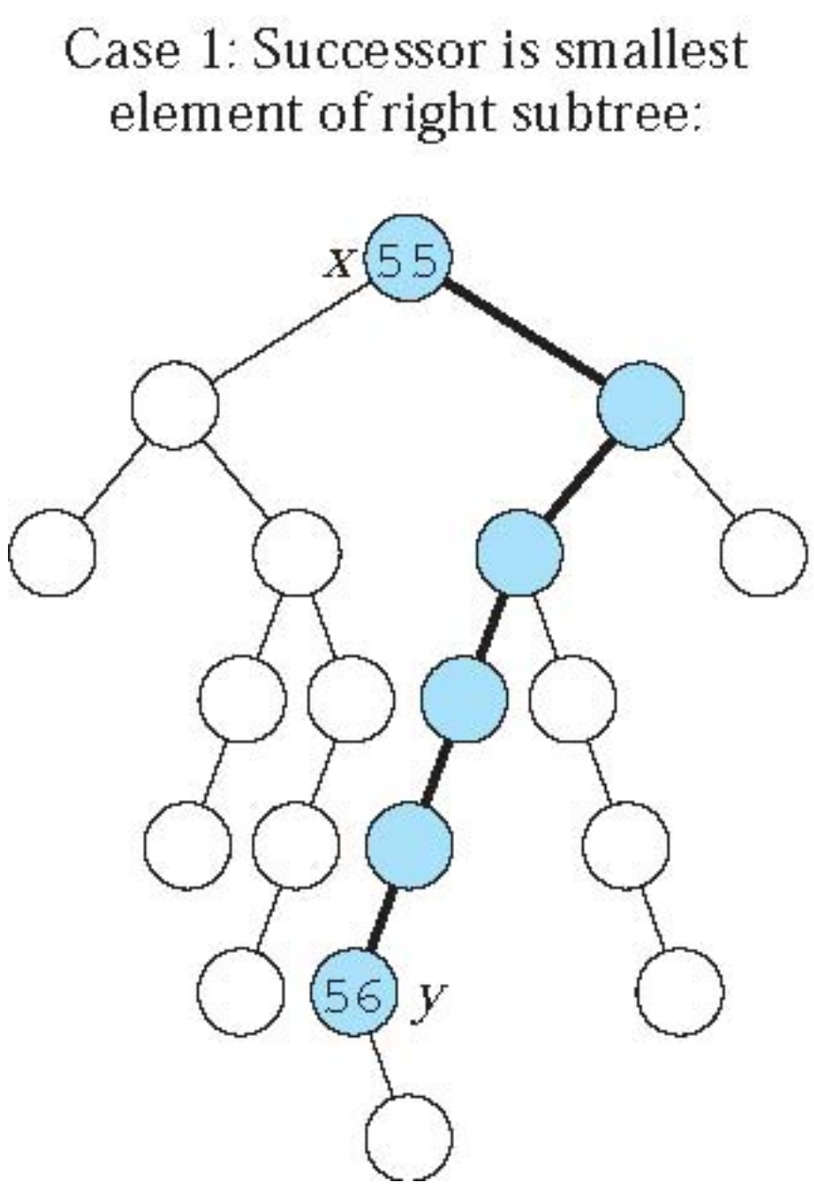
(이진트리의 특성을 잃지 않도록! 트리의 변동성 최소화)

Ⅰ. 오른쪽 서브트리에서 가장 왼쪽에 있는 노드이다. (가장 작은 값)

Ⅱ. 왼쪽 서브트리에서 가장 오른쪽에 있는 노드이다. (가장 큰 값)

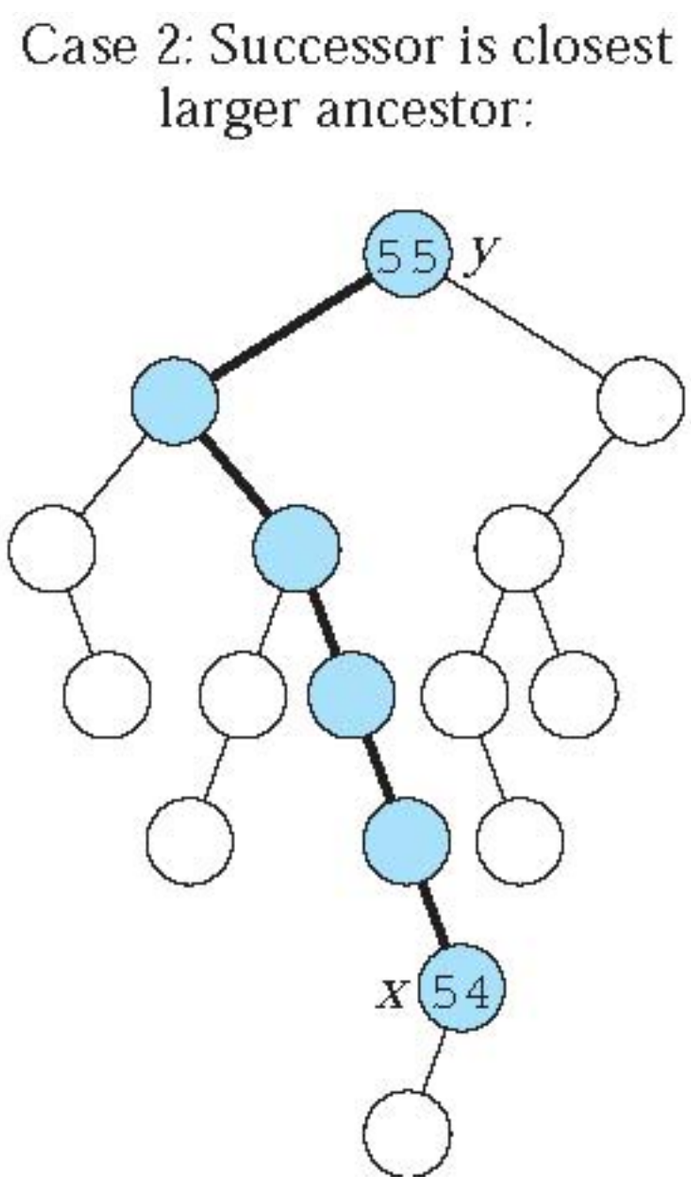
- BST 삭제

ㅈ.오른쪽 서브트리에서 가장 왼쪽에 있는 노드이다. (가장 작은 값)



- BST 삭제

II. 왼쪽 서브트리에서 가장 오른쪽에 있는 노드이다. (가장 큰 값)



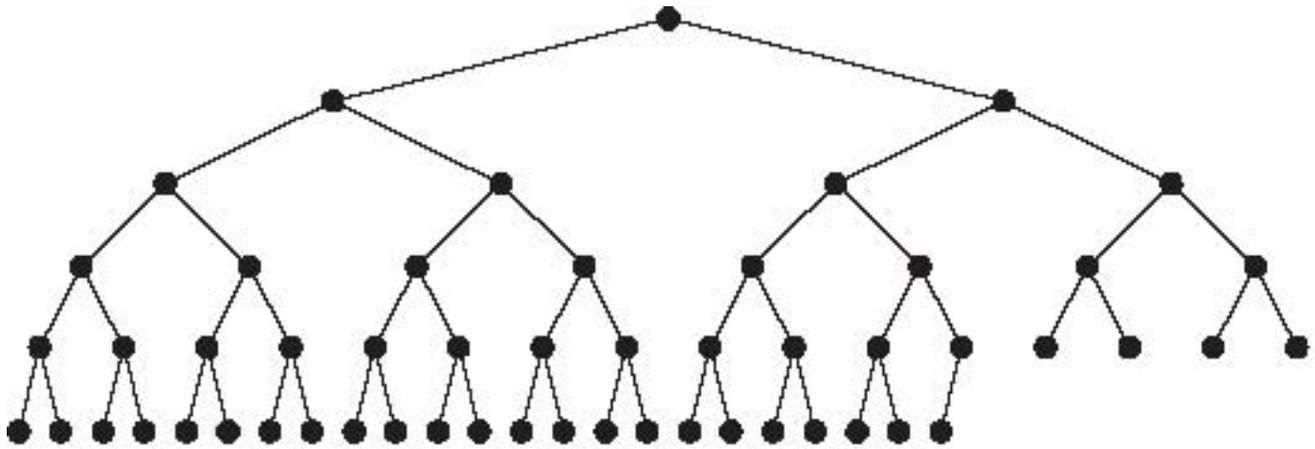
BST의 탐색과 삽입 알고리즘에 대한 시간 복잡도

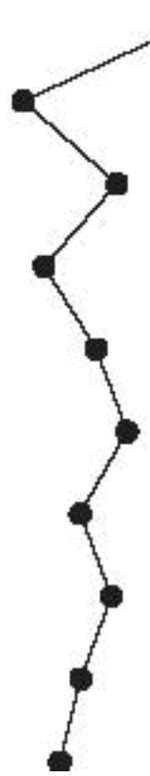
Best: O(1)

Average: **O(lg n)**

Worst: O(n)

BST의 최선과 최악





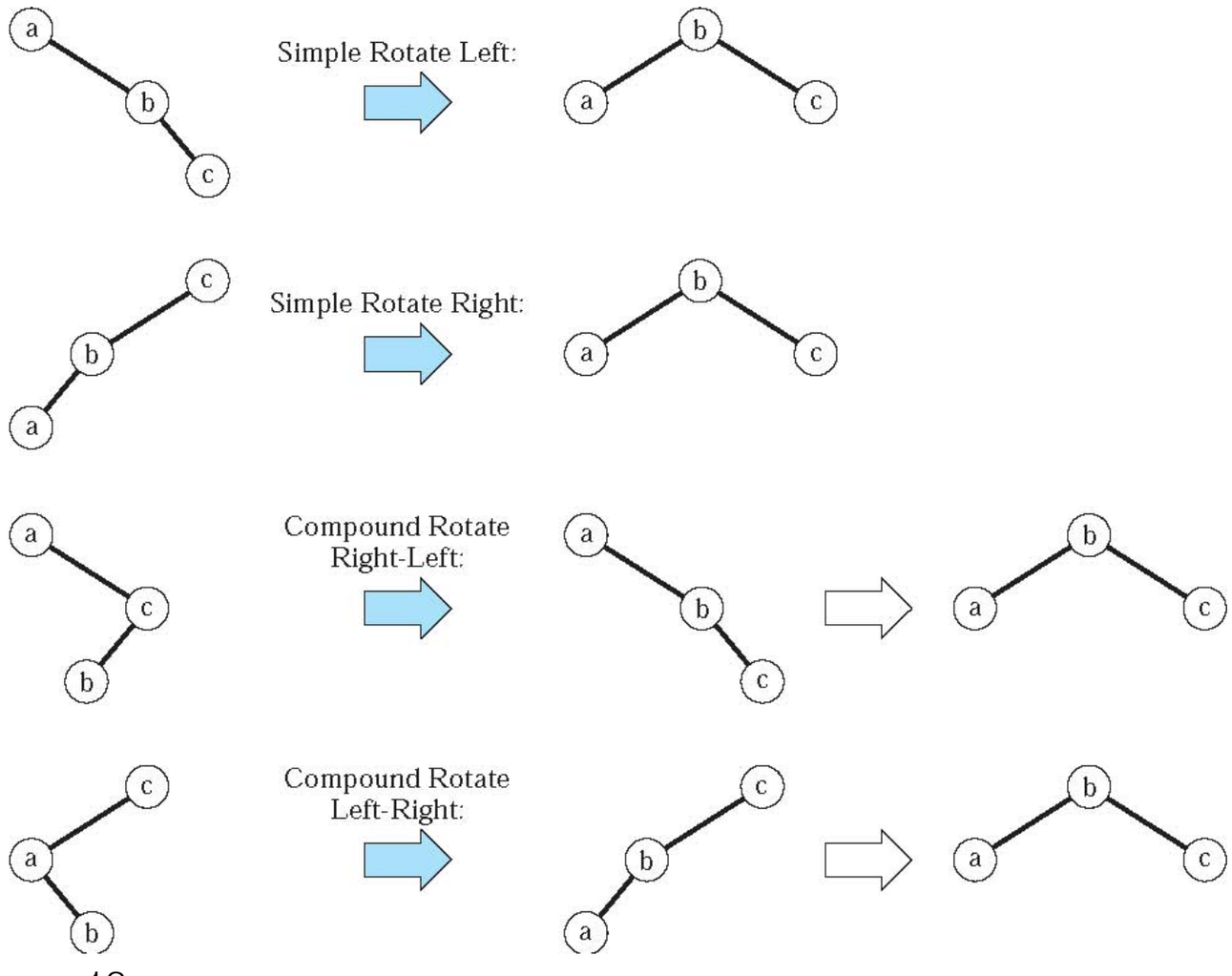
최악의 경우, BST의 장점 사라짐 -> AVL트리 탄생

AVL트리

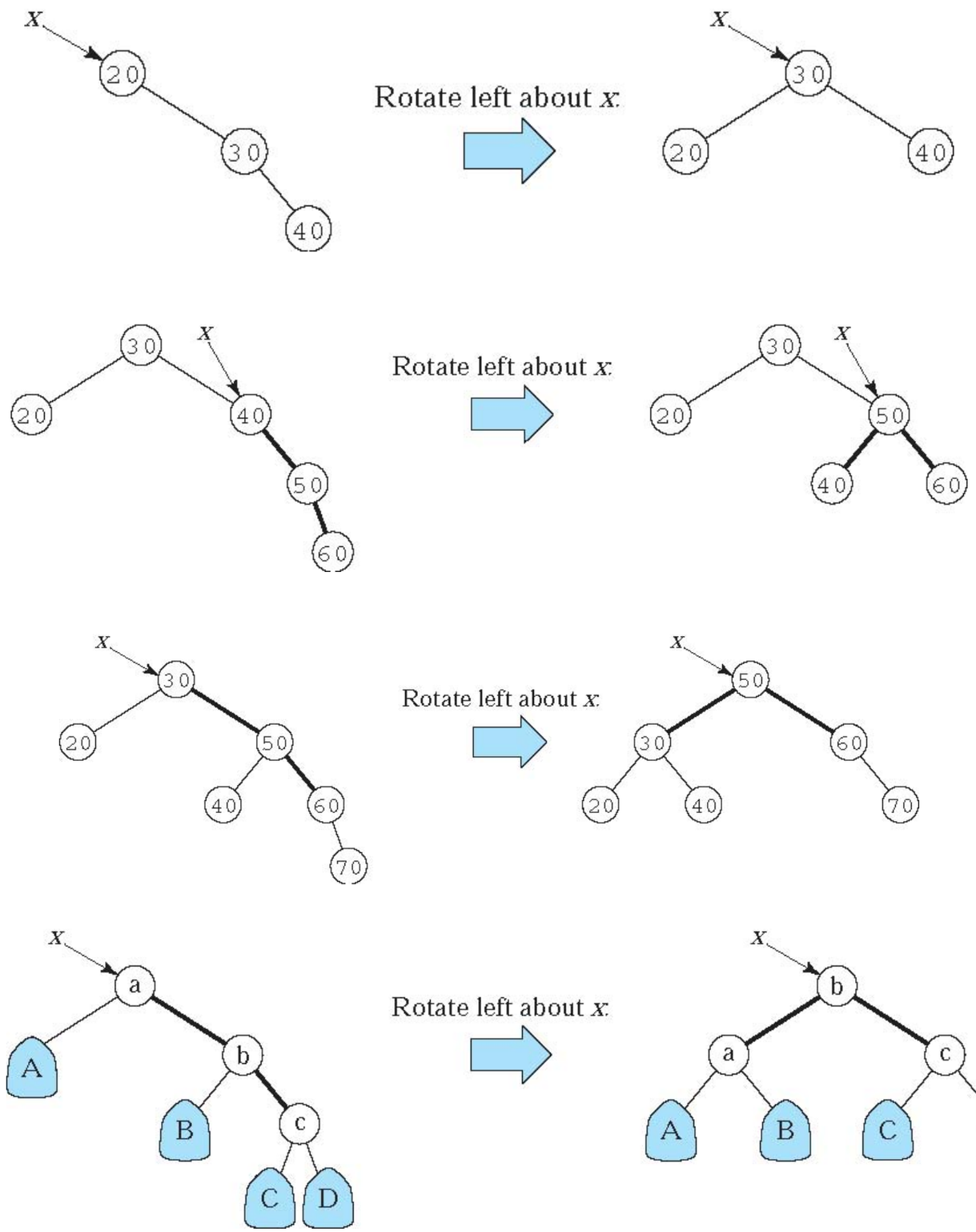
최악의 경우 O(lg n)

: 어떤 노드에서도 두 서브트리가 거의 같은 높이를 갖도록 강제하여 균형을 유지하는 이진 탐색 트리

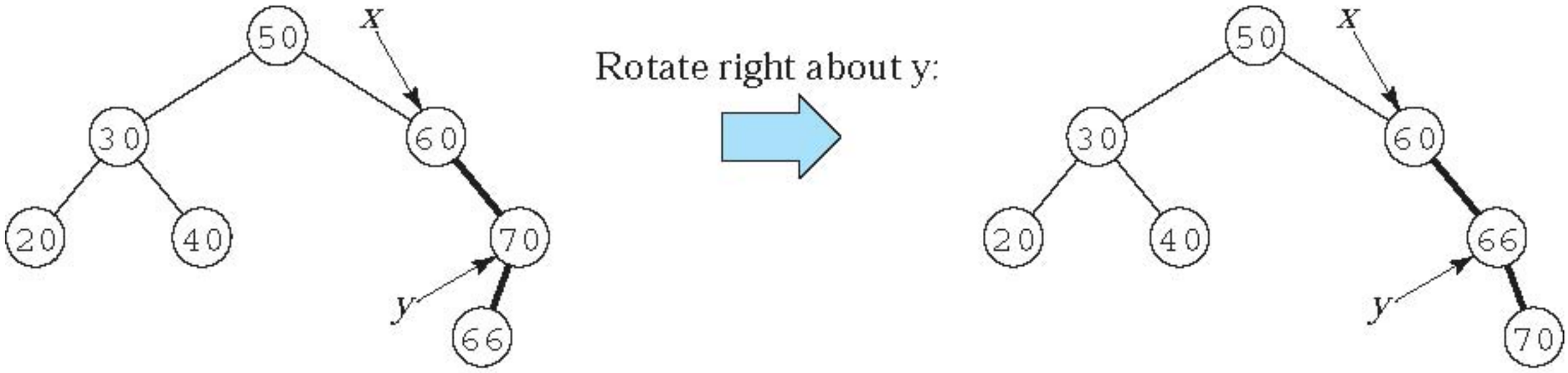
- AVL 회전의 패턴
  - 왼쪽 단순 회전
  - 오른쪽 단순 회전
  - 복합 오른쪽-왼쪽 회전
  - 복합 왼쪽-오른쪽 회전

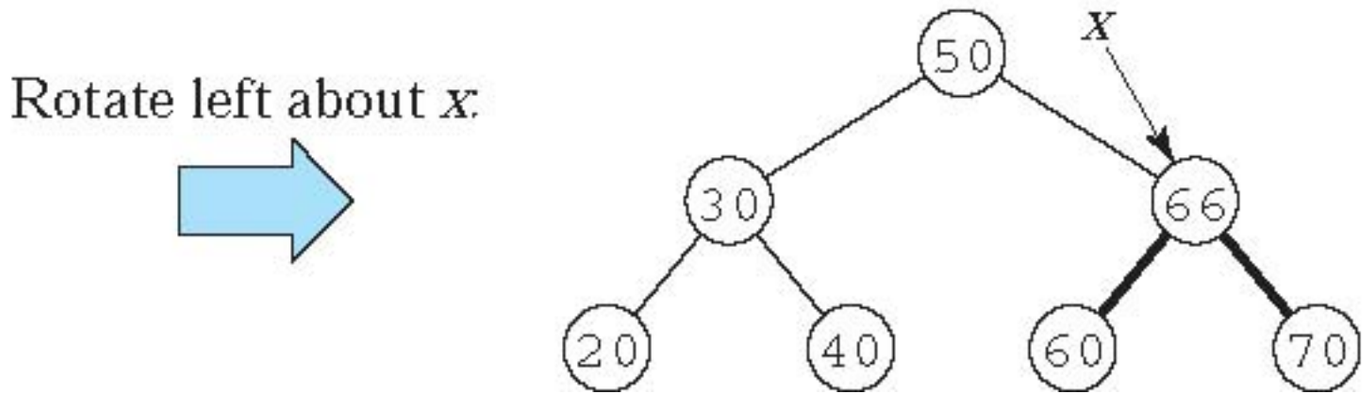


- 왼쪽 단순 회전

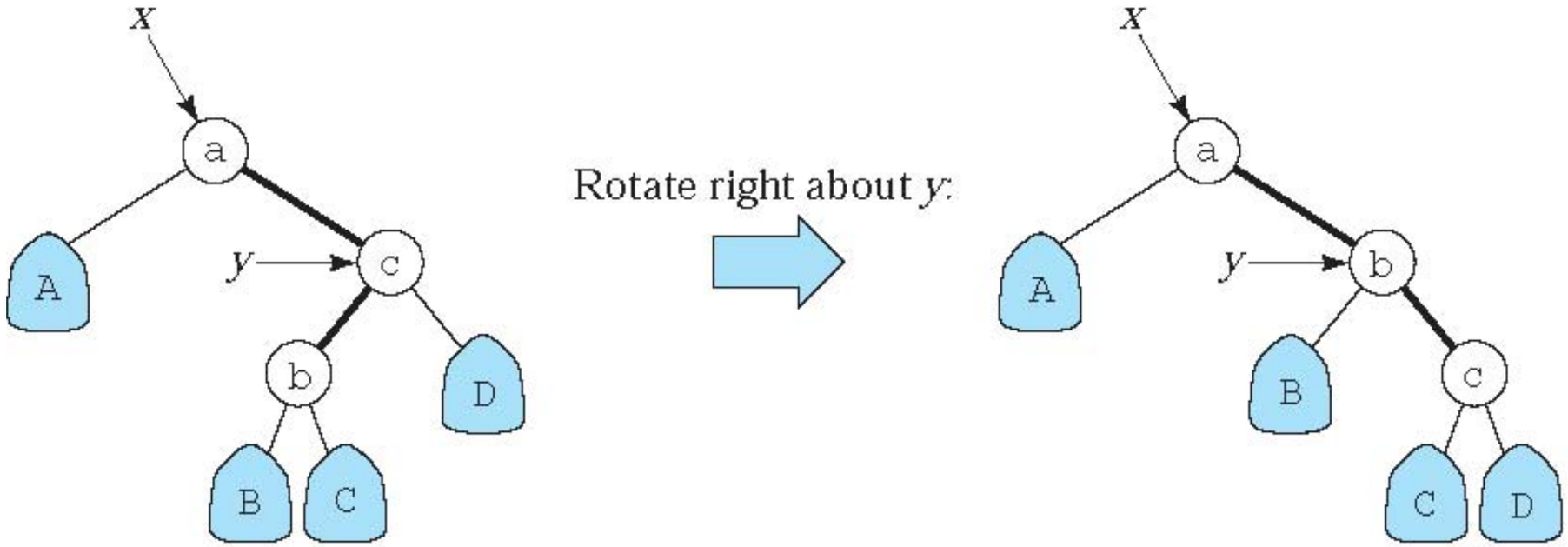

X
20
30
40
Rotate left about x:
X
30
20
40
X
30
20
40
50
60
Rotate left about x:
30
X
20
50
40
60
X
30
20
50
60
70
Rotate left about x:
X
30
20
50
40
60
70
X
a
A
b
B
c
C
D
Rotate left about x:
X
b
a
c
A
B
C
D

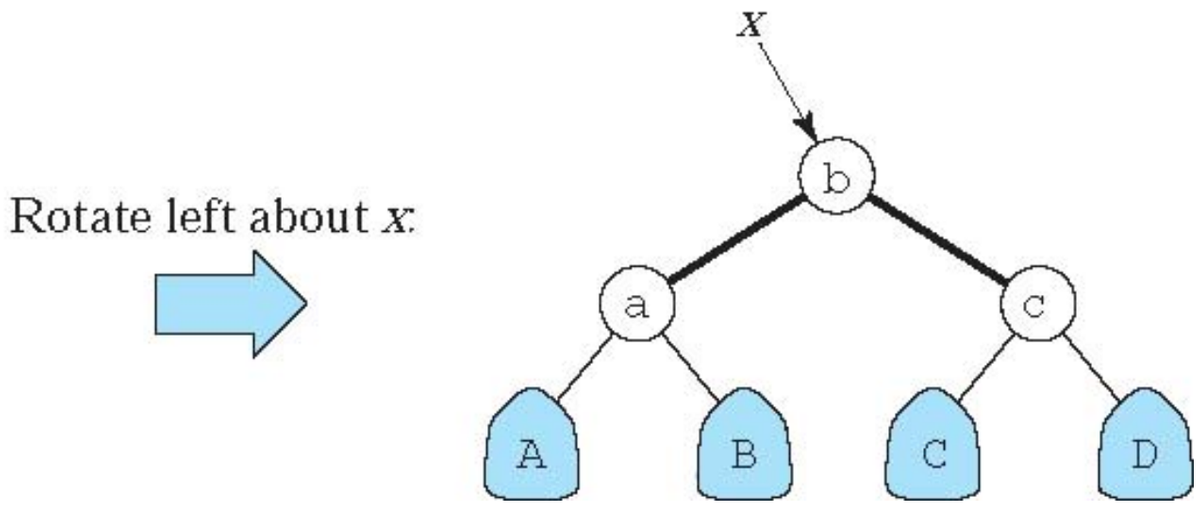
- 복합 오른쪽-왼쪽 회전


50
X
30
60
20
40
70
y
66
Rotate right about y:
50
X
30
60
20
40
66
y
70

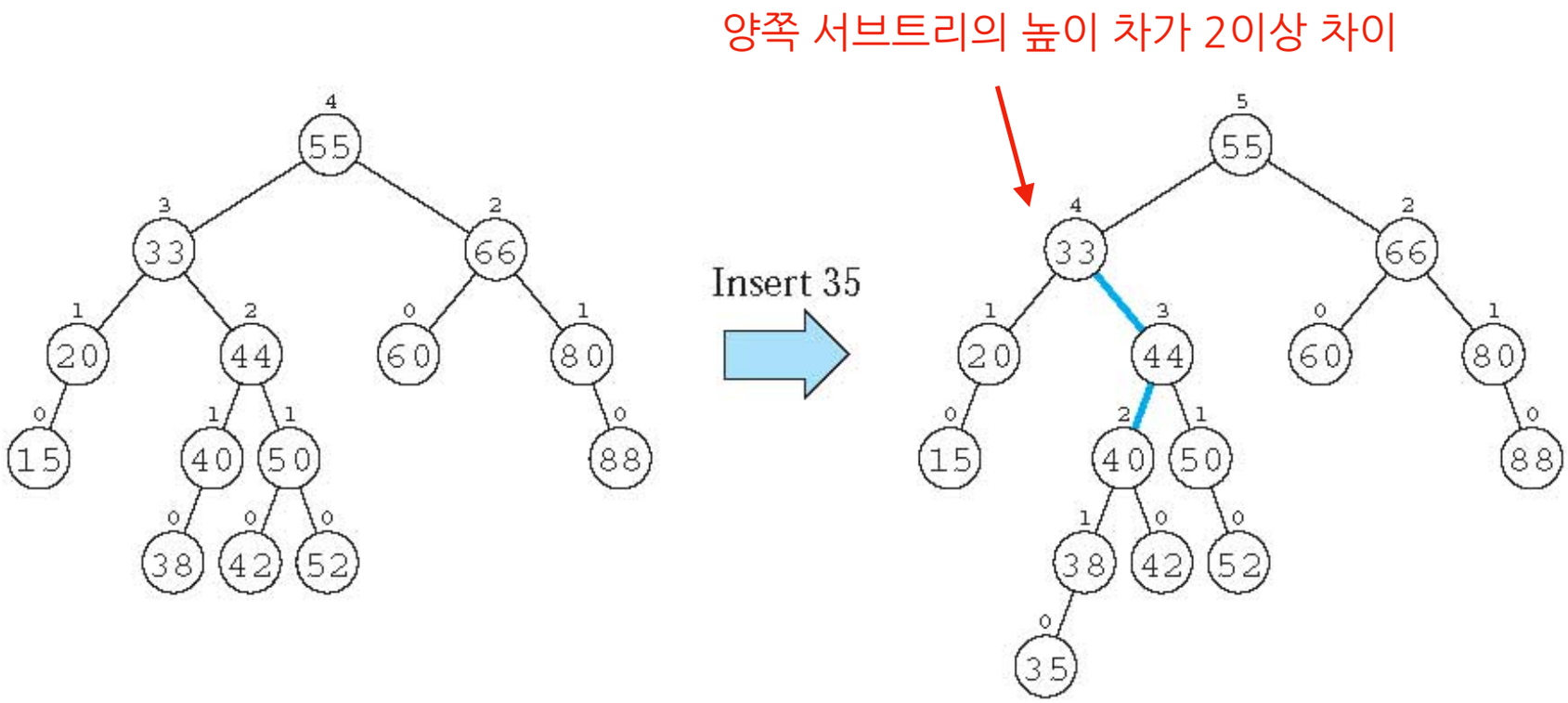

Rotate left about x:
50
X
30
66
20
40
60
70

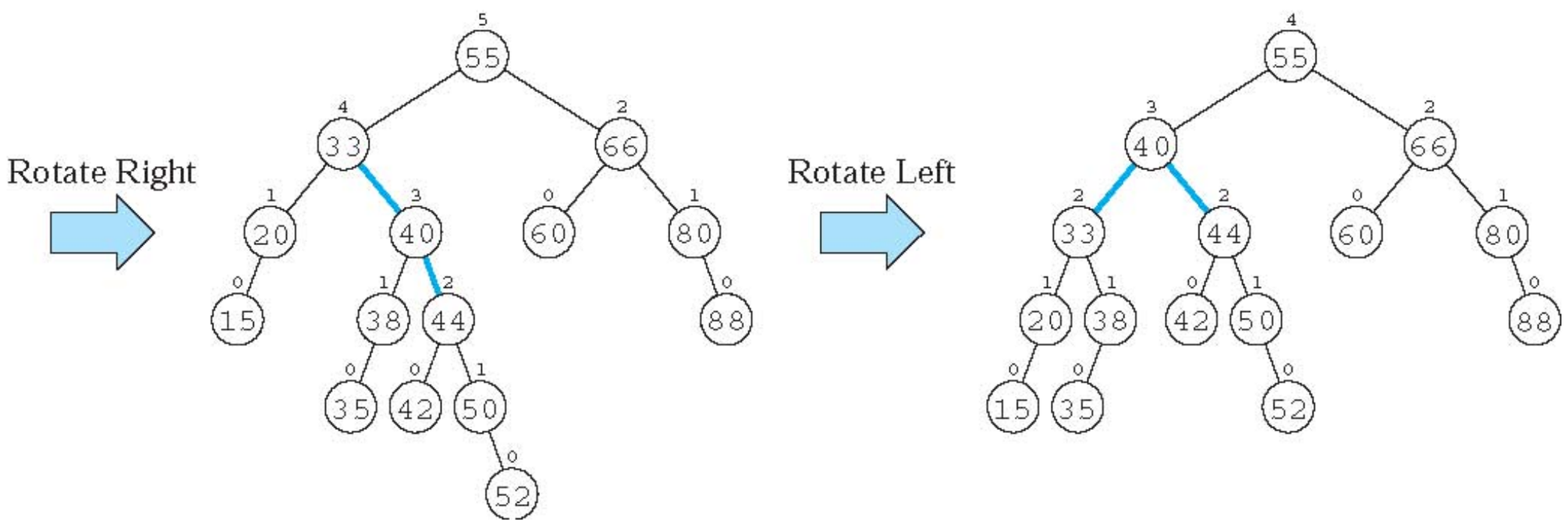
- 복합 오른쪽-왼쪽 회전


X
a
A
y
c
b
D
B
C
Rotate right about y:
X
a
A
y
b
B
c
C
D

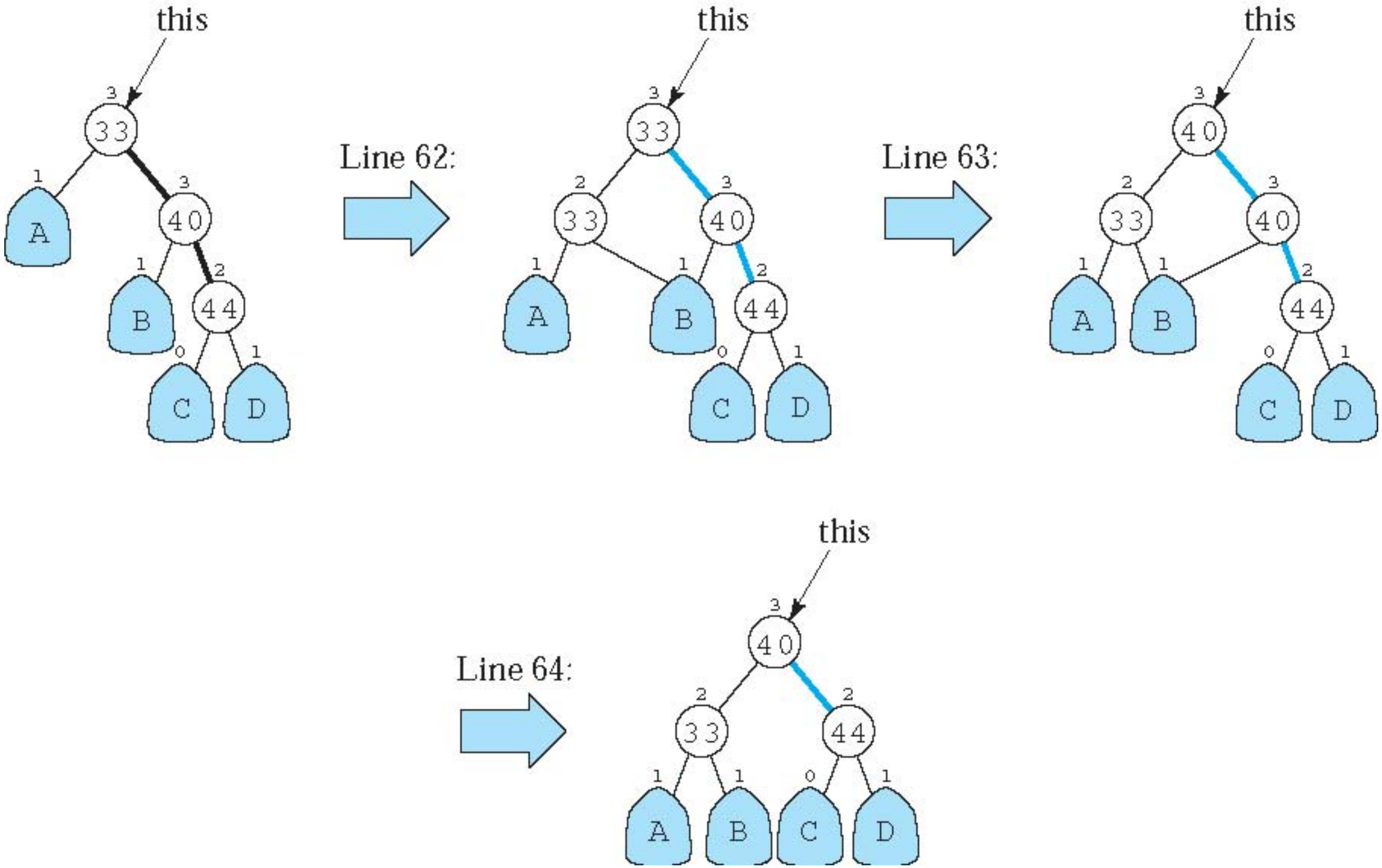

Rotate left about x:
X
b
a
c
A
B
C
D

- AVL트리 삽입


양쪽 서브트리의 높이 차가 2이상 차이
Insert 35
55
33
66
20
44
60
80
15
40
50
88
38
42
52
35


Rotate Right
Rotate Left
55
33
66
20
40
60
80
15
38
44
88
35
42
50
52
40
33
44
20
38
42
50
15
35
52

- AVL트리 삽입

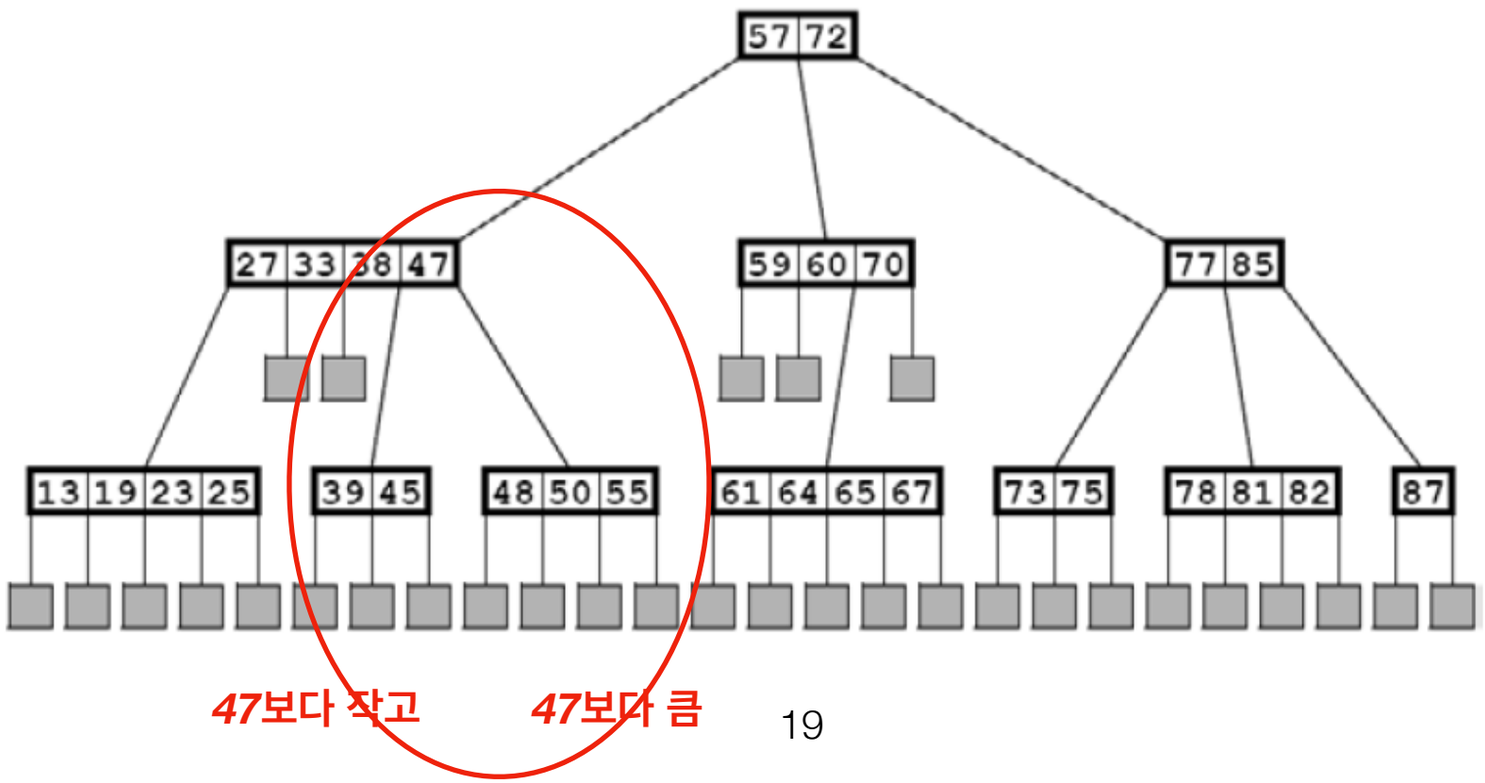

this
3
33
1
A
3
40
1
B
2
44
0
C
1
D
Line 62:
this
3
33
2
33
3
40
1
A
1
B
2
44
0
C
1
D
Line 63:
this
3
40
2
33
3
40
1
A
1
B
2
44
0
C
1
D
Line 64:
this
3
40
2
33
2
44
1
A
1
B
0
C
1
D

다원 탐색 트리(MST: Multiway Search Tree)

: 만일 T_i에 있는 각각의 x_i에 대해 $x_0, x_1, x_2, ..., x_{d-1}$을 서브트리에 있는 키라고 하면, $x_0<k_0<x_1<k_1<x_2<k_2<...<x_{d-2}<k_{d-2}<x_{d-1}$를 만족하는 트리.

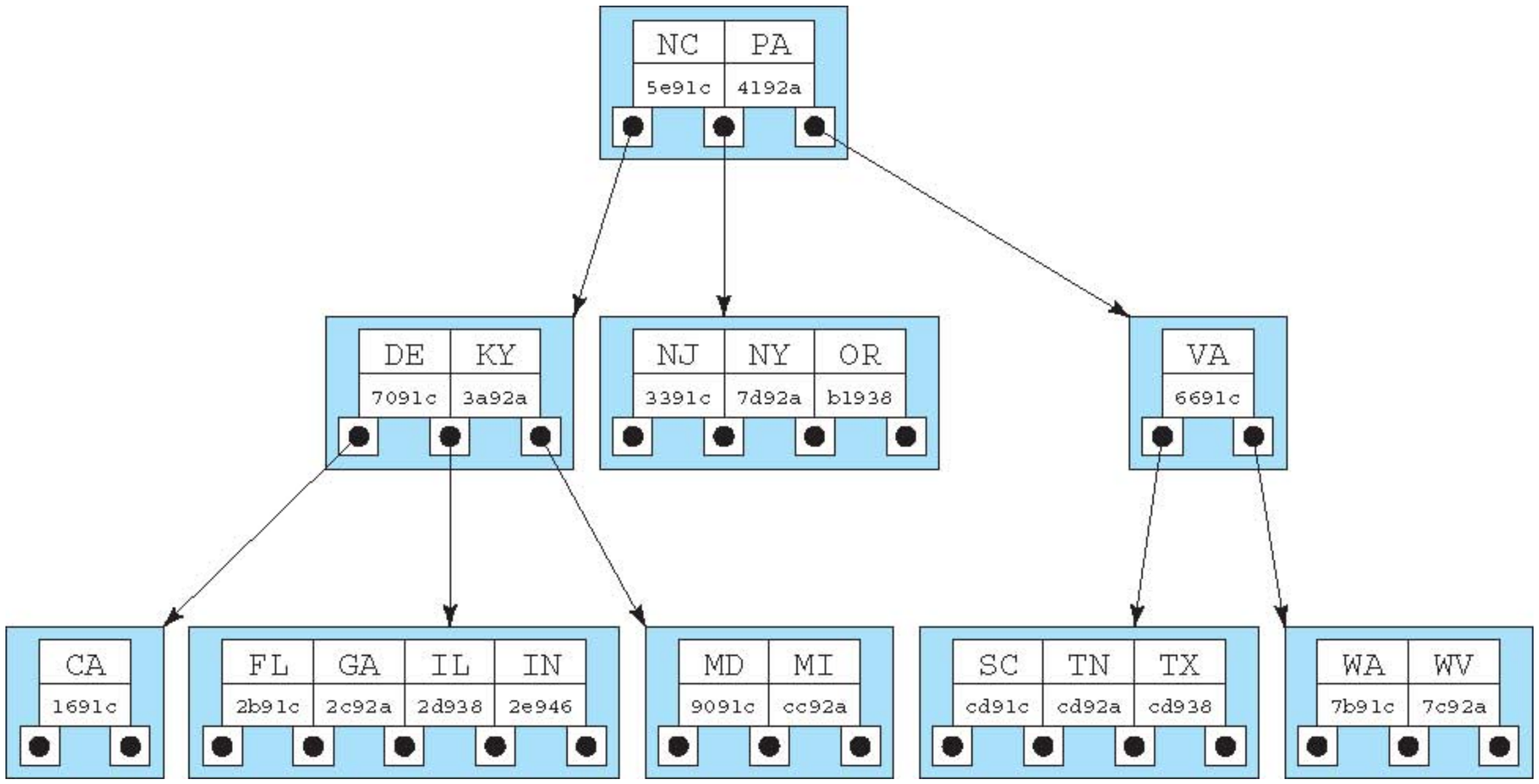
k는 (k_0, k_1, . . ., k_{d-2}) of n –1 keys

EX) 5-way search tree



◆차수가**5**인다원탐색트리

-> 가장 큰 애의 차수

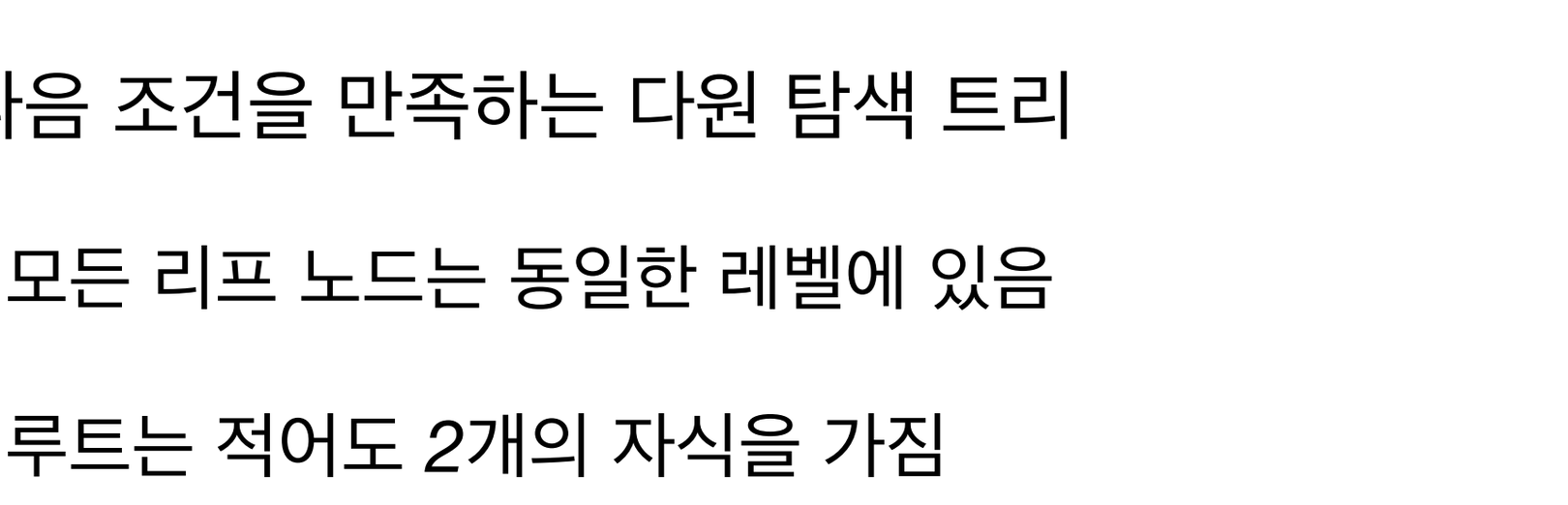


B-트리

: 다음 조건을 만족하는 다원 탐색 트리

- 모든 리프 노드는 동일한 레벨에 있음
- 루트는 적어도 2개의 자식을 가짐
- 루트가 아닌 모든 내부 노드는 최소한 $\lceil m/2 \rceil$ 의 차수를 가짐
  *(m*은 트리의 차수*)*

EX)



높이가 2이고
차수가 4인 B-트리

<참고> B-트리 장점

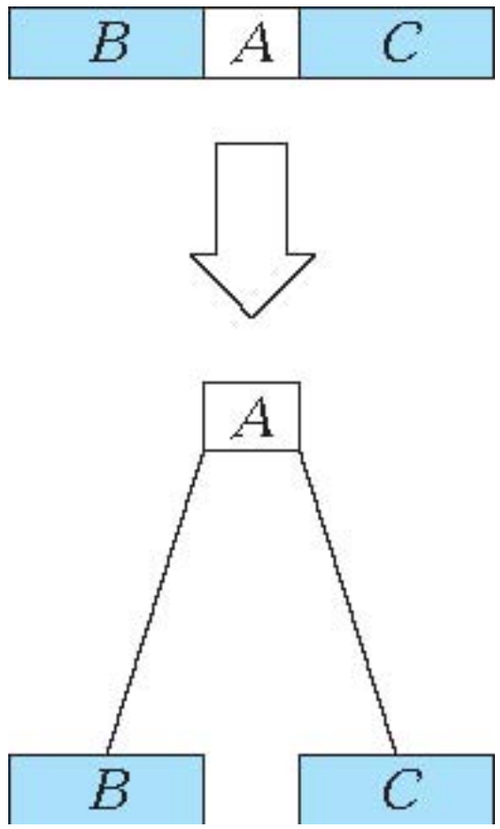
데이터베이스와 파일 시스템에서 널리 사용되는 트리 자료구조의 일종

- B-트리는 자료를 정렬된 상태로 보관하고, 삽입 및 삭제를 대수 시간으로 할 수 있다.
- B-트리는 노드 접근시간이 노드에서의 연산시간에 비해 훨씬 길 경우, 다른 구현 방식에 비해 상당한 이점을 가지고 있다. 이는 대부분의 노드가 하드디스크와 같은 2차 저장장치에 있을 때 일반적으로 일어난다. 각 내부 노드에 있는 자식 노드의 수를 최대화함으로써, 트리의 높이는 감소하며, 균형맞춤은 덜 일어나고, 효율은 증가하게 된다. 대개 이 값은 각 노드가 완전한 하나의 디스크 블록 혹은 2차 저장장치에서의 유사한 크기를 차지하도록 정해진다.

- B-트리 삽입

탐색을 통해 위치 찾아감. 노드가 꽉 찼을 경우 노드 분할

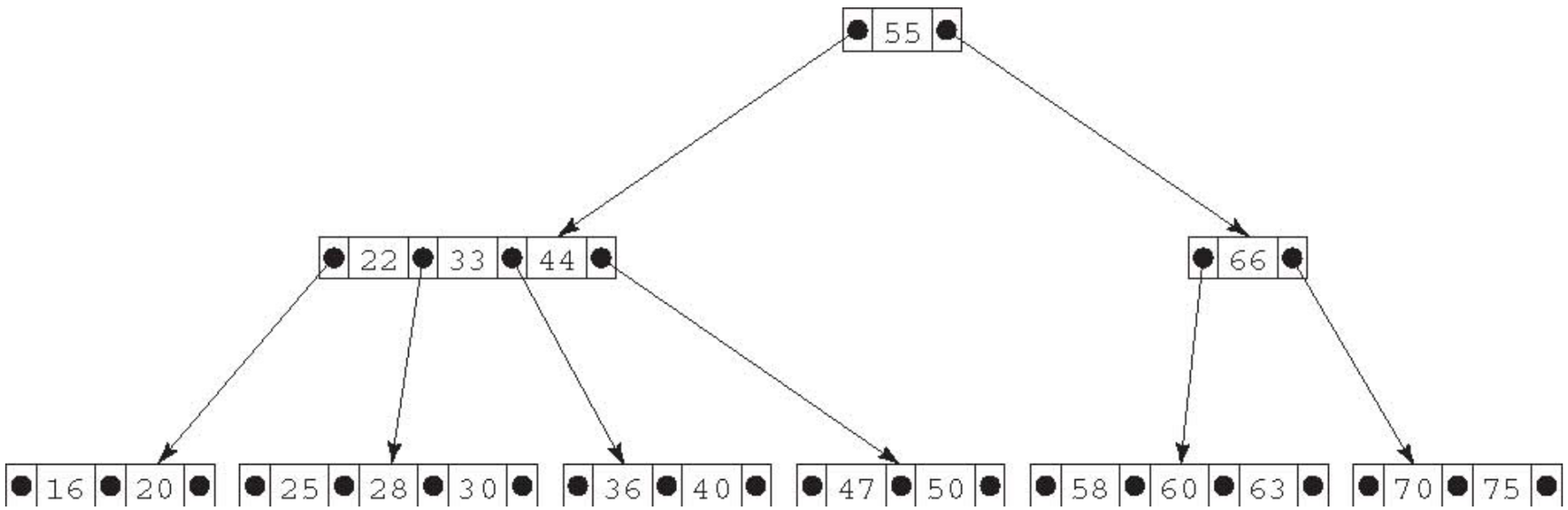
(원래 노드에 있는 키의 중앙값이 부모가 됨)



- B-트리 삭제

탐색을 통해 위치 찾아감.

삭제로 인해 한 노드의 자료수가 m/2보다 작으면 형제에게 빌리거나 형제와 결합



<2-3-4트리>

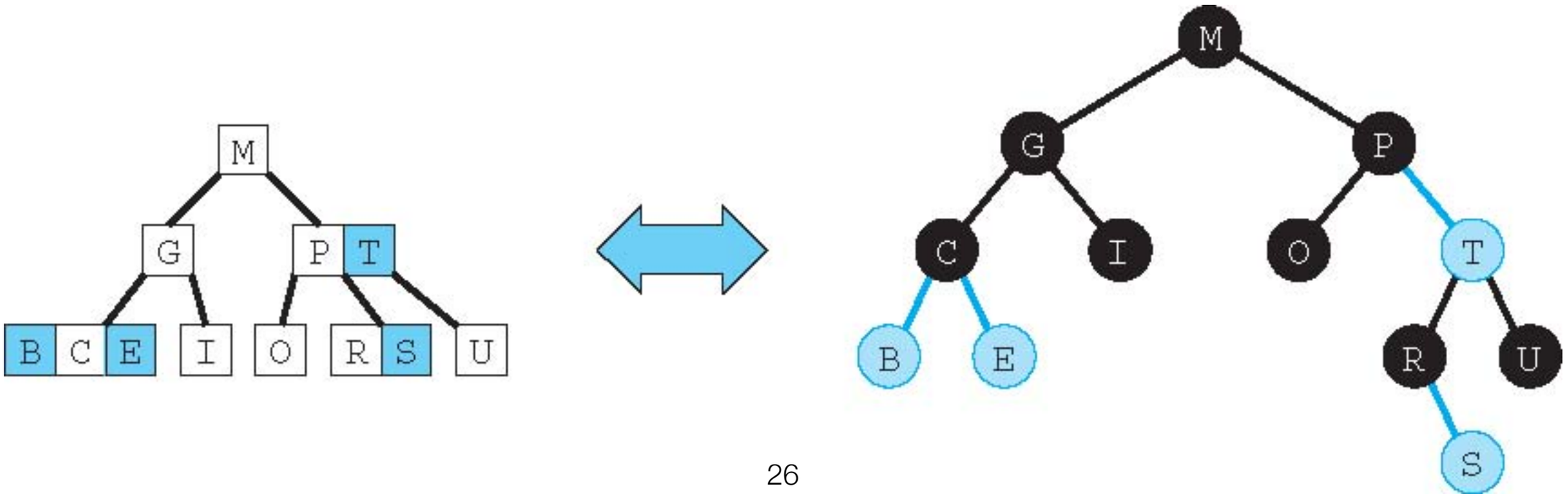
차수가 4인 B-트리는 각각의 내부 노드가 2, 3, 4개의 자식을 가지고 있기 때문에 2-3-4트리라고 불림. 3개의 자식을 가진 노드를 3-노드, 4개의 자식을 가진 노드를 4-노드라고 부름.

레드-블랙 트리

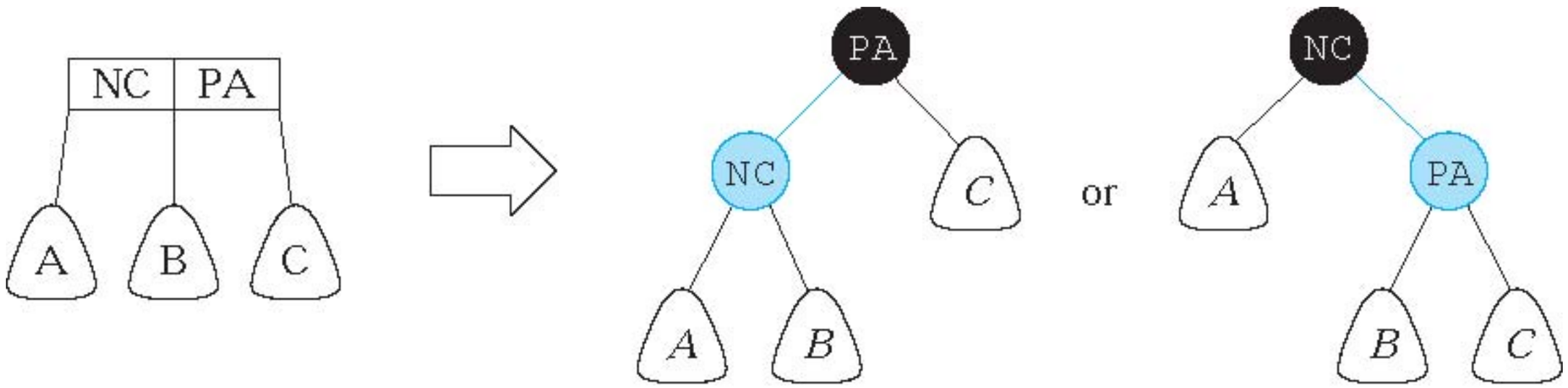
: 이진 트리의 일반적인 노드는 블랙으로 표현하고, 3-노드와 4-노드는 레드로 표현한 트리.

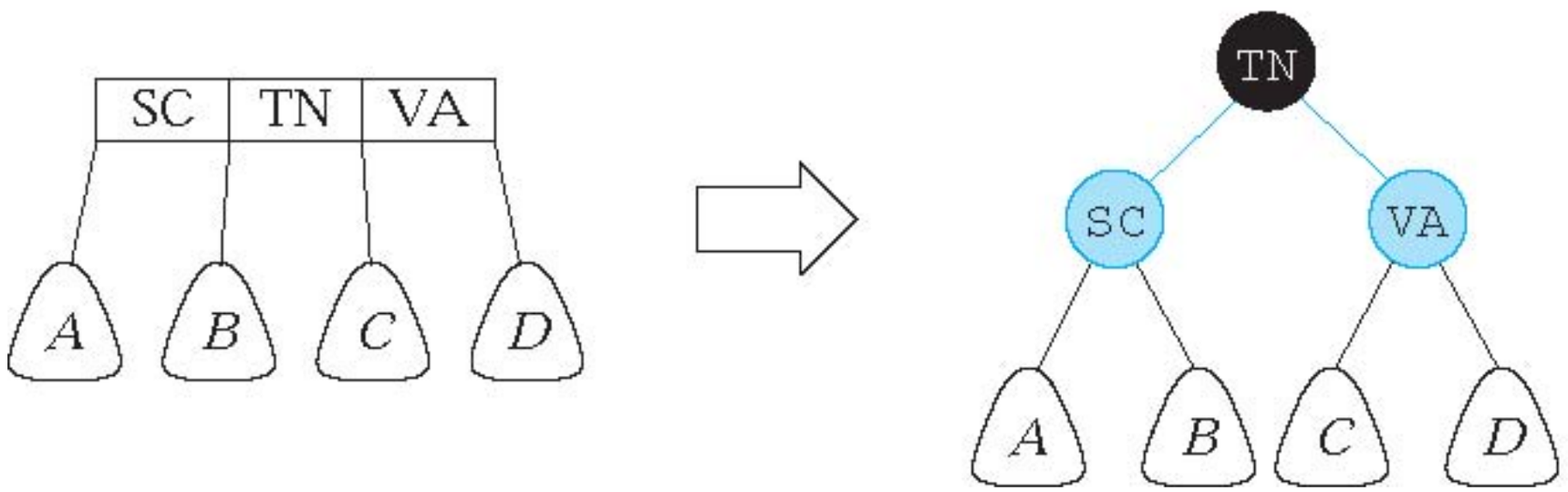
모든 단독 노드는 블랙이고, 3-노드와 4-노드는 레드-블랙 서브트리로 교체

EX) B-트리를 표현하는 레드-블랙 트리

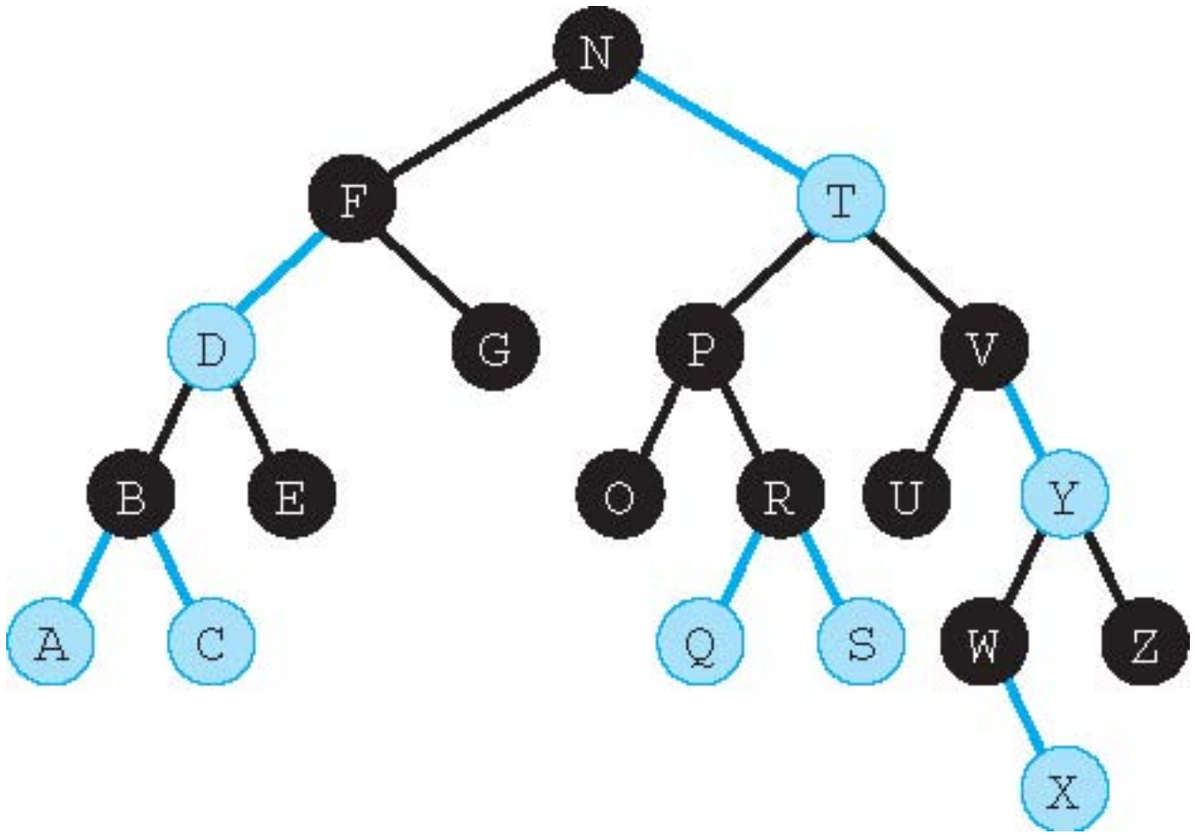


- 레드-블랙 트리로 변환 방법





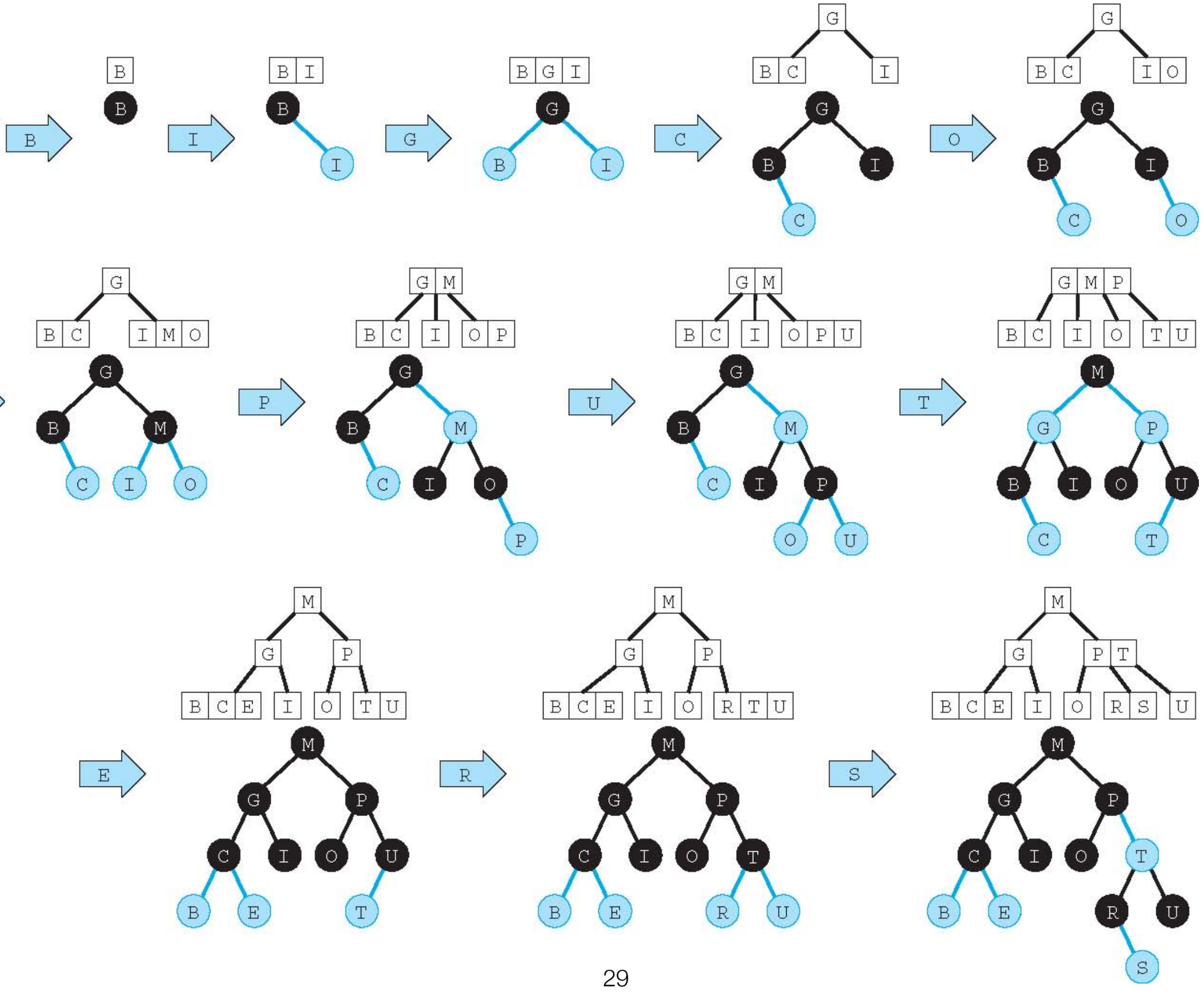
- 블랙 높이가 2인 레드-블랙 트리



- 레드-블랙 트리의 특성

1. 각각의 노드는 레드 또는 블랙이다.
2. 루트와 NIL 노드는 블랙이다.
3. 각각의 레드 노드의 두 자식은 블랙이다.
4. 모든 루트-리프 경로는 동일한 개수의 블랙 노드를 가지고 있다.

- 레드-블랙 트리 삽입


B
B
B I
B
I
B G I
G
B
I
C
G
B C
I
B
G
I
C
O
G
B C
I O
B
G
I
C
O
M
G
B C
I M O
G
B
M
C
I
O
P
G M
B C
I
O P
G
B
M
C
I
O
U
G M
B C
I
O P U
G
B
M
C
I
P
O
U
T
G M P
B C
I
O
T U
M
G
P
B
I
O
U
C
T
E
M
G
P
B C E
I
O
T U
C
I
O
U
B
E
T
R
M
G
P
B C E
I
O
R T U
C
I
O
T
B
E
R
U
S
M
G
P
B C E
I
O
R S
U
C
I
O
T
B
E
R
U
S