

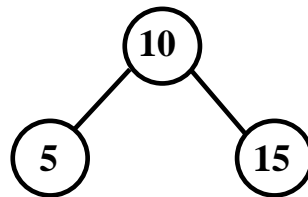
제 **10** 장

효율적인 이원 탐색 트리

최적 이원 탐색 트리(1)

◆ 개요

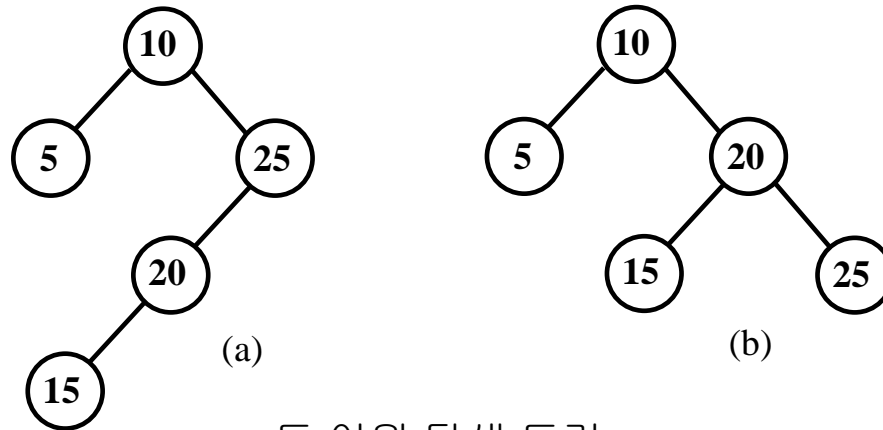
- 정적 원소들의 집합에 대한 이원 탐색트리 구조
 - ◆ 삽입이나 삭제는 하지 않고 탐색만 수행
- 정렬된 리스트
 - ◆ 함수 Get(프로그램 5.19 참고) 이용
- 비용 측정 방법
 - ◆ 함수 Get 이용 for 루프를 1번 반복



리스트(5,10,15)에서의 이원 탐색에 해당되는 이원 탐색 트리

최적 이원 탐색 트리(2)

◆ 예제 10.1



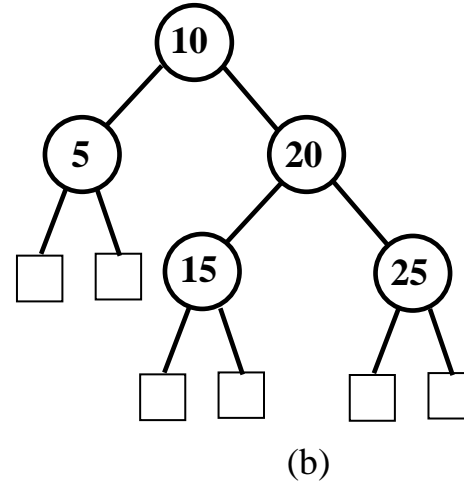
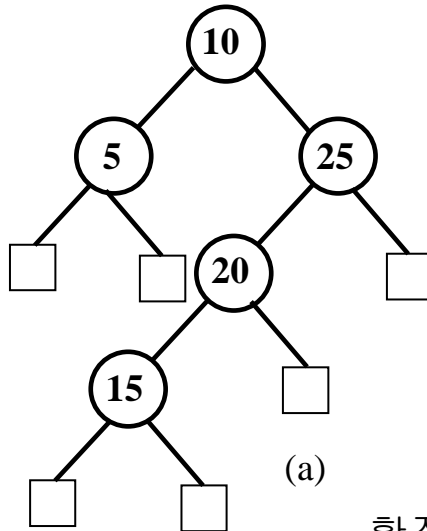
두 이원 탐색 트리

- 최악의 경우 탐색시간: (a) 4번, (b) 3번
- 성공적인 탐색에 필요한 평균: (a) 2.4번, (b) 2.2번
- 5,10,15,20,25가 각각 0.3,0.3,0.05,0.05,0.3의 확률로 탐색
: (a) 1.85, (b) 2.05

최적 이원 탐색 트리(3)

◆ 이원탐색트리의 평가

- 특별한 사각형 노드(square node) 추가시 유용

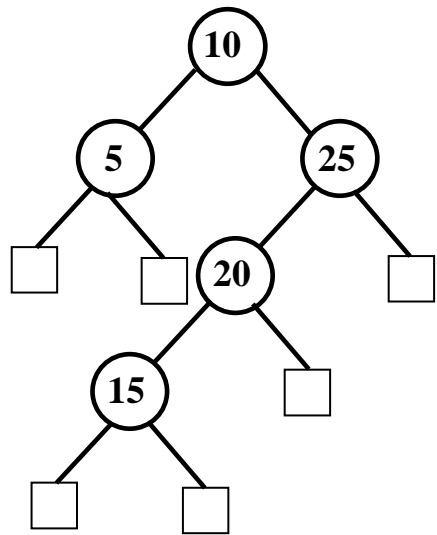


확장 이진 트리

- 외부노드(실패노드) : $n+1$ 개의 사각노드
- 내부노드 : 원래 노드

최적 이원 탐색 트리(4)

- ◆ 외부경로길이(**external path length**)
 - 루트로부터 각 외부노드 경로 길이의 합
- ◆ 내부경로길이(**internal path length**)
 - 루트로부터 각 내부노드 경로 길이의 합



• 내부경로길이(I) = $0 + 1 + 1 + 2 + 3 = 7$

• 외부경로길이(E) = $2 + 2 + 4 + 4 + 3 + 2 = 17$

• I 와 E 의 관례(n 개의 내부노드를 가질 경우)

$$E = I + 2n$$

E 가 최고일때 I 도 최고

최적 이원 탐색 트리(5)

◆ I의 최소값

- 최악의 경우: 트리가 편향될 때

$$I = \sum_{j=0}^{n-1} i = n(n-1)/2$$

- 일반적인 경우

$$0 + 2 \times 1 + 4 \times 2 + 8 \times 3 + \dots +$$

- 완전 이진 트리일 경우

$$i = \sum_{1 \leq i \leq n} \log_2 i = O(n \log_2 n)$$

최적 이원 탐색 트리(6)

◆ 정적 원소 집합을 이원탐색 트리로 표현

- 탐색이 성공적일때 비용

$$\sum_{1 \leq i \leq n} p_i * \text{level}(a_i) \quad (a_i \text{ 를 탐색할 확률이 } p_i \text{ 일 경우})$$

- 탐색이 실패적일때 비용

$$\sum_{1 \leq i \leq n} q_i * (\text{level}(\text{실패노드 } i) - 1)$$

(탐색중인 원소가 E_i 에 있을 확률이 q_i 일 경우)

- 이원탐색트리의 총 비용

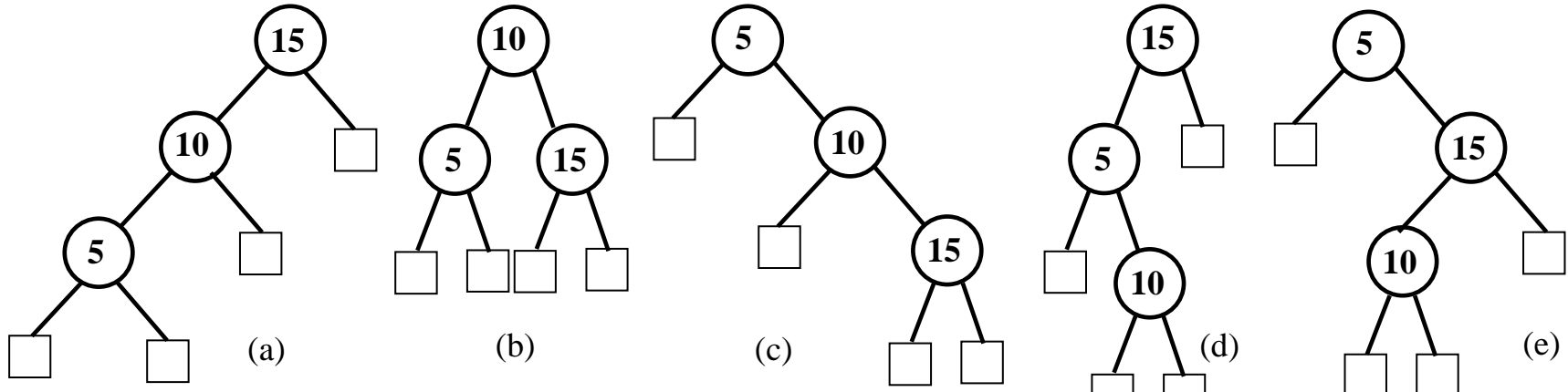
$$\sum_{1 \leq i \leq n} p_i * \text{level}(a_i) + \sum_{1 \leq i \leq n} q_i * (\text{level}(\text{실패노드 } i) - 1)$$

- 최적 이원탐색트리의 성립

$$\sum_{1 \leq i \leq n} p_i + \sum_{1 \leq i \leq n} q_i = 1$$

최적 이원 탐색 트리(7)

◆ 예제 10.2



3개의 원소를 가진 이원 탐색 트리

– 똑같은 확률 $p_i=q_i=1/7$ 일 경우

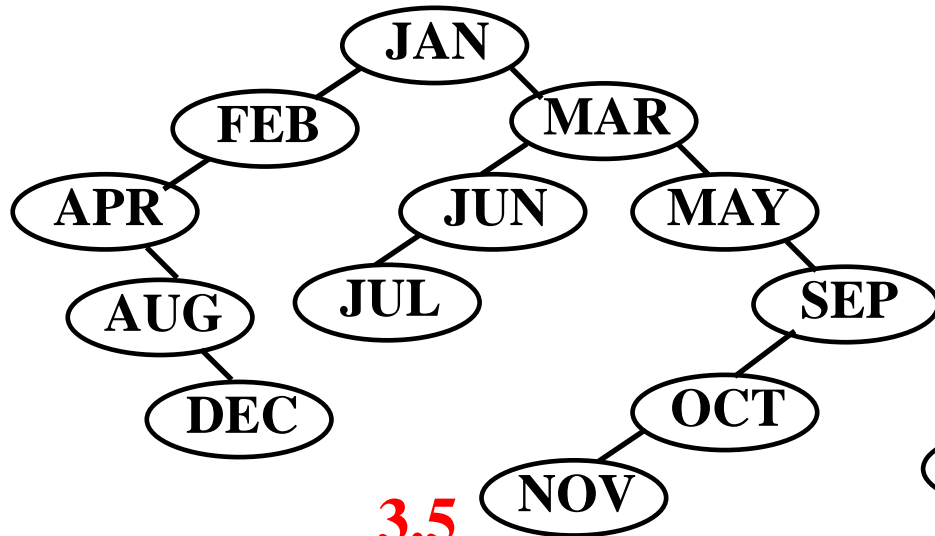
- ◆ $\text{cost}(\text{tree a}) = 15/7$; **$\text{cost}(\text{tree b}) = 13/7$**
 $\text{cost}(\text{tree c}) = 15/7$; $\text{cost}(\text{tree d}) = 15/7$; $\text{cost}(\text{tree e}) = 15/7$

– $p_1=0.5, p_2=0.1, p_3=0.05, q_0=0.15, q_1=0.1, q_2=0.05, q_3=0.05$

- ◆ $\text{cost}(\text{tree a}) = 2.65$; $\text{cost}(\text{tree b}) = 1.9$
 $\text{cost}(\text{tree c}) = 1.5$; $\text{cost}(\text{tree d}) = 2.05$; $\text{cost}(\text{tree e}) = 1.6$

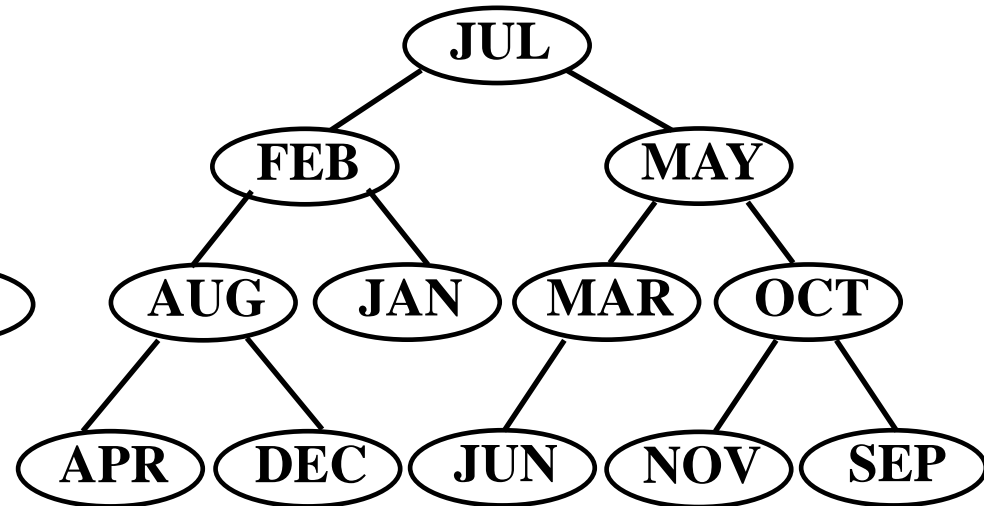
AVL 트리(1/3)

◆ 어떤 원소를 찾기 위한 최대 비교 횟수



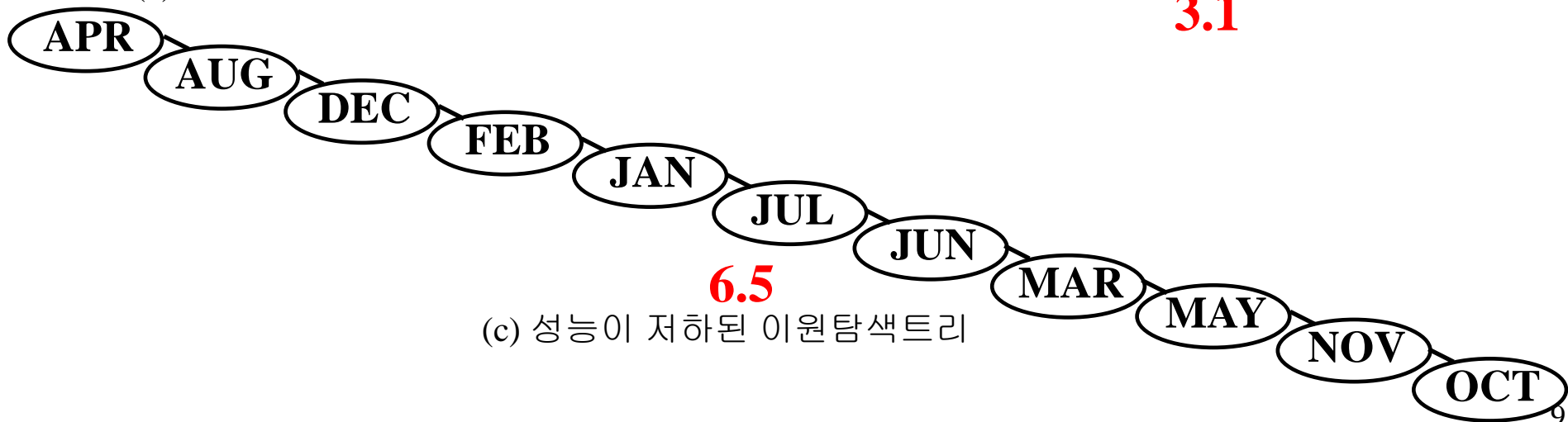
(a) 1년 12달에 대한 이원탐색 트리

3.5



(b) 1년 12달에 대한 균형트리

3.1



(c) 성능이 저하된 이원탐색트리

6.5

AVL 트리(2/3)

◆ Definition

- An empty tree is height-balanced.
- If T is a nonempty binary tree with T_L and T_R as its left and right subtrees respectively, then T is height-balanced iff
 - (1) T_L and T_R are height-balanced and
 - (2) $|h_L - h_R| \leq 1$ where h_L and h_R are the heights of T_L and T_R , respectively.

◆ 앞 장에서

- (a), (c)는 높이균형을 이루지 못하는 반면
- (b)는 높이균형을 이룸

AVL 트리(3/3)

◆ 균형인수(balance factor)

- 왼쪽, 오른쪽 서브트리 사이의 높이 차
- $BF(T) = h_L - h_R$
- AVL 트리의 어떠한 노드 T에 대해서도 $BF(T) = -1, 0, 1$

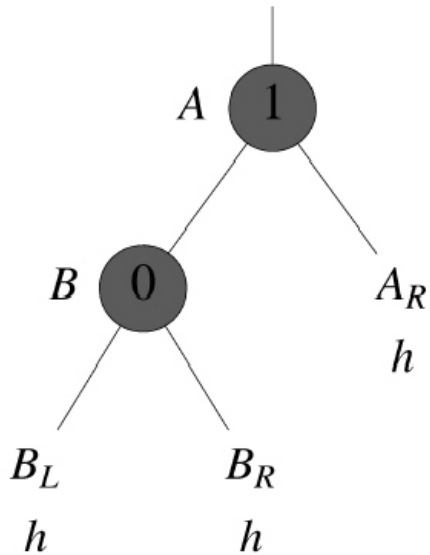
◆ 삽입된 노드 Y에 가장 가까우면서 균형인수가 ± 2 인 nearest ancestor A에 대한 회전 성질

- LL: 새 노드 Y는 A의 왼쪽 서브트리의 왼쪽 서브트리에 삽입
- LR: Y는 A의 왼쪽 서브트리의 오른쪽 서브트리에 삽입
- RR: Y는 A의 오른쪽 서브트리의 오른쪽 서브트리에 삽입
- RL: Y는 A의 오른쪽 서브트리의 왼쪽 서브트리에 삽입

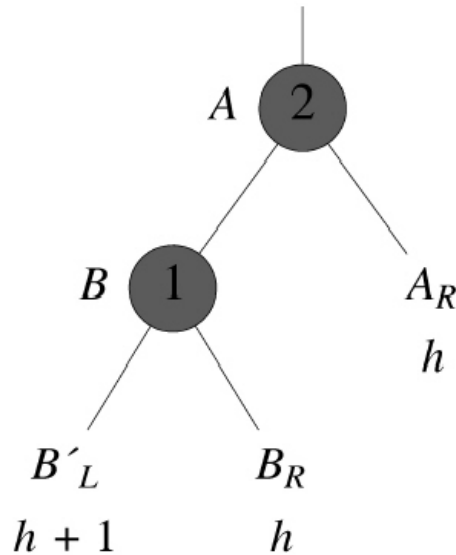
◆ 단일회전(Single rotation) : LL과 RR 불균형을 바로잡는 변환

◆ 이중회전(Double rotation) : LR과 RL 불균형을 바로잡는 변환

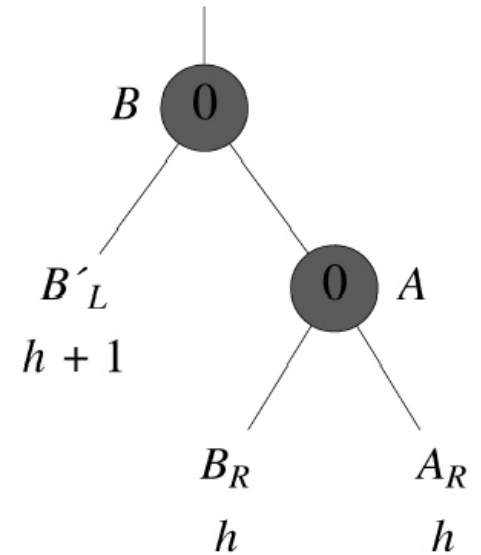
LL 회전



삽입 전



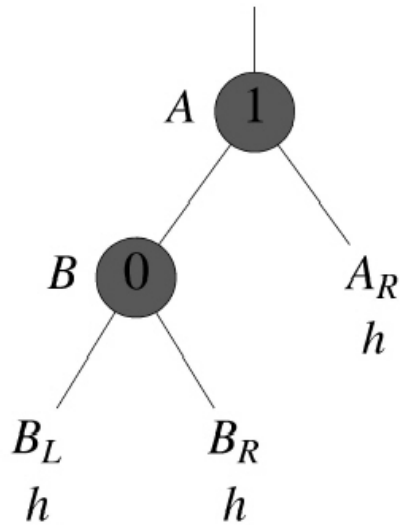
B_L에 삽입한 뒤



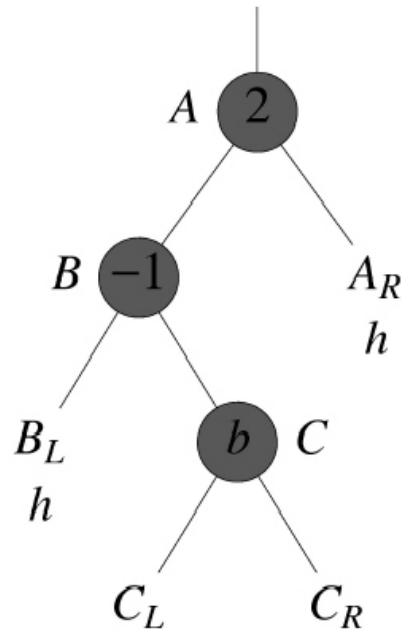
LL 회전 뒤

균형 인수는 노드 안에 있음
서브트리 높이는 서브트리 이름 밑에 있음

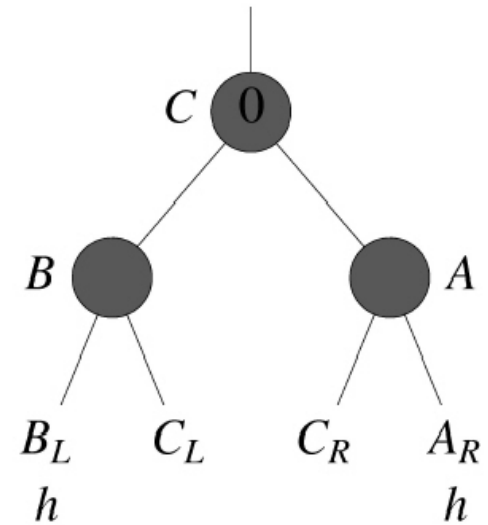
LR 회전



삽입 전



B_R 에 삽입한 뒤



LR 회전 뒤

$b = 0 \Rightarrow \text{bf}(B) = \text{bf}(A) = 0$ 회전 뒤

$b = 1 \Rightarrow \text{bf}(B) = 0$ and $\text{bf}(A) = -1$ 회전 뒤

$b = -1 \Rightarrow \text{bf}(B) = 1$ and $\text{bf}(A) = 0$ 회전 뒤

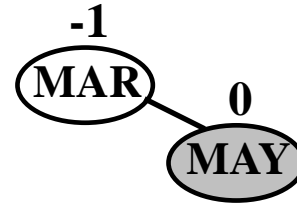
트리 확장 및 균형유지 과정(1/5)

◆ 삽입순서

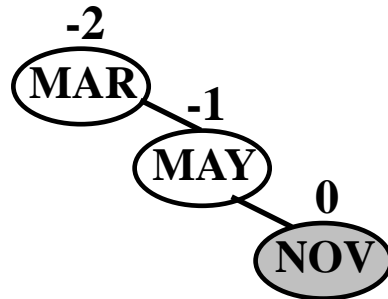
- MAR, MAY, NOV, AUG, APR, JAN, DEC, JUL, FEB, JUN
OCT, SEP 순



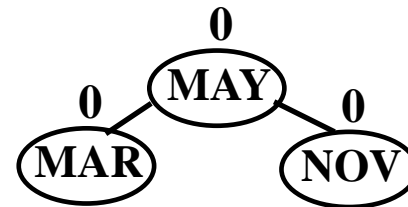
(a) 삽입 MARCH



(b) 삽입 MAY

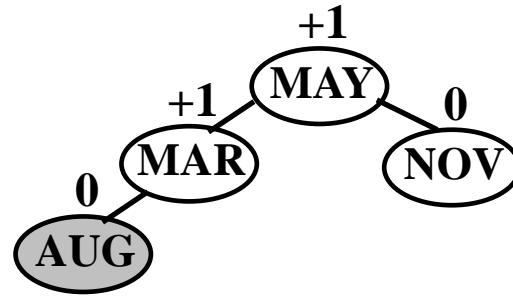


RR →

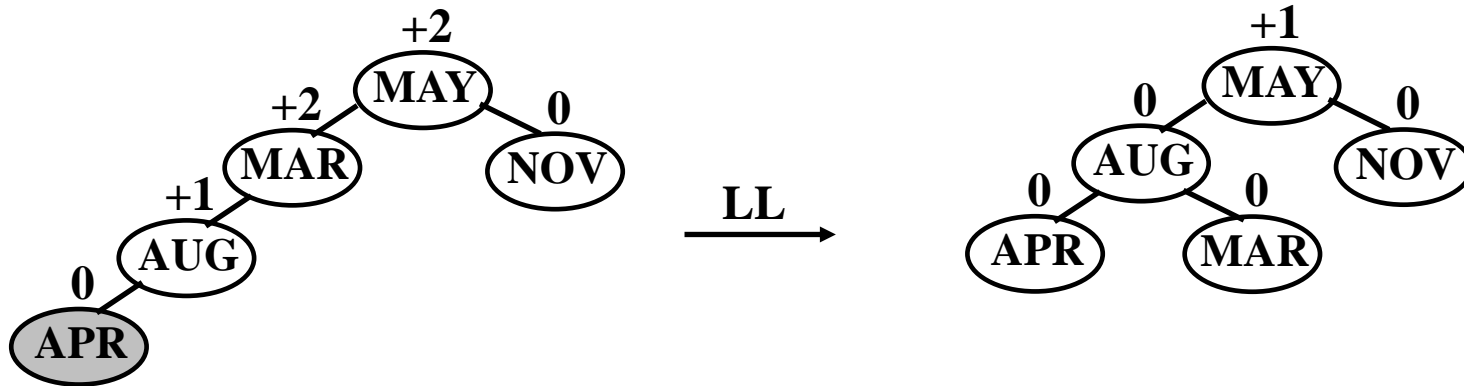


(c) 삽입 NOV

트리 확장 및 균형유지 과정(2/5)

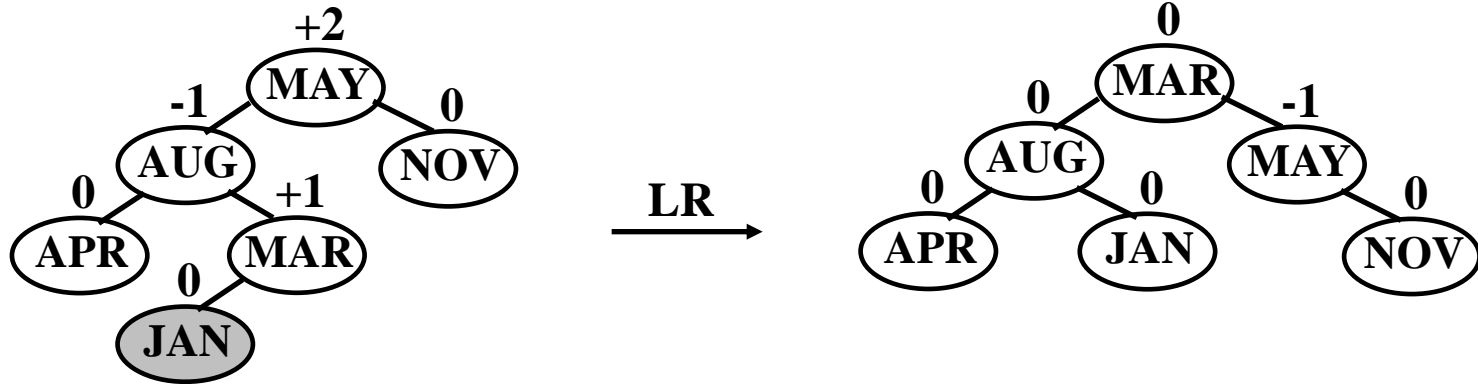


(d) 삽입 AUGUST

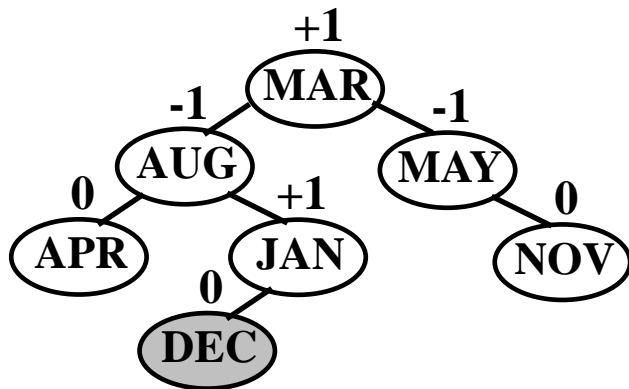


(e) 삽입 APRIL

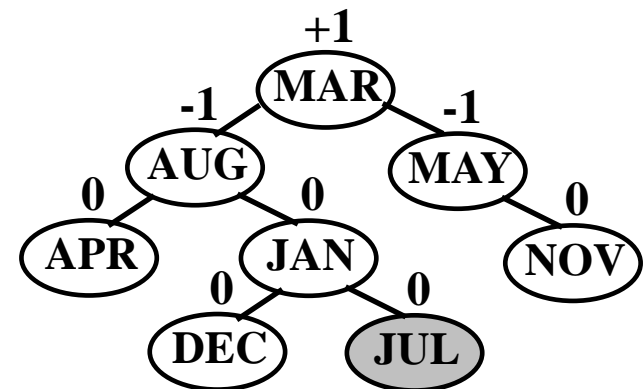
트리 확장 및 균형유지 과정(3/5)



(e) 삽입 JANUARY

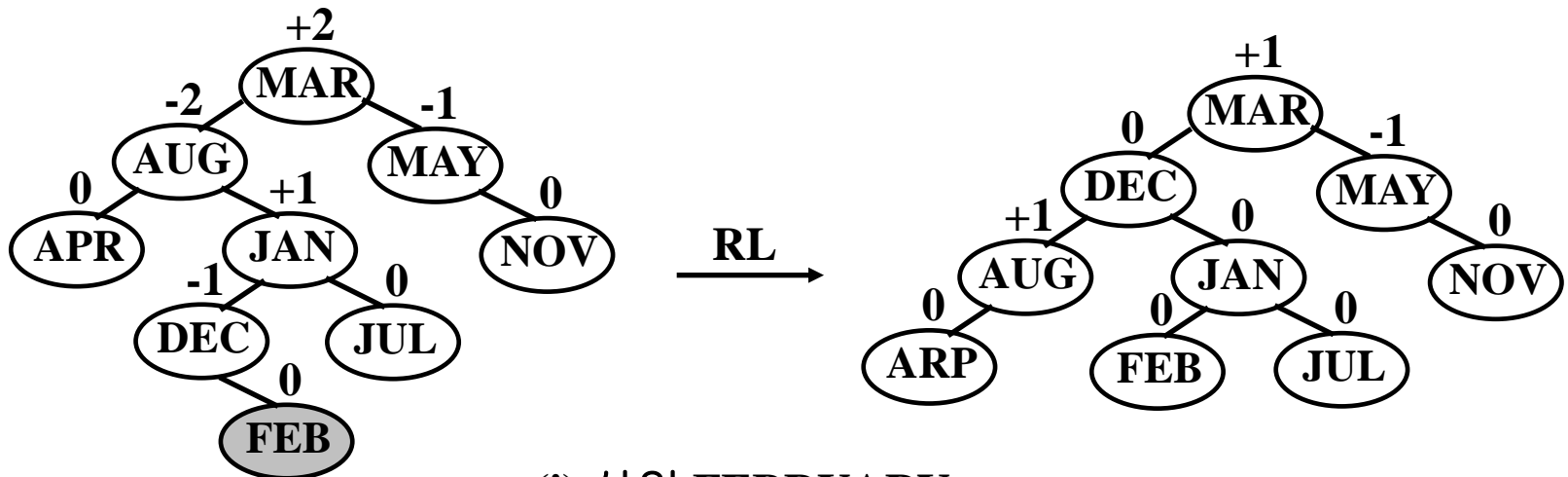


(g) 삽입 DECEMBER

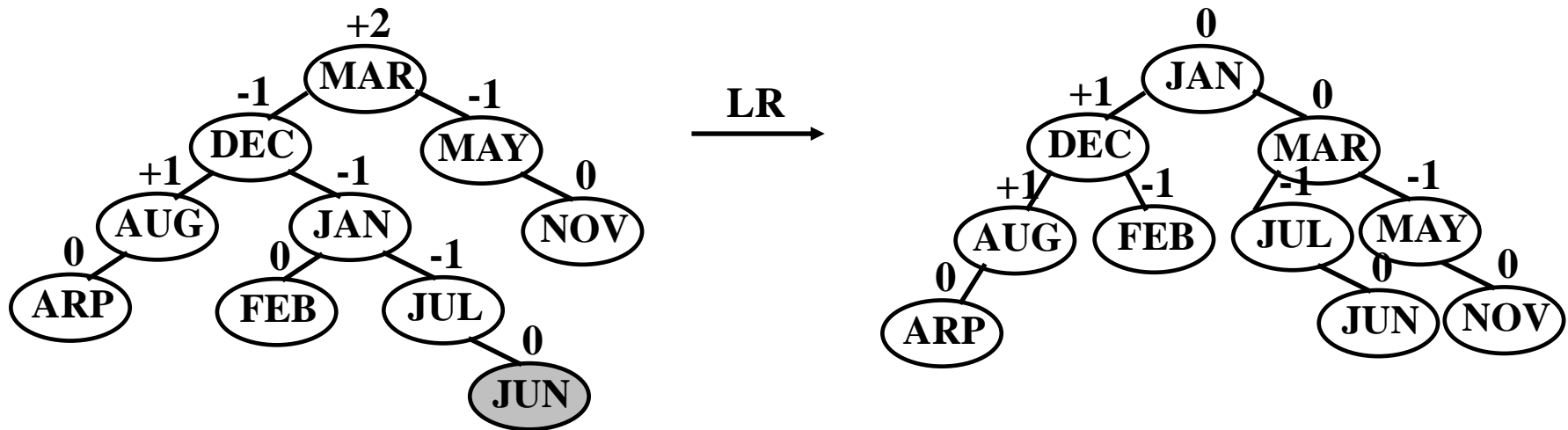


(h) 삽입 JULY

트리 확장 및 균형유지 과정(4/5)

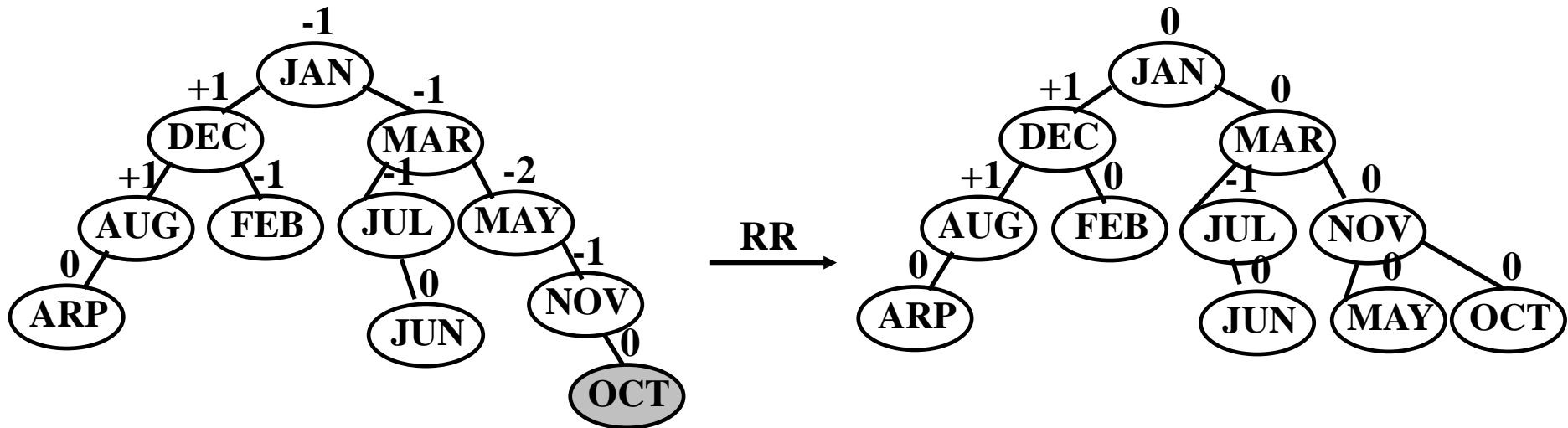


(i) 삽입 FEBRUARY

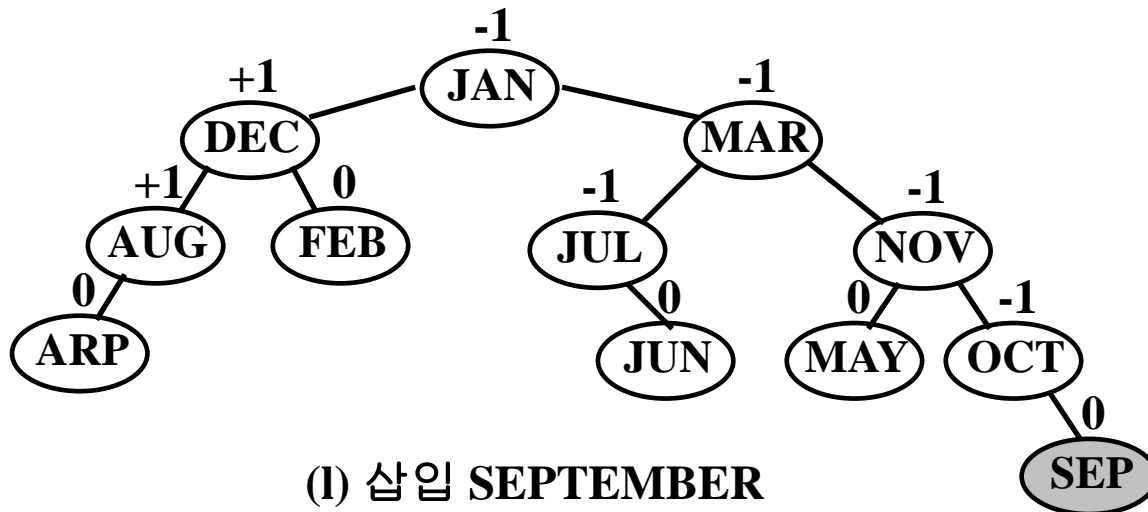


(j) 삽입 JUNE

트리 확장 및 균형유지 과정(5/5)



(k) 삽입 OCTOBER



(l) 삽입 SEPTEMBER

AVL 트리에서의 삽입

◆ 여러 구조들의 비교

연 산	순차리스트	연결리스트	AVL트리
키가 k인 원소탐색	$O(\log n)$	$O(n)$	$O(\log n)$
j번째 원소탐색	$O(1)$	$O(j)$	$O(\log n)$
키가 k인 원소삭제	$O(n)$	$O(1)^1$	$O(\log n)$
j번째 원소삭제	$O(n - j)$	$O(j)$	$O(\log n)$
삽입	$O(n)$	$O(1)^2$	$O(\log n)$
순서대로 출력	$O(n)$	$O(n)$	$O(n)$

1. k의 위치가 알려진 이중 연결 리스트
2. 삽입할 위치가 알려진 경우