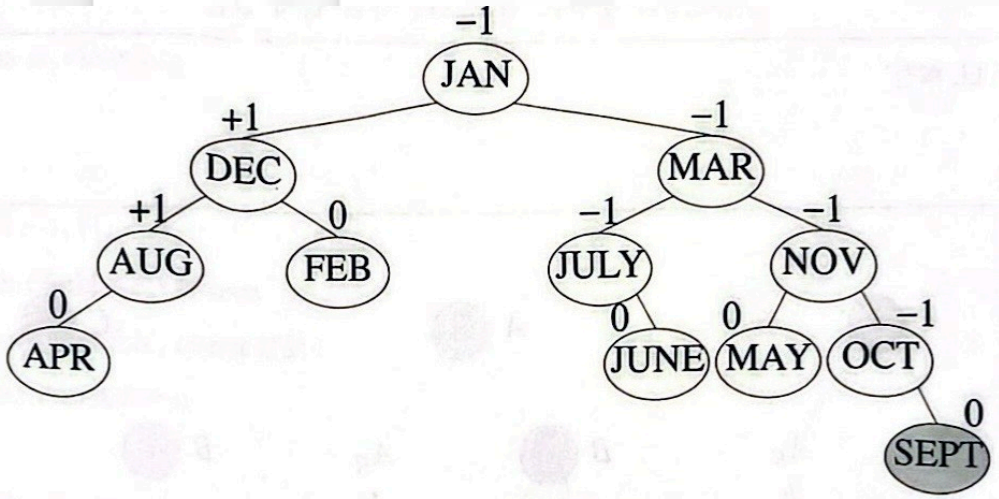

Pervasive Elastic MetaLearning Laboratory
Department of Computer Engineering
Hongik University

# HW10 실습

AVL Tree

# AVL Tree

- 서브트리들의 높이가 균형을 이루는 이진 트리 구조
- n개의 노드를 가진 트리에 대해 동적 검색, 새로운 키 삽입이나 삭제 → O(log n) 시간 안에 할 수 있다



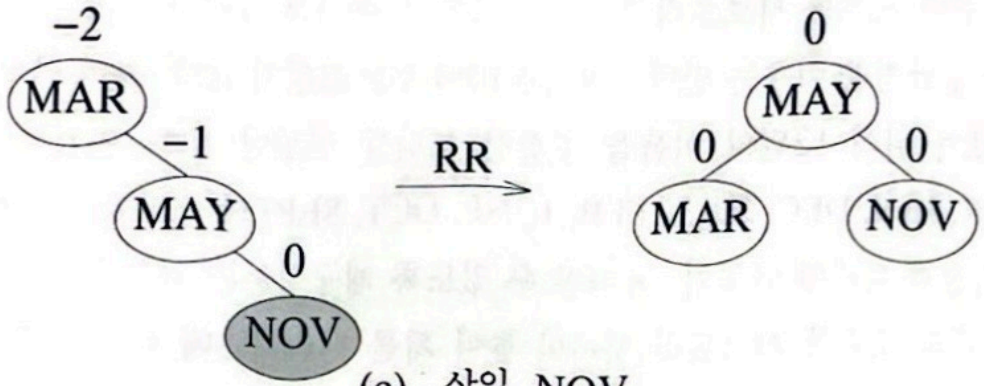
# Balance Factor (BF)

- 왼쪽 서브트리의 높이에서 오른쪽 서브트리의 높이를 뺀 값

- BF = 1 : 왼쪽 서브트리가 오른쪽 서브트리보다 높이가 한단계 높다
- BF = 0 : 왼쪽 서브트리가 오른쪽 서브트리의 높이가 같다
- BF = -1 : 왼쪽 서브트리가 오른쪽 서브트리보다 높이가 한단계 낮다

# 회전 rotation

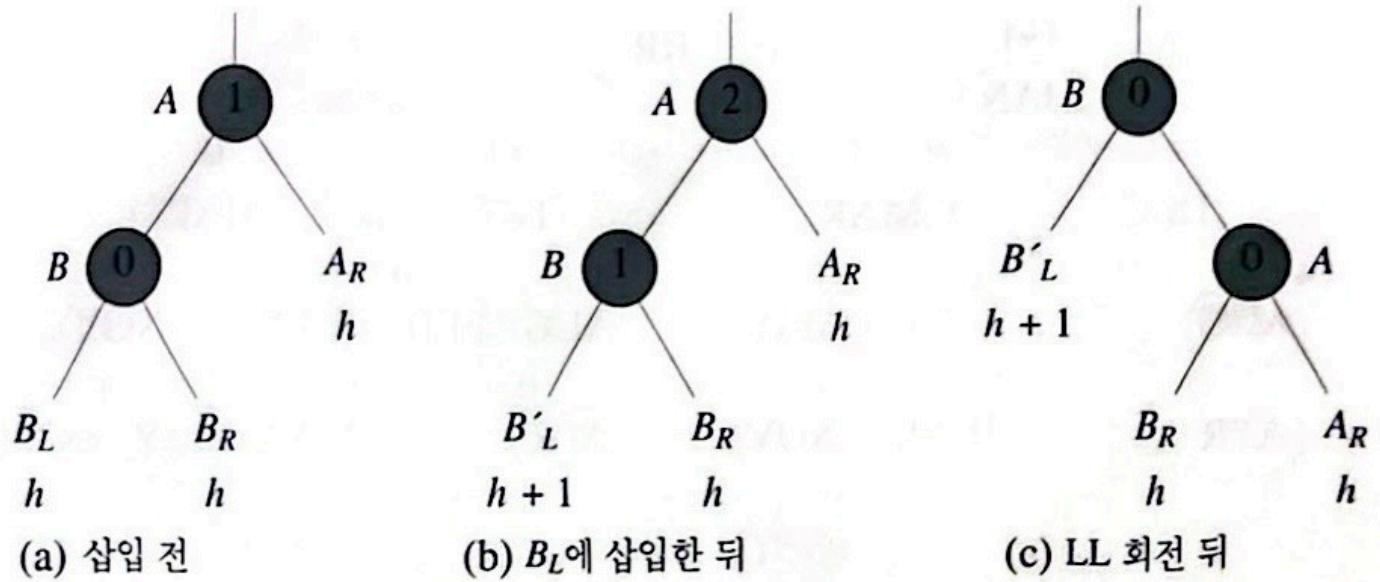
- 삽입, 삭제 시 불균형 (BF가 -1, 0, 1이 아닌 경우) 상태가 되면 AVL 트리는 불균형 노드를 기준으로 서브 트리의 위치를 변경하는 회전 작업을 수행하여 트리의 균형을 맞춘다



(c) 삽입 NOV

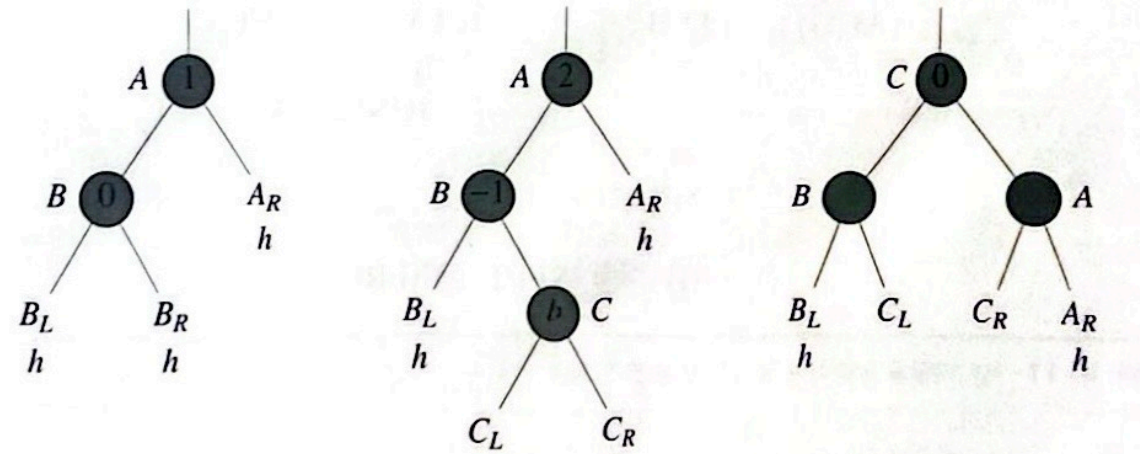
# Single rotation

- LL, RR 불균형을 바로잡는 변환
- LL : 새 노드 Y가 A의 왼쪽 서브트리의 왼쪽 서브트리에 삽입된 경우 ⇒ right rotation
- RR : Y가 A의 오른쪽 서브트리의 오른쪽 서브트리에 삽입된 경우 ⇒ left rotation



# Double rotation

- LR, RL 불균형을 바로잡는 변환
- LR : Y가 A의 왼쪽 서브트리의 오른쪽 서브트리에 삽입된 경우
  ⇒ left, right 순으로 두번 rotation
- RL : Y가 A의 오른쪽 서브트리의 왼쪽 서브트리에 삽입된 경우
  ⇒ right, left 순으로 두번 rotation




(a) 삽입 전 (b) $B_R$에 삽입한 뒤 (c) LR 회전 뒤

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

# 실습 내용

- Binary Search Tree 구현
- Binary Search Tree → AVL 트리로!

# 질문

- pemds81718@gmail.com
- 간단한 구글링으로 알 수 있는 내용은 답변하지 않습니다.