21강, 이진 탐색 트리(Binary Search Trees) (2)

1) 이진 탐색 트리에서 원소 삭제

(1) 키(key)를 이용해서 노드를 찾는다.

: 해당 키의 노드가 없으면, 삭제할 것도 없음

: 찾은 노드의 부모 노드도 알고 있어야 함

-> 찾은 노드를 제거하고도 이진 탐색 트리 성질을 만족하도록 트리의 구조를 정리한다.

2) 인터페이스의 설계

입력 : 키(key)

출력: 삭제한 경우 True, 해당 키의 노드가 없는 경우 False

3) 이진 탐색 트리 구조의 유지

삭제되는 노드가

1, 말단(leaf) 노드인 경우

: 그냥 그 노드를 없애면 됨

-> 부모 노드의 링크를 조정(좌?,우?)

2, 자식을 하나 가지고 있는 경우

: 삭제되는 노드 자리에 그 자식을 대신 배치

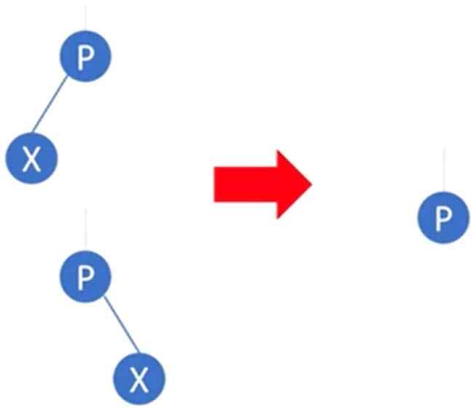
->자식이 왼쪽인지 오른쪽인지

-> 부모 노드의 링크를 조정

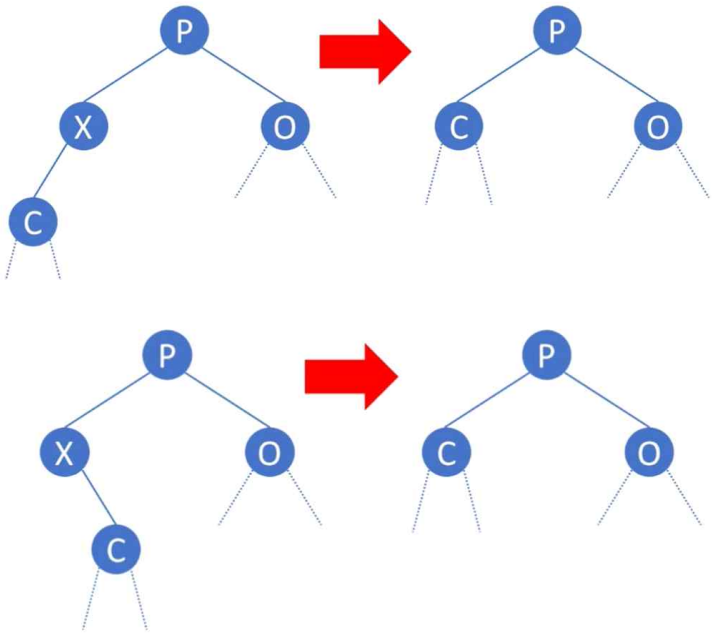
3, 삭제되는 노드가 자식을 둘 가지고 있는 경우

: 삭제되는 노드보다 바로 다음 (큰) 키를 가지는 노드를 찾아 그 노드를 삭제되는 노드 자리 대신 배치하고 이 노드를 대신 삭제

3-1) 말단(Leaf) 노드의 삭제

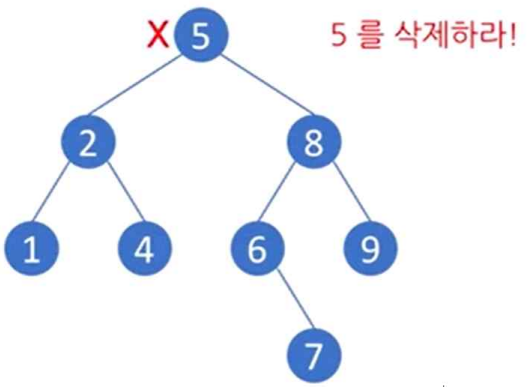


3-2) 자식을 하나 가지고 있는 경우

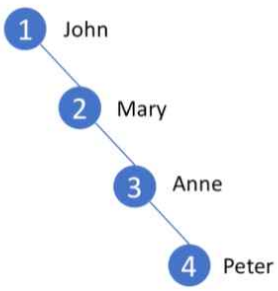


삭제되는 노드(x)가 root node인 경우는 어떻게? -> 대신 들어오는 자식이 새로 root가 됨

3-3) 자식이 둘인 노드의 삭제



4) 이진 탐색 트리가 별로 효율적이지 못한 경우



한 쪽으로 치우친 경우 선형 배열이랑 다를게 없음

높이의 균형을 유지함으로써 O(logn) 의 탐색 복잡도 보장, 삽입, 삭제 연산이 보다 복잡