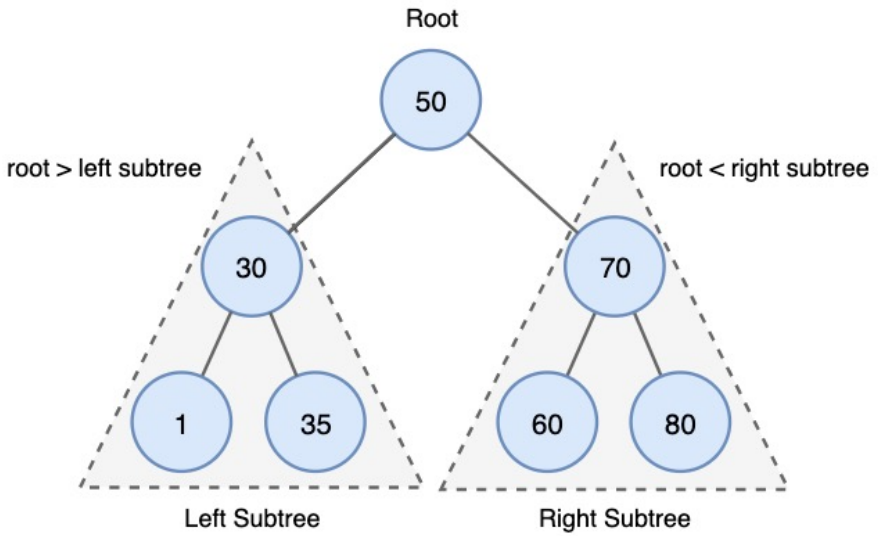
Pervasive Elastic MetaLearning Laboratory
Department of Computer Engineering
Hongik University

HW9

Binary Search Tree

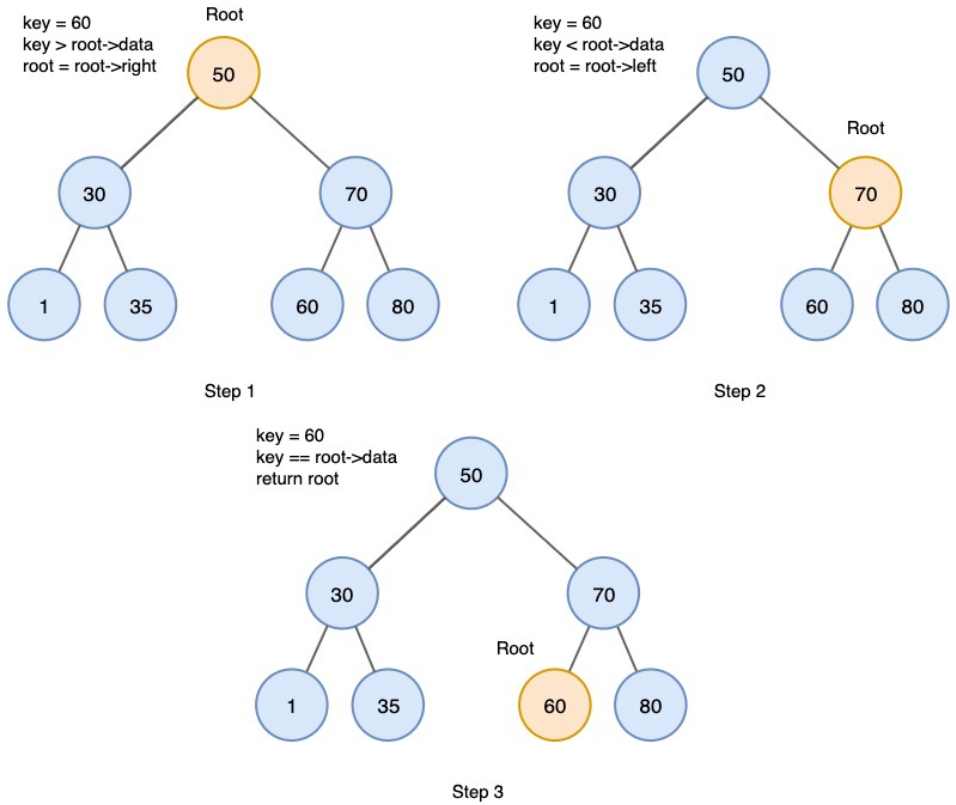
Binary Search Tree

- 왼쪽 서브트리에 있는 키들은 그 루트의 키보다 작다
- 오른쪽 서브트리에 있는 키들은 그 루트의 키보다 크다
- 왼쪽과 오른쪽 서브트리도 모두 이진 검색 트리이다
- 중복된 키를 허용하지 않는다



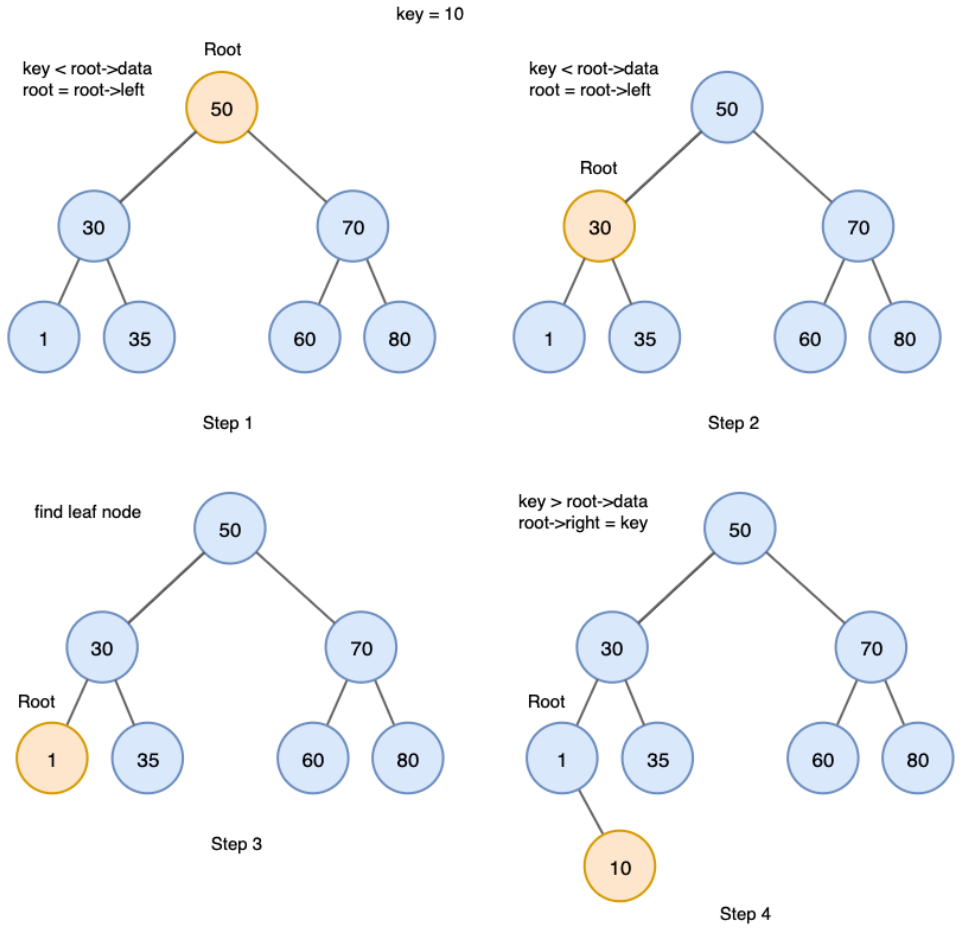
검색 Get

1. 루트에서 시작한다.
2. 검색 값을 루트와 비교한다.
    1. 루트보다 작으면 왼쪽으로
    2. 루트보다 크면 오른쪽으로
3. 일치하는 값을 찾을 때까지 2를 반복한다.
4. 검색 값이 없으면 null을 반환한다.



삽입 Insert

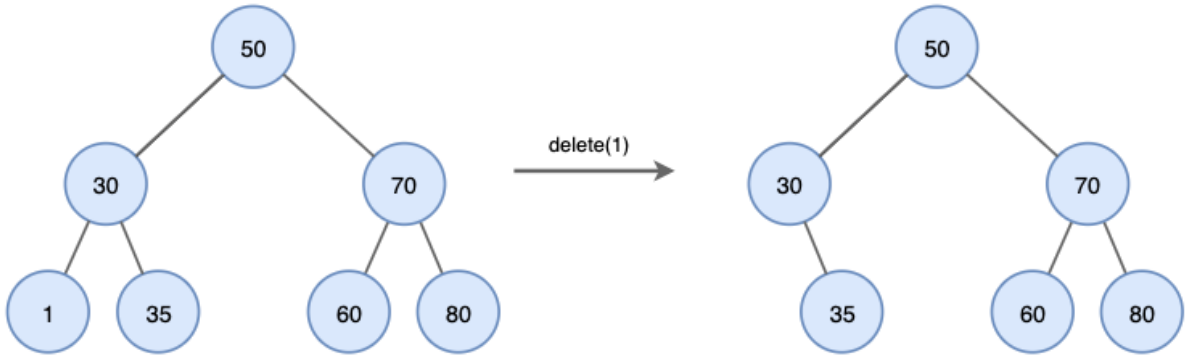
1. 루트에서 시작한다.
2. 삽입 값을 루트와 비교한다.
    1. 루트보다 작으면 왼쪽
    2. 루트보다 크면 오른쪽
3. 리프 노드에 도달한 후
    1. 해당 노드보다 작으면 왼쪽에
    2. 크면 오른쪽에



삭제 Delete

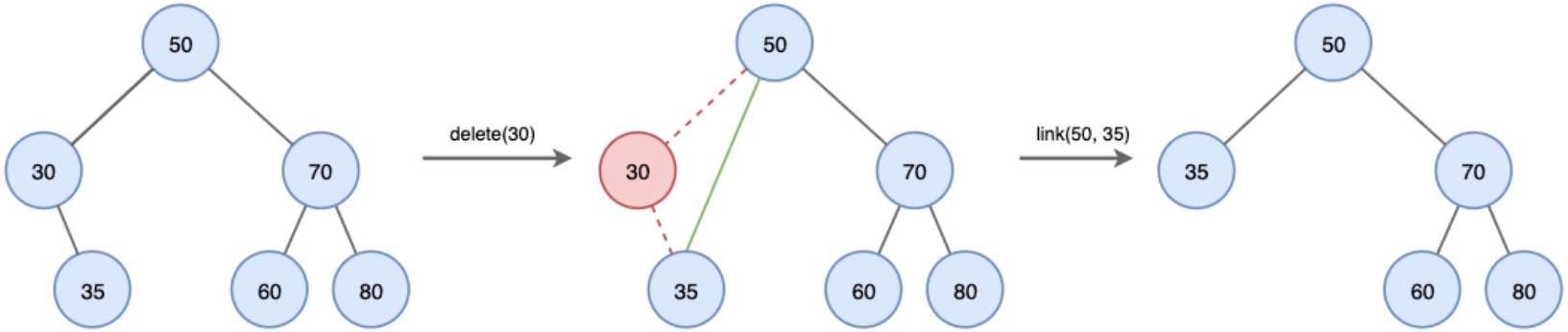
- 삭제할 노드가 리프노드인 경우 :

그 노드의 부모의 삭제된 자식 필드를 0으로 만들고 삭제된 노드를 반환하면 된다



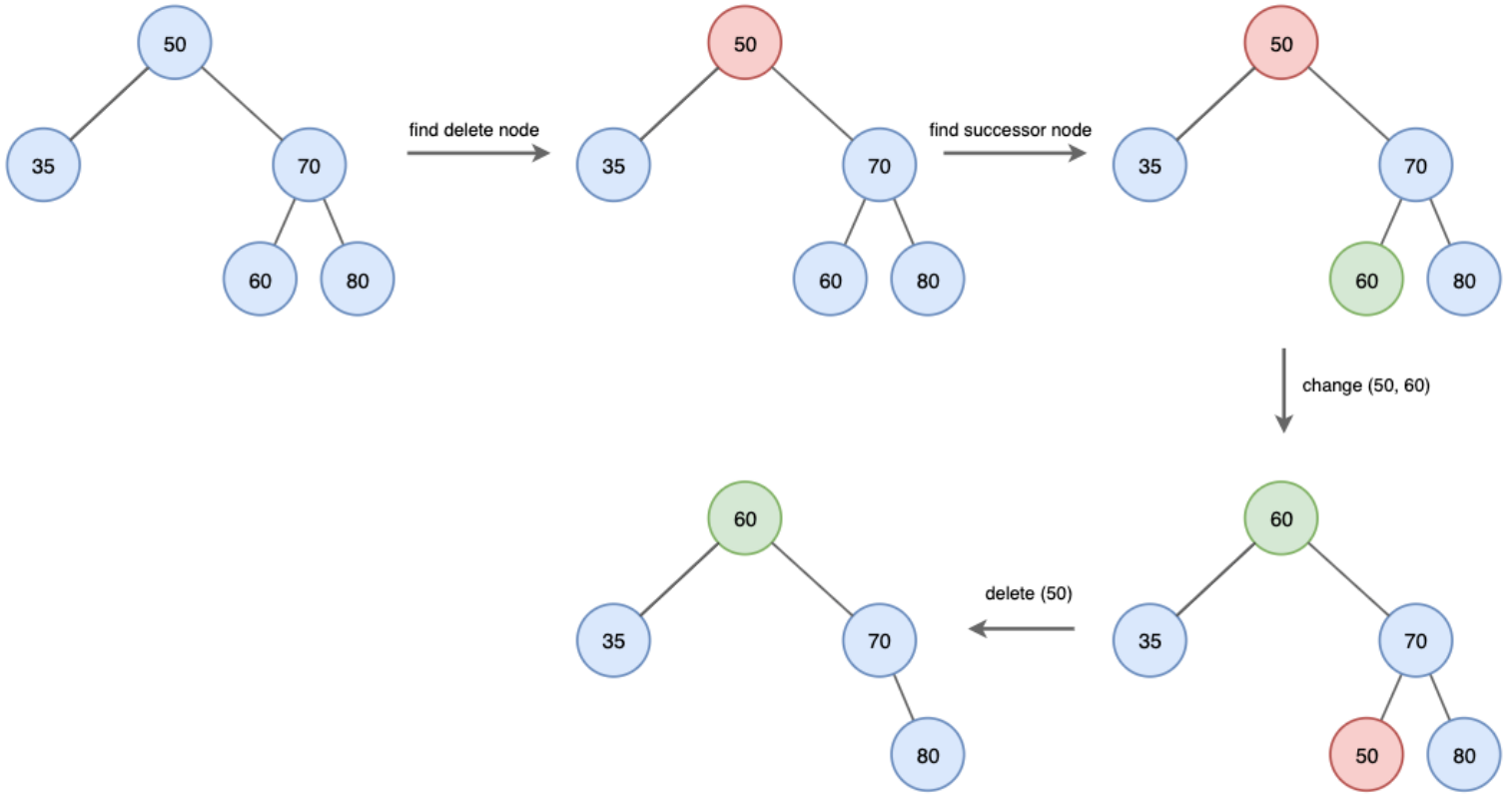
삭제 Delete

- 삭제할 노드에 자식이 하나만 있는 경우 :
  삭제될 노드는 반환되고 삭제된 노드의 자식을 삭제된 노드의 자리에 위치시킨다



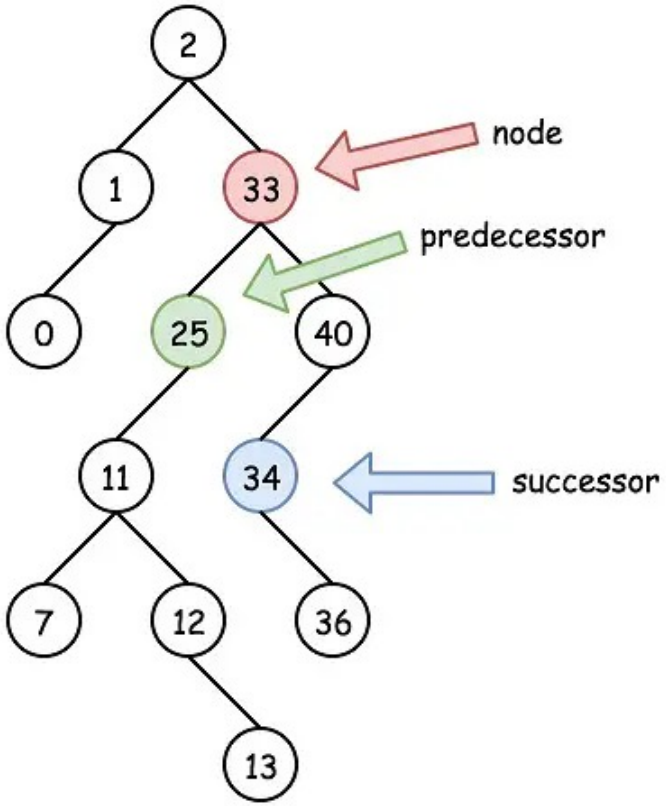
삭제 Delete

- 삭제할 노드에 자식이 둘 있는 경우 :
  그 원소를 왼쪽 서브트리에서 가장 큰 원소 or 오른쪽 서브트리에서 가장 작은 원소로 대체하고 이를 삭제한다



Successor, Predecessor

- Successor : 선택한 노드의 오른쪽 서브트리 중 가장 작은 값을 가진 노드
  - 중위 순회 시 자기 뒤에 오는 노드
- Predecessor : 선택한 노드의 왼쪽 서브트리 중 가장 큰 값을 가지는 노드
  - 중위 순회 시 자기 앞에 오는 노드
- … 25, 33, 34, …



predecessor =
one step left and then right till you can

successor =
one step right and then left till you can

실습 내용

- Binary Search Tree의 연산을 구현
- Insert : BST에 데이터를 삽입한다
- Get : BST에서 특정 요소의 위치를 찾는다
- Delete : BST에서 특정 노드를 삭제한다

질문

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