COSE213: Data Structure

Lecture 8 review

Minseok Jeon

2024 Fall

이진 탐색 트리 (Binary Search Tree)

- 이진 탐색 트리(BST, Binary Search Tree)는 이진트리 기반의 탐색을 위한 자료구조임

- 이진 탐색 트리는 다음과 같이 (재귀적으로) 정의됨

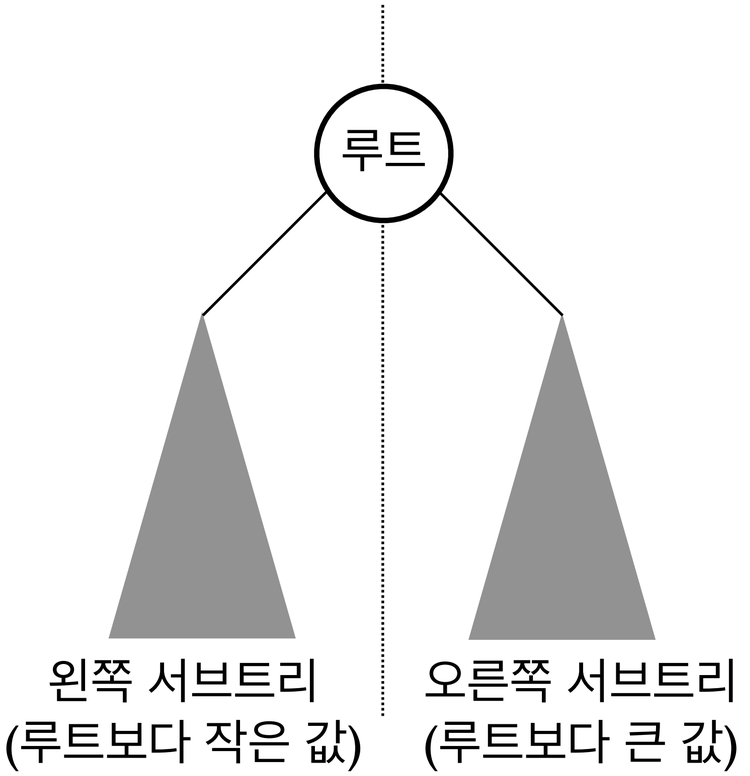
Definition

(1) 모든 노드는 유일한 키(key)를 갖는다.

(2) 왼쪽 서브트리의 키들은 루트의 키보다 작다.

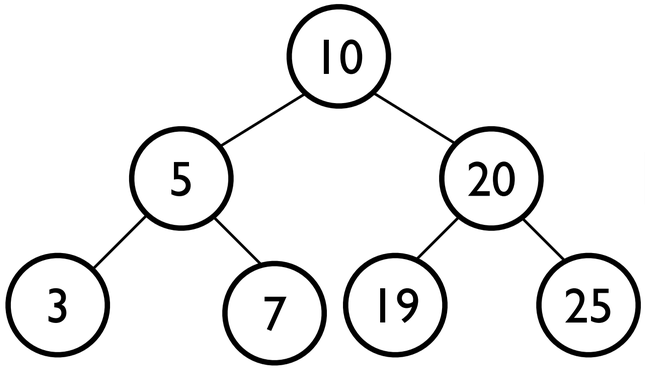
(3) 오른쪽 서브트리의 키들은 루트의 키보다 크다.

(4) 왼쪽과 오른쪽 서브트리도 이진 탐색 트리이다.



이진 탐색 트리의 순회 (Traversal)

- Example: 이진 탐색트리의 키값들을 오름차순으로 출력하고싶다 어떻게 순회를 해야 할까?

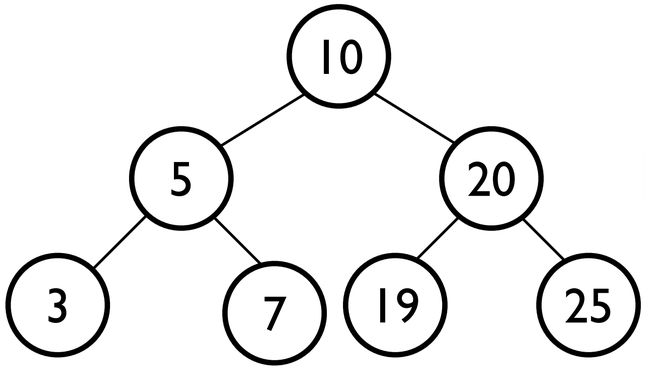


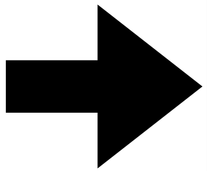
중위순회 (LVR)

```
procedure traversal(node)
    if node is not null then
        traversal(node.left)
        print(node)
        traversal(node.right)
    end if
end procedure
```

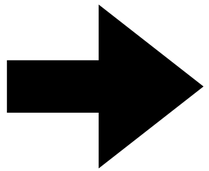
이진 탐색 트리의 순회 (Traversal)

- Example: 이진 탐색트리의 키값들을 내림차순으로 출력하고싶다 어떻게 순회를 해야 할까?



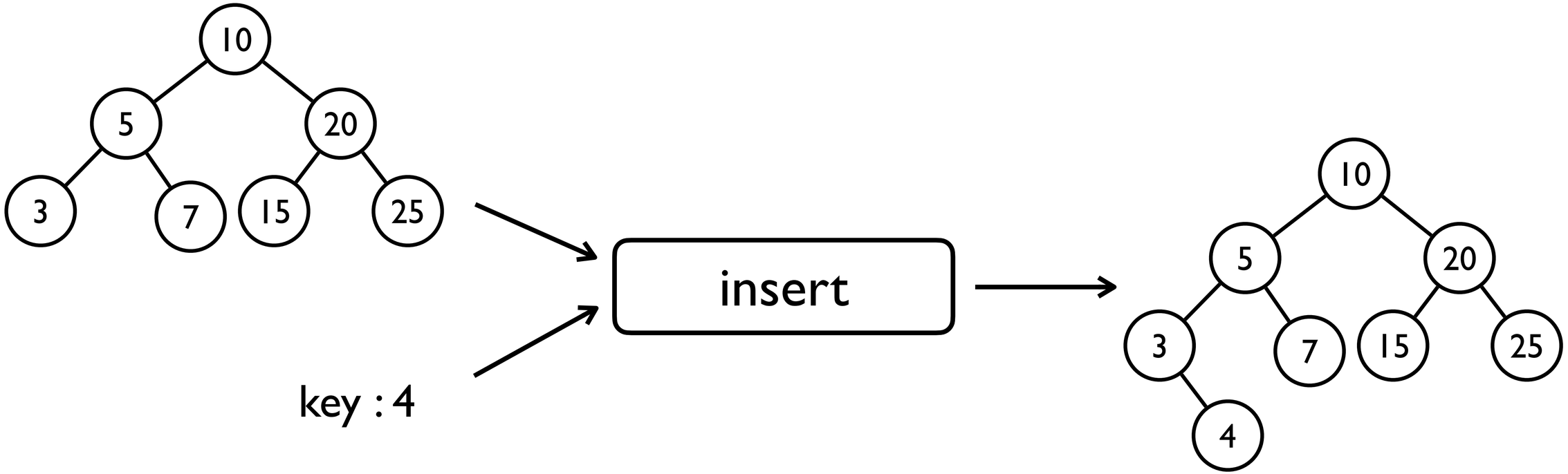


```
procedure traversal(node)
    if node is not null then
        traversal(node.right)
        print(node)
        traversal(node.left)
    end if
end procedure
```



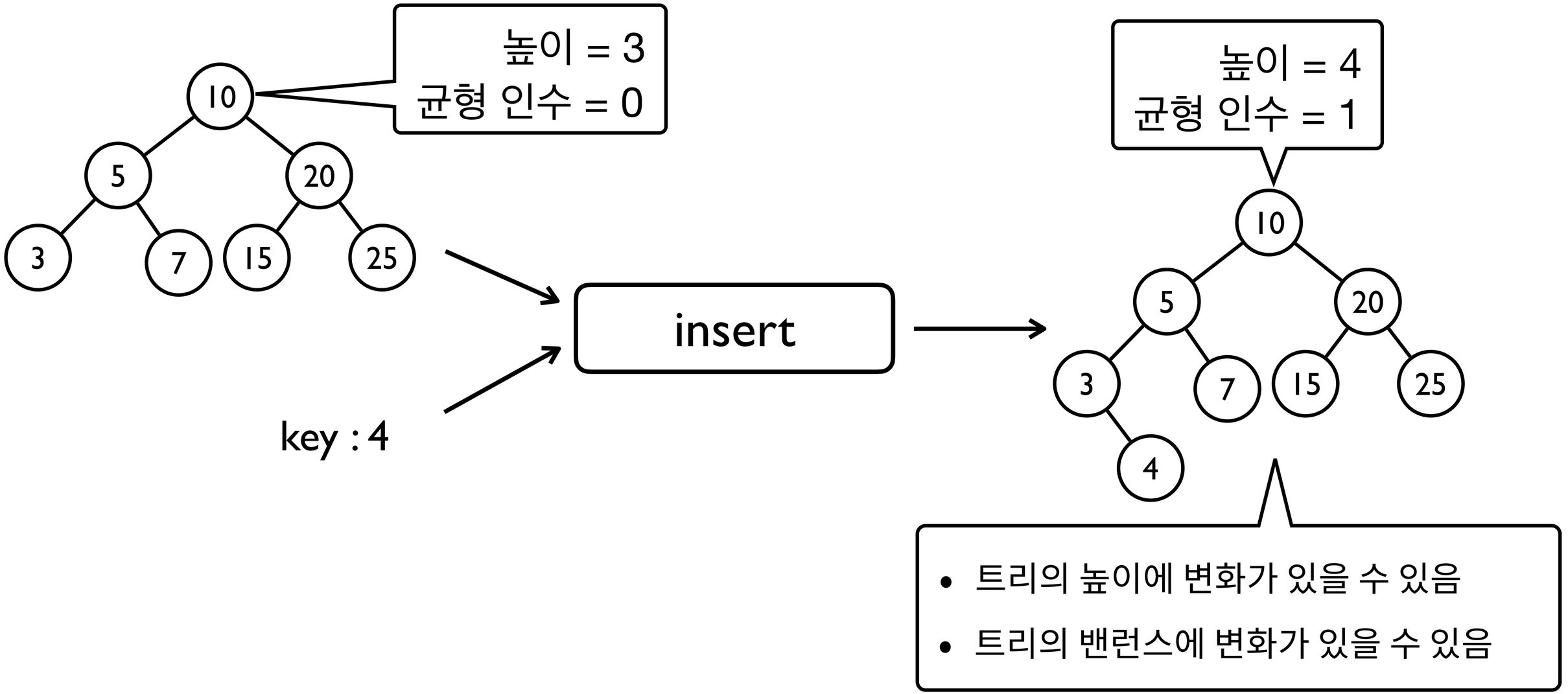
insert

- insert: 이진 탐색 트리의특성을 유지하면서 주어진 키를 가지는 새로운 노드를 이진 탐색 트리에 삽입함



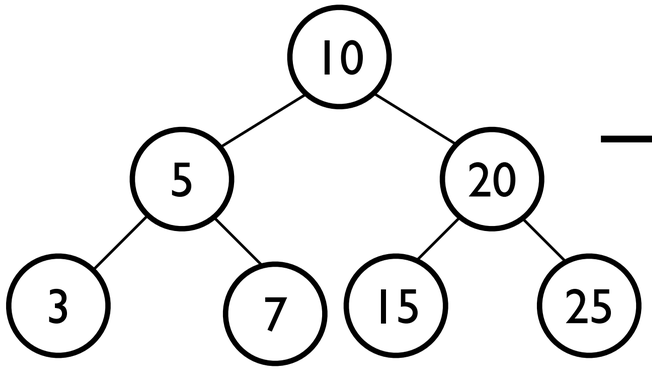
insert

- insert: 이진 탐색 트리의특성을 유지하면서 주어진 키를 가지는 새로운 노드를 이진 탐색 트리에 삽입함



insert

- insert: 이진 탐색 트리의특성을 유지하면서 주어진 키를 가지는 새로운 노드를 이진 탐색 트리에 삽입함

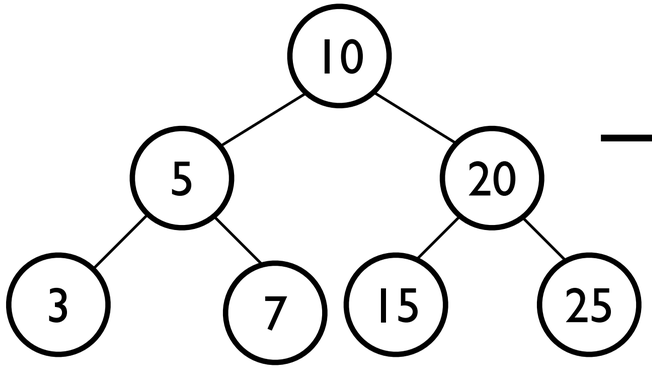


key : 4

```
procedure insert(root, key)
  if root = NULL then
    root ← allocateNode()
    root.key ← key
    root.left ← NULL
    root.right ← NULL
    return root
  end if
  if key < root.key then
    root.left ← insert(root.left, key)
  elif key > root.key then
    root.right ← insert(root.right, key)
  end if
  return root
end procedure
```

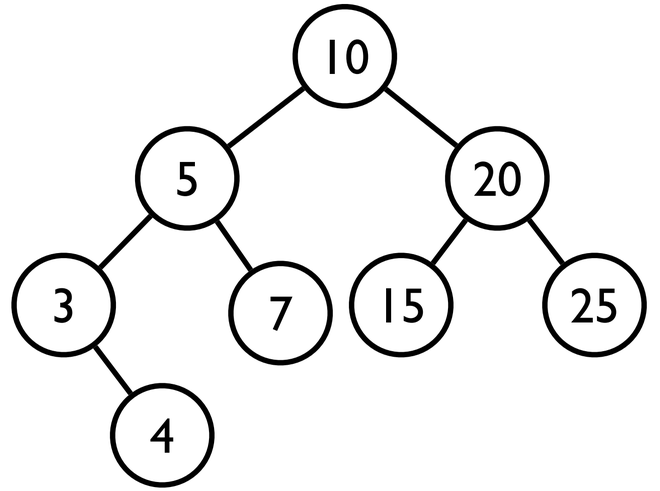
insert

- insert: 이진 탐색 트리의특성을 유지하면서 주어진 키를 가지는 새로운 노드를 이진 탐색 트리에 삽입함



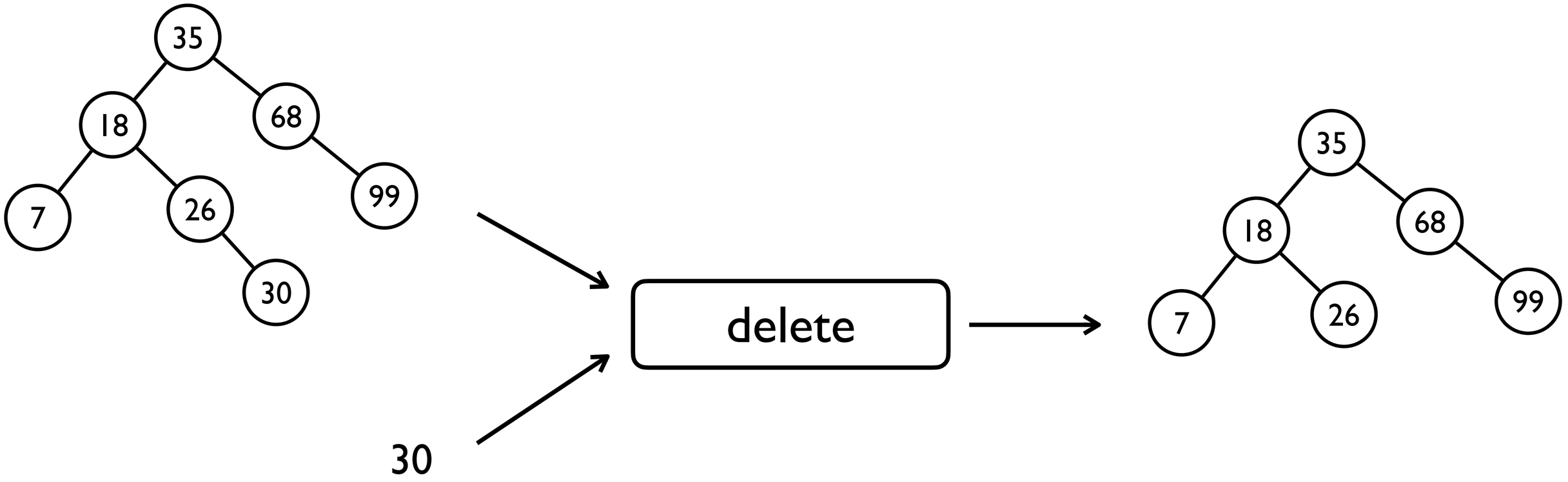
key : 4

```
procedure insert(root, key)
  if root = NULL then
    root ← allocateNode()
    root.key ← key
    root.left ← NULL
    root.right ← NULL
    return root
  end if
  if key < root.key then
    root.left ← insert(root.left, key)
  elif key > root.key then
    root.right ← insert(root.right, key)
  end if
  return root
end procedure
```



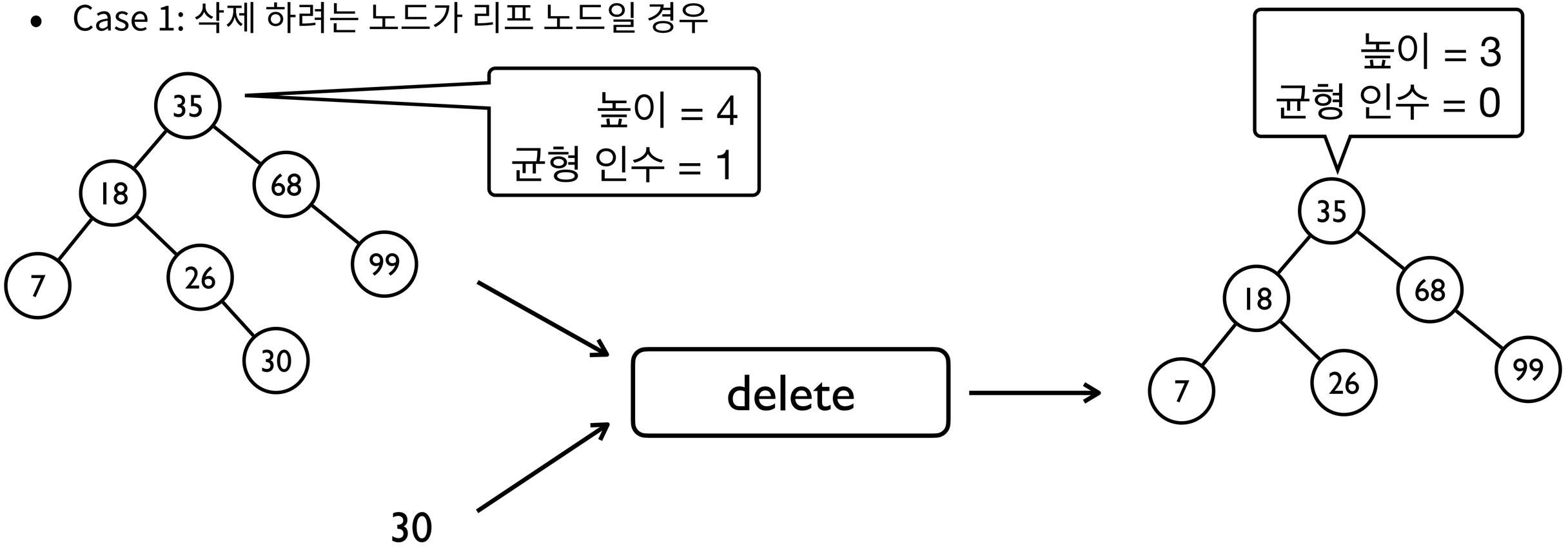
delete

- delete: 이진 탐색 트리의특성을 유지하면서 주어진 키값을 가진 노드를 이진 탐색 트리에서 삭제함
  - Case 1: 삭제 하려는 노드가 리프 노드일 경우



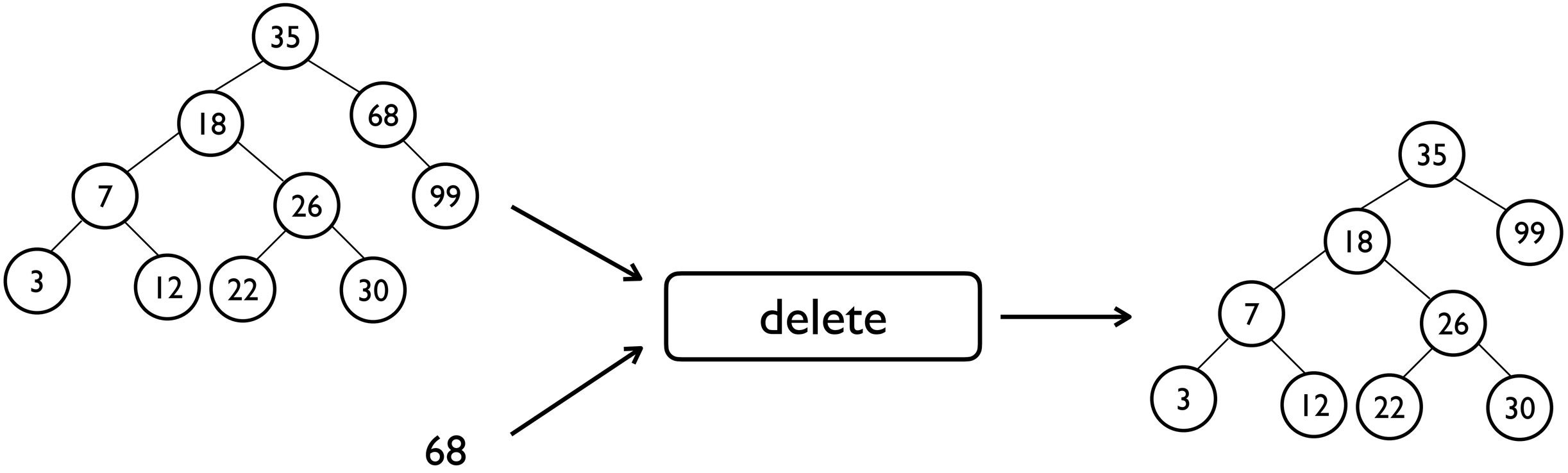
delete

- delete: 이진 탐색 트리의특성을 유지하면서 주어진 키값을 가진 노드를 이진 탐색 트리에서 삭제함
  - Case 1: 삭제 하려는 노드가 리프 노드일 경우



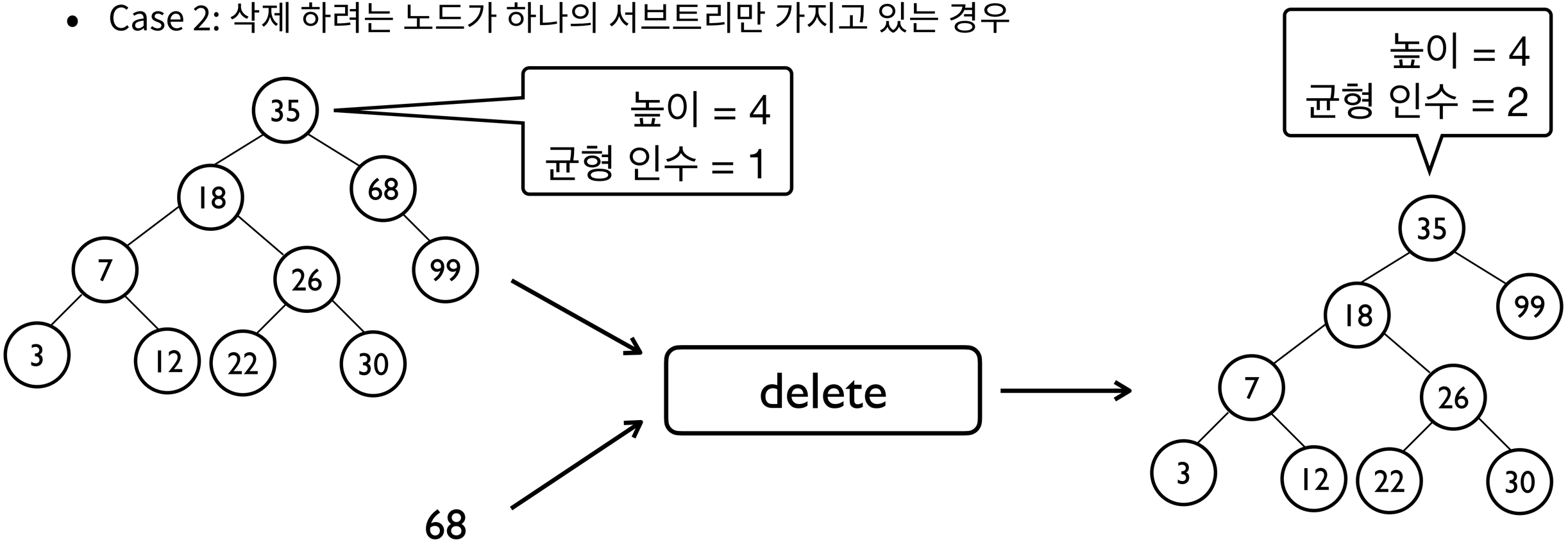
delete

- delete: 이진 탐색 트리의특성을 유지하면서 주어진 키값을 가진 노드를 이진 탐색 트리에서 삭제함
  - Case 2: 삭제 하려는 노드가 하나의 서브트리만 가지고 있는 경우



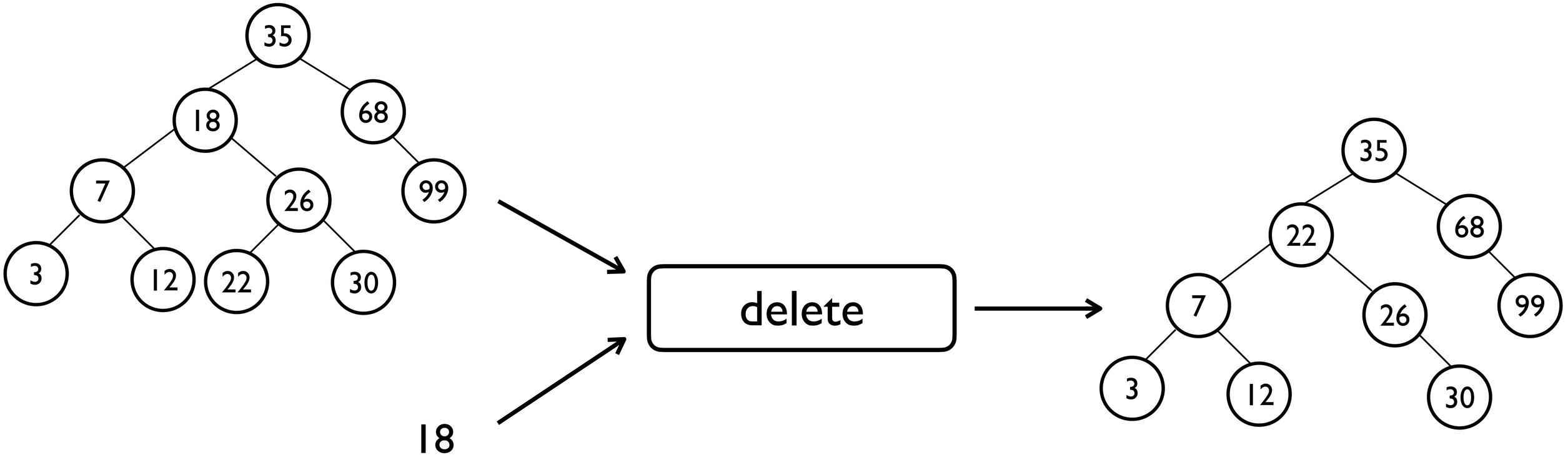
delete

- delete: 이진 탐색 트리의특성을 유지하면서 주어진 키값을 가진 노드를 이진 탐색 트리에서 삭제함
  - Case 2: 삭제 하려는 노드가 하나의 서브트리만 가지고 있는 경우

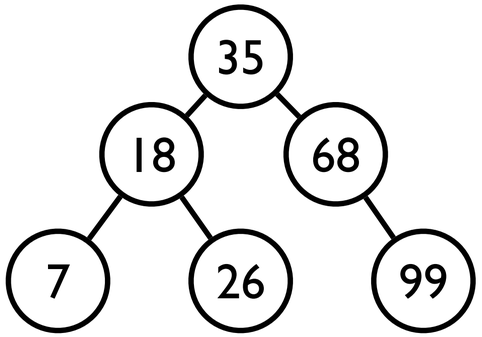


delete

- delete: 이진 탐색 트리의특성을 유지하면서 주어진 키값을 가진 노드를 이진 탐색 트리에서 삭제함
  - Case 3: 삭제 하려는 노드가 두개의 서브트리를 가지고 있는 경우

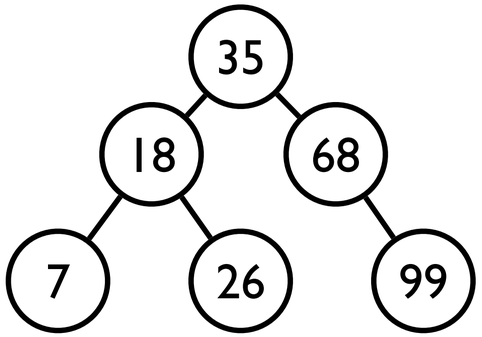


delete




```
procedure delete(root, key)
  if root = NULL then
    return NULL
  end if
  if key < root.key then
    root.left ← deleteNode(root.left, key)
  elif key > root.key then
    root.right ← deleteNode(root.right, key)
  else
    if root.left = NULL then
      node ← root.right
      free(root)
      return node
    elif root.right = NULL then
      node ← root.left
      free(root)
      return node
    end if
    node ← findMin(root.right)
    root.key ← node.key
    root.right ← delete(root.right, node.key)
  end if
  return root
end procedure
```

delete



18

```
procedure delete(root, key)
  if root = NULL then
    return NULL
  end if
  if key < root.key then
    root.left ← deleteNode(root.left, key)
  elif key > root.key then
    root.right ← deleteNode(root.right, key)
  else
    if root.left = NULL then
      node ← root.right
      free(root)
      return node
    elif root.right = NULL then
      node ← root.left
      free(root)
      return node
    end if
    node ← findMin(root.right)
    root.key ← node.key
    root.right ← delete(root.right, node.key)
  end if
  return root
end procedure
```