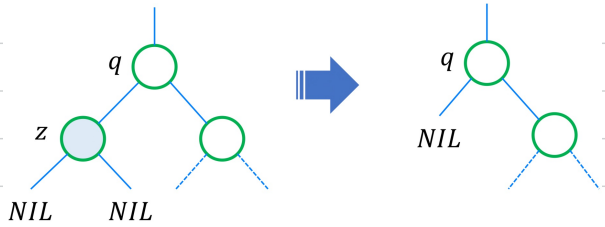
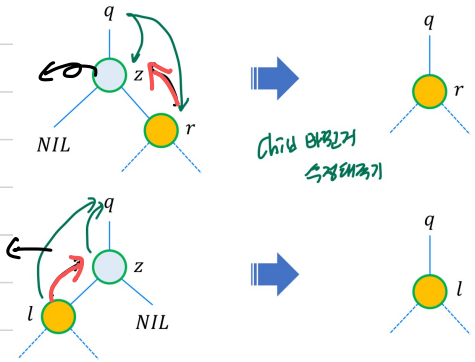
Deletion

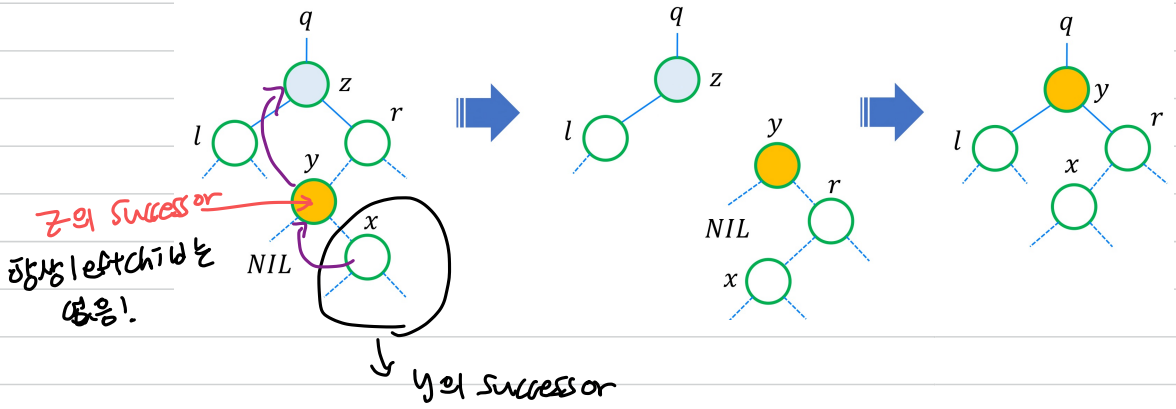
① z가 no children (leaf) 이면 simply remove
   ⓐ 그 z의 parent의 child가 없어졌으니 NIL로 바꿔줌



② z가 one child 가지고 있으면 child를 parent 위치로 수정함



① z는 날라가고
z의 child가 z위치로 바뀜
⇒ ② parent가 x로 바뀜
⇒ ③ x의 child가 r로 바뀜



③ z가 two children이 있으면,
   ① z의 successor찾기 = right subtree에서 가장 왼쪽 node (min (left child가 없음!)) ← right subtree가 있는 경우
   ② y (z의 successor)을 z 위치로 replace 함
      → z의 right subtree와 left subtree는
      y의 새로운 right subtree와 left subtree가 됨
   ③ y는 z의 successor이기 때문에 right subset에서 가장 min key를 갖고 있음 (no left child)
      → right child를 y 위치로 replace 함
      → y의 원래 right subtree는 자동으로 따라옴

Deletion - transplant

- The subroutine *TRANSPLANT* replaces one subtree as a child of its parent with another subtree
  - $O(1)$
- $TRANSPLANT(T, u, v)$

```
1.  if u.p == NIL
2.      T.root = v
3.  elseif u == u.p.left
4.      u.p.left = v
5.  else
6.      u.p.right = v
7.  if v ≠ NIL
8.      v.p = u.p
```

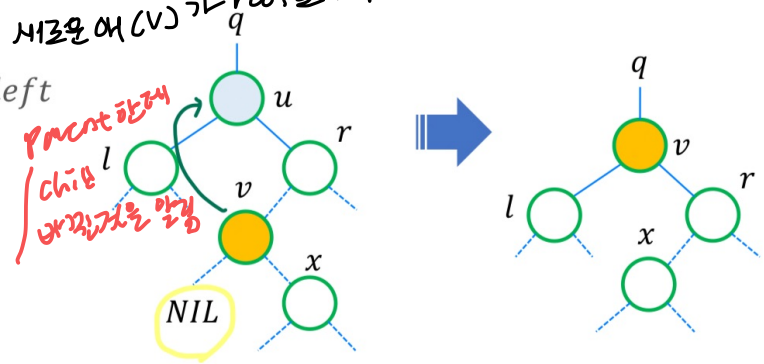
원래 node를 다른 애로 replace

내가 내의 parent 관점에서 left에 있었다면 그 left인 내에 v로 replace

right 였다면 right 내에 v로 replace

u = root

u가 root를 replace



바뀐 아이의 parent를 원래 아이 u의 parent로 replace 해줌

실제 code 보기

- $O(h)$ time on a tree of height h
- $TREE-DELETE(T, z)$

```
1.  if z.left == NIL and z.right == NIL // case 1
2.      if z.p.left == z
3.          z.p.left = NIL
4.      else
5.          z.p.right = NIL
6.  elseif z.left == NIL // case 2 (left)
7.      TRANSPLANT(T, z, z.right) // replace z by its right child
8.  elseif z.right == NIL // case 2 (right)
9.      TRANSPLANT(T, z, z.left) // replace z by its left child
10. else // case3
11.     y = TREE-MINIMUM(z.right) // y is z's successor
12.     if y ≠ z.right // is y farther down the tree?
13.         TRANSPLANT(T, y, y.right) // replace y by its right child
14.         y.right = z.right // z's right child becomes
15.         y.right.p = y     // y's right child
16.     TRANSPLANT(T, z, y) // replace z by its successor y
17.     y.left = z.left     // and give z's left child to y
18.     y.left.p = y        // which had no left child
```

remove child

a, b, ※

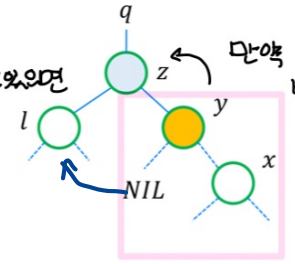
root->left == nullptr

(x->left) x = x->left

Parent의 left, right 수정 ⇒ child 없어져서

자리를 다른 아이로 바꾸고 Parent 수정 (가르키는 pointer 바꾸기)

y을 z 위치로 바꿔주는 것



⇒ 만약 successor가 z에 바로 안 붙어있으면

만약 successor가 바로 z의 right child 라면 그대로 z 위치로 옮기면 됨 (successor의 following right subtree 함께)

→ y의 parent만 바꿔주면 됨