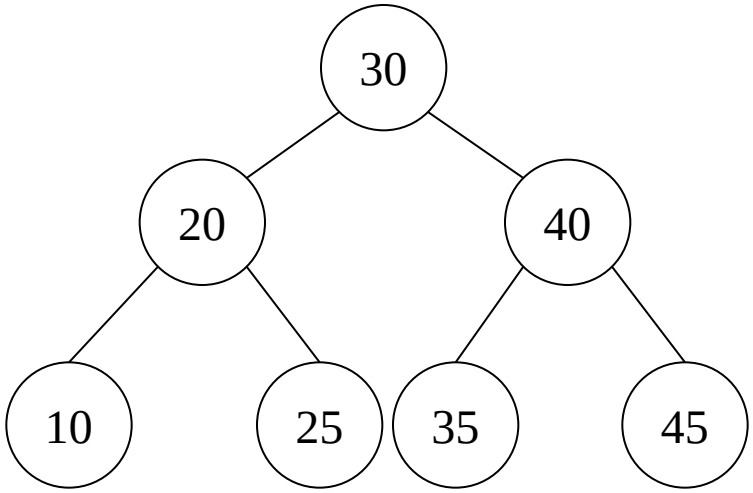
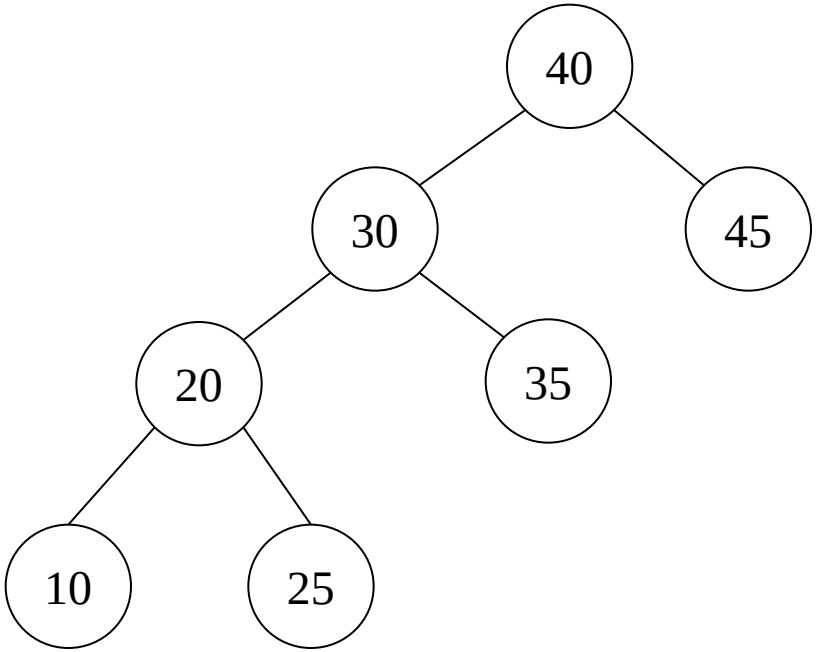
이진검색트리

- 이진검색트리의 각 노드는 키값을 하나씩 갖는다. 각 노드의 키값은 모두 달라야 한다.
- 최상위 레벨에 루트 노드가 있고, 각 노드는 최대 두 개의 자식을 갖는다.
- 임의의 노드의 키값은 자신의 왼쪽 자식 노드의 키값보다 크고, 오른쪽 자식의 키값보다 작다.

이진검색트리의 예

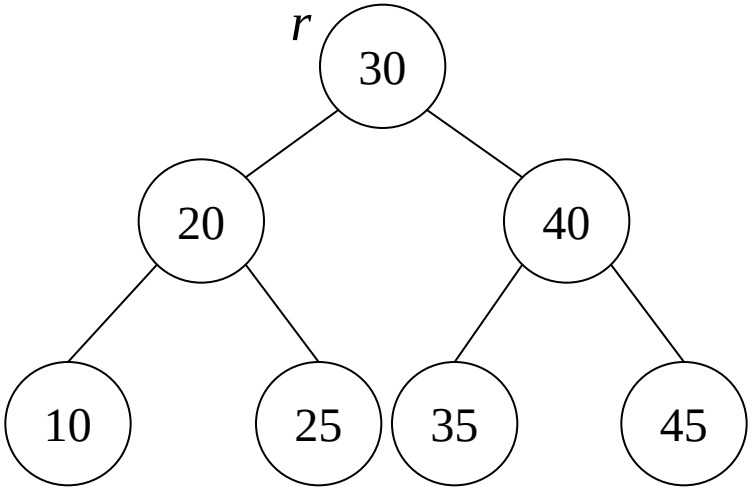


(a)

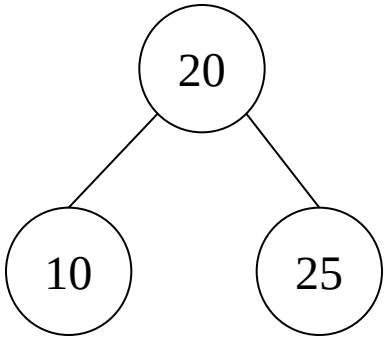


(b)

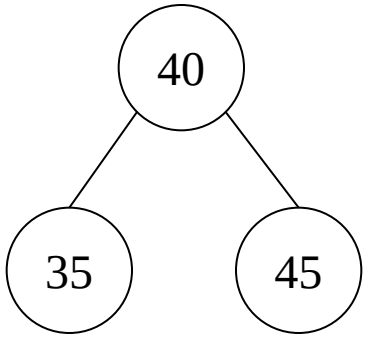
서브트리의 예



(a)



(b) 노드 *r*의 왼쪽 서브트리

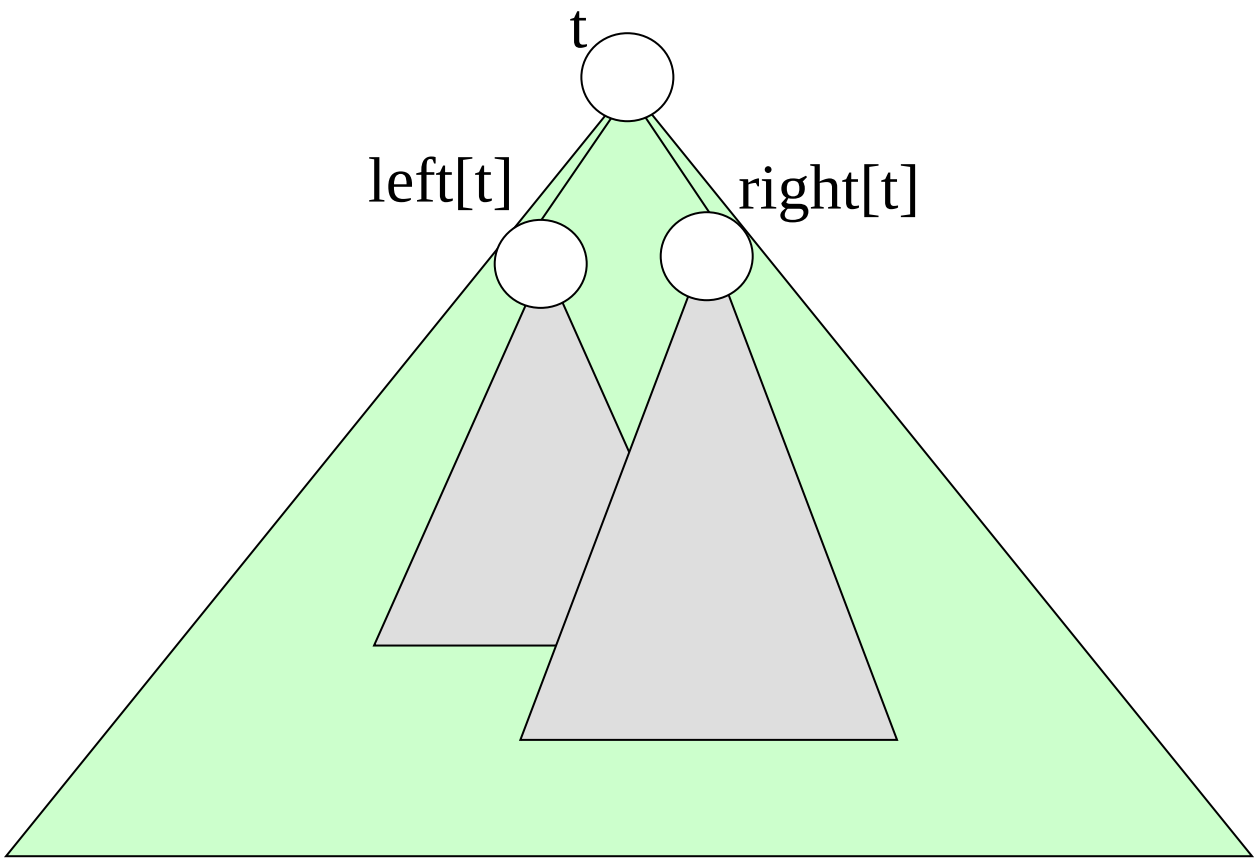


(c) 노드 *r*의 오른쪽 서브트리

이진검색트리에서의 검색

```
treeSearch(t, x)
▷ t: 트리의 루트 노드
▷ x: 검색하고자 하는 키
{
        if (t=NIL or key[t]=x) then return t;
        if (x < key[t])
                then return treeSearch(left[t], x);
                else return treeSearch(right[t], x);
}
```

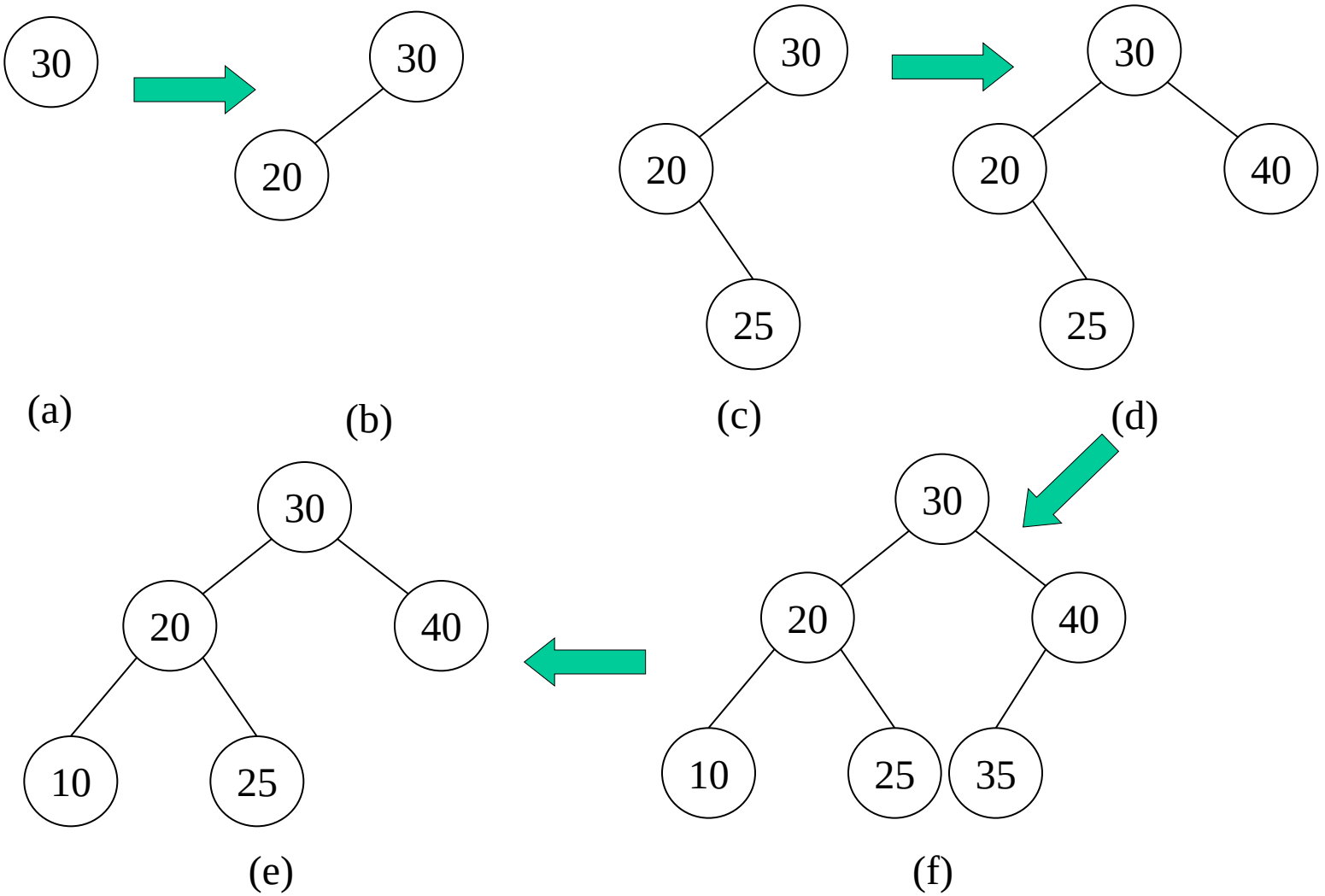
검색에서 재귀적 관점



이진검색트리에서의 삽입

```
treeInsert(t, x)
▷ t: 트리의 루트 노드
▷ x: 삽입하고자 하는 키
▷ 작업 완료 후 루트 노드의 포인터를 리턴한다
{
    if (t=NIL) then {
        key[r] ← x;  left[r] ← NIL;  right[r] ← NIL;    ▷ r : 새 노드
        return r ;
    }
    if (x < key(t))
        then {left[t] ← treeInsert(left[t], x); return t;}
        else {right[t] ← treeInsert(right[t], x); return t;}
}
```

삽입의 예



이진검색트리에서의 삭제

t: 트리의 루트 노드
r: 삭제하고자 하는 노드

- 3가지 경우에 따라 다르게 처리한다
  - Case 1 : *r*이 리프 노드인 경우
  - Case 2 : *r*의 자식 노드가 하나인 경우
  - Case 3 : *r*의 자식 노드가 두 개인 경우

이진검색트리에서의 삭제

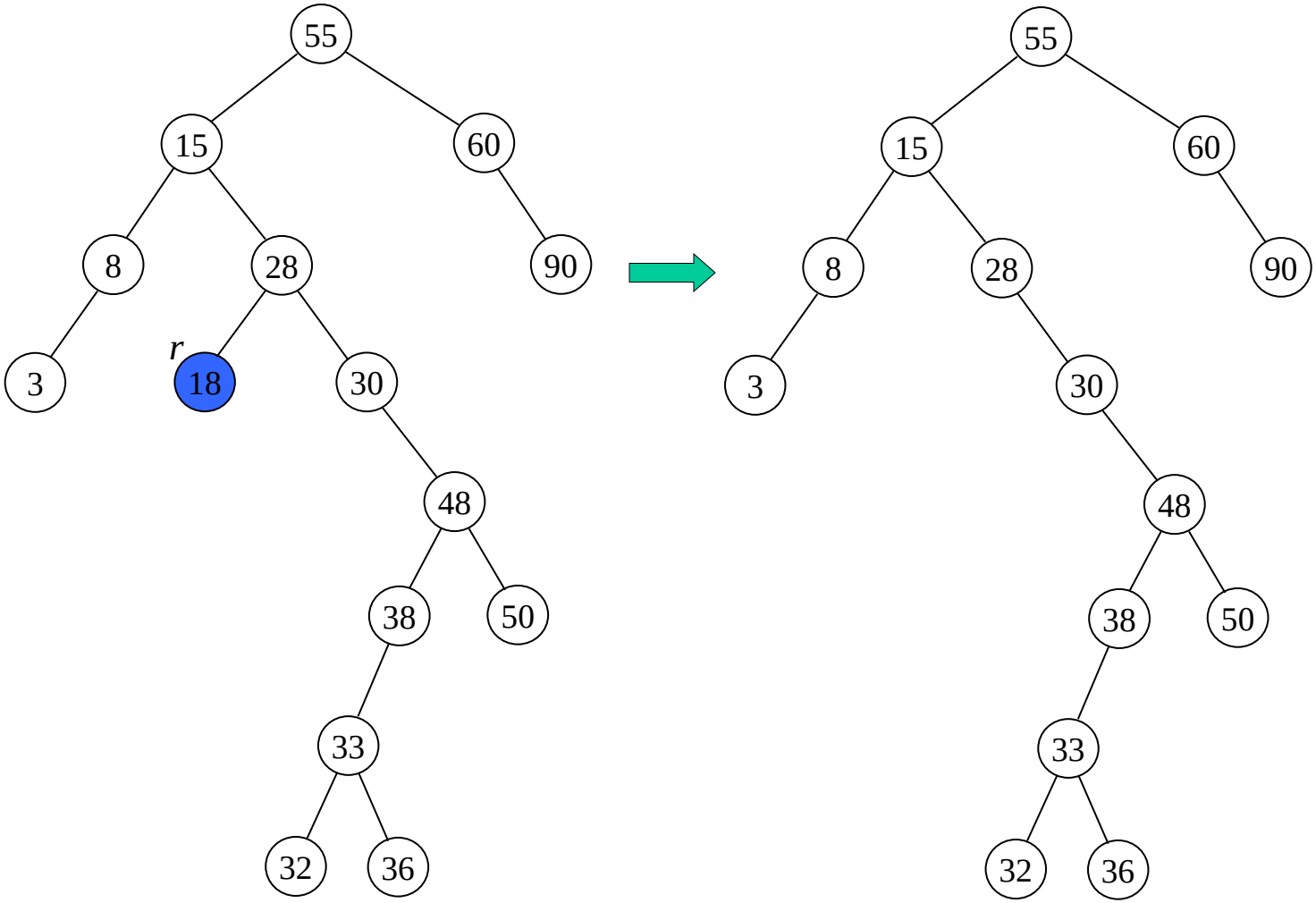
```
Sketch-TreeDelete(t, r)
▷ t: 트리의 루트 노드
▷ x: 삭제하고자 하는 키
{
    if (r이 리프 노드) then                                  ▷ Case 1
        그냥 r을 버린다;
    else if (r의 자식이 하나만 있음) then                    ▷ Case 2
        r의 부모가 r의 자식을 직접 가리키도록 한다;
    else                                                     ▷ Case 3
        r의 오른쪽 서브트리의 최소원소 노드 s를 삭제하고,
        s를 r 자리에 놓는다;
}
```

이진검색트리에서의 삭제

t: 트리의 루트 노드
r: 삭제하고자 하는 노드
p: *r*의 부모 노드

```
treeDelete(t, r, p)
{
    if (r = t) then root ← deleteNode(t);          ▷ r이 루트 노드인 경우
    else if (r = left[p])                          ▷ r이 루트가 아닌 경우
            then left[p] ← deleteNode(r);                  ▷ r이 p의 왼쪽 자식
            else right[p] ← deleteNode(r);                 ▷ r이 p의 오른쪽 자식
}
deleteNode(r)
{
    if (left[r] = right[r] = NIL) then return NIL;                          ▷ Case 1
    else if (left[r] = NIL and right[r] ≠ NIL) then return right[r];        ▷ Case 2-1
    else if (left[r] ≠ NIL and right[r] = NIL) then return left[r];         ▷ Case 2-2
    else {                                                                  ▷ Case 3
            s ← right[r];
            while (left[s] ≠ NIL)
                    {parent ← s;  s ← left[s];}
            key[r] ← key[s];
            if (s = right[r]) then right[r] ← right[s];
                    else left[parent] ← right[s];
            return r;
    }
}
```

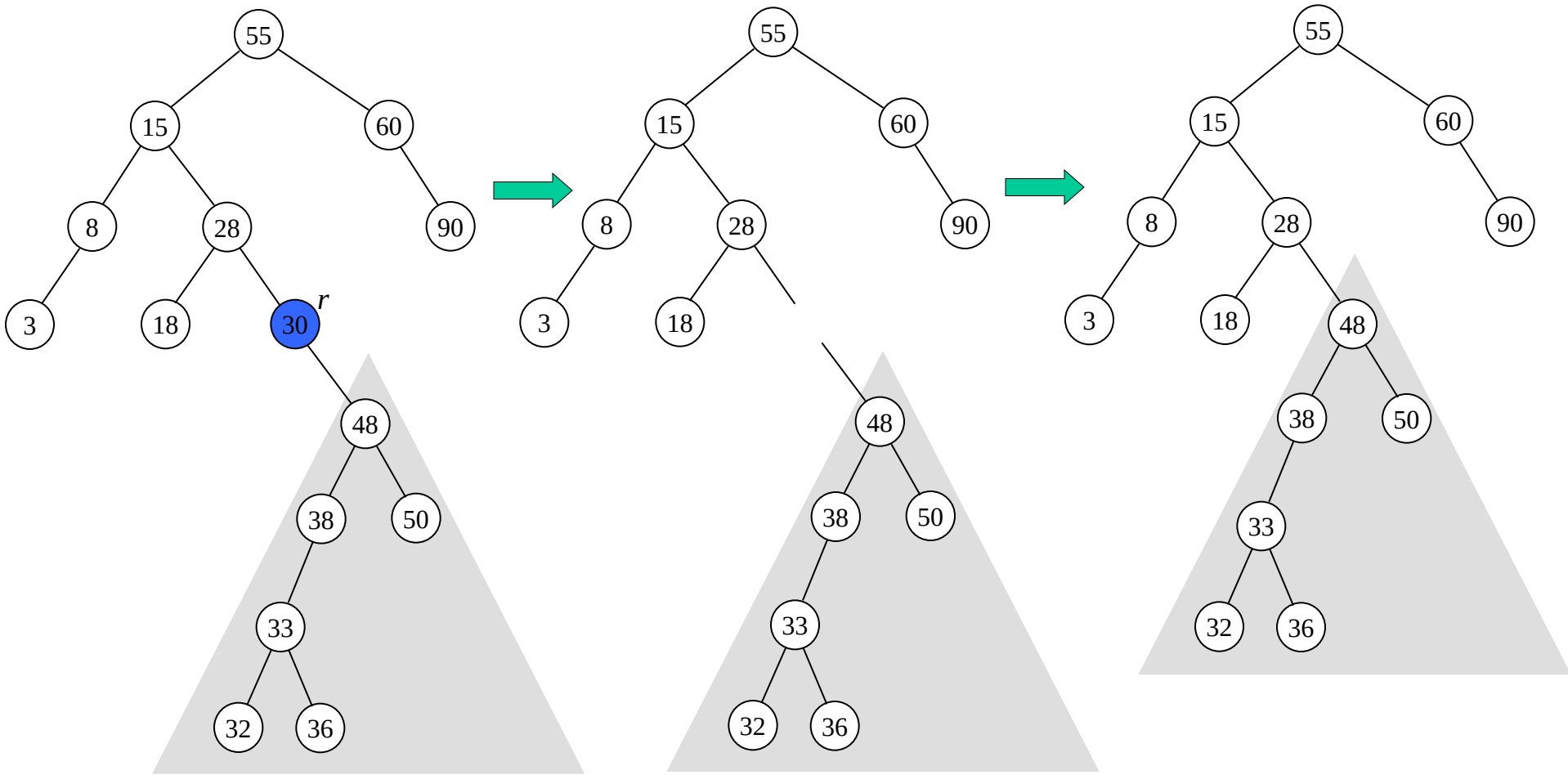
삭제의 예: Case 1



(a) r의 자식이 없음

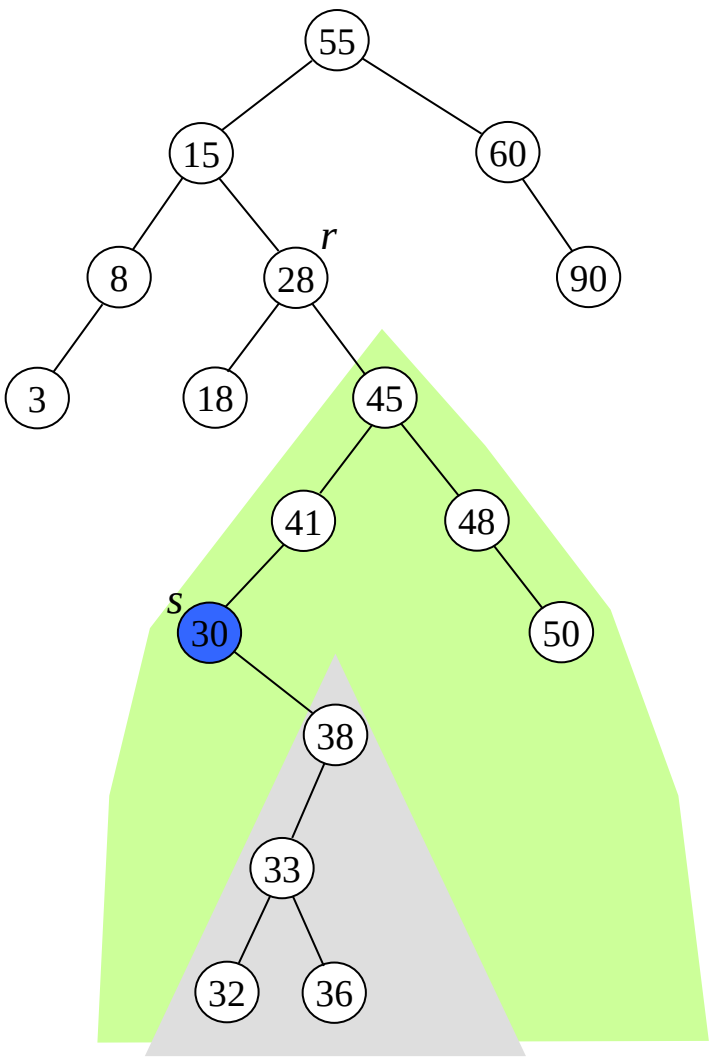
(b) 단순히 r을 제거한다

삭제의 예: Case 2

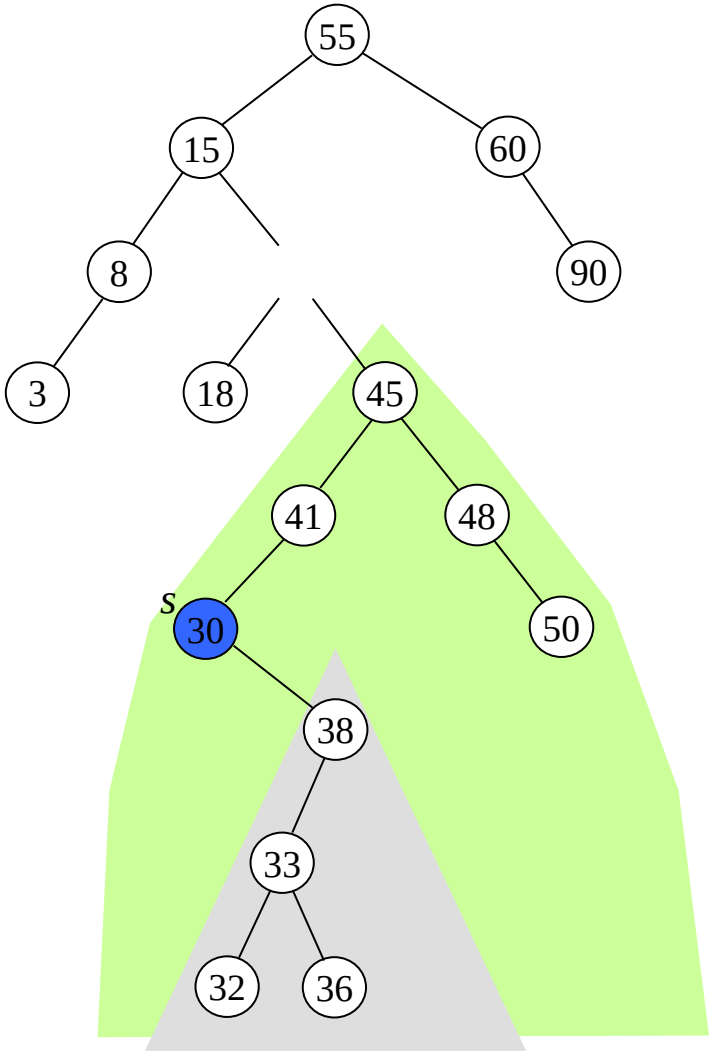


(a) r의 자식이 하나뿐임 (b) r을 제거 (c) r 자리에 r의 자식을 놓는다

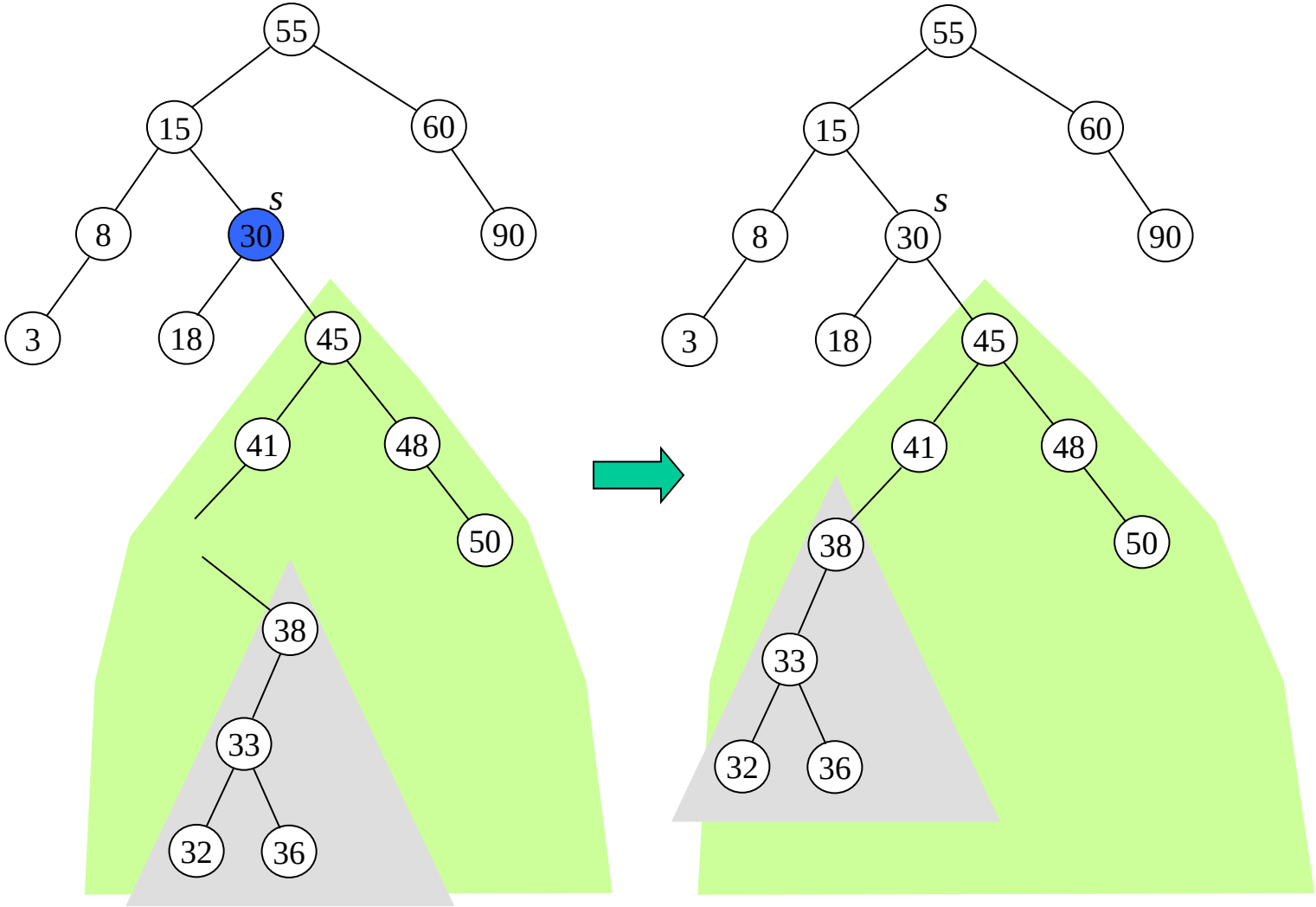
삭제의 예: Case 3



(a) r의 직후원소 s를 찾는다



(b) r을 없앤다



(c) s를 r자리로 옮긴다

(d) s가 있던 자리에 s의 자식을 놓는다