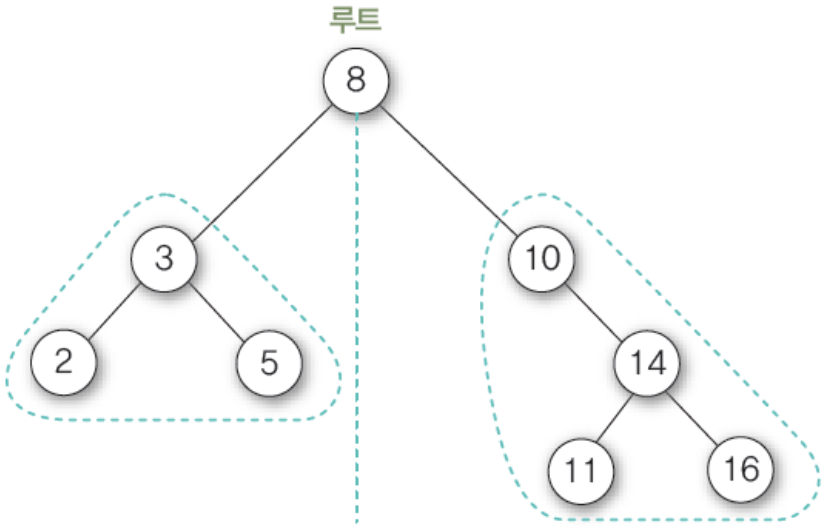
5. 이진 탐색 트리

❖ 이진 탐색 트리binary search tree

- 이진 트리를 탐색용 자료구조로 사용하기 위해 원소 크기에 따라 노드 위치를 정의한 것

- 모든 원소는 서로 다른 유일한 키를 갖는다.
- 왼쪽 서브 트리에 있는 원소들의 키는 그 루트의 키보다 작다.
- 오른쪽 서브 트리에 있는 원소들의 키는 그 루트의 키보다 크다.
- 왼쪽 서브 트리와 오른쪽 서브 트리도 이진 탐색 트리이다.

그림 7-32 이진 탐색 트리의 정의



왼쪽 서브 트리의 키값 〈 루트의 키값 〈 오른쪽 서브 트리의 키값

그림 7-33 이진 탐색 트리의 구조

5. 이진 탐색 트리

❖ 이진 탐색 트리의 탐색 연산

- 루트에서 시작
- 탐색할 키값 x를 루트 노드의 키값과 비교
  - (키값 x = 루트 노드의 키값)인 경우 : 원하는 원소를 찾았으므로 탐색연산 성공
  - (키값 x < 루트 노드의 키값)인 경우 : 루트노드의 왼쪽 서브트리에 대해서 탐색연산 수행
  - (키값 x > 루트 노드의 키값)인 경우 : 루트노드의 오른쪽 서브트리에 대해서 탐색연산 수행
- 서브트리에 대해서 순환적으로 탐색 연산을 반복

5. 이진 탐색 트리

알고리즘 7-4 이진 탐색 트리의 노드 탐색

```
searchBST(bsT, x)
    p ← bsT;
    if (p = NULL) then
        return NULL;
    if (x = p.key) then
        return p;
    if (x < p.key) then
        return searchBST(p.left, x);
    else return searchBST(p.right, x);
end searchBST()
```

5. 이진 탐색 트리

- 이진 탐색 트리에서 원소 11을 탐색

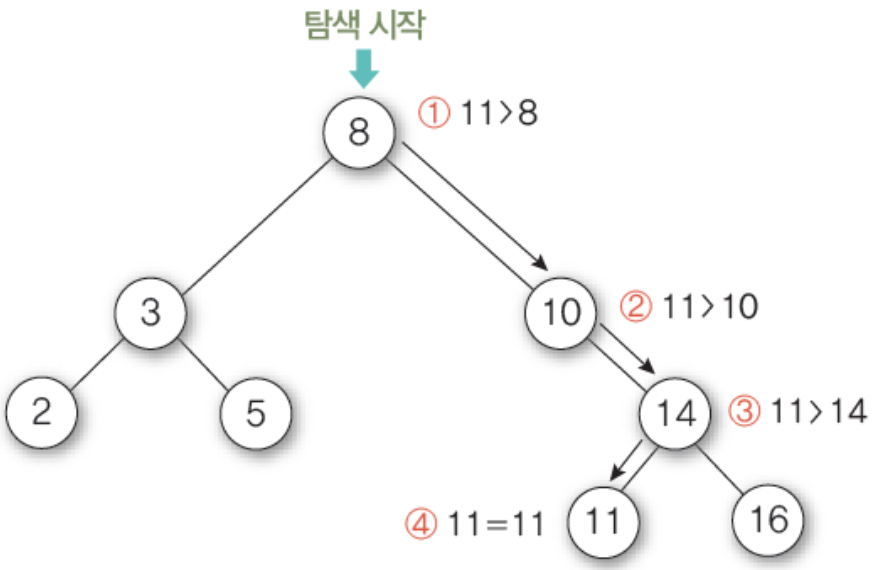


그림 7-34 이진 탐색 트리에서의 탐색 과정 예

- ① (찾는 키값 11 > 루트 노드의 키값 8)이므로 오른쪽 서브 트리 탐색
- ② (찾는 키값 11 > 노드의 키값 10)이므로 다시 오른쪽 서브 트리 탐색
- ③ (찾는 키값 11 < 노드의 키값 14)이므로 왼쪽 서브 트리 탐색
- ④ (찾는 키값 11 = 노드의 키값 11)이므로 탐색 성공으로 연산 종료

5. 이진 탐색 트리

❖ 이진 탐색 트리의 삽입 연산

1) 먼저 탐색 연산을 수행

- 삽입할 원소와 같은 원소가 트리에 있으면 삽입할 수 없으므로, 같은 원소가 트리에 있는지 탐색하여 확인
- 탐색에서 탐색 실패가 결정되는 위치가 삽입 위치가 됨

2) 탐색 실패한 위치에 원소를 삽입

5. 이진 탐색 트리

- 이진 탐색 트리에서 삽입 연산을 하는 알고리즘
  - 삽입할 자리를 찾기 위해 포인터 p를 사용, 삽입할 노드의 부모 노드를 지정하기 위해 포인터q 를 사용

알고리즘 7-5 이진 탐색 트리의 노드 삽입

```
insertBST(bsT, x)
    p ← bsT
    while (p≠NULL) do {
        if (x = p.key) then return;
        q ← p;
        if (x < p.key) then p ← p.left;
        else p ← p.right;
    }

    new ← getNode();
    new.key ← x;
    new.left ← NULL;
    new.right ← NULL;
```

❶ 삽입할 노드 탐색

❷ 삽입할 노드 생성

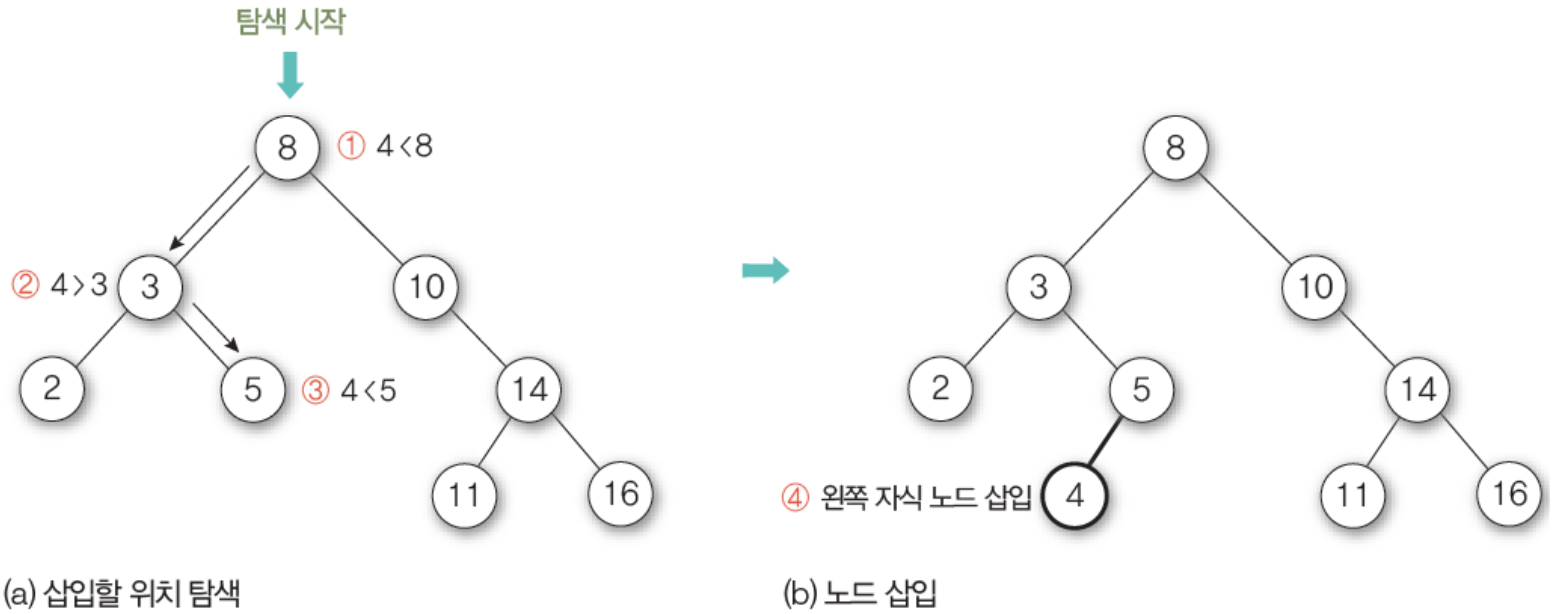
5. 이진 탐색 트리


```
    if (bsT = NULL) then bsT←new;
    else if (x < q.key) then q.left ← new;
    else q.right ← new;
    return;
end insertBST()
```

❸ 삽입 노드 연결

5. 이진 탐색 트리

- 이진 탐색 트리에 원소 4를 삽입 하기



(a) 삽입할 위치 탐색 (b) 노드 삽입

그림 7-35 이진 탐색 트리에서의 삽입 과정 예

① (찾는 키값 4 < 루트 노드의 키값 8)이므로 왼쪽 서브 트리를 탐색

② (찾는 키값 4 > 노드의 키값 3)이므로 오른쪽 서브 트리를 탐색

③ (찾는 키값 4 < 노드의 키값 5)이므로 왼쪽 서브 트리를 탐색해야 하지만, 왼쪽 자식 노드가 없으므로 노드 5의 왼쪽 자식 노드에서 탐색 실패가 발생

④ 실패가 발생한 자리, 즉 노드 5의 왼쪽 자식 노드 자리에 노드 4를 삽입

5. 이진 탐색 트리

- 이진 탐색 트리에 원소 4를 삽입하는 과정을 연결 자료구조로 표현

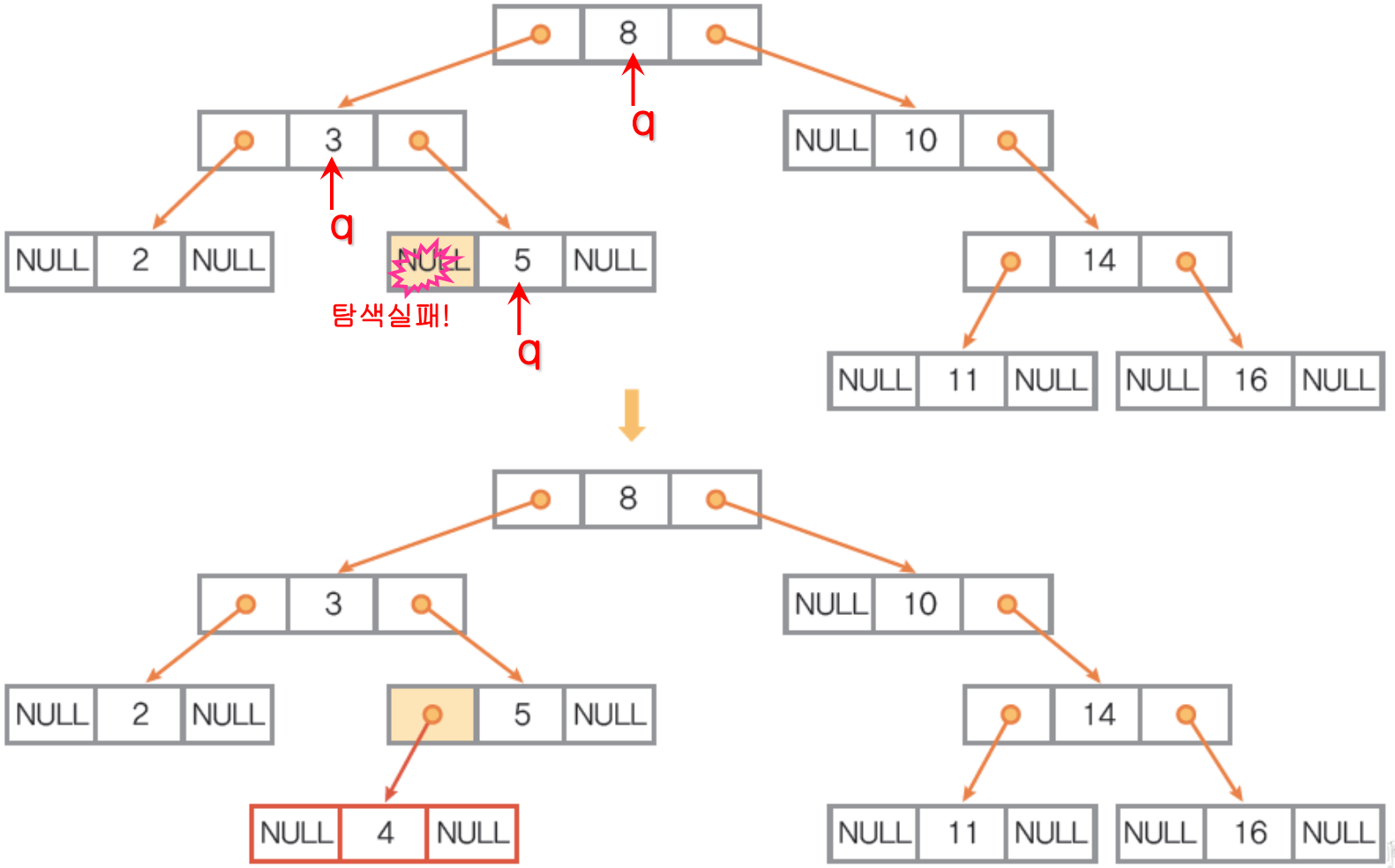


그림 7-36 이진 탐색 트리에 원소 4를 삽입하기 전과 후로 나눠 연결 자료구조로 표현

5. 이진 탐색 트리

❖ 이진 탐색 트리의 삭제 연산

1) 먼저 탐색 연산을 수행

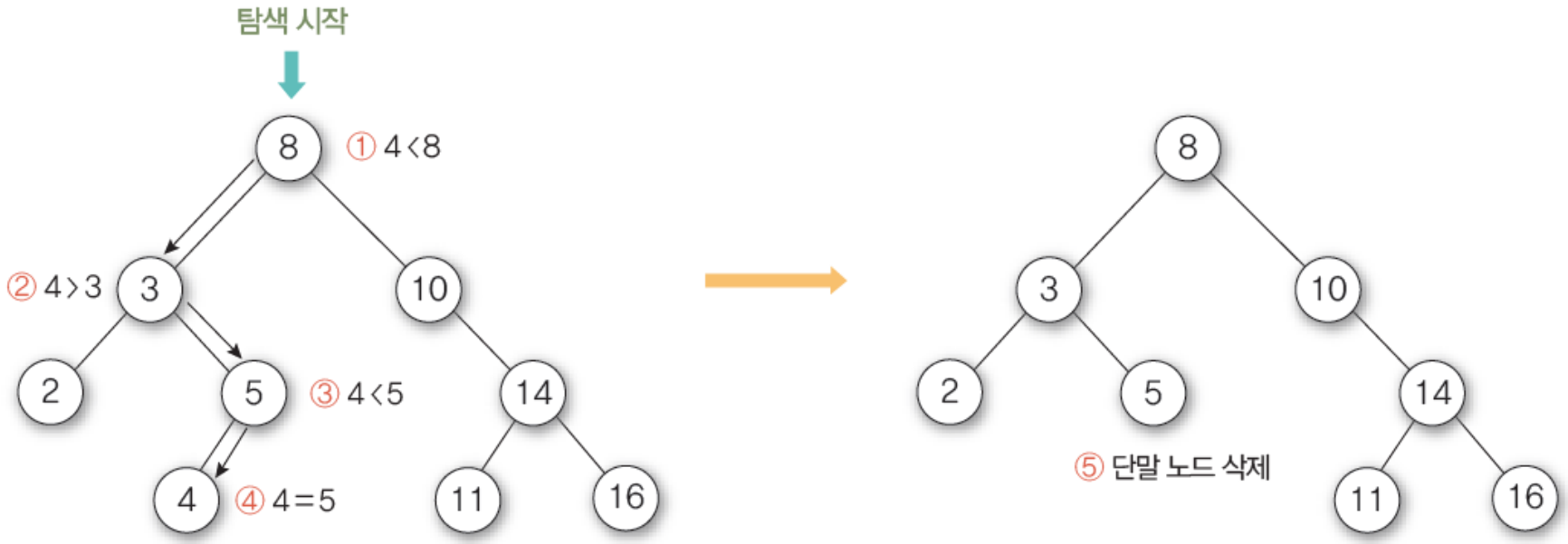
- 삭제할 노드의 위치를 알아야 하므로 트리를 탐색

2) 탐색하여 찾은 노드를 삭제

- 노드의 삭제 후에도 이진 탐색 트리를 유지해야 하므로 삭제 노드의 경우에 대한 후속 처리(이진 탐색 트리의 재구성 작업)가 필요함
  - 삭제할 노드의 경우
    - 삭제할 노드가 단말 노드인 경우(차수 = 0)
    - 삭제할 노드가 자식 노드를 한 개 가진 경우(차수 = 1)
    - 삭제할 노드ㄴ가 자식 노드를 두 개 가진 경우(차수 = 2)

5. 이진 탐색 트리

- 삭제할 노드가 단말 노드인 경우(차수 =0)의 삭제 연산
  - 노드 4를 삭제하는 경우



(a) 삭제할 노드 탐색 (b) 노드 4 삭제

그림 7-37 이진 탐색 트리에서 단말 노드 4를 삭제하는 예

5. 이진 탐색 트리

- 4를 삭제하기 전과 후로 연결 자료구조를 표현

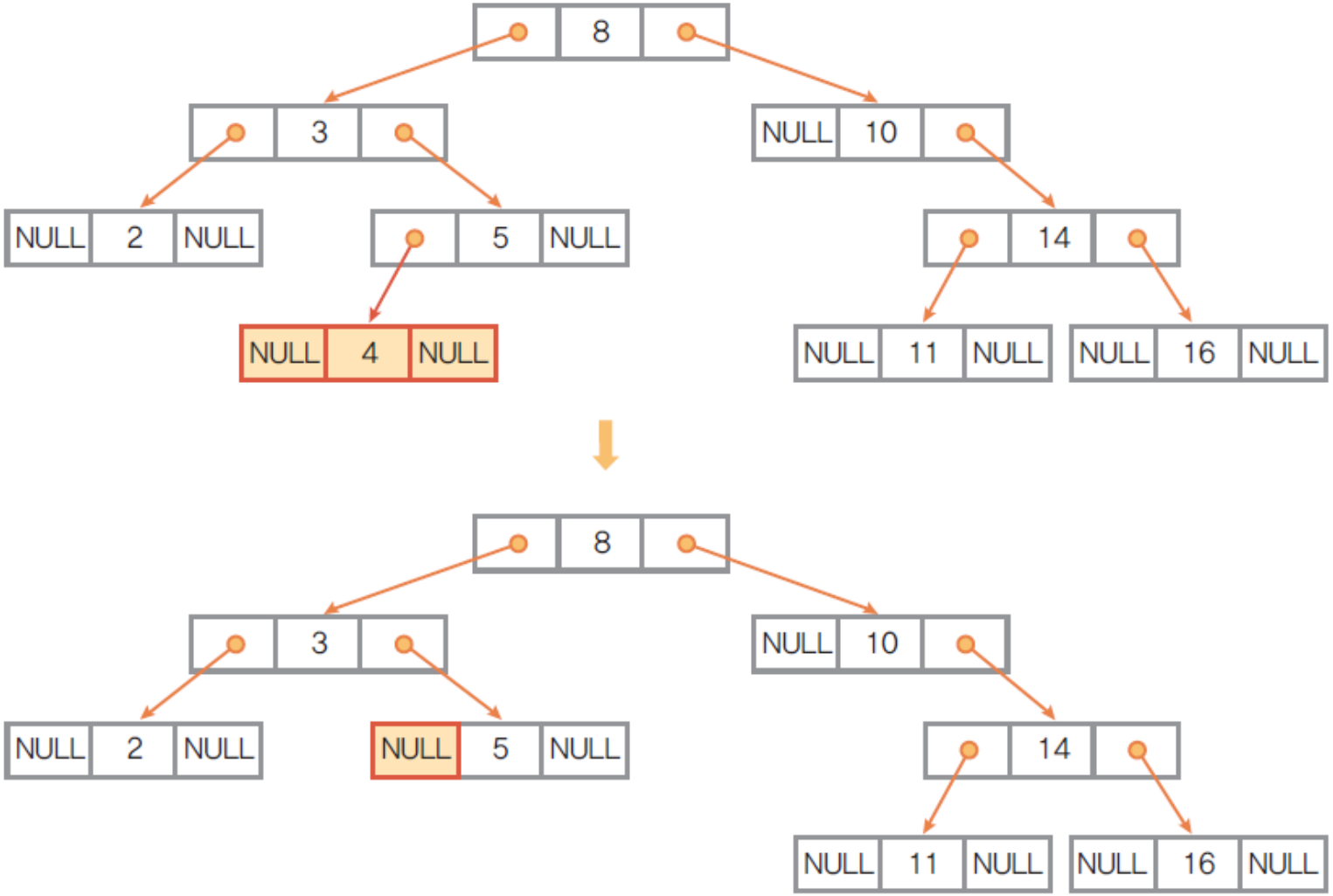
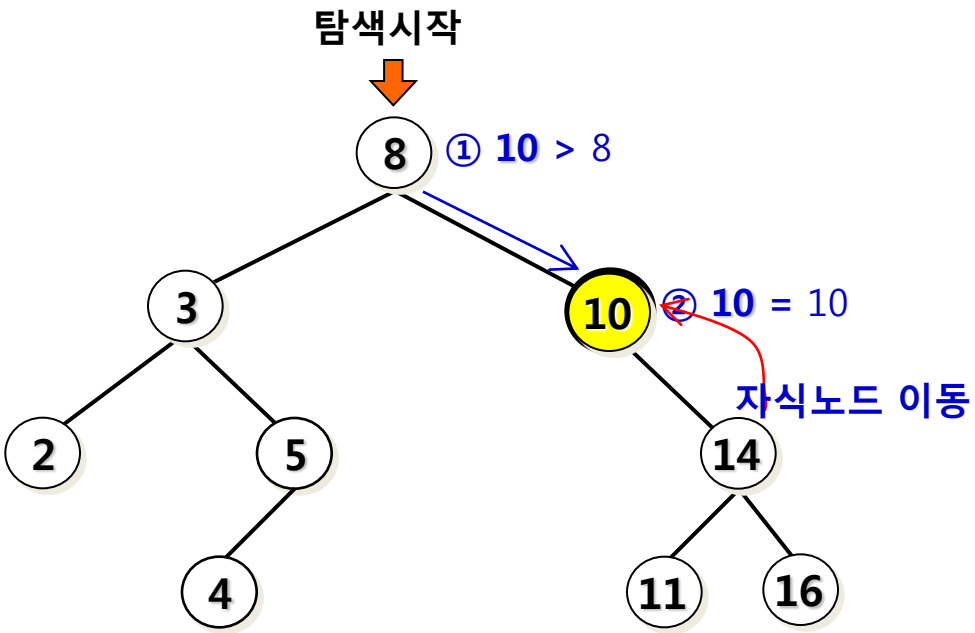


그림 7-38 이진 탐색 트리에서 단말 노드 4를 삭제하기 전과 후로 나눠 연결 자료구조로 표현

5. 이진 탐색 트리

- 삭제할 노드가 자식 노드를 한 개 가진 경우(차수 =1)의 삭제 연산
  - 노드를 삭제하면, 자식 노드는 트리에서 연결이 끊어져서 고아가 됨
  - 후속 처리 : 삭제한 부모노드의 자리를 자식노드에게 물려줌
  - 노드 10을 삭제하는 경우

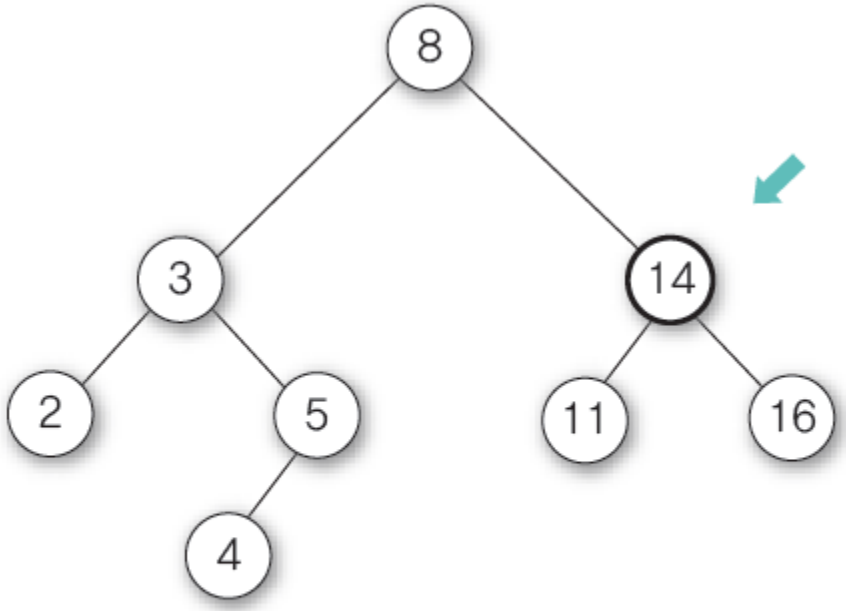


1단계: 삭제할 노드 탐색

2단계: 탐색한 노드 삭제

3단계: 후속처리

5. 이진 탐색 트리



(c) 자식 노드의 위치 조정

그림 7-39 이진 탐색 트리에서 자식 노드가 하나인 노드 10을 삭제하는 예

5. 이진 탐색 트리

- 노드 10을 삭제하는 경우에 대한 단순 연결 리스트 표현

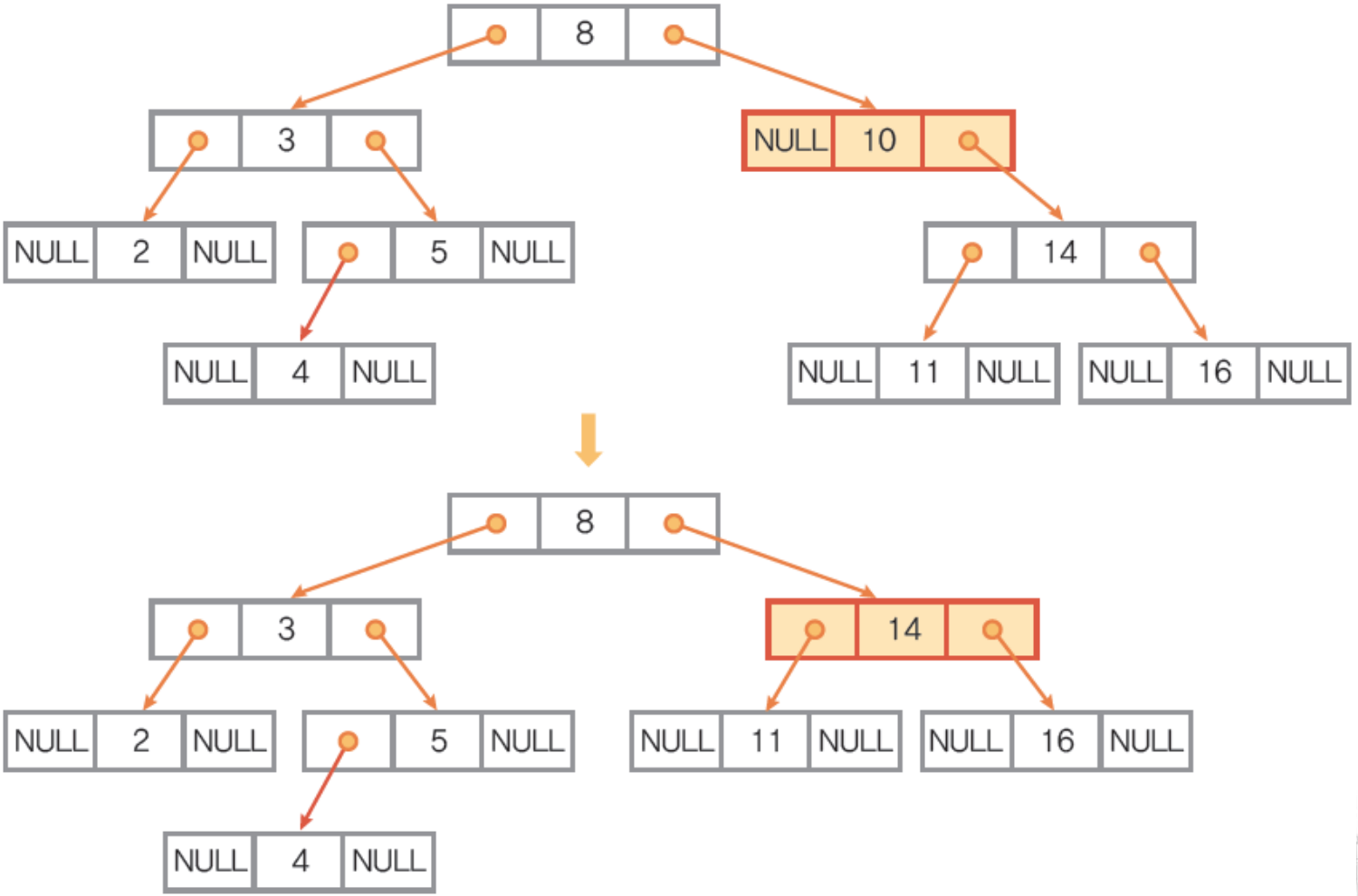


그림 7-40 이진 탐색 트리에서 자식 노드가 하나인 노드 10을 삭제하기 전과 후로 나눠 연결 자료구조로 표현

5. 이진 탐색 트리

- 삭제할 노드가 자식 노드를 두 개 가진 경우(차수 =2)의 삭제 연산
  - 노드를 삭제하면, 자식 노드들은 트리에서 연결이 끊어져서 고아가 됨
  - 후속 처리 : 삭제한 노드의 자리를 자손 노드들 중에서 선택한 후계자에게 물려줌
  - 후계자 선택 방법
    - 왼쪾 서브트리에서 가장 큰 자손노드 선택 : 왼쪽 서브트리의 오른쪽 링크를 따라 계속 이동하여 오른쪽 링크 필드가 NULL인 노드 즉, 가장 오른쪽 노드가 후계자가 됨
    - 오른쪽 서브트리에서 가장 작은 자손노드 선택 : 오른쪽 서브트리에서 왼쪽 링크를 따라 계속 이동하여 왼쪽 링크 필드가 NULL인 노드 즉, 가장 왼쪽에 있는 노드가 후계자가 됨

5. 이진 탐색 트리

- 삭제한 노드의 자리를 물려받을 수 있는 자손 노드

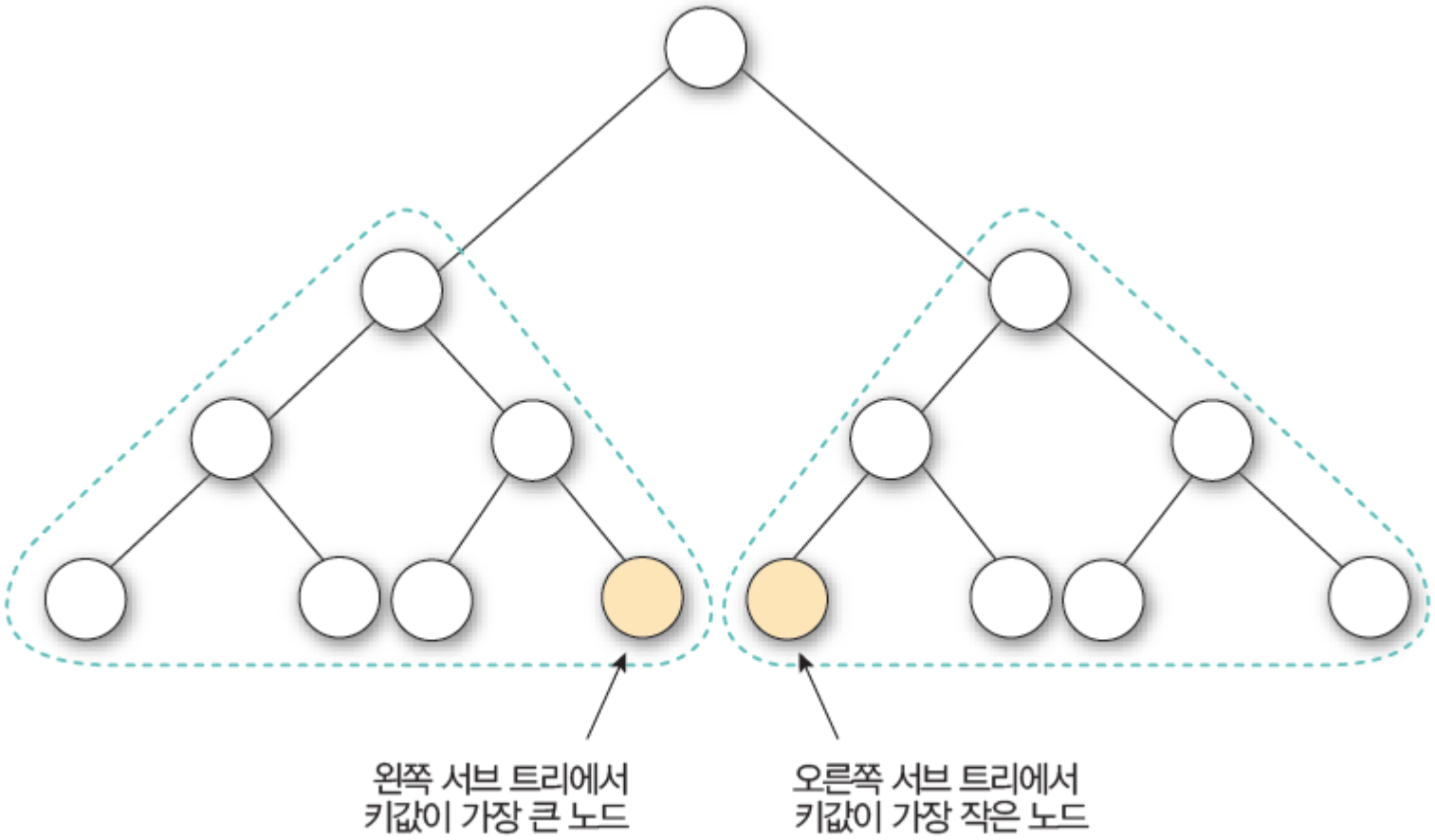
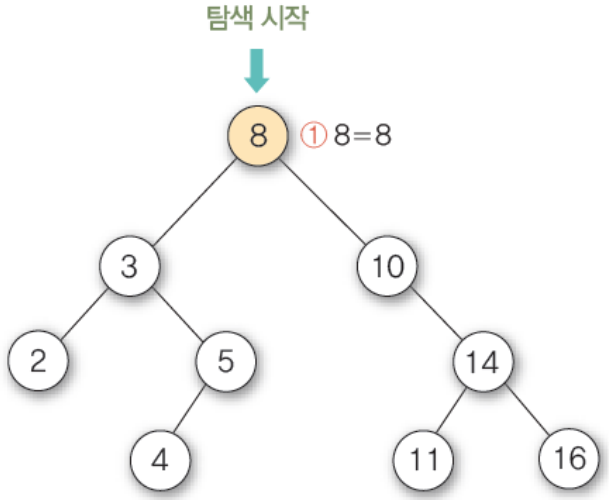


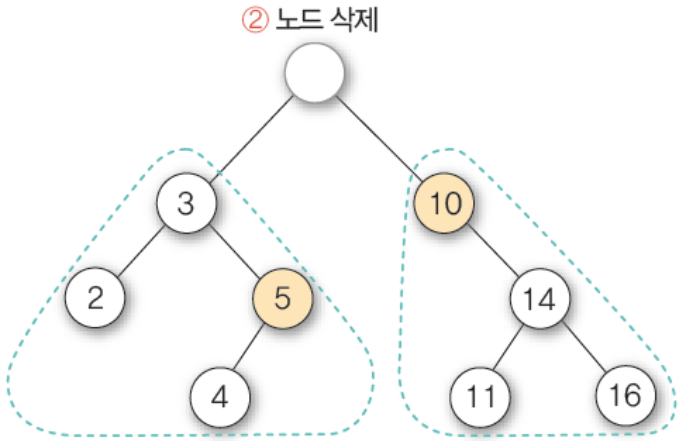
그림 7-41 삭제할 노드의 자리를 물려받을 수 있는 자손 노드

5. 이진 탐색 트리

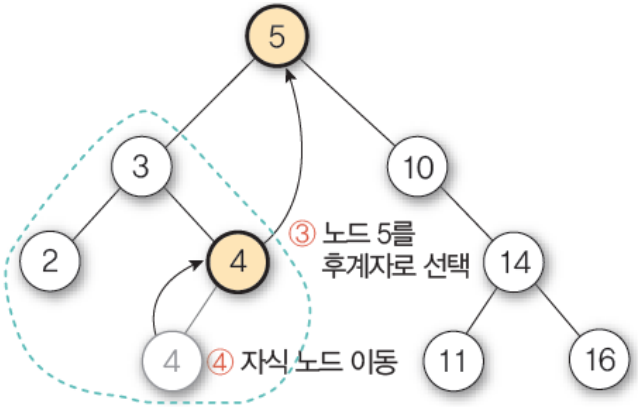
- 노드 8을 삭제하는 경우



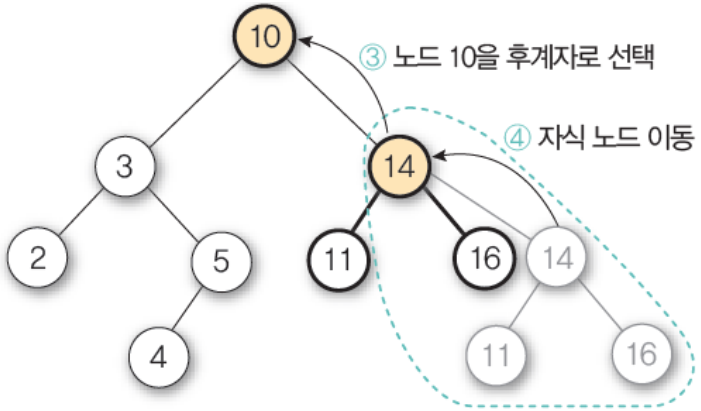
(a) 삭제할 노드 탐색



(b) 노드 삭제



(c) –1 노드 5를 후계자로 선택하는 경우



(c) –2 노드 10을 후계자로 선택하는 경우

그림 7-42 이진 탐색 트리에서 자식 노드가 둘인 노드 8을 삭제하는 예

5. 이진 탐색 트리

- 노드 5를 후계자로 선택한 경우

  ① 후계자 노드 5를 원래자리에서 삭제하여, 삭제노드 8의 자리를 물려줌

  ② 후계자 노드 5의 원래자리는 자식노드 4에게 물려주어 이진 탐색 트리를 재구성 (자식노드가 하나인 노드 삭제 연산의 후속처리 수행)

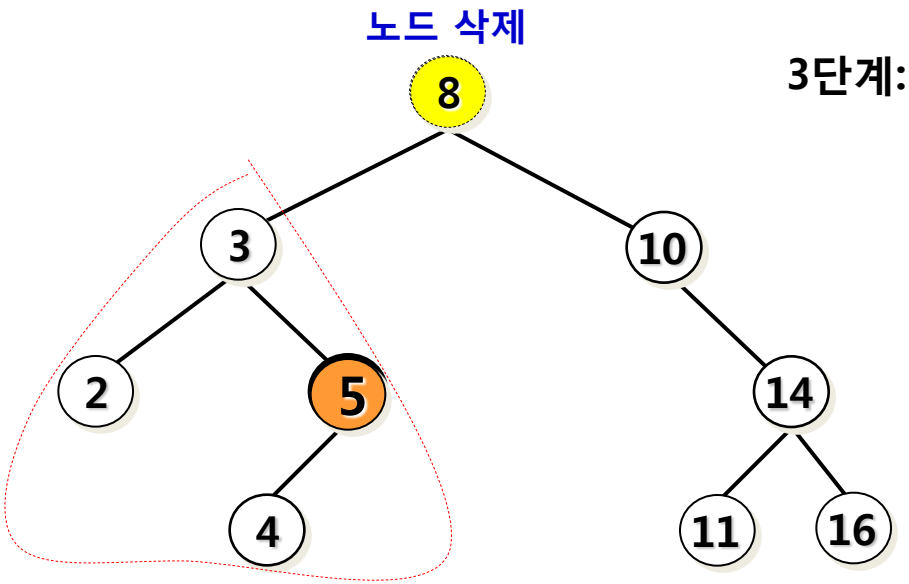
- 노드 10을 후계자로 선택한 경우

  ① 후계자 노드 10을 원래자리에서 삭제하여, 삭제노드 8의 자리를 물려줌

  ② 후계자 노드 10의 원래자리는 자식노드 14에게 물려주어 이진 탐색 트리를 재구성(자식노드가 하나인 노드 삭제 연산의 후속처리 수행)

5. 이진 탐색 트리

- 노드 5를 후계자로 선택한 경우



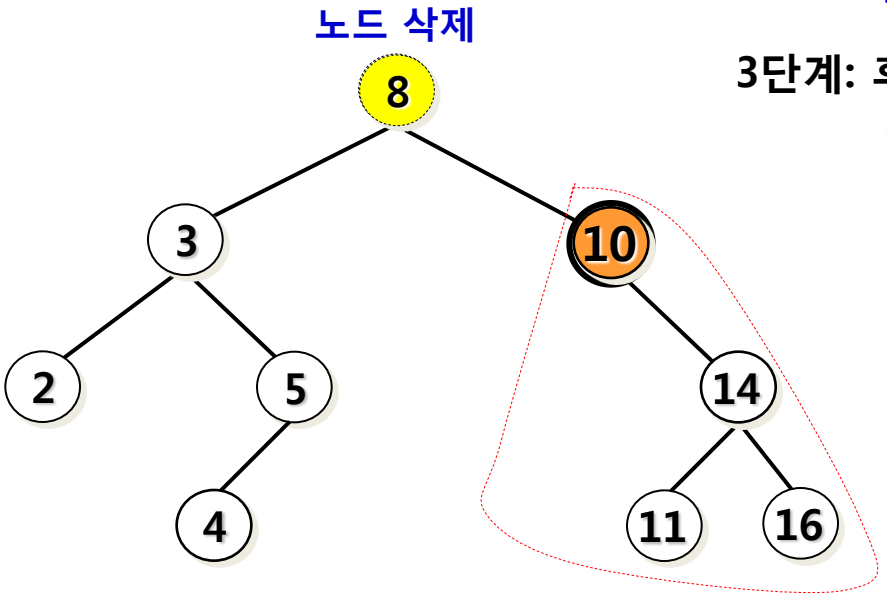
1단계: 노드 삭제

2단계: 삭제한 노드의 자리를
후계자에게 물려주기

3단계: 후계자노드의 원래자리를
자식노드에게 물려주기

5. 이진 탐색 트리

- 노드 8을 후계자로 선택한 경우



1단계: 노드 삭제

2단계: 삭제한 노드의 자리를
후계자에게 물려주기

3단계: 후계자노드의 원래자리를
자식노드에게 물려주기

5. 이진 탐색 트리

- 이진 탐색 트리의 노드 삭제

알고리즘 7-6 이진 탐색 트리의 노드 삭제

```
deleteBST(bsT, x)
    p ← 삭제할 노드;
    parent ← 삭제할 노드의 부모 노드;

    // 삭제할 노드가 없는 경우
    if (p = NULL) then return;

    // 삭제할 노드의 차수가 0인 경우
    if (p.left = NULL and p.right = NULL) then {
        if (parent.left = p) then parent.left ← NULL;
        else parent.right ← NULL;
    }
```

5. 이진 탐색 트리


```
    // 삭제할 노드의 차수가 1인 경우
    else if (p.left = NULL or p.right = NULL) then {
        if (p.left ≠ NULL) then{
            if (parent.left = p) then parent.left ← p.left;
            else parent.right ← p.left;
        }
        else {
            if (parent.left = p) then parent.left ← p.right;
            else parent.right ← p.right;
        }
    }
    // 삭제할 노드의 차수가 2인 경우
    else if (p.left ≠ NULL and p.right ≠ NULL) then {
        q ← maxNode(p.left);      // 왼쪽 서브 트리에서 후계자 노드를 포인터 q로 지정
        p.key ← q.key;
        deleteBST(p.left, p.key); // 후계자 노드 자리에 대한 2차 재구성
    }
end deleteBST()
```