11주차 1차시 이진 탐색 트리 1

【학습목표】

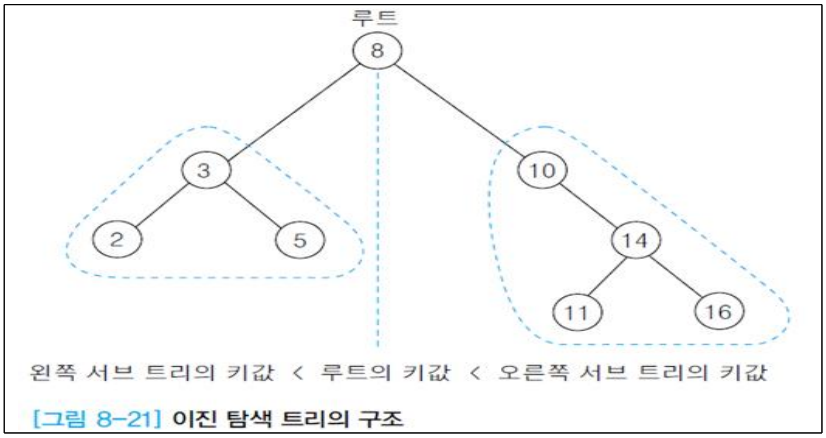
1. 이진 탐색 트리의 탐색, 삽입, 삭제 연산을 구분할 수 있다.
2. 이진 탐색 트리의 연산을 예를 들어 설명할 수 있다.

학습내용1 : 이진 탐색 트리의 탐색연산

1. 이진 탐색 트리(binary search tree)

* 이진 트리에 탐색을 위한 조건을 추가하여 정의한 자료구조

* 이진 탐색 트리의 정의
(1) 모든 원소는 서로 다른 유일한 키를 갖는다
(2) 왼쪽 서브트리에 있는 원소의 키들은 그 루트의 키보다 작다
(3) 오른쪽 서브트리에 있는 원소의 키들은 그 루트의 키보다 크다
(4) 왼쪽 서브트리와 오른쪽 서브트리도 이진 탐색 트리다



[그림 8-21] 이진 탐색 트리의 구조

2. 이진 탐색 트리의 탐색 연산

* 루트에서 시작한다

* 탐색할 킷값 x를 루트 노드의 킷값과 비교한다
- (킷값 x = 루트노드의 킷값)인 경우 : 원하는 원소를 찾았으므로 탐색연산 성공
- (킷값 x < 루트노드의 킷값)인 경우 : 루트노드의 왼쪽 서브트리에서 탐색연산 수행
- (킷값 x > 루트노드의 킷값)인 경우 : 루트노드의 오른쪽 서브트리에서 탐색연산 수행

3. 탐색 연산 알고리즘

알고리즘 8-4 이진 탐색 트리에서의 탐색 연산 알고리즘

```
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( )
```

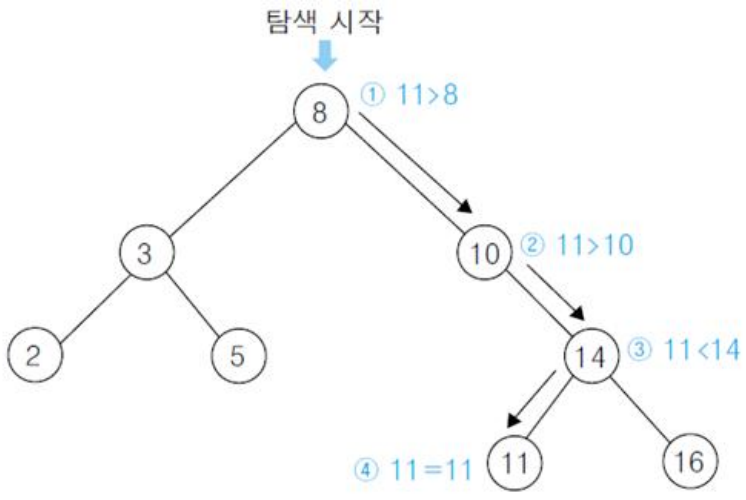
* 탐색 연산 예제 - 원소 11 탐색하기

① 찾는 킷값 11을 루트노드의 킷값 8과 비교 (찾는 킷값 11 > 노드의 킷값 8) 이므로 오른쪽 서브트리를 탐색

② (찾는 킷값 11 > 노드의 킷값 10) 이므로, 다시 오른쪽 서브트리를 탐색

③ (찾는 킷값 11 < 노드의 킷값 14) 이므로, 왼쪽 서브트리를 탐색

④ (찾는 킷값 11 = 노드의 킷값 11) 이므로, 탐색 성공! (연산 종료)



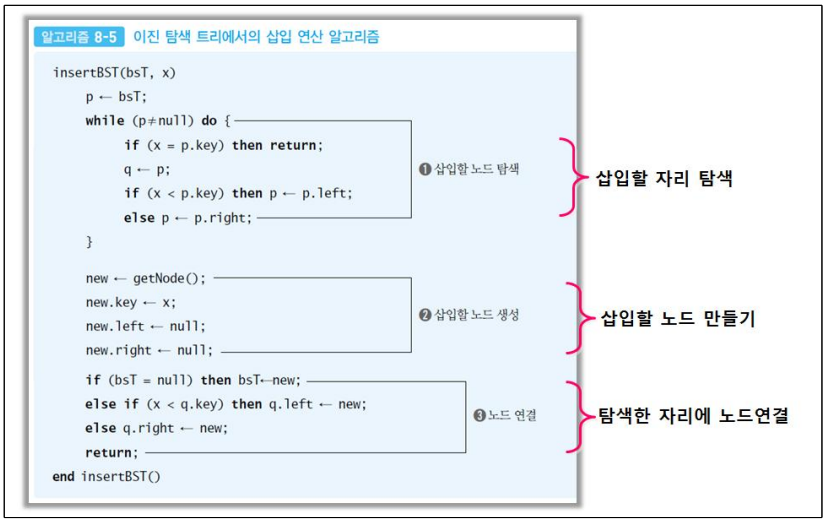
[그림 8-22] 이진 탐색 트리에서의 탐색 과정

학습내용2 : 이진 탐색 트리의 삽입 연산

1. 이진 탐색 트리의 삽입 연산 절차

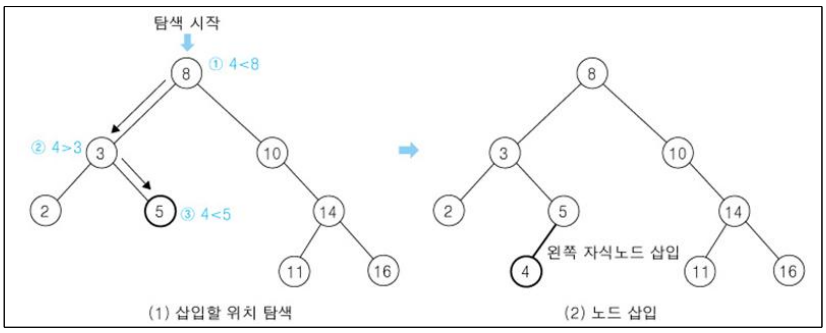
(1) 먼저 탐색 연산을 수행
- 삽입할 원소와 같은 원소가 트리에 있으면 삽입할 수 없으므로 같은 원소가 트리에 있는지 탐색하여 확인한다
- 탐색에서 탐색 실패가 결정되는 위치가 삽입 위치가 된다

(2) 탐색 실패한 위치에 원소를 삽입한다

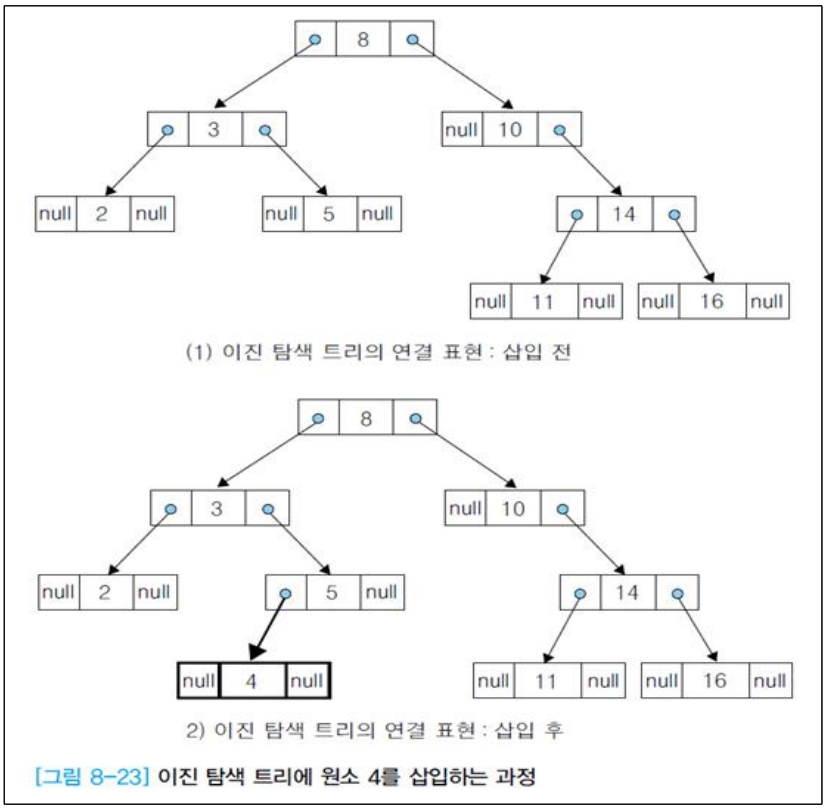


* 삽입 연산 예제 - 원소 4 삽입하기

① 찾는 킷값 4를 루트노드의 킷값 8과 비교하여, (찾는 킷값 4 < 노드의 킷값 8) 이므로, 왼쪽 서브트리를 탐색한다.
② (찾는 킷값 4 > 노드의 킷값 3) 이므로, 오른쪽 서브트리를 탐색한다.
③ (찾는 킷값 4 < 노드의 킷값 5) 이므로, 왼쪽 서브트리를 탐색해야하는데 왼쪽 자식노드가 없으므로 탐색 실패! 이때, 탐색 실패가 결정된 위치 즉, 왼쪽 자식노드의 위치가 삽입 위치가 된다.
④ 탐색작업으로 찾은 자리 즉, 노드 5의 왼쪽 자식노드 자리에 노드 4를 삽입한다.



* 단순 연결 리스트로 표현한 이진 트리에서의 원소 4 삽입하기



[그림 8-23] 이진 탐색 트리에 원소 4를 삽입하는 과정

학습내용3 : 이진 탐색 트리의 삭제 연산

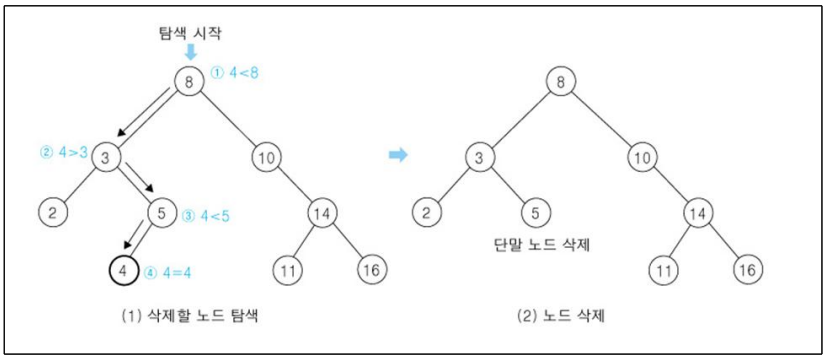
1. 이진 탐색 트리의 삭제 연산 절차

(1) 먼저 탐색 연산을 수행
- 삭제할 노드의 위치를 알아야 하므로 트리를 탐색한다.

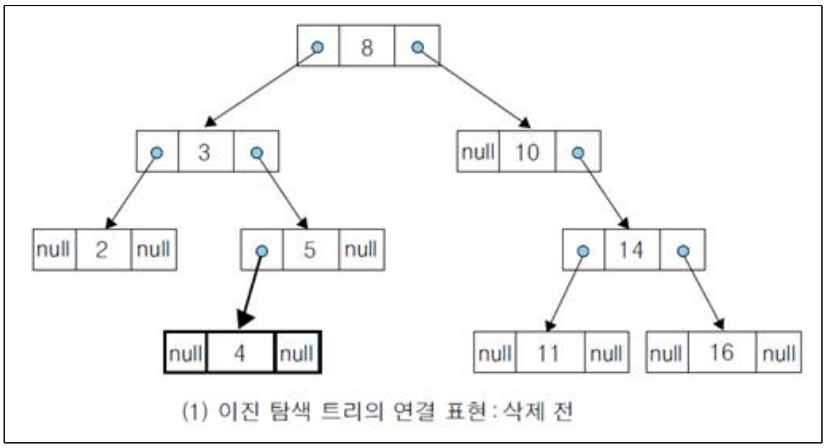
(2) 탐색하여 찾은 노드를 삭제한다.
- 노드의 삭제 후에도 이진 탐색 트리를 유지해야 하므로 삭제 노드의 경우에 대한 후속 처리(이진 탐색 트리의 재구성 작업)가 필요하다.

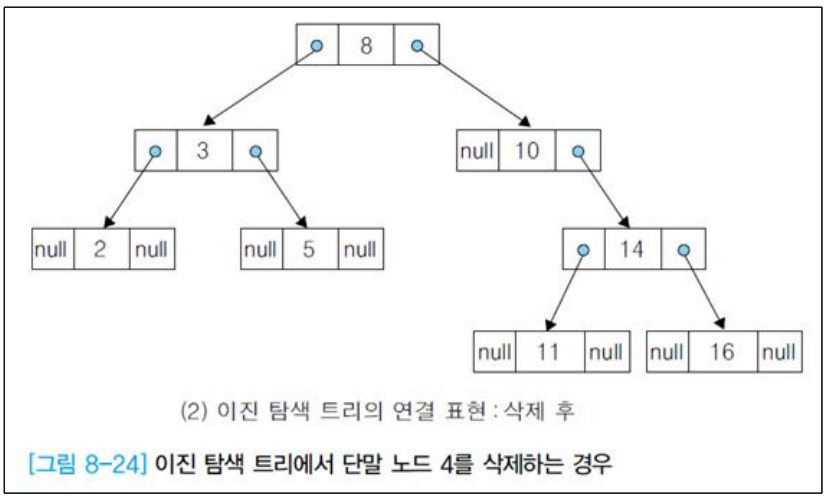
- 삭제할 노드의 경우
    - 삭제할 노드가 단말노드인 경우 : 차수가 0인 경우
    - 삭제할 노드가 하나의 자식노드를 가진 경우 : 차수가 1인 경우
    - 삭제할 노드가 두개의 자식노드를 가진 경우 : 차수가 2인 경우

* 삭제 연산 예제 - 노드 4 삭제하기



* 노드 4를 삭제하는 경우에 대한 단순 연결 리스트 표현
- 노드를 삭제하고 삭제한 노드의 부모 노드의 링크 필드에 null 설정

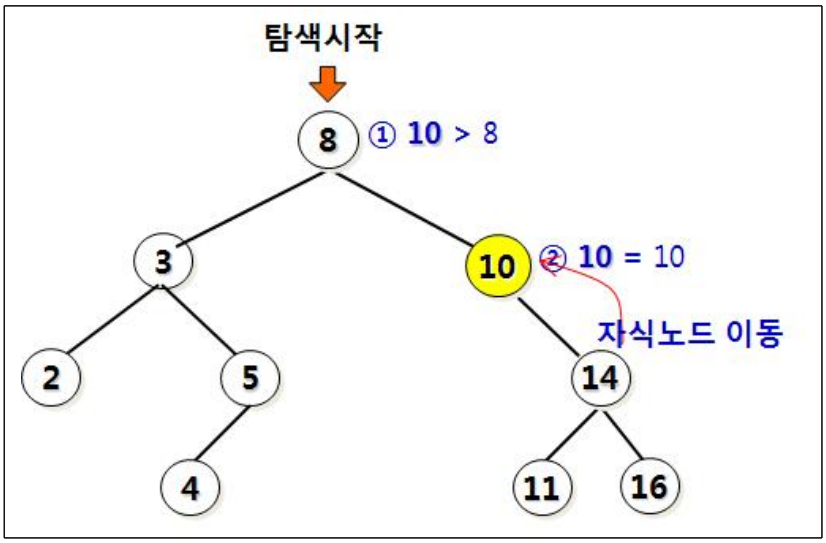


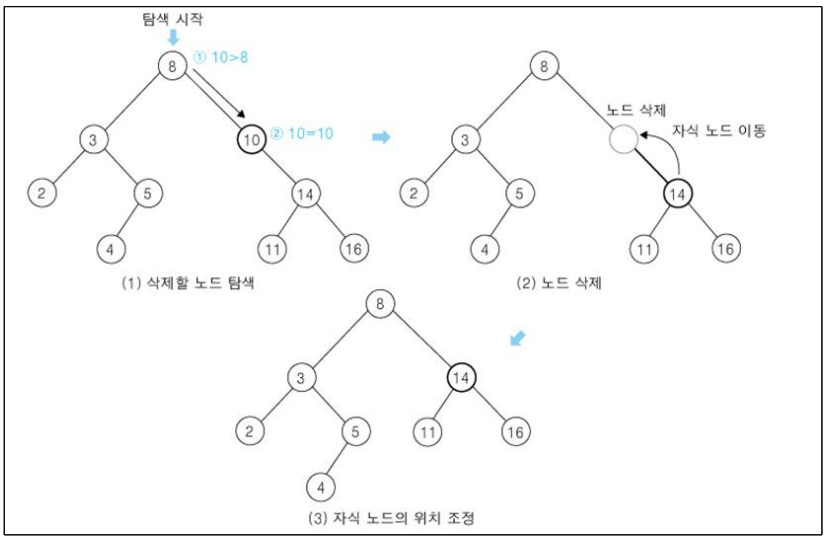


[그림 8-24] 이진 탐색 트리에서 단말 노드 4를 삭제하는 경우

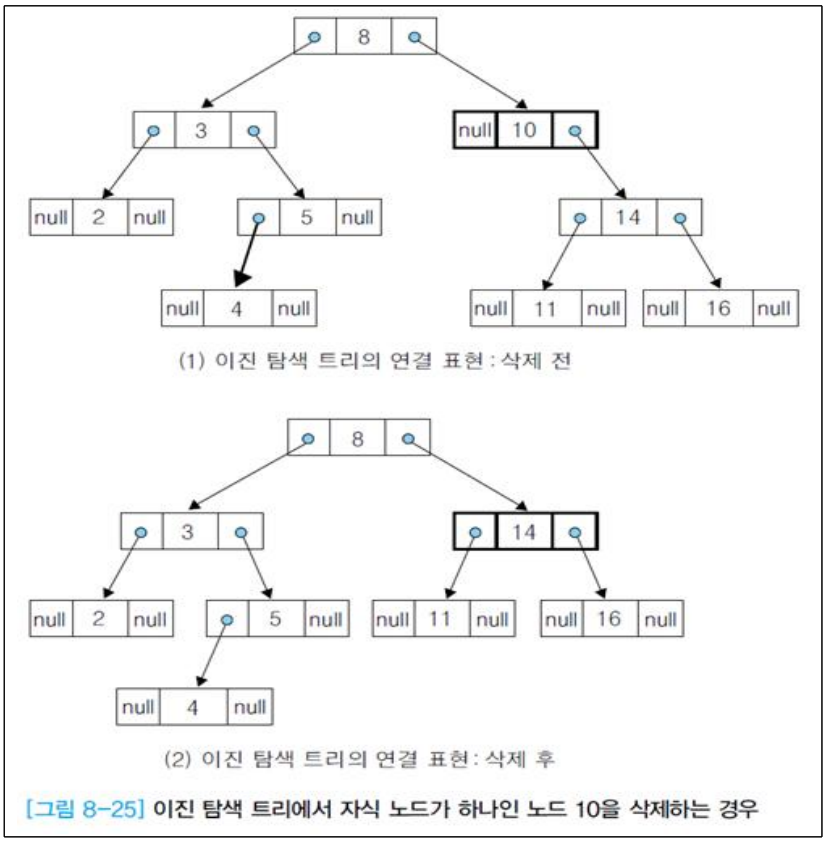
* 삭제 연산 예제 - 자식 노드가 하나인 노드, 즉 차수가 1인 노드의 삭제 연산
- 노드를 삭제하면 자식 노드는 트리에서 연결이 끊어져서 고아가 된다
- 후속처리 : 이진 탐색 트리의 재구성
    - 삭제한 부모노드의 자리를 자식노드에게 물려준다

- 예) 노드 10을 삭제하는 경우
  - 1단계 : 삭제할 노드 탐색
  - 2단계 : 탐색한 노드 삭제
  - 3단계 : 후속처리





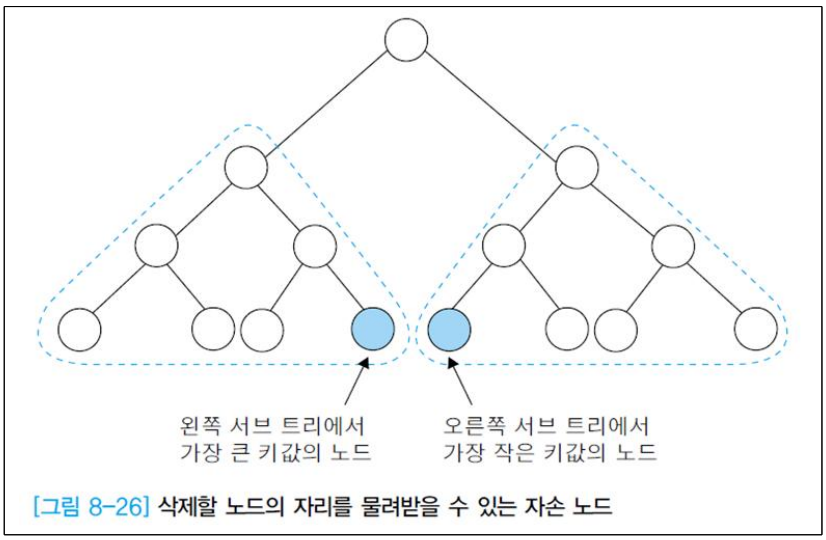
* 예) 노드 10을 삭제하는 경우에 대한 단순 연결 리스트 표현



[그림 8-25] 이진 탐색 트리에서 자식 노드가 하나인 노드 10을 삭제하는 경우

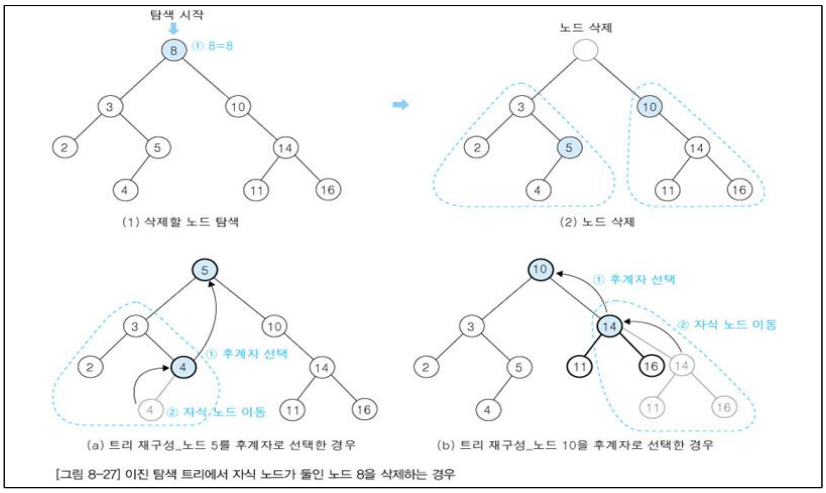
* 삭제 연산 예제 - 자식 노드가 둘인 노드, 즉 차수가 2인 노드의 삭제 연산

- 노드를 삭제하면 자식 노드들은 트리에서 연결이 끊어져서 고아가 된다
- 후속처리 : 이진 탐색 트리의 재구성
  - 삭제한 노드의 자리를 자손 노드들 중에서 선택한 후계자에게 물려준다
- 후계자 선택방법
  - 방법1) 왼쪽 서브트리에서 가장 큰 자손노드 선택
    : 왼쪽 서브트리의 오른쪽 링크를 따라 계속 이동하여 오른쪽 링크 필드가 null인 노드 즉, 가장 오른쪽에 있는 노드가 후계자가 된다
  - 방법2) 오른쪽 서브트리에서 가장 작은 자손노드 선택
    : 오른쪽 서브트리에서 왼쪽 링크를 따라 계속 이동하여 왼쪽 링크 필드가 null인 노드 즉, 가장 왼쪽에 있는 노드가 후계자가 된다



[그림 8-26] 삭제할 노드의 자리를 물려받을 수 있는 자손 노드

* 예) 노드 8을 삭제하는 경우

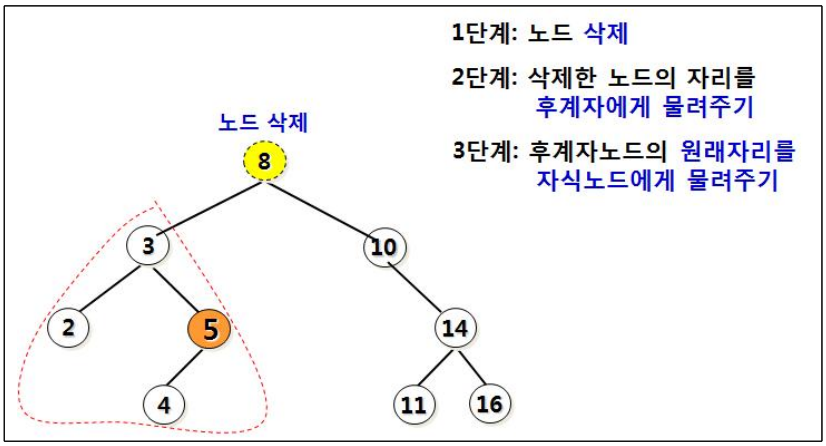


[그림 8-27] 이진 탐색 트리에서 자식 노드가 둘인 노드 8을 삭제하는 경우

- 노드 5를 후계자로 선택한 경우

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

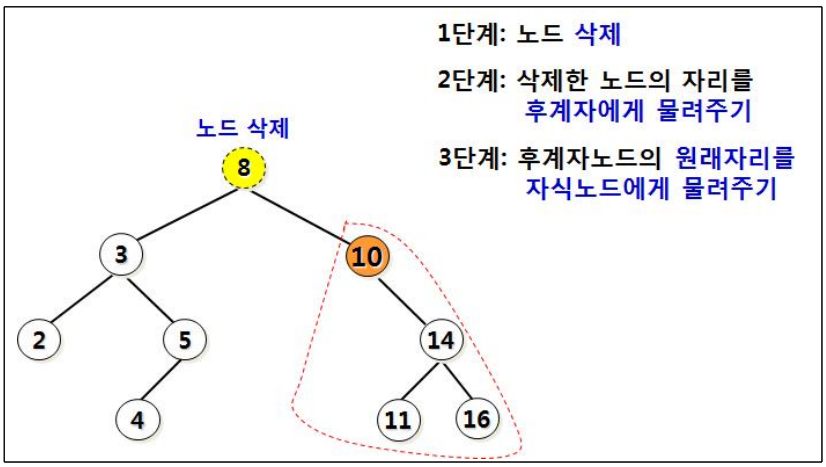
② 후계자 노드 5의 원래자리는 자식노드 4에게 물려주어 이진 탐색 트리를 재구성한다. (☞ 자식노드가 하나인 노드 삭제 연산의 후속처리 수행!)



- 노드 10을 후계자로 선택한 경우

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

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



【학습정리】

1. 이진 트리를 탐색을 위한 자료구조로 사용하기 위해서 저장할 데이터의 크기에 따라 노드의 위치를 정하도록 정의한 이진 탐색트리는 다음의 조건을 만족한다.

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