트리

유튜브 주소 : https://youtu.be/Cm5qB_baqPk

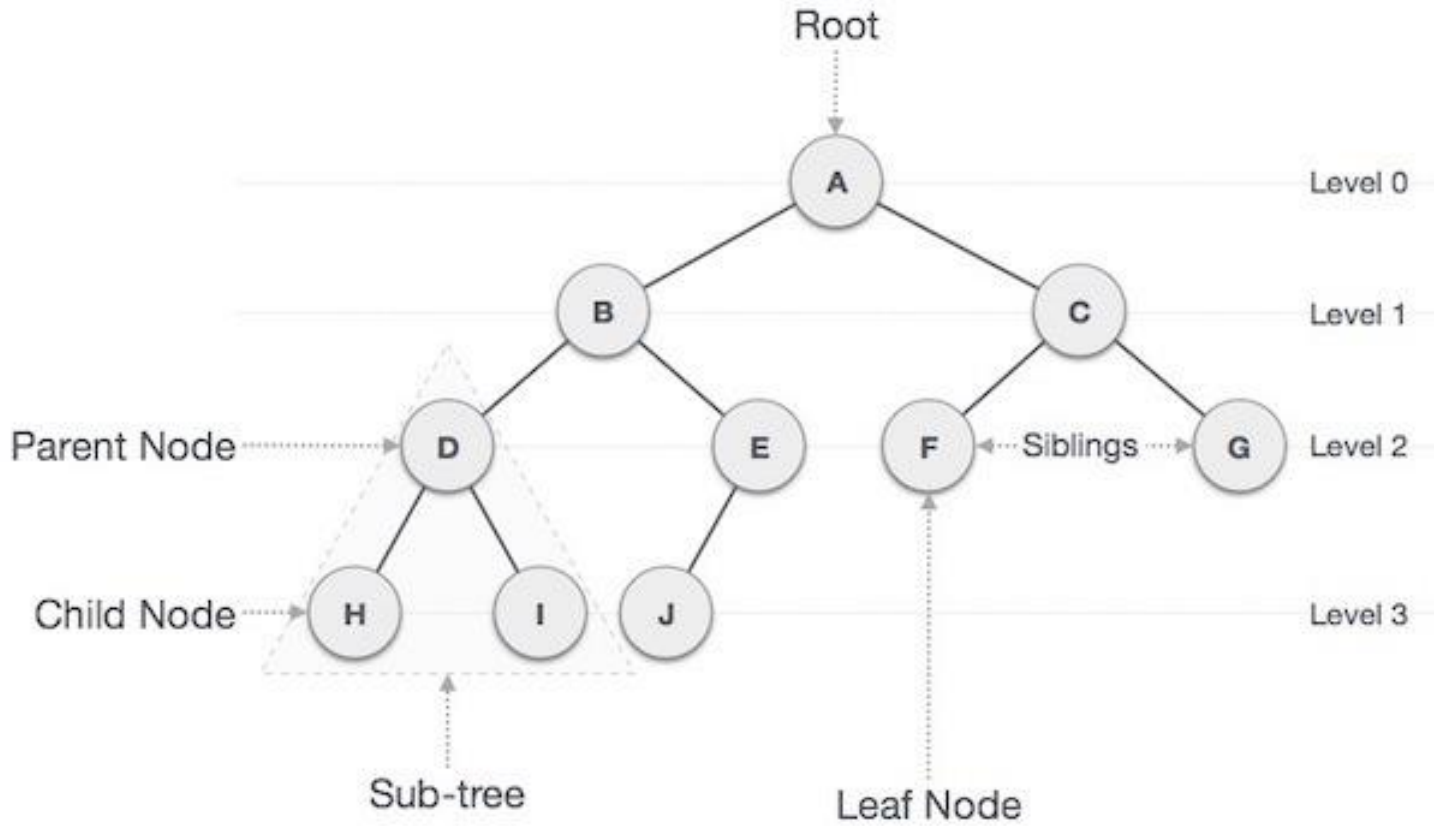
트리 개요

트리 순회 방법

이진 탐색 트리

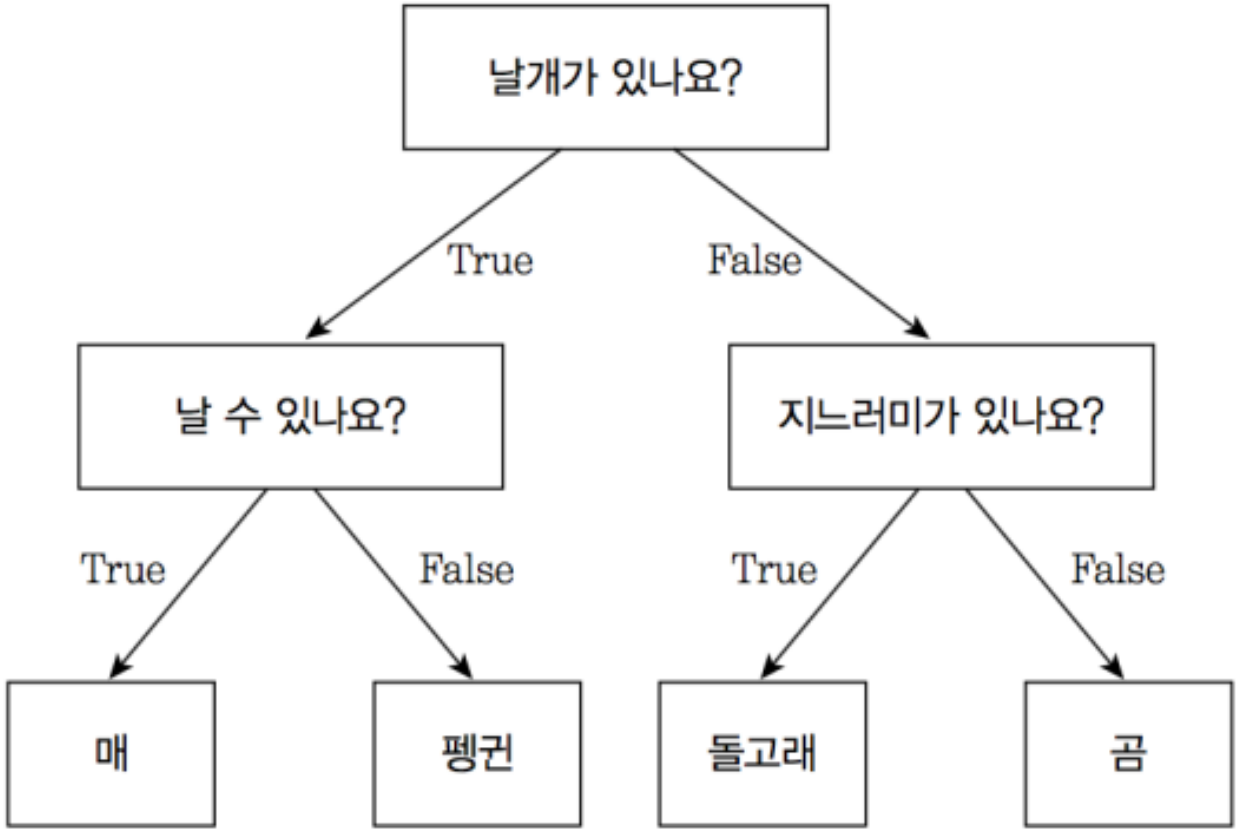
트리 개요

- 트리(tree)
- 계층적 자료 표현에 적합



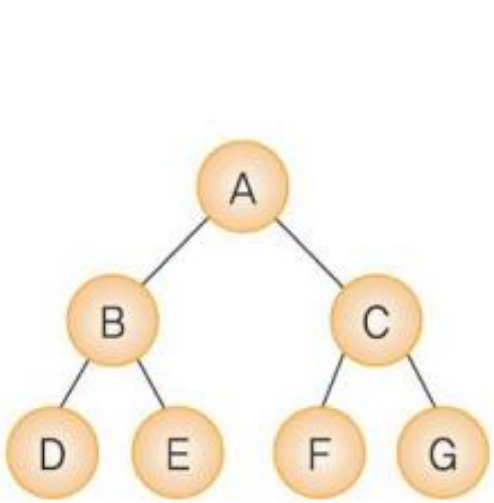
트리 개요

- 결정 트리
- 인공 지능 문제에서 사용

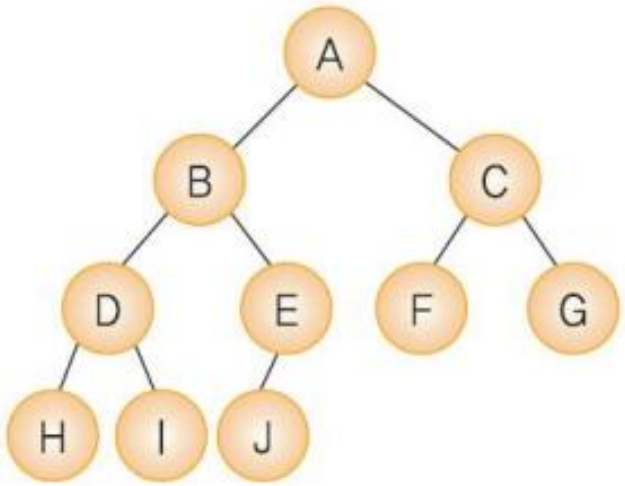


트리 개요

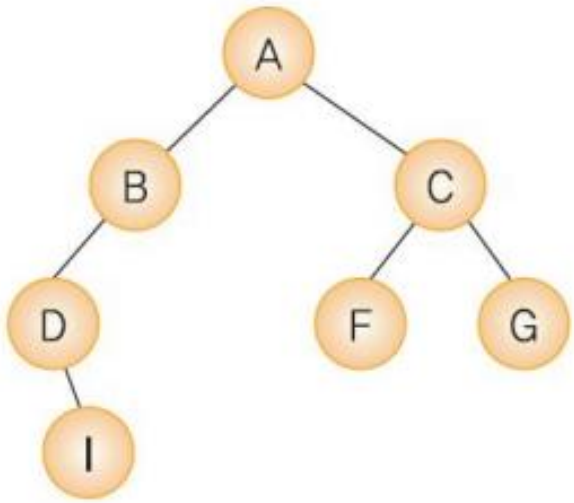
- 이진 트리
- 모든 노드가 두 개의 서브 트리를 지님
- 서브 트리는 공집합일 수 있음
- 3 종류로 구분



(a) 포화 이진 트리



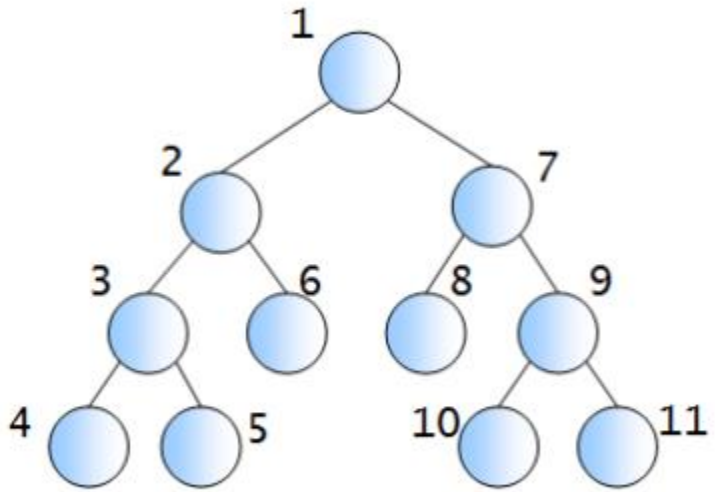
(b) 완전 이진 트리



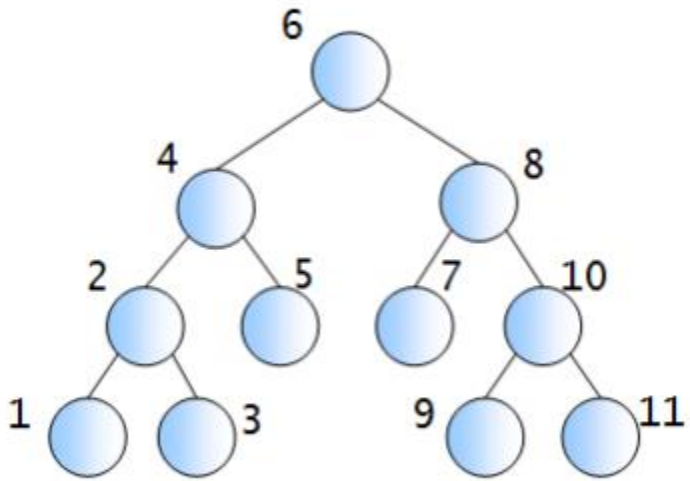
(c) 기타 이진 트리

트리 순회 방법

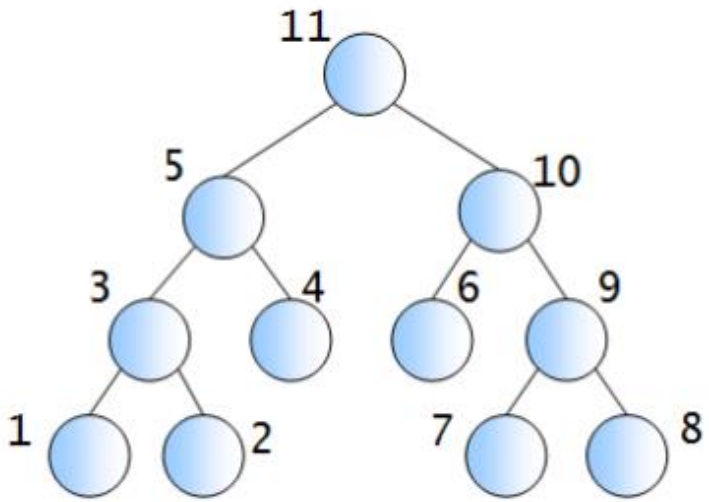
전위 순회



중위 순회

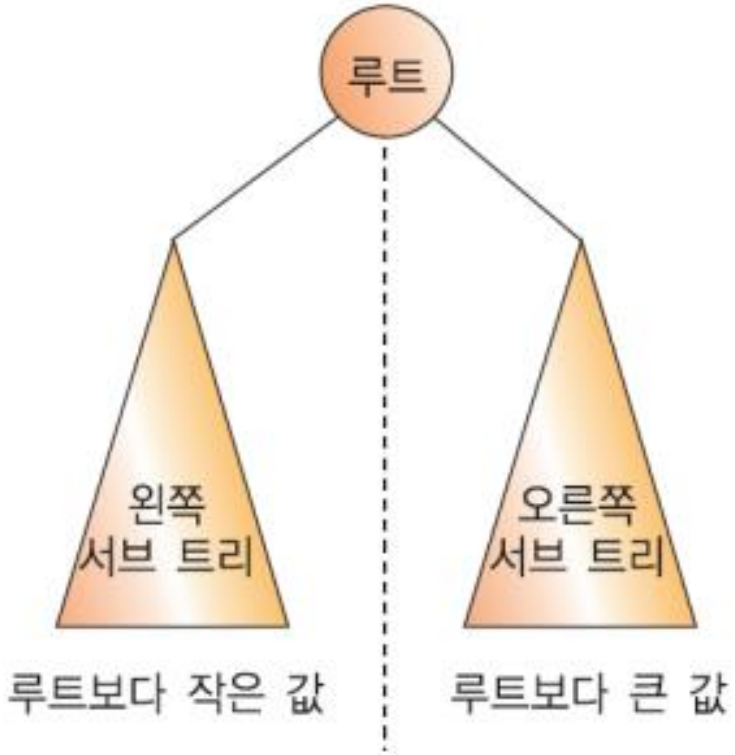


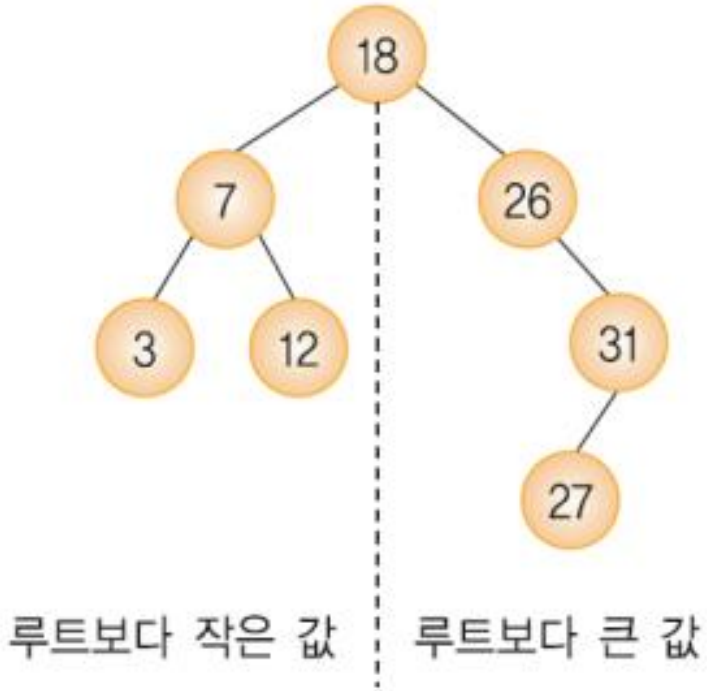
후위 순회



이진 탐색 트리

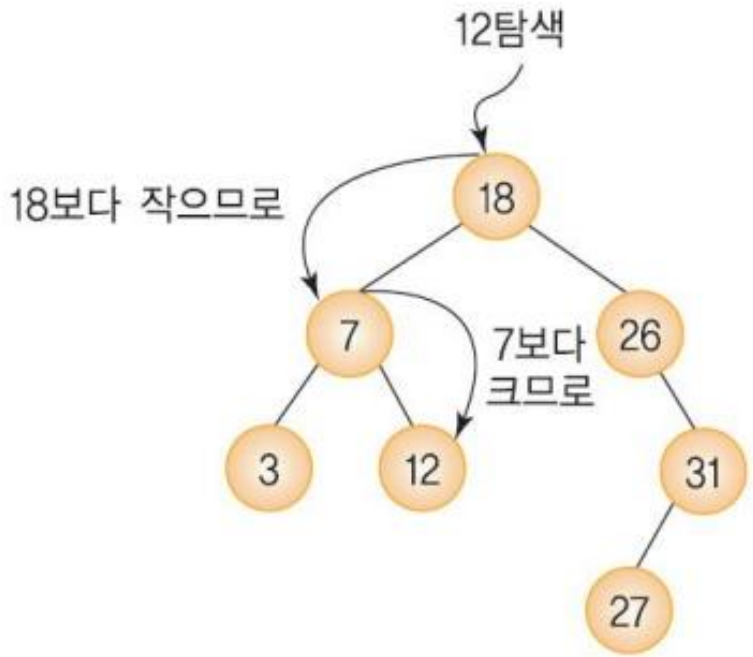
- 효율적인 탐색을 위한 트리
- 중위 순회 시 오름차순으로 정렬된 값 획득 가능



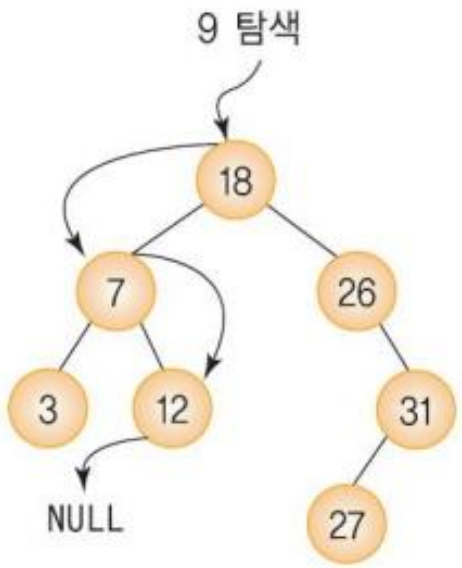


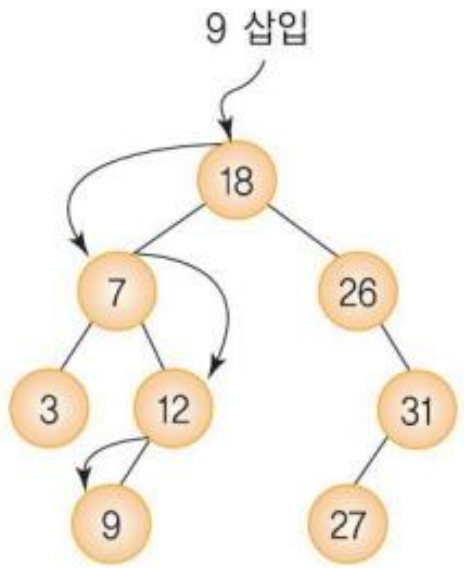
이진 탐색 트리

탐색 연산



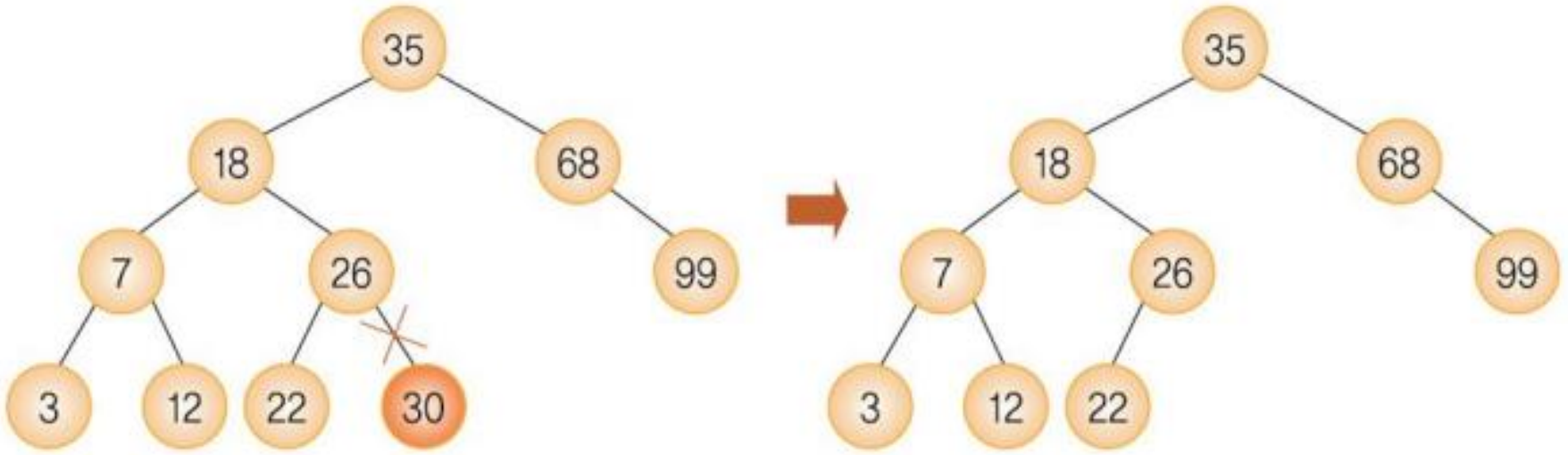
삽입 연산





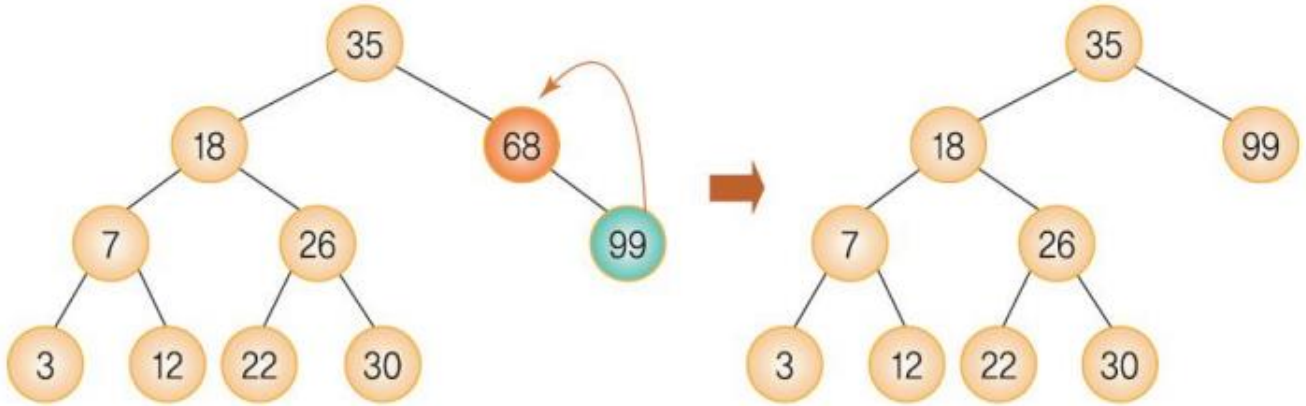
이진 탐색 트리

- 삭제 연산
- 삭제하려는 노드가 단말 노드일 경우

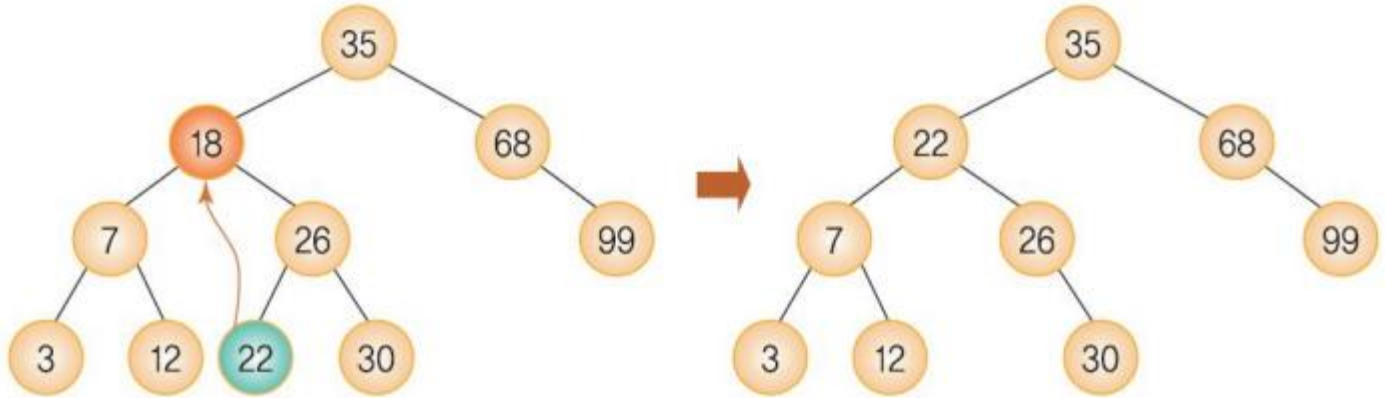


이진 탐색 트리

- 삭제하려는 노드가 왼쪽이나 오른쪽 서브 트리 중 하나만 가지는 경우



- 삭제하려는 노드가 두 개의 서브트리 모두 가지고 있는 경우



이진 탐색 트리 구현 코드 분석

```python
class Node:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None
```

```python
class NodeMgmt:
    def __init__(self, head):
        self.head = head

    def insert(self, value):
        self.current_node = self.head
        while True:
            if value < self.current_node.value:
                if self.current_node.left != None:
                    self.current_node = self.current_node.left
                else:
                    self.current_node.left = Node(value)
                    break
            else:
                if self.current_node.right != None:
                    self.current_node = self.current_node.right
                else:
                    self.current_node.right = Node(value)
                    break
```

이진 탐색 트리 구현 코드 분석

```python
def search(self, value):
    self.current_node = self.head
    while self.current_node:
        if self.current_node.value == value:
            return True
        elif value < self.current_node.value:
            self.current_node = self.current_node.left
        else:
            self.current_node = self.current_node.right
    return False
```

```python
def delete(self, value):
    searched = False
    self.current_node = self.head
    self.parent = self.head
    while self.current_node:
        if self.current_node.value == value:
            searched = True
            break
        elif value < self.current_node.value:
            self.parent = self.current_node
            self.current_node = self.current_node.left
        else:
            self.parent = self.current_node
            self.current_node = self.current_node.right

    if searched == False:
        return False

        # case1
    if self.current_node.left == None and self.current_node.right == None:
        if value < self.parent.value:
            self.parent.left = None
        else:
            self.parent.right = None
```

이진 탐색 트리 구현 코드 분석

```
# case2
elif self.current_node.left != None and self.current_node.right == None:
    if value < self.parent.value:
        self.parent.left = self.current_node.left
    else:
        self.parent.right = self.current_node.left
elif self.current_node.left == None and self.current_node.right != None:
    if value < self.parent.value:
        self.parent.left = self.current_node.right
    else:
        self.parent.right = self.current_node.right


        # case 3
elif self.current_node.left != None and self.current_node.right != None:
    # case3-1
    if value < self.parent.value:
        self.change_node = self.current_node.right
        self.change_node_parent = self.current_node.right
        while self.change_node.left != None:
            self.change_node_parent = self.change_node
            self.change_node = self.change_node.left
        if self.change_node.right != None:
            self.change_node_parent.left = self.change_node.right
        else:
            self.change_node_parent.left = None
        self.parent.left = self.change_node
        self.change_node.right = self.current_node.right
        self.change_node.left = self.change_node.left
```

```
    # case 3-2
    else:
        self.change_node = self.current_node.right
        self.change_node_parent = self.current_node.right
        while self.change_node.left != None:
            self.change_node_parent = self.change_node
            self.change_node = self.change_node.left
        if self.change_node.right != None:
            self.change_node_parent.left = self.change_node.right
        else:
            self.change_node_parent.left = None
        self.parent.right = self.change_node
        self.change_node.right = self.current_node.right
        self.change_node.left = self.current_node.left

return True
```

이진 탐색 트리 구현 코드 분석

```python
import random

# 0 ~ 999 중, 10 개의 숫자 랜덤 선택
bst_nums = set()
while len(bst_nums) != 10:
    bst_nums.add(random.randint(0, 999))
    print(bst_nums)

# 선택된 10개의 숫자를 이진 탐색 트리에 입력, 루트 노드는 임의로 500을 넣음
head = Node(500)
binary_tree = NodeMgmt(head)
for num in bst_nums:
    binary_tree.insert(num)

# 입력한 10개의 숫자 검색 (검색 기능 확인)
for num in bst_nums:
    if binary_tree.search(num) == True:
        print('search ', num)
```

```
{271}
{49, 271}
{49, 358, 271}
{49, 461, 358, 271}
{358, 461, 271, 49, 699}
{358, 461, 271, 49, 690, 699}
{580, 358, 461, 271, 49, 690, 699}
{580, 358, 635, 461, 271, 49, 690, 699}
{450, 580, 358, 635, 461, 271, 49, 690, 699}
{450, 580, 358, 635, 461, 271, 49, 690, 699, 94}
```

```python
# 입력한 10개의 숫자 중 5개의 숫자를 랜덤 선택
delete_nums = set()
bst_nums = list(bst_nums)
while len(delete_nums) != 5:
    delete_nums.add(bst_nums[random.randint(0, 9)])

# 선택한 5개의 숫자를 삭제 (삭제 기능 확인)
for del_num in delete_nums:
    if binary_tree.delete(del_num) == True:
        print('delete', del_num)

# 삭제한 5개의 숫자를 탐색 (삭제 기능 확인)
for del_num in delete_nums:
    if binary_tree.search(del_num) == False:
        print('deleted', del_num)
```

```
search  450
search  580
search  358
search  635
search  461
search  271
search  49
search  690
search  699
search  94
```

```
delete 461
delete 271
delete 690
delete 635
delete 94
deleted 461
deleted 271
deleted 690
deleted 635
deleted 94

Process finished with exit code 0
```

Q & A