

CHAPTER.16

이진 탐색 트리의 일반 구현 및 응용





학습 내용

[1] 이진 탐색 트리의 일반 구현

[2] 이진 탐색 트리의 응용



학습 목표

- ④ 이진 트리 중 활용도가 높은 이진 탐색 트리를 이해하고 활용할 수 있다.
- ④ 이진 트리를 활용하여 다양한 응용 프로그램을 작성할 수 있다.

01

이진 탐색 트리의 일반 구현

- 1) 이진 탐색 트리의 특징
- 2) 이진 탐색 트리의 생성
- 3) 이진 탐색 트리에서 데이터를 삽입하는 일반적인 형태
- 4) 이진 탐색 트리에서 데이터 검색
- 5) 이진 탐색 트리에서 데이터 검색

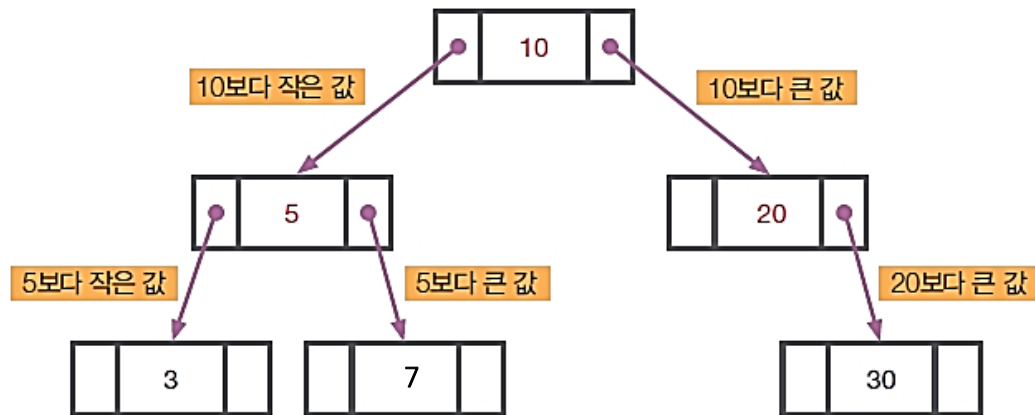


1] 이진 탐색 트리의 특징



이진 트리 중 활용도가 높은 트리

- 데이터 크기를 기준으로 일정 형태로 구성
- 이진 탐색 트리의 대표적인 형태



이진 탐색 트리의 특징

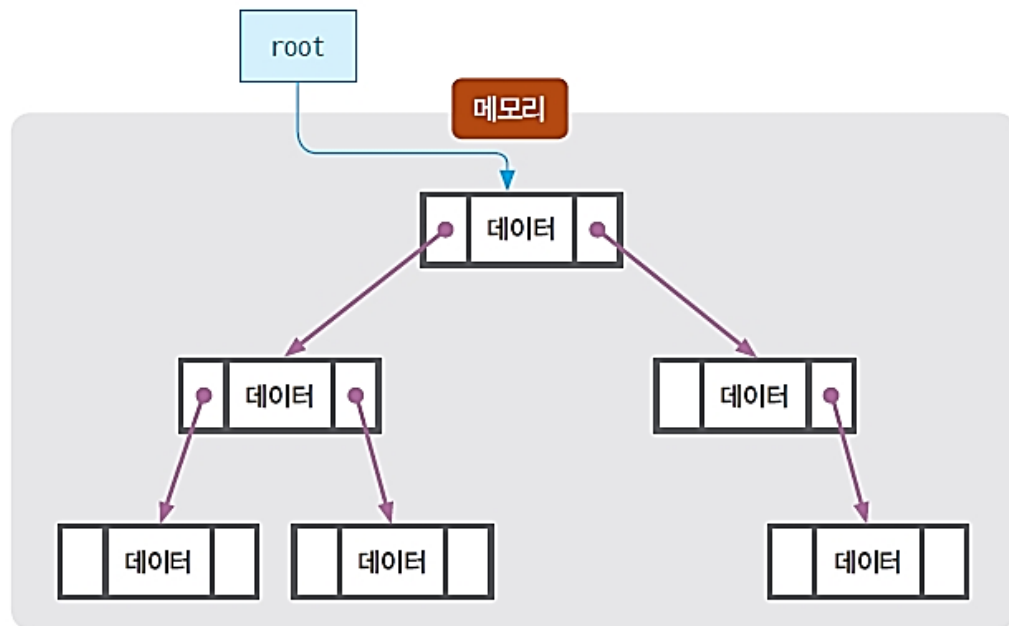
- 1 왼쪽 서브 트리는 루트 노드보다 모두 작은 값을 가짐
- 2 오른쪽 서브 트리는 루트 노드보다 모두 큰 값을 가짐
- 3 각 서브 트리도 ①, ② 특징을 가짐
- 4 모든 노드 값은 중복되지 않음

▪ 즉, 중복된 값은 이진 탐색 트리에 저장할 수 없음

2] 이진 탐색 트리의 생성



생성할 이진 탐색 트리



2] 이진 탐색 트리의 생성



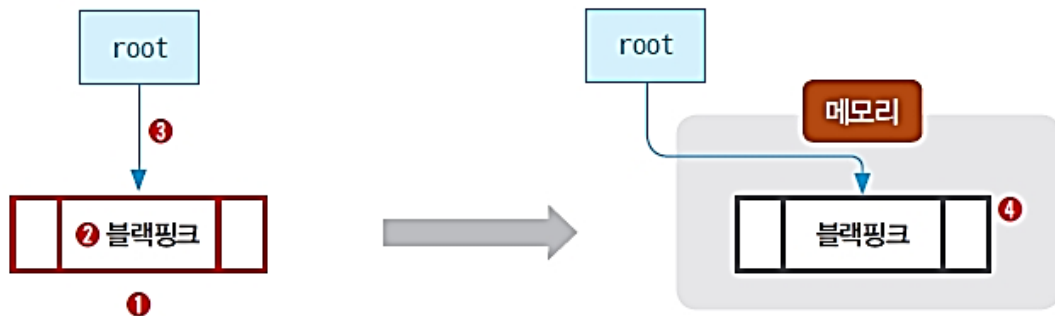
1 메모리를 준비하고 root는 None으로 초기화

```
memory = []  
root = None
```

2 배열에 있는 데이터를 차례대로 이진 탐색 트리에 삽입

```
nameArr = ['블랙핑크', '레드벨벳', '에이핑크', '걸스데이', '트와이스', '마마무']
```

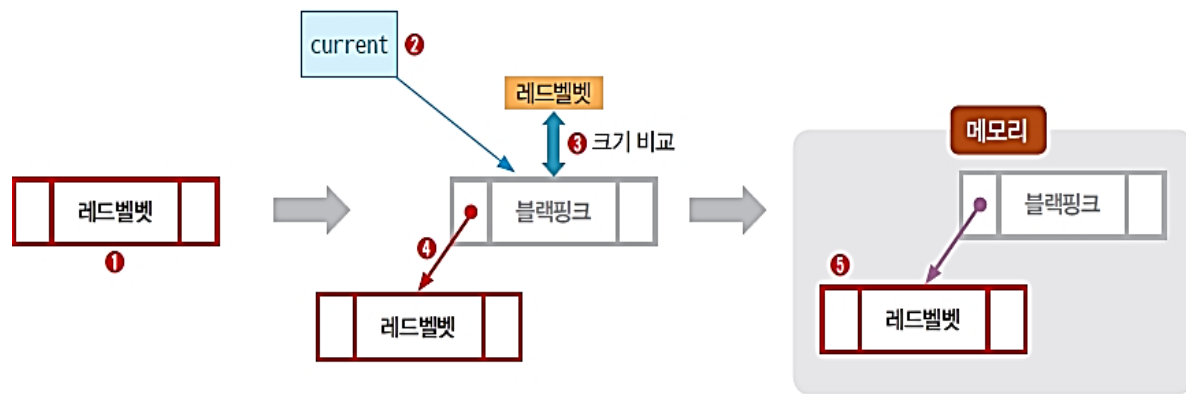

3 첫 번째 데이터 삽입



- ① node = TreeNode() # 노드 생성
- ② node.data = nameAry[0] # 데이터 입력
- ③ root = node # 첫 번째 노드를 루트 노드로 지정
- ④ memory.append(node) # 생성한 노드를 메모리에 저장

4 두 번째 이후 데이터 삽입

▪ 두 번째 데이터 삽입



4 두 번째 이후 데이터 삽입

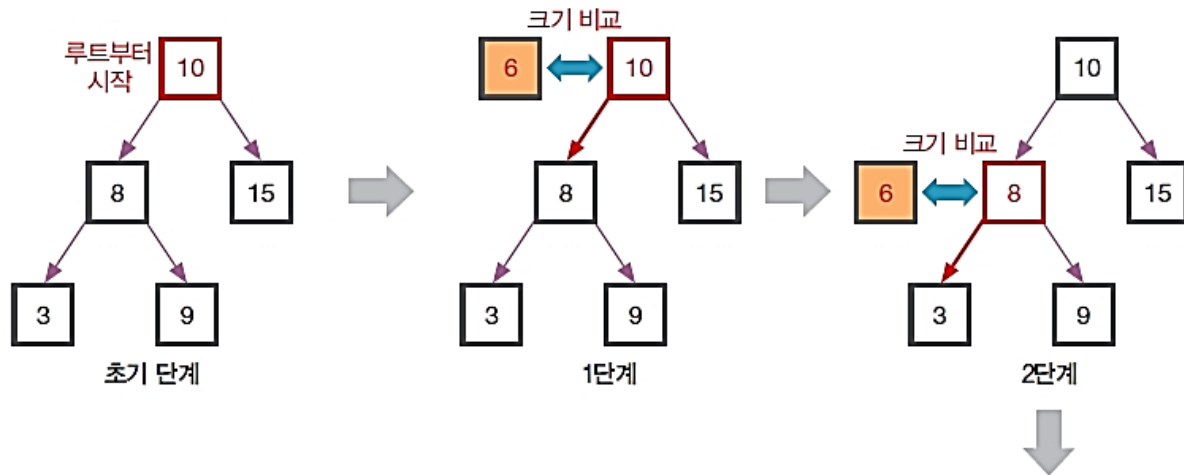
▪ 두 번째 데이터 삽입

1	name = '레드벨벳'	# 두 번째 데이터
	node = TreeNode()	# 새 노드 생성
	node.data = name	# 새 노드에 데이터 입력
	current = root	# 현재 작업 노드를 루트 노드로 지정
	if name < current.data :	# 입력할 값을 현재 작업 노드의 값과 비교
2~4	current.left = node	# 작으면 새 노드를 왼쪽 링크로 연결
	else :	
	current.right = node	# 크면 새 노드를 오른쪽 링크로 연결
5	memory.append(node)	# 새 노드를 메모리에 저장

3] 이진 탐색 트리에서 데이터를 삽입하는 일반적인 형태



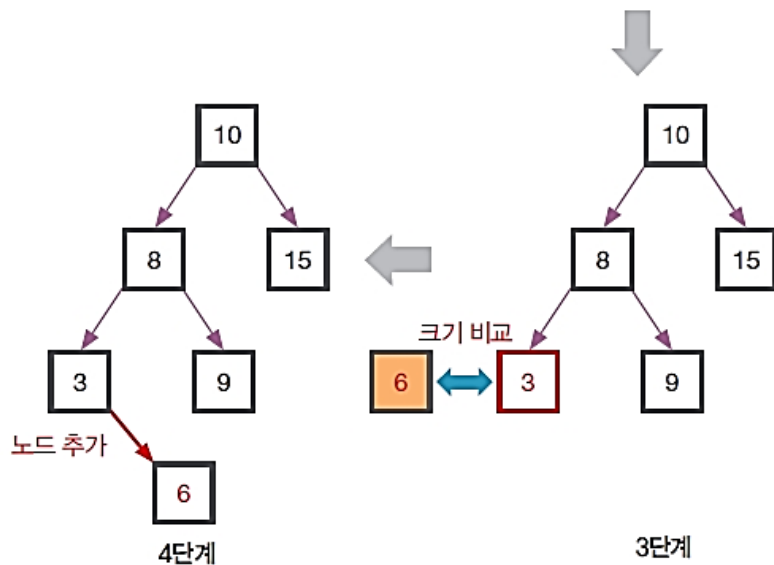
레벨 2의 초기 상태인 이진 탐색 트리에 6 데이터를 삽입하는 과정



3] 이진 탐색 트리에서 데이터를 삽입하는 일반적인 형태



레벨 2의 초기 상태인 이진 탐색 트리에 6 데이터를 삽입하는 과정



3] 이진 탐색 트리에서 데이터를 삽입하는 일반적인 형태



앞 페이지의 그림과 같이 반복적으로 자리를 찾아가는 과정을
코드 형태로 구현

```
name = 6                                # 위치를 찾을 새 데이터
node = TreeNode()                       # 새 노드 생성
node.data = name

❶ current = root                        # 루트부터 시작
❷ while True :                          # 무한 반복
    ❸ if name < current.data :           # 1단계
        {
            if current.left == None :   # 4단계
                ❹ current.left = node   # 4단계
                break
        }
    ❺ current = current.left             # 2~3단계
```

3] 이진 탐색 트리에서 데이터를 삽입하는 일반적인 형태



앞 페이지의 그림과 같이 반복적으로 자리를 찾아가는 과정을
코드 형태로 구현

```
else :  
    {  
        if current.right == None : # 4단계  
        ⑥    current.right = node    # 4단계  
        break  
    }  
    current = current.right
```

이진 탐색 트리에서 데이터를 삽입하는 과정 통합의 코드

■ 이진 탐색 트리의 삽입 작동

```
1  ## 함수 선언 부분 ##
2  class TreeNode() :
3      def __init__(self) :
4          self.left = None
5          self.data = None
6          self.right = None
7
8  ## 전역 변수 선언 부분 ##
9  memory = []
10 root = None
11 nameAry = ['블랙핑크', '레드벨벳', '마마무', '에이핑크', '걸스데이', '트와이스']
12
```


이진 탐색 트리에서 데이터를 삽입하는 과정 통합의 코드

■ 이진 탐색 트리의 삽입 작동

```
13 ## 메인 코드 부분 ##
14 node = TreeNode()
15 node.data = nameAry[0]
16 root = node
17 memory.append(node)
18
19 for name in nameAry[1:] :
20
21     node = TreeNode()
22     node.data = name
23
24     current = root
25     while True :
```

이진 탐색 트리에서 데이터를 삽입하는 과정 통합의 코드

■ 이진 탐색 트리의 삽입 작동

```
26         if name < current.data :
27             if current.left == None :
28                 current.left = node
29                 break
30             current = current.left
31         else :
32             if current.right == None :
33                 current.right = node
34                 break
35             current = current.right
36
```

이진 탐색 트리에서 데이터를 삽입하는 과정 통합의 코드

■ 이진 탐색 트리의 삽입 작동

```
37     memory.append(node)
38
39 print("이진 탐색 트리 구성 완료!")
```

실행 결과

이진 탐색 트리 구성 완료!

4] 이진 탐색 트리에서 데이터 검색



```
1 { if 현재 작업 노드의 데이터 == 찾을 데이터 :  
   탐색 종료  
2 { elif 현재 작업 노드의 데이터 > 찾을 데이터 :  
   왼쪽 서브 트리 탐색  
3 { else :  
   오른쪽 서브 트리 탐색
```

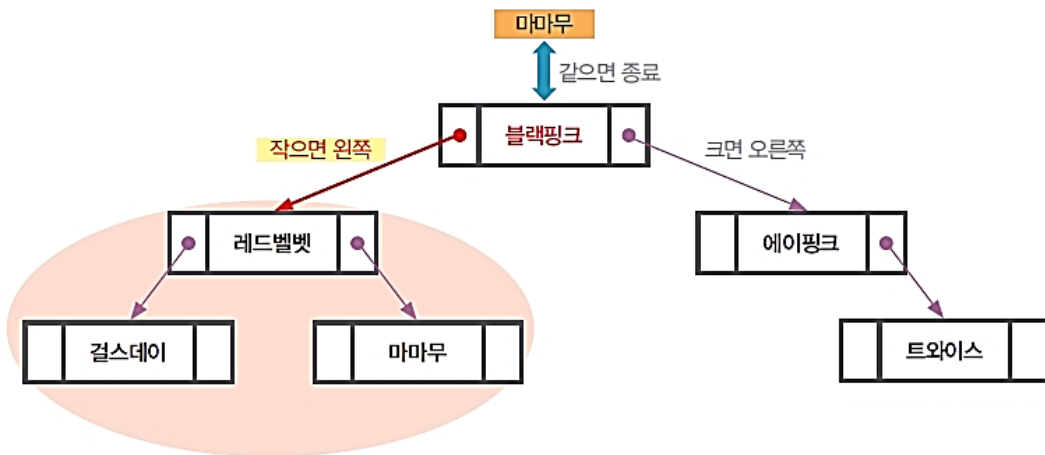
4] 이진 탐색 트리에서 데이터 검색



완성된 이진 탐색 트리에서 '마마무'를 찾는 예

1 찾고자 하는 '마마무'를 루트 노드의 데이터와 비교

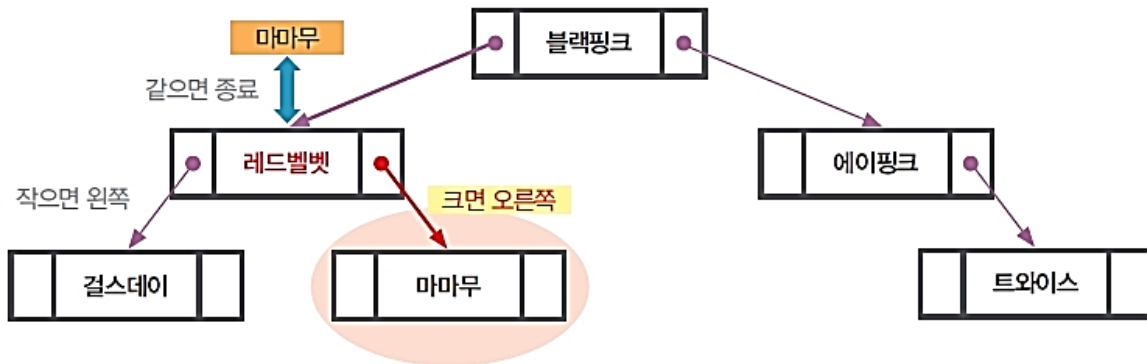
- '마마무'가 루트 노드의 데이터보다 작아 왼쪽으로 이동



완성된 이진 탐색 트리에서 '마마무'를 찾는 예

2 왼쪽 서브 트리에서도 동일하게 처리

- 찾고자 하는 '마마무'가 왼쪽 서브 트리의 루트 노드보다 커 오른쪽으로 이동



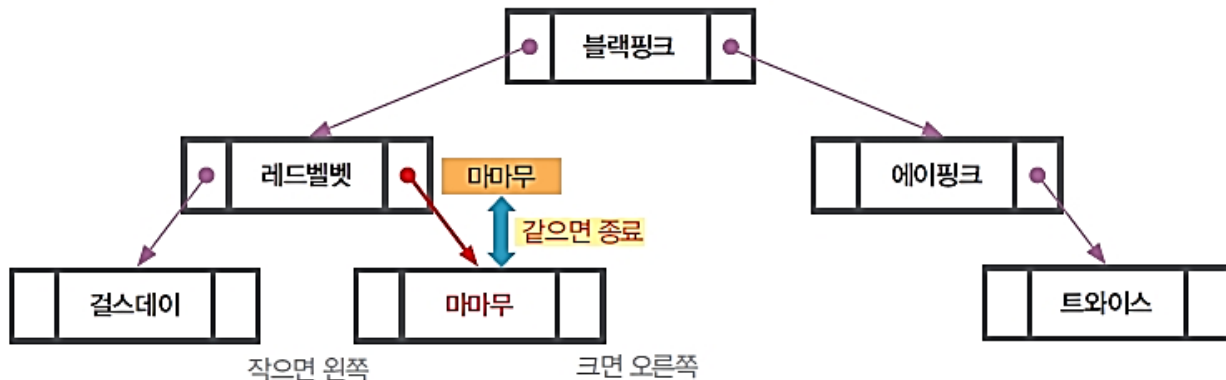
4] 이진 탐색 트리에서 데이터 검색



완성된 이진 탐색 트리에서 '마마무'를 찾는 예

3 오른쪽 서브 트리에서도 동일하게 처리

- 여기에서는 '마마무'를 찾았으므로 종료



이진 탐색 트리의 검색 작동

```
1  ## 함수 선언 부분 ##  
... # 생략(    앞쪽의    2~37행과 동일)  
38  
39 findName = '마마무'  
40  
41 current = root  
42 while True :  
43     if findName == current.data :  
44         print(findName, '을(를) 찾음.')  
45         break  
46     elif findName < current.data :  
47         if current.left == None :  
48             print(findName, '이(가) 트리에 없음')  
49             break
```


이진 탐색 트리의 검색 작동

```
50     current = current.left
51     else :
52         if current.right == None :
53             print(findName, '이(가) 트리에 없음')
54             break
55         current = current.right
```

실행 결과

마마무 을(를) 찾음.

4] 이진 탐색 트리에서 데이터 검색



이진 탐색 트리에서 데이터 검색 실습

앞쪽의 소스를 수정해서 nameAry에 잇지와 여자친구를 추가하자. 그리고 검색할 이름을 input() 함수로 입력받은 후 검색하도록 하자.

실행 결과

찾을 그룹이름-->잇지

잇지 을(를) 찾았음.

찾을 그룹이름-->소녀시대

소녀시대 이(가) 트리에 없음

4] 이진 탐색 트리에서 데이터 검색

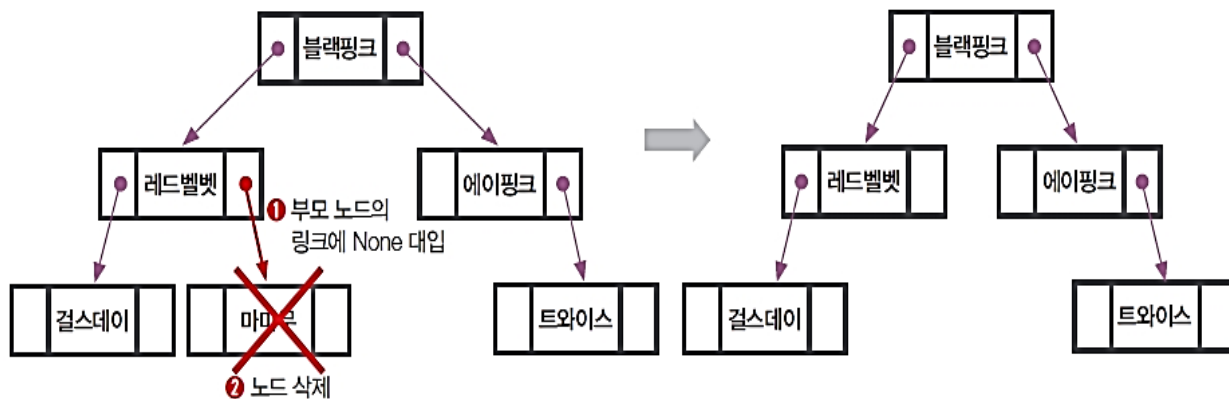


이진 탐색 트리에서 데이터 검색 실습



1 리프 노드 (맨 아래쪽 노드)를 삭제하는 경우

▪ 리프 노드 삭제



1 리프 노드(맨 아래쪽 노드)를 삭제하는 경우

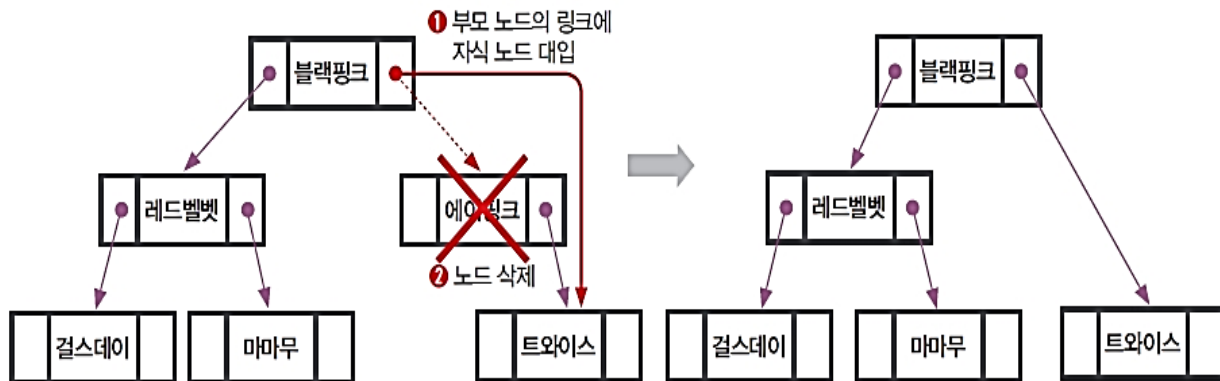
- 현재 노드가 부모 노드의 왼쪽 링크인지, 오른쪽 링크인지 구분하고자 코드 형태로 구현

```
if parent.left == current :      # 부모 노드 왼쪽 링크와 삭제할 노드가 같으면
    parent.left = None           # 부모 노드의 왼쪽에 None 대입
else :                           # 부모 노드 오른쪽 링크와 삭제할 노드가 같으면
    parent.right = None          # 부모 노드의 오른쪽에 None 대입

del(current)                     # 노드 삭제
```

2 자식 노드가 하나인 노드를 삭제하는 경우

▪ 자식 노드가 1개인 노드 삭제



2 자식 노드가 하나인 노드를 삭제하는 경우

▪ 왼쪽 자식 노드가 있는 경우

```
if parent.left == current :      # 부모 노드 왼쪽 링크와 삭제할 노드가 같으면
    parent.left = current.left   # 부모 노드의 왼쪽 링크에 왼쪽 자식 노드 대입
else :                          # 부모 노드 오른쪽 링크와 삭제할 노드가 같으면
    parent.right = current.left  # 부모 노드의 오른쪽 링크에 왼쪽 자식 노드 대입

del(current)
```

2 자식 노드가 하나인 노드를 삭제하는 경우

▪ 오른쪽 자식 노드가 있는 경우

```
if parent.left == current :      # 부모 노드 왼쪽 링크와 삭제할 노드가 같으면
    parent.left = current.right  # 부모 노드의 왼쪽 링크에 오른쪽 자식 노드 대입
else :                          # 부모 노드 오른쪽 링크와 삭제할 노드가 같으면
    parent.right = current.right # 부모 노드의 오른쪽 링크에 오른쪽 자식 노드 대입

del(current)
```

3 자식 노드가 둘 있는 노드를 삭제하는 경우

▪ 재귀를 사용해야 편리

이진 탐색 트리에서 데이터 삭제 완성(코드로 구현)

■ 이진 탐색 트리의 삭제 작동

```
1  ## 함수 선언 부분 ##
... # 생략(    앞쪽의    2~37행과 동일)
38
39 deleteName = '마마무'
40
41 current = root
42 parent = None
43 while True :
44     if deleteName == current.data :
45
```

이진 탐색 트리에서 데이터 삭제 완성(코드로 구현)

■ 이진 탐색 트리의 삭제 작동

```
46     if current.left == None and current.right == None :
47         if parent.left == current :
48             parent.left = None
49         else :
50             parent.right = None
51         del(current)
52
53     elif current.left != None and current.right == None :
54         if parent.left == current :
55             parent.left = current.left
```

이진 탐색 트리에서 데이터 삭제 완성(코드로 구현)

■ 이진 탐색 트리의 삭제 작동

```
56         else :  
57             parent.right = current.left  
58             del(current)  
59  
60         elif current.left == None and current.right != None :  
61             if parent.left == current :  
62                 parent.left = current.right  
63             else :  
64                 parent.right = current.right  
65             del(current)  
66
```

이진 탐색 트리에서 데이터 삭제 완성(코드로 구현)

■ 이진 탐색 트리의 삭제 작동

```
67         print(deleteName, '이(가) 삭제됨.')
68         break
69     elif deleteName < current.data :
70         if current.left == None :
71             print(deleteName, '이(가) 트리에 없음')
72             break
73         parent = current
74         current = current.left
75     else :
76         if current.right == None :
77             print(deleteName, '이(가) 트리에 없음')
```

이진 탐색 트리에서 데이터 삭제 완성(코드로 구현)

■ 이진 탐색 트리의 삭제 작동

```
78         break
79         parent = current
80         current = current.right
```

실행 결과

마마무 이(가) 삭제됨.



02

이진 탐색 트리의 응용

- 1) 이진 탐색 트리는 데이터를 보관하고
검색할 때 효율적
- 2) 이진 탐색 트리의 응용 실습

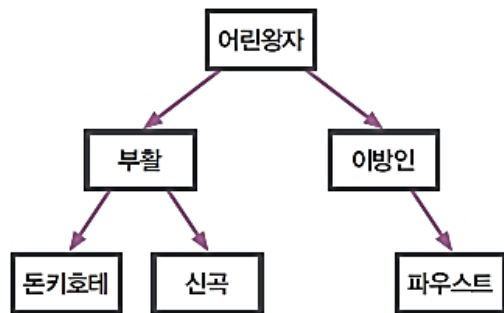


1] 이진 탐색 트리는 데이터를 보관하고 검색할 때 효율적

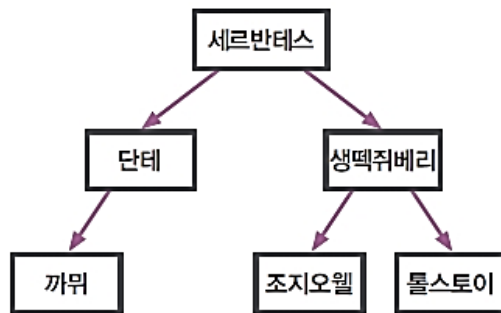


도서관에 새로 입고된 책 정보를 이진 탐색 트리에 보관해서 검색하는 예

- 책 이름 및 작가 이름으로 생성한 이진 탐색 트리



책 이름 이진 탐색 트리



작가 이름 이진 탐색 트리

도서관에 새로 입고된 책 정보를 이진 탐색 트리에 보관해서 검색하는 예

- 책 이름 및 작가 이름으로 생성한 이진 탐색 트리

```
import random
bookAry = [['어린왕자', '생텍쥐베리'], ['이방인', '까뮈'], ['부활', '톨스토이'],
           ['신곡', '단테'], ['돈키호테', '세르반테스'], ['동물농장', '조지오웰'],
           ['데미안', '헤르만헤세'], ['파우스트', '괴테'], ['대지', '펼벅']]
bookAry = random.shuffle(bookAry)

rootBook, rootAuth = None, None
```


책 이름 트리와 작가 이름 트리를 따로 구현

▪ 도서관 책 목록의 보관과 검색

```
1 import random
2 ## 함수 선언 부분 ##
3 class TreeNode() :
4     def __init__ (self) :
5         self.left = None
6         self.data = None
7         self.right = None
8
9 ## 전역 변수 선언 부분 ##
10 memory = []
11 rootBook, rootAuth = None, None
```

책 이름 트리와 작가 이름 트리를 따로 구현

▪ 도서관 책 목록의 보관과 검색

```
12 bookAry = [['어린왕자', '생텍쥐페리'], ['이방인', '까뮈'], ['부활', '톨스토이'],  
13           ['신곡', '단테'], ['돈키호테', '세르반테스'], ['동물농장', '조지오웰'],  
14           ['데미안', '헤르만헤세'], ['파우스트', '괴테'], ['대지', '펼벅']]  
15 random.shuffle(bookAry)  
16  
17 ## 메인 코드 부분 ##  
18  
19 ### 책 이름 트리 ###  
20 node = TreeNode()  
21 node.data = bookAry[0][0]  
22 rootBook = node
```

책 이름 트리와 작가 이름 트리를 따로 구현

▪ 도서관 책 목록의 보관과 검색

```
23 memory.append(node)
24
25 for book in bookAry[1:] :
26     name = book[0]
27     node = TreeNode()
28     node.data = name
29
30     current = rootBook
31     while True :
32         if name < current.data :
33             if current.left == None :
```

책 이름 트리와 작가 이름 트리를 따로 구현

▪ 도서관 책 목록의 보관과 검색

```
34         current.left = node
35         break
36         current = current.left
37     else :
38         if current.right == None :
39             current.right = node
40             break
41         current = current.right
42
43     memory.append(node)
44
```

책 이름 트리와 작가 이름 트리를 따로 구현

▪ 도서관 책 목록의 보관과 검색

```
45 print("책 이름 트리 구성 완료!")
46
47 ### 작가 이름 트리 ###
48 node = TreeNode()
49 node.data = bookAry[0][1]
50 rootAuth = node
51 memory.append(node)
52
53 for book in bookAry[1:] :
54     name = book[1]
55     node = TreeNode()
```

책 이름 트리와 작가 이름 트리를 따로 구현

▪ 도서관 책 목록의 보관과 검색

```
56     node.data = name
57
58     current = rootAuth
59     while True :
60         if name < current.data :
61             if current.left == None :
62                 current.left = node
63                 break
64             current = current.left
65         else :
66             if current.right == None :
```

책 이름 트리와 작가 이름 트리를 따로 구현

▪ 도서관 책 목록의 보관과 검색

```
67             current.right = node
68             break
69             current = current.right
70
71     memory.append(node)
72
73     print("작가 이름 트리 구성 완료!")
74
75     ## 책 이름 및 작가 이름 검색 ##
76     bookOrAuth = int(input('책검색(1), 작가검색(2)-->'))
77     findName = input('검색할 책 또는 작가-->')
```

책 이름 트리와 작가 이름 트리를 따로 구현

▪ 도서관 책 목록의 보관과 검색

```
78
79 if bookOrAuth == 1 :
80     root = rootBook
81 else :
82     root = rootAuth
83
84 current = root
85 while True :
86     if findName == current.data :
87         print(findName, '을(를) 찾음.')
88         findYN = True
```


책 이름 트리와 작가 이름 트리를 따로 구현

▪ 도서관 책 목록의 보관과 검색

```
89         break
90     elif findName < current.data :
91         if current.left == None :
92             print(findName, '이(가) 목록에 없음')
93             break
94         current = current.left
95     else :
96         if current.right == None :
97             print(findName, '이(가) 목록에 없음')
98             break
99         current = current.right
```

책 이름 트리와 작가 이름 트리를 따로 구현

▪ 도서관 책 목록의 보관과 검색

실행 결과

책 이름 트리 구성 완료!

작가 이름 트리 구성 완료!

책검색(1), 작가검색(2)-->1

검색할 책 또는 작가-->대지

대지 을(를) 찾음.

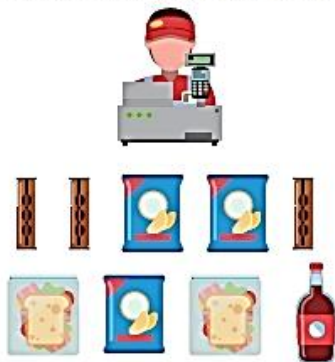
2] 이진 탐색 트리의 응용 실습



편의점에서 판매된 물건 목록 출력하기

편의점에서는 매일 다양한 물품을 판매한다. 하루에 판매하는 물건은 당연히 중복해서 여러 개 판매한다. 마감 시간에 오늘 판매된 물건 종류를 살펴볼 때는 중복된 것은 하나만 남기도록 한다. 이진 탐색 트리를 활용해서 중복된 물품은 하나만 남기자.

편의점에서 하루 판매된 물건(중복○)



오늘 판매된 물건(중복×)



2] 이진 탐색 트리의 응용 실습



편의점에서 판매된 물건 목록 출력하기

실행 결과

```
Python
File Edit Shell Debug Options Window Help
***** RESTART: C:\CookData\WEx08-01.py *****
오늘 판매된 물건(중복O) --> ['레쓰비캔커피', '레쓰비캔커피', '레쓰비캔커피', '도시락', '도시락', '삼각김밥', '레쓰비캔커피', '도시락', '코카콜라',
'삼다수', '레쓰비캔커피', '레쓰비캔커피', '레쓰비캔커피', '츄파춥스', '츄파춥스', '레쓰비캔커피', '코카콜라', '츄파춥스', '삼각김밥', '코카콜라']

이진 탐색 트리 구성 완료

오늘 판매된 종류(중복X)-> 레쓰비캔커피 도시락 삼각김밥 코카콜라 삼다수 츄파춥스
>>> |
```

2] 이진 탐색 트리의 응용 실습



세종사이버대학교

 편의점에서 판매된 물건 목록 출력하기



Q1

Q2

Q3

Q1

이진 탐색 트리의 특징과 가장 거리가 먼 것은?

- 1 왼쪽 서브 트리는 루트 노드보다 모두 작은 값을 가진다.
- 2 오른쪽 서브 트리는 루트 노드보다 모두 큰 값을 가진다.
- 3 각 서브 트리도 1,2 특징을 갖는다.
- 4 모든 노드 값은 필요하다면 중복될 수도 있다.


Q1

Q2

Q3

Q1

이진 탐색 트리의 특징과 가장 거리가 먼 것은?

- 1 왼쪽 서브 트리는 루트 노드보다 모두 작은 값을 가진다.
- 2 오른쪽 서브 트리는 루트 노드보다 모두 큰 값을 가진다.
- 3 각 서브 트리도 1,2 특징을 갖는다.
-  4 모든 노드 값은 필요하다면 중복될 수도 있다.

정답

4 모든 노드 값은 필요하다면 중복될 수도 있다.

해설

모든 노드 값은 중복되지 않습니다. 즉, 중복된 값은 이진 탐색 트리에 저장할 수 없습니다.

Q1

Q2

Q3

Q2

이진 탐색 트리의 삭제 동작 중에서 재귀 함수를
사용해야 하는 경우는?

- 1 리프 노드(맨 아래쪽 노드)를 삭제하는 경우
- 2 자식 노드가 하나(왼쪽)인 노드를 삭제하는 경우
- 3 자식 노드가 하나(오른쪽)인 노드를 삭제하는 경우
- 4 자식 노드가 둘 있는 노드를 삭제하는 경우


Q1

Q2

Q3

Q2

이진 탐색 트리의 삭제 동작 중에서 재귀 함수를
사용해야 하는 경우는?

- 1 리프 노드(맨 아래쪽 노드)를 삭제하는 경우
- 2 자식 노드가 하나(왼쪽)인 노드를 삭제하는 경우
- 3 자식 노드가 하나(오른쪽)인 노드를 삭제하는 경우
-  4 자식 노드가 둘 있는 노드를 삭제하는 경우

정답

4 자식 노드가 둘 있는 노드를 삭제하는 경우

해설

자식 노드가 둘 있는 노드를 삭제할 때는 재귀를 사용해야
편리합니다.

Q1

Q2

Q3

Q3

이진 탐색 트리의 삽입 작동 코드의 일부이다.
(1)에 적합한 코드는 무엇인가?

```
node = TreeNode()  
node.data = nameAry[0]  
root = node  
memory.append(node)
```

```
for name in ( 1 ):  
    node = TreeNode()  
    node.data = name
```

- 1 nameAry
- 2 nameAry[0:]
- 3 nameAry[1:]
- 4 nameAry[2:]

Q1

Q2

Q3

Q3

이진 탐색 트리의 삽입 작동 코드의 일부이다.
(1)에 적합한 코드는 무엇인가?

```
node = TreeNode()  
node.data = nameAry[0]  
root = node  
memory.append(node)
```

```
for name in (    1    ):  
    node = TreeNode()  
    node.data = name
```

- 1 nameAry
- 2 nameAry[0:]
- ☒ 3 nameAry[1:]
- 4 nameAry[2:]

정답

3 nameAry[1:]

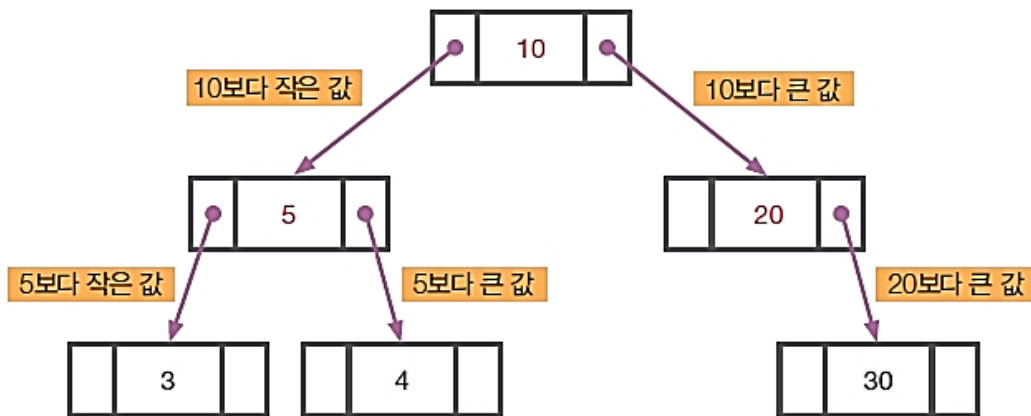
해설

이름 배열의 두 번째부터 처리합니다. 첫 번째는 이미 처리했기 때문입니다.

이진 탐색 트리의 일반 구현

④ 이진 탐색 트리의 특징

- 이진 트리 중 활용도가 높은 트리
- 데이터 크기를 기준으로 일정 형태로 구성함



이진 탐색 트리의 일반 구현

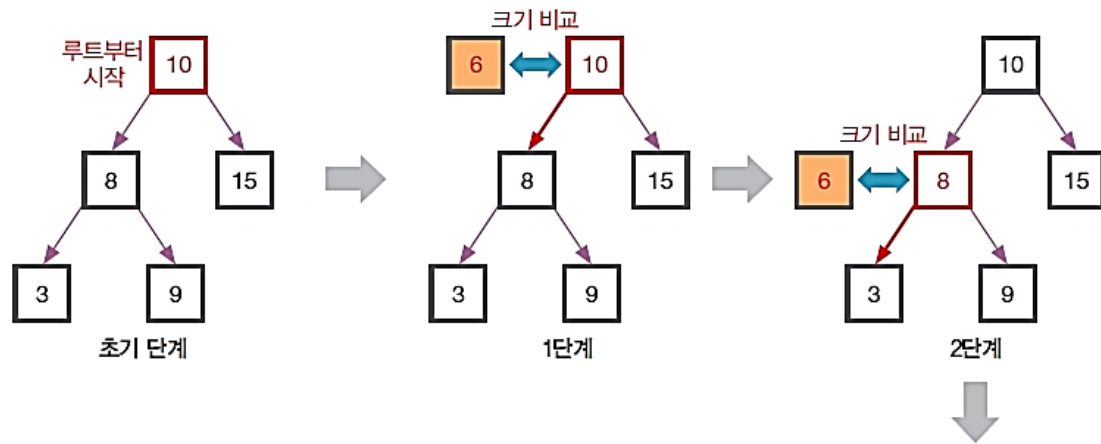
☑ 이진 탐색 트리의 특징

- ① 왼쪽 서브 트리는 루트 노드보다 모두 작은 값을 가짐
- ② 오른쪽 서브 트리는 루트 노드보다 모두 큰 값을 가짐
- ③ 각 서브 트리도 ①, ② 특징을 가짐
- ④ 모든 노드 값은 중복되지 않음
즉, 중복된 값은 이진 탐색 트리에 저장할 수 없음

이진 탐색 트리의 일반 구현

④ 이진 탐색 트리에서 데이터를 삽입하는 일반적인 형태

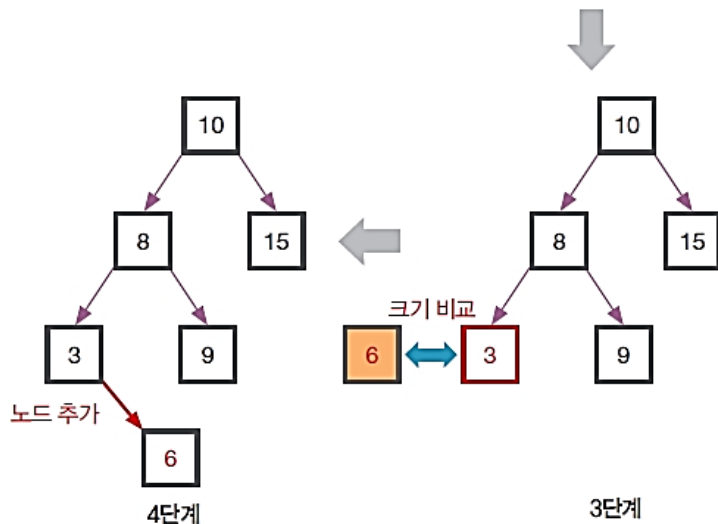
- 레벨 2인 이진 탐색 트리에서 6 데이터를 삽입하는 과정



이진 탐색 트리의 일반 구현

④ 이진 탐색 트리에서 데이터를 삽입하는 일반적인 형태

- 레벨 2인 이진 탐색 트리에서 6 데이터를 삽입하는 과정



이진 탐색 트리의 일반 구현

④ 이진 탐색 트리에서 데이터 검색

```
1 { if 현재 작업 노드의 데이터 == 찾을 데이터 :  
   탐색 종료  
2 { elif 현재 작업 노드의 데이터 < 찾을 데이터 :  
   왼쪽 서브 트리 탐색  
3 { else :  
   오른쪽 서브 트리 탐색
```