



# 이진 트리

자료구조와 알고리즘  
9주차 강의

---



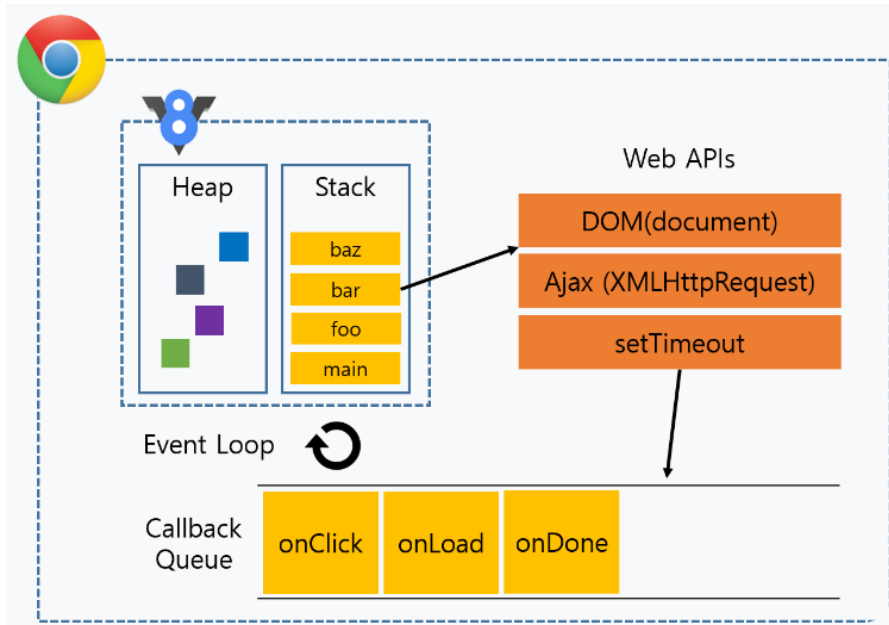
## 강의 계획표 / C# 프로그래밍 2판

주	주제
1	자료구조와 알고리즘 소개
2	파이썬 기초 문법과 데이터 형식
3	선형 리스트
4	단순 연결 리스트
5	원형 연결 리스트
6	스택
7	큐
8	중간고사
<b>9</b>	<b>이진 트리</b>
10	그래프
11	재귀 호출
12	정렬 기본
13	정렬 고급
14	검색
15	동적 계획법
16	기말고사



## 자바스크립트 런타임 작동에 스택과 큐가 사용됨

- > 자바스크립트는 웹브라우저에서 동작하는 프로그래밍 언어(Node.js 등의 일반 환경도 있음)
- > 런타임은 자바스크립트가 구동되는 환경 (인터프리터 언어)
- > 콜 스택(Call Stack)은 함수가 실행되는 순서를 기억
- > 콜백 큐(Callback Queue)는 WebAPI 결과값을 저장
  - > 스택으로 처리한 함수에서 이벤트를 쌓아둠(예: 타이머로 호출할 함수를 등록하는 것)



자바스크립트 런타임 작동 방식

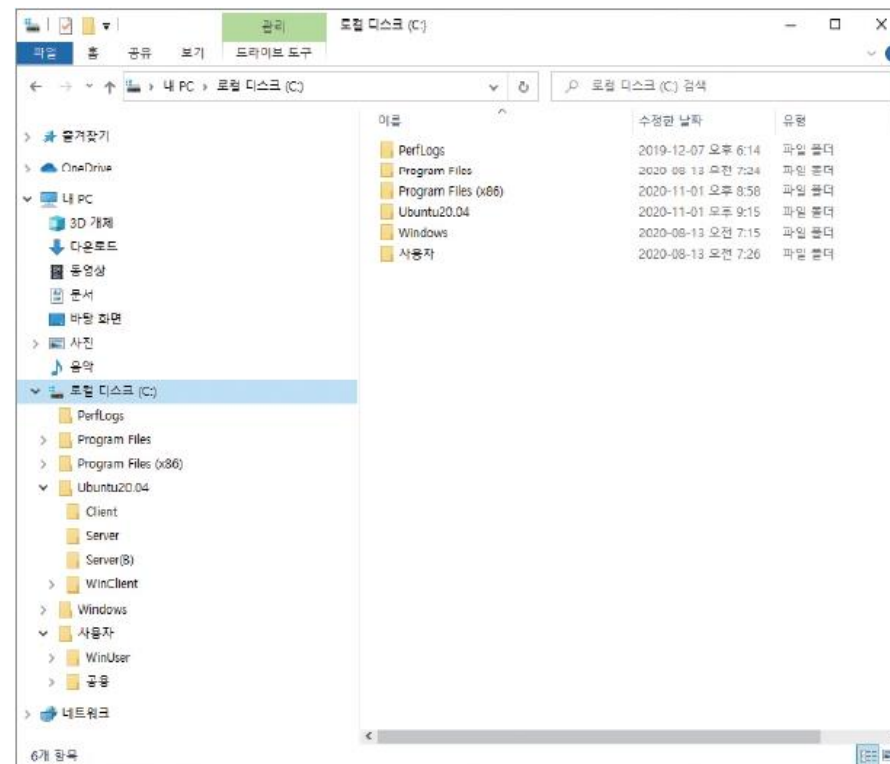
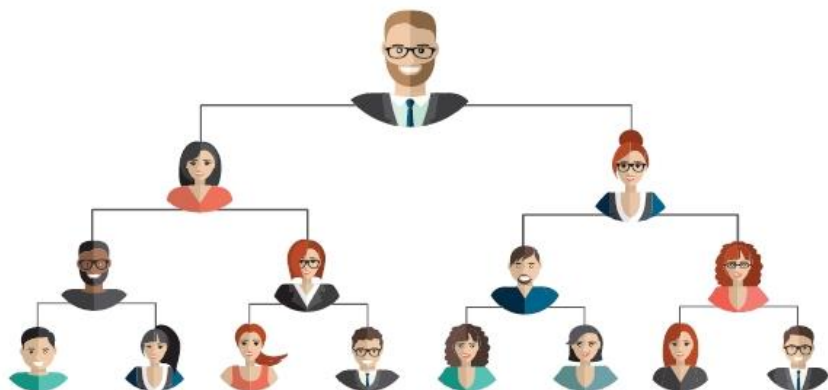
소프트웨어로 구현한  
스택과 큐가 동작함  
(V8 엔진)

# 이진 트리 기본

---

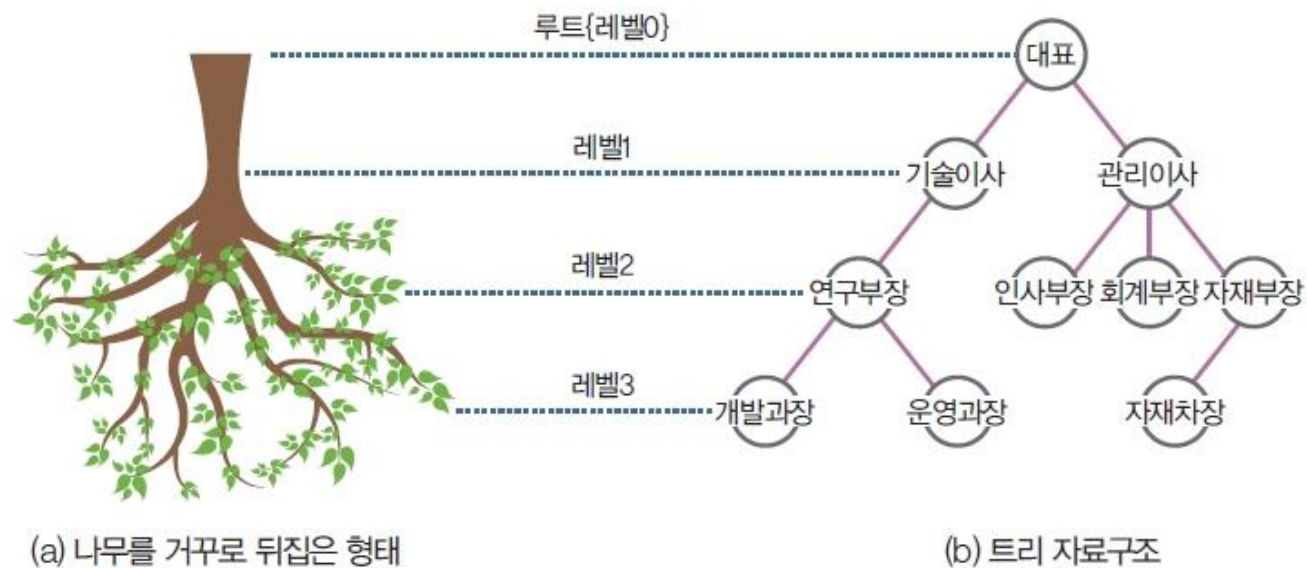
## 트리 구조란?

-> 회사 사장을 필두로 그 아래 직책들이 구성되어 있는 조직표 또는 컴퓨터의 상위 폴더 안에 하위 폴더들이 계속 이어져 있는 구조와 같은 구성

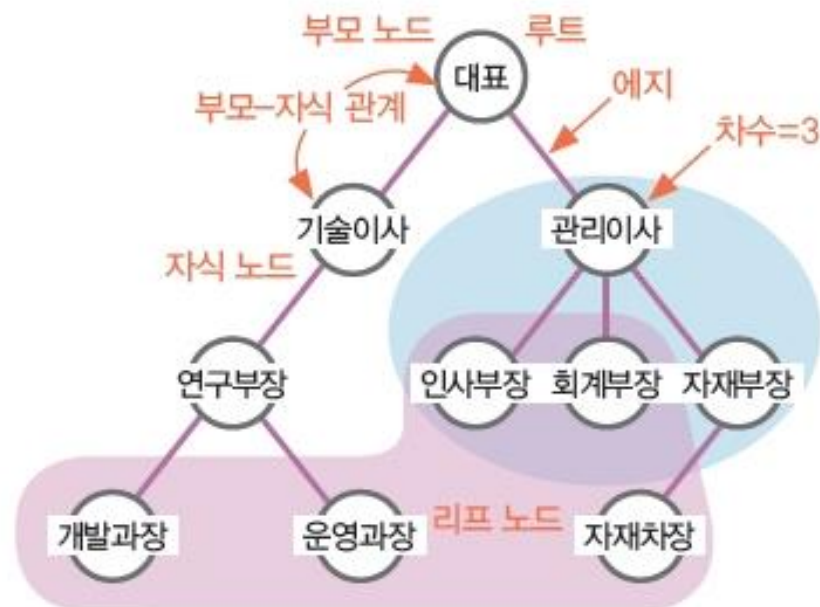


## 이진 트리의 개념

-> 트리(Tree) 자료구조는 나무를 거꾸로 뒤집어 놓은 형태



## 트리 자료구조 용어



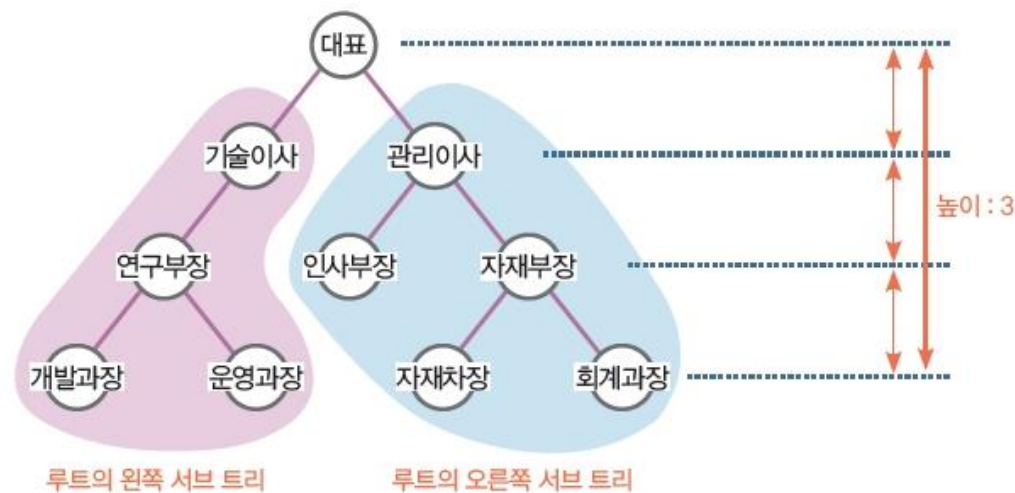


## 이진 트리의 개념

-> 모든 노드의 자식이 최대 2개인 트리(자식이 2개 이하로 구성)

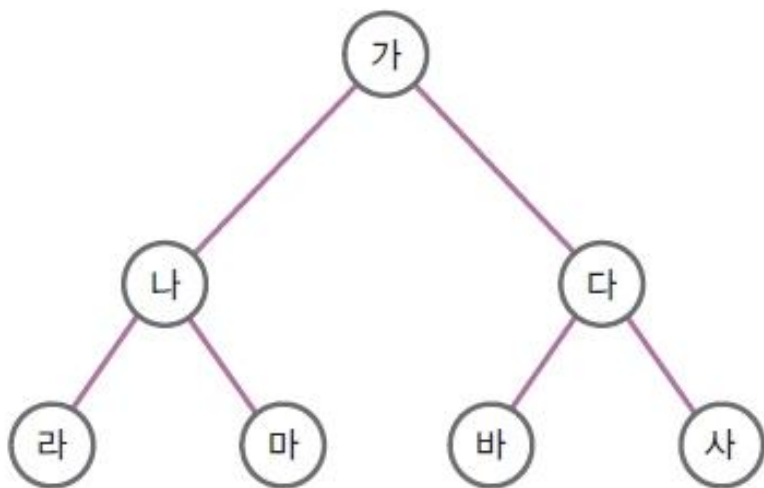


-> 전형적인 이진 트리

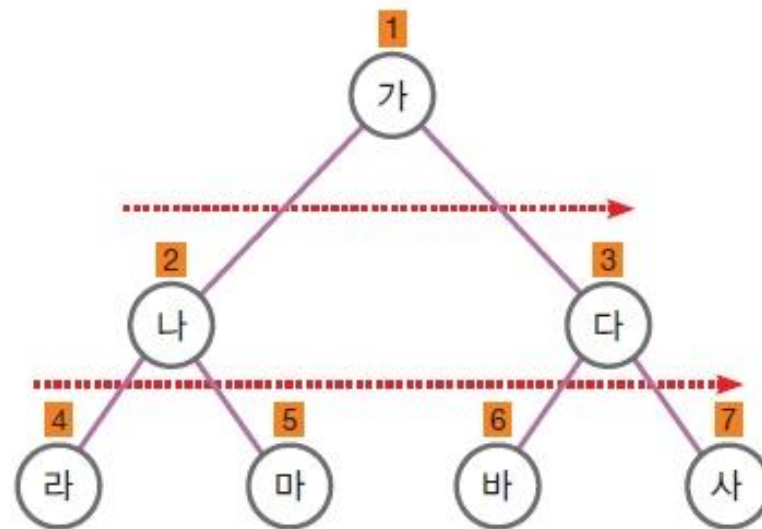


## 이진 트리의 종류

-> 포화 이진 트리(full binary tree)



포화 이진 트리 예



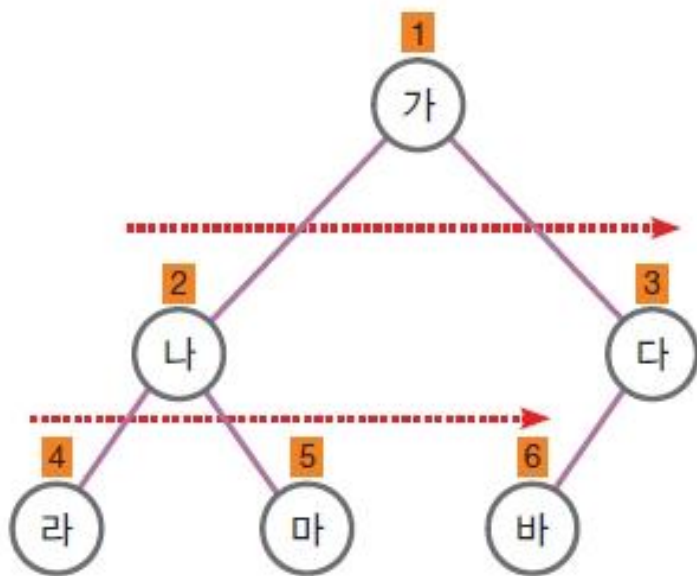
포화 이진 트리의 번호 부여 순서



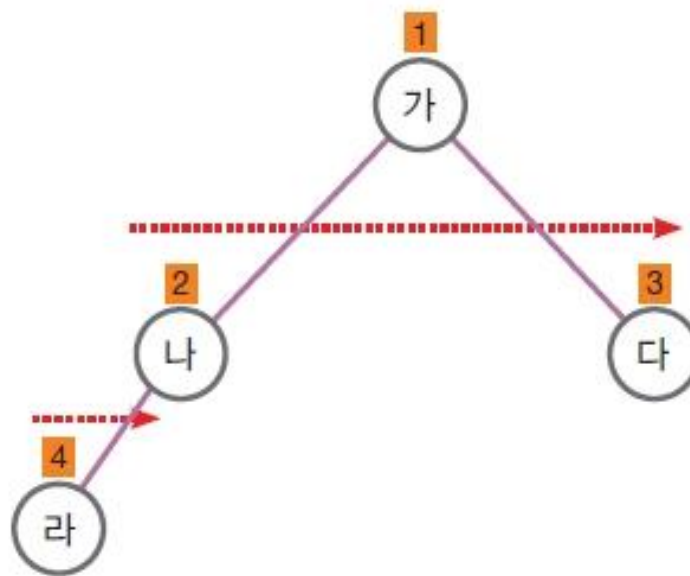


## 이진 트리의 종류

-> 완전 이진 트리(complete binary tree)

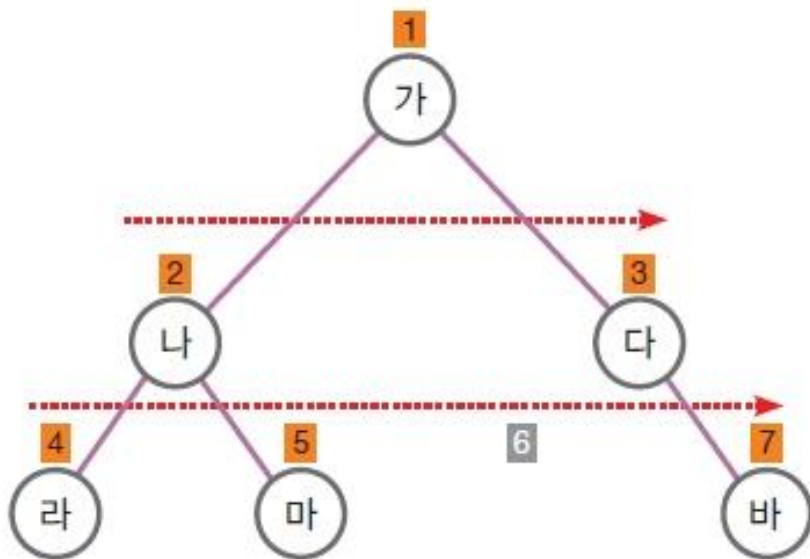


완전 이진 트리 예

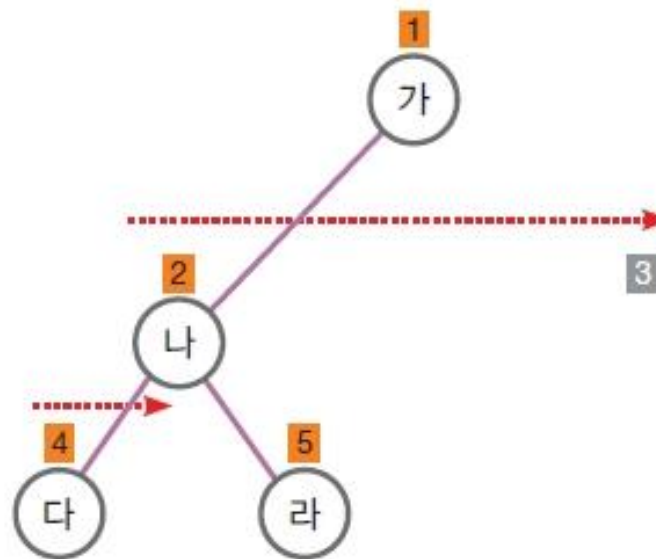




## 이진 트리의 종류 -> 일반 이진 트리



일반 이진 트리 예

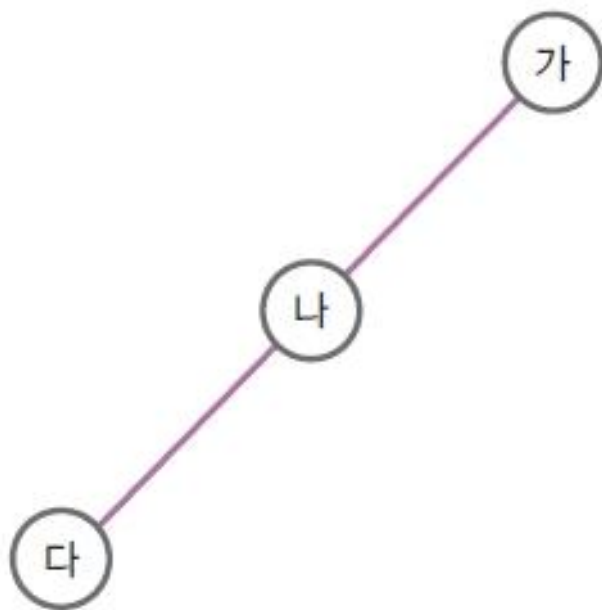


번호가 불완전

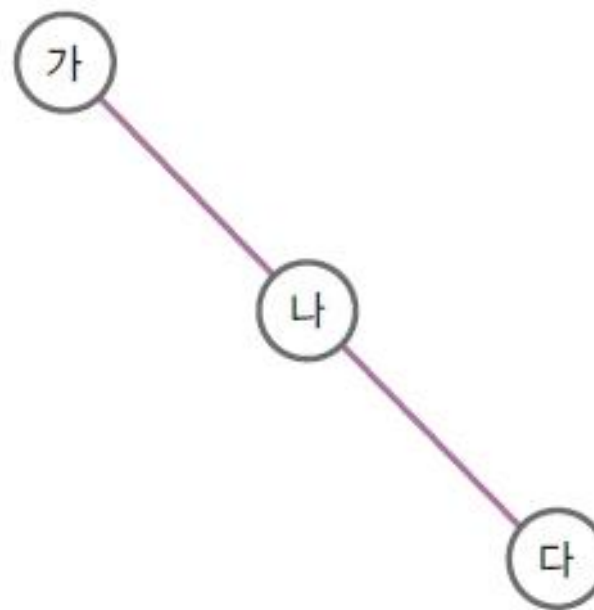


## 이진 트리의 종류

-> 편향 이진 트리(skewed binary tree)



(a) 왼쪽 편향 이진 트리



(b) 오른쪽 편향 이진 트리

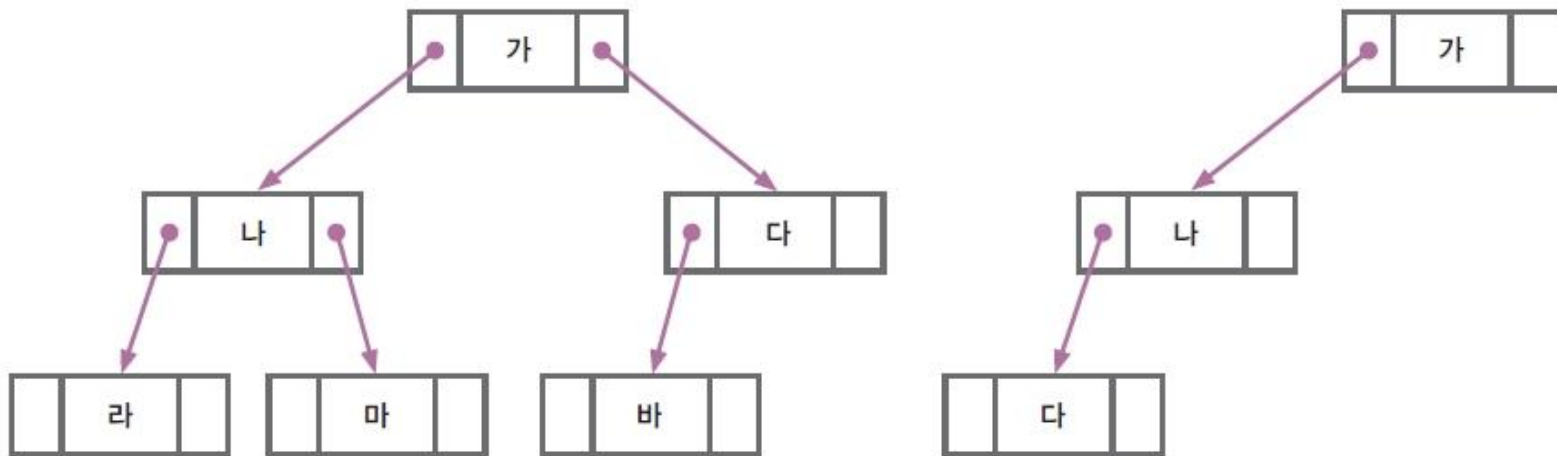


## 이진 트리의 노드 구조

-> 이중 연결 리스트를 이용한 트리 노드 표현



(a) 트리 노드의 구현을 위한 이중 연결 리스트



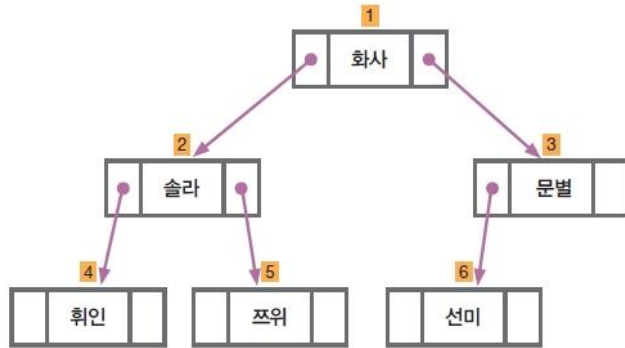
# 이진 트리 간단 구현

---



## 이진 트리의 생성

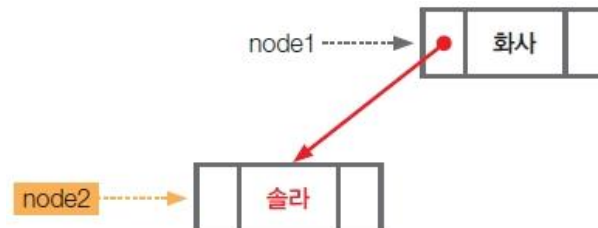
-> 높이가 2고 데이터가 6개인 완전 이진 트리 생성 예



### 1 루트 노드(화사)를 생성



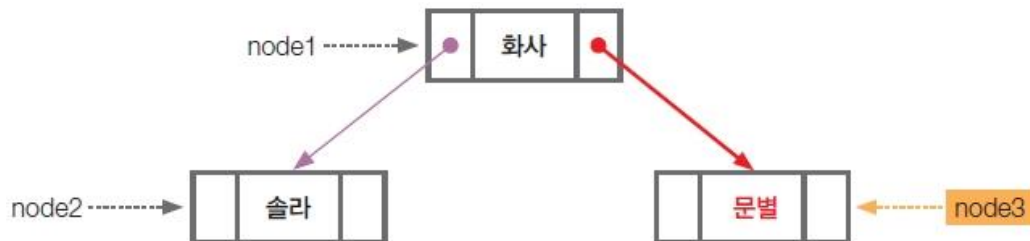
### 2 두 번째 노드(솔라)를 생성하고 루트 노드의 왼쪽 노드로 지정한다.



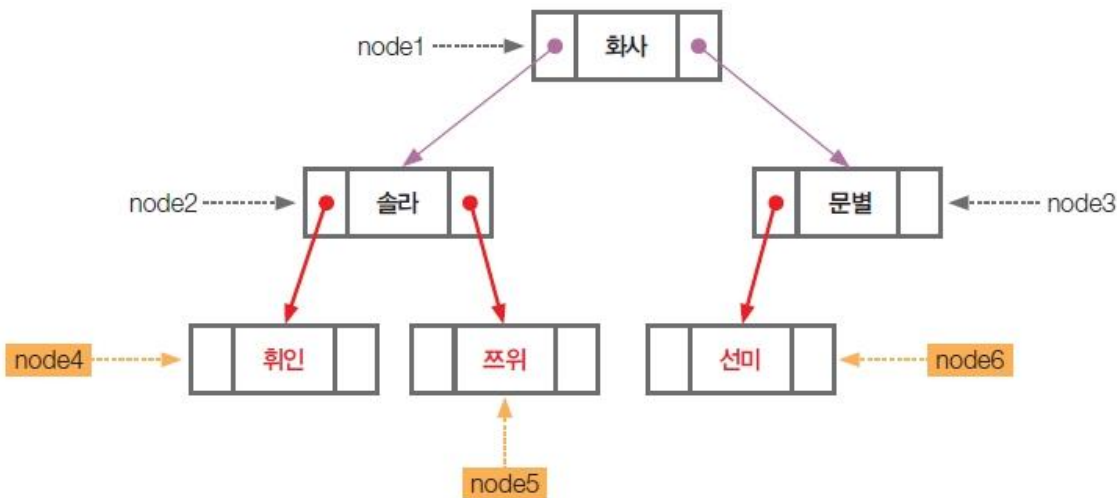


## 이진 트리의 생성

3 세 번째 노드(문별)를 생성하고 루트 노드의 오른쪽 노드로 지정한다.



4 네 번째부터 여섯 번째까지 노드를 생성하고 부모 노드와 연결한다.





## 이진 트리의 생성

```
1 class TreeNode():          # 이진 트리 노드 생성
2     def __init__(self):
3         self.left = None
4         self.data = None
5         self.right = None
6
7     node1 = TreeNode() } 1
8     node1.data = '화사'
9
10    node2 = TreeNode() } 2
11    node2.data = '솔라'
12    node1.left = node2
13
14    node3 = TreeNode() } 3
15    node3.data = '문별'
16    node1.right = node3
17
```

```
18 node4 = TreeNode()
19 node4.data = '휘인'
20 node2.left = node4
21
22 node5 = TreeNode()
23 node5.data = '쯔위' } 4
24 node2.right = node5
25
26 node6 = TreeNode()
27 node6.data = '선미'
28
29 node3.left = node6
30 print(node1.data, end = ' ')
31 print()
32 print(node1.left.data, node1.right.data, end = ' ')
33 print()
34 print(node1.left.left.data, node1.left.right.data, node1.right.left.data, end = ' ')
```

### 실행 결과

화사  
솔라 문별  
휘인 쯔위 선미





## 이진 트리의 순회의 종류

-> 이진 트리의 노드 전체를 한 번씩 방문하는 것을 순회(traversal)라고 함

-> 노드 데이터를 처리하는 순서에 따라 전위 순회, 중위 순회, 후위 순회

- ① 현재 노드 데이터 처리
- ② 왼쪽 서브 트리로 이동
- ③ 오른쪽 서브 트리로 이동

전위 순회(preorder traversal)

- ① 왼쪽 서브 트리로 이동
- ② 현재 노드 데이터 처리
- ③ 오른쪽 서브 트리로 이동

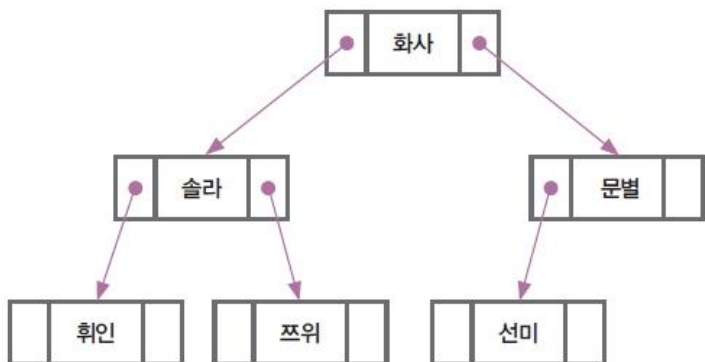
중위 순회(inorder traversal)

- ① 왼쪽 서브 트리로 이동
- ② 오른쪽 서브 트리로 이동
- ③ 현재 노드 데이터 처리

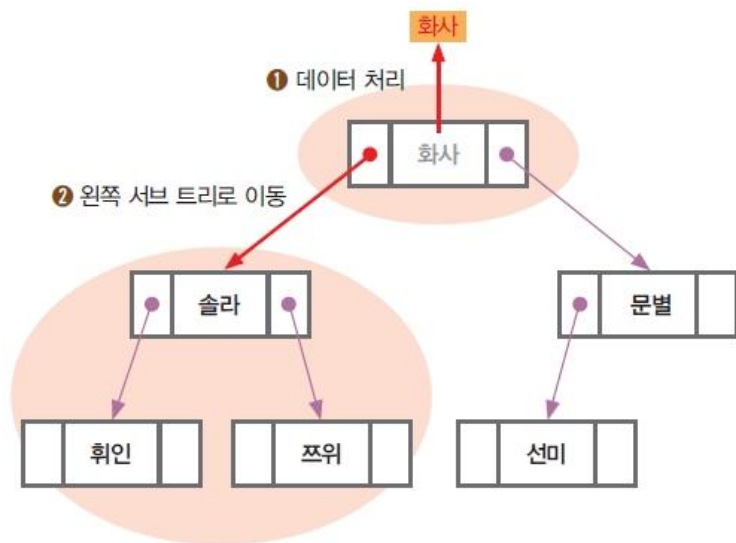
후위 순회(postorder traversal)

## 전위 순회 작동

### 0 전위 순회할 이진 트리



### 1 루트 노드(화사)의 데이터를 처리하고 왼쪽 서브 트리로 이동한다.

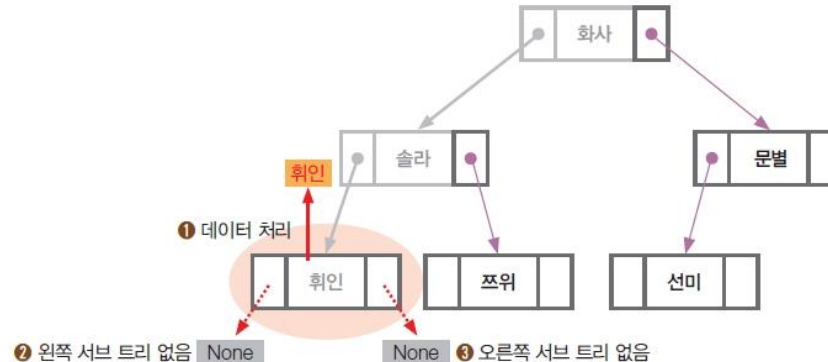


## 전위 순회 작동

**2** 이동한 왼쪽 서브 트리의 술라 데이터를 처리하고 다시 왼쪽 서브 트리으로 이동한다.



**3** 이동한 왼쪽 서브 트리의 휘인 데이터를 먼저 처리하고, 다시 왼쪽 서브 트리를 처리하려고 하나 왼쪽 서브 트리가 없어 오른쪽 서브 트리를 처리하려고 한다. 그런데 오른쪽 서브 트리도 없으므로 휘인 노드는 처리가 완료되었다.



## 전위 순회 작동

- 4** 현재 노드(취인 노드)는 더 이상 처리할 것이 없으므로 앞 노드로 올라가서 처리하지 않았던 오른쪽 서브 트리로 내려간다.



- 5** 이동한 오른쪽 서브 트리의 쯔위 데이터를 먼저 처리하고, 왼쪽 서브 트리를 처리하려고 하나 왼쪽 서브 트리가 없어 오른쪽 서브 트리를 처리하려고 한다. 그런데 오른쪽 서브 트리도 없으므로 쯔위 노드는 처리가 완료되었다.



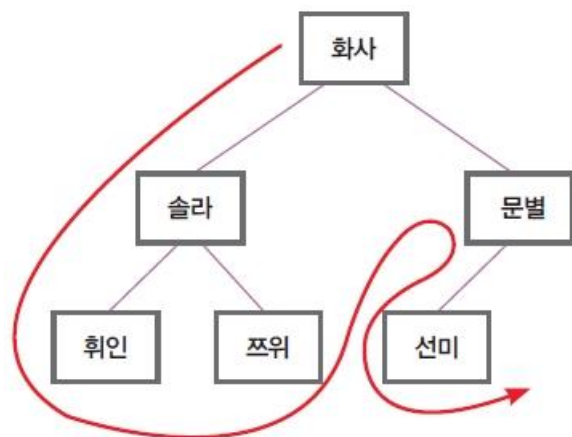
## 전위 순회 작동

- 8** 이동한 왼쪽 서브 트리의 선미 데이터를 먼저 처리하고, 다시 왼쪽 서브 트리를 처리하려고 하나 왼쪽 서브 트리가 없어 오른쪽 서브 트리를 처리하려고 한다. 그런데 오른쪽 서브 트리도 없으므로 선미 노드는 처리가 완료되었다. 그리고 트리의 모든 노드 순회가 완료되었다.

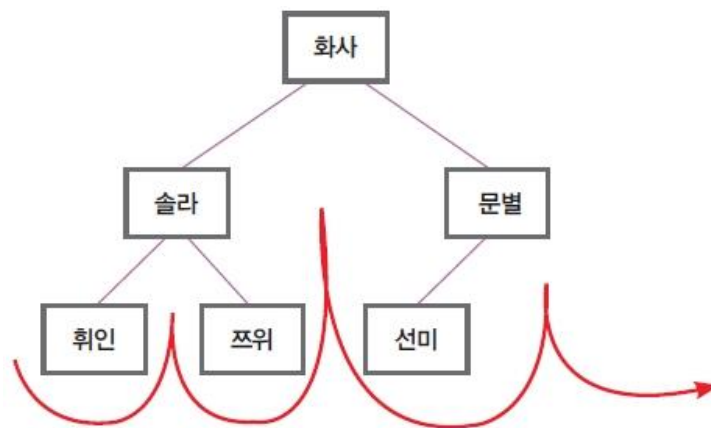


- 1 ~ 8** 에서 출력된 데이터를 확인하면 화사, 솔라, 휘인, 쯔위, 문별, 선미 순이다. 즉, 전위 순회인 현재 데이터 → 왼쪽 서브 트리 → 오른쪽 서브 트리 순서로 출력된 것을 확인할 수 있다.

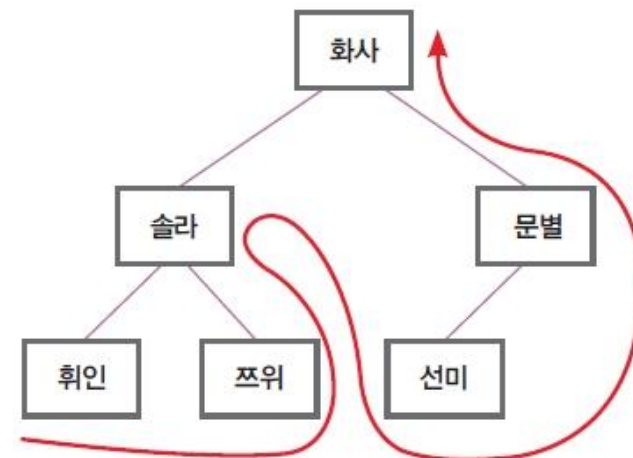
## ■ 좀 더 간단한 트리 순회



■ 전위 순회 : 루트 → 왼쪽 → 오른쪽



■ 중위 순회 : 왼쪽 → 루트 → 오른쪽



■ 후위 순회 : 왼쪽 → 오른쪽 → 루트



## 이진 트리 순회 구현

```
1 class TreeNode() :
2     def __init__(self) :
3         self.left = None
4         self.data = None
5         self.right = None
6
7 node1 = TreeNode()
8 node1.data = '화사'
9
10 node2 = TreeNode()
11 node2.data = '솔라'
12 node1.left = node2
13
14 node3 = TreeNode()
15 node3.data = '문별'
16 node1.right = node3
17
18 node4 = TreeNode()
19 node4.data = '휘인'
20 node2.left = node4
21
22 node5 = TreeNode()
23 node5.data = '쯔위'
24 node2.right = node5
25
26 node6 = TreeNode()
27 node6.data = '선미'
28 node3.left = node6
29
```



## 이진 트리 순회 구현

```
30 def preorder(node) :
31     if node == None:
32         return
33     print(node.data, end='->')
34     preorder(node.left)
35     preorder(node.right)
36
37 def inorder(node):
38     if node == None :
39         return
40     inorder(node.left)
41     print(node.data, end='->')
42     inorder(node.right)
43
44 def postorder(node):
45     if node == None :
46         return
47     postorder(node.left)
48     postorder(node.right)
49     print(node.data, end='->')
50
```

```
51 print('전위 순회 : ', end = '')
52 preorder(node1)
53 print('끝')
54
55 print('중위 순회 : ', end = '')
56 inorder(node1)
57 print('끝')
58
59 print('후위 순회 : ', end = '')
60 postorder(node1)
61 print('끝')
```

### 실행 결과

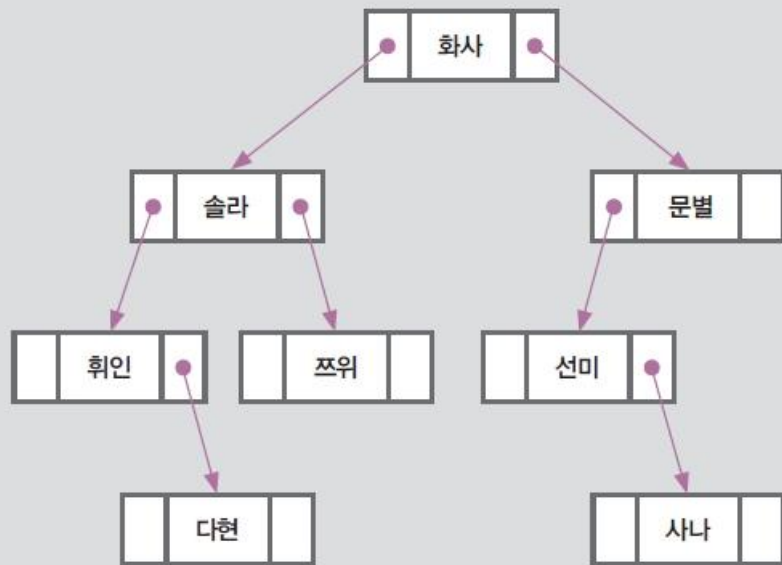
```
전위 순회 : 화사->솔라->휘인->쯔위->문별->선미->끝
중위 순회 : 휘인->솔라->쯔위->화사->선미->문별->끝
후위 순회 : 휘인->쯔위->솔라->선미->문별->화사->끝
```





## 연습문제

수정해서 다음 그림과 같이 만들고, 전위/중위/후위 순회를 시켜 보자.



### 실행 결과

전위 순회 : 화사->솔라->휘인->다현->쯔위->문별->선미->사나->끝

중위 순회 : 휘인->다현->솔라->쯔위->화사->선미->사나->문별->끝

후위 순회 : 다현->휘인->쯔위->솔라->사나->선미->문별->화사->끝

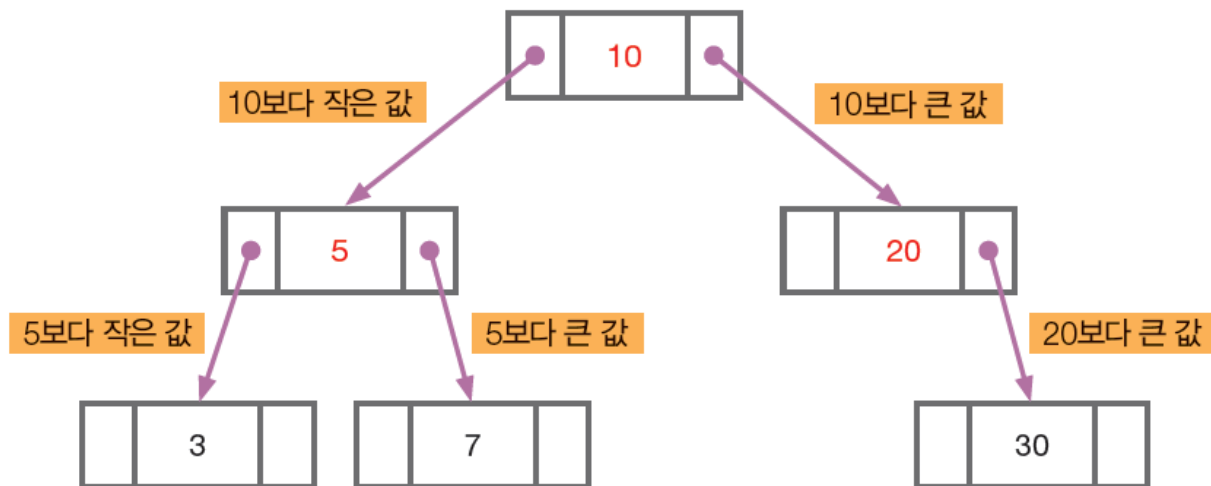
# 이진 탐색 트리 일반 구현

---



## 이진 탐색 트리의 특징

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

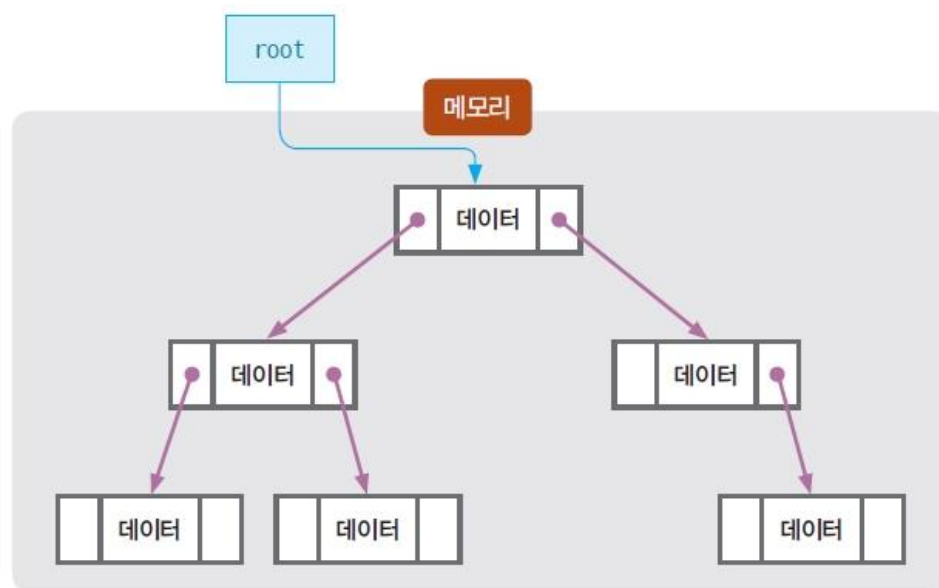


이진 탐색 트리 특징

- 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 중복되지 않는다. 즉, 중복된 값은 이진 탐색 트리에 저장할 수 없다.



## 이진 탐색 트리의 생성



-> 메모리를 준비하고 root는 None으로 초기화

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



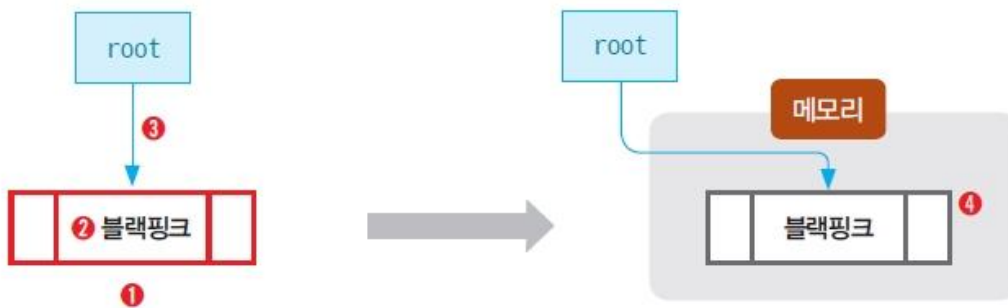
## 이진 탐색 트리의 생성

-> 배열에 있는 데이터를 차례대로 이진 탐색 트리에 삽입

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

-> 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
```



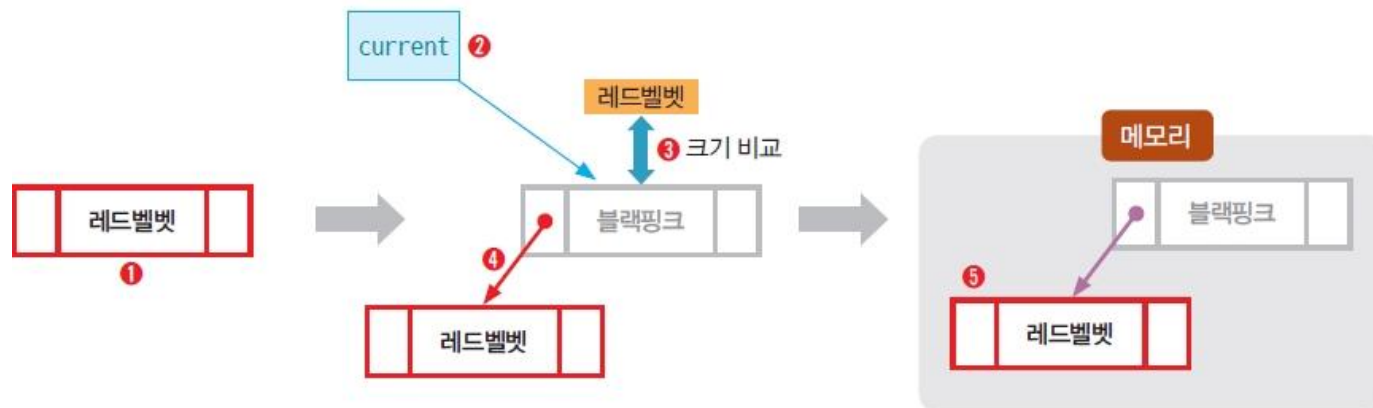


## 이진 탐색 트리의 생성

-> 두 번째 이후 데이터 삽입

```
1 { name = '레드벨벳'           # 두 번째 데이터
    node = TreeNode()          # 새 노드 생성
    node.data = name            # 새 노드에 데이터 입력

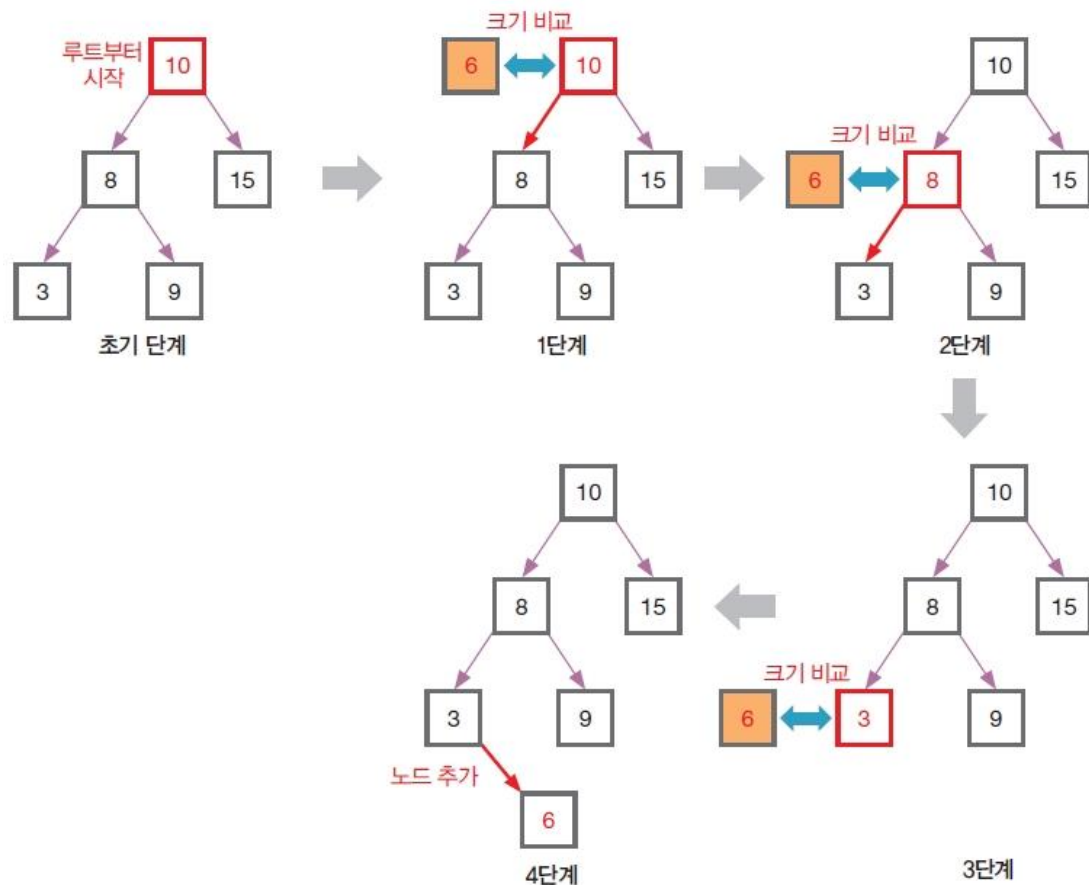
2~4 { current = root           # 현재 작업 노드를 루트 노드로 지정
    if name < current.data :    # 입력할 값을 현재 작업 노드의 값과 비교
        current.left = node    # 작으면 새 노드를 왼쪽 링크로 연결
    else :
        current.right = node    # 크면 새 노드를 오른쪽 링크로 연결
5 memory.append(node)          # 새 노드를 메모리에 저장
```





## 이진 탐색 트리에서 데이터를 삽입하는 일반적인 형태

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



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

```
① current = root # 루트부터 시작
② while True : # 무한 반복
    ③ if name < current.data : # 1단계
        {
            if current.left == None : # 4단계
                ④ current.left = node # 4단계
                break
            ⑤ current = current.left # 2~3단계
        }
    else :
        {
            if current.right == None : # 4단계
                ⑥ current.right = node # 4단계
                break
            current = current.right
        }
```



## 이진 탐색 트리에서 데이터를 삽입하는 코드

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

실행결과

이진 탐색 트리 구성 완료!



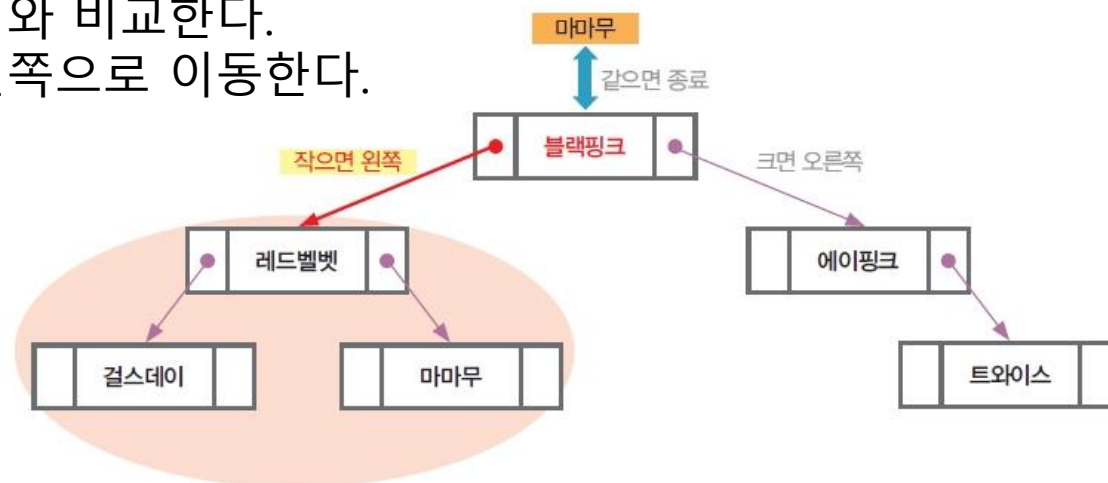


## 이진 탐색 트리에서 데이터를 탐색

- 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

### -> 완성된 이진 탐색 트리에서 마마무를 찾는 예

- 1 찾고자 하는 마마무를 루트 노드의 데이터와 비교한다.  
마마무가 루트 노드의 데이터보다 작아 왼쪽으로 이동한다.

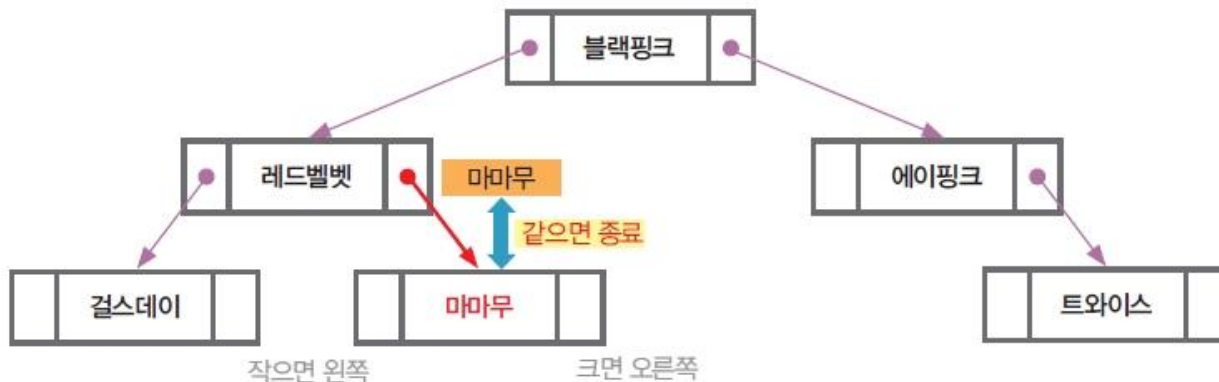


## 이진 탐색 트리에서 데이터를 탐색

- 2 왼쪽 서브 트리에서도 동일하게 처리한다. 찾고자 하는 마마무가 왼쪽 서브 트리의 루트 노드보다 커 오른쪽으로 이동한다.



- 3 오른쪽 서브 트리에서도 동일하게 처리한다. 그런데 여기에서는 마마무를 찾았으므로 종료한다.





## 이진 탐색 트리에서 데이터를 탐색 코드 1 (32번 슬라이드와 동일)

```
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
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 :
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
37     memory.append(node)
38
```



## 이진 탐색 트리에서 데이터를 탐색 코드 2

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

실행결과

마마무 을(를) 찾음.



## 연습문제

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

### 실행 결과

찾을 그룹이름-->잇지

잇지 을(를) 찾았음.

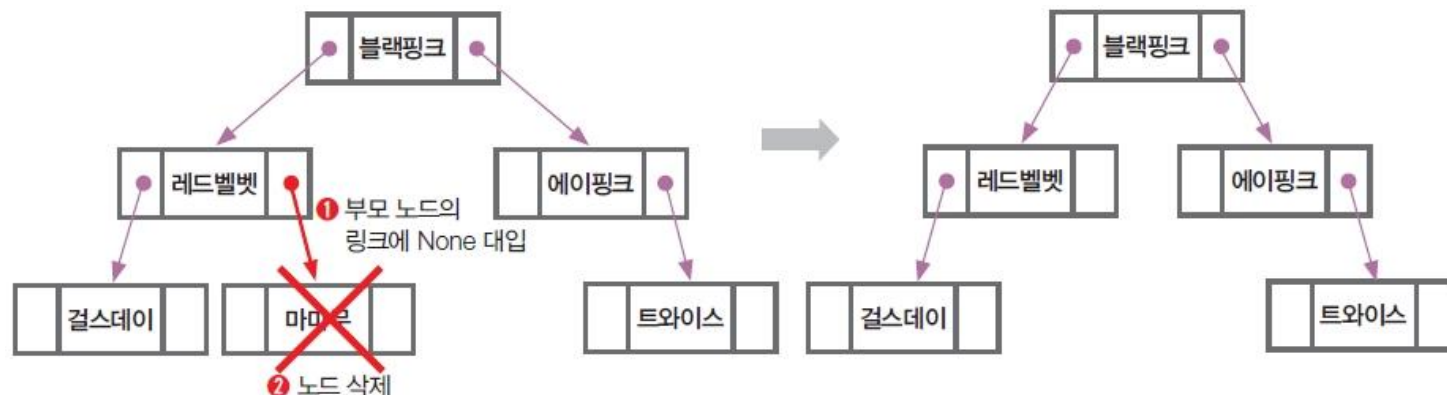
찾을 그룹이름-->소녀시대

소녀시대 이(가) 트리에 없음



## 이진 탐색 트리에서 데이터 삭제

-> 리프 노드(맨 아래쪽 노드)를 삭제하는 경우 - मामु를 삭제하는 경우



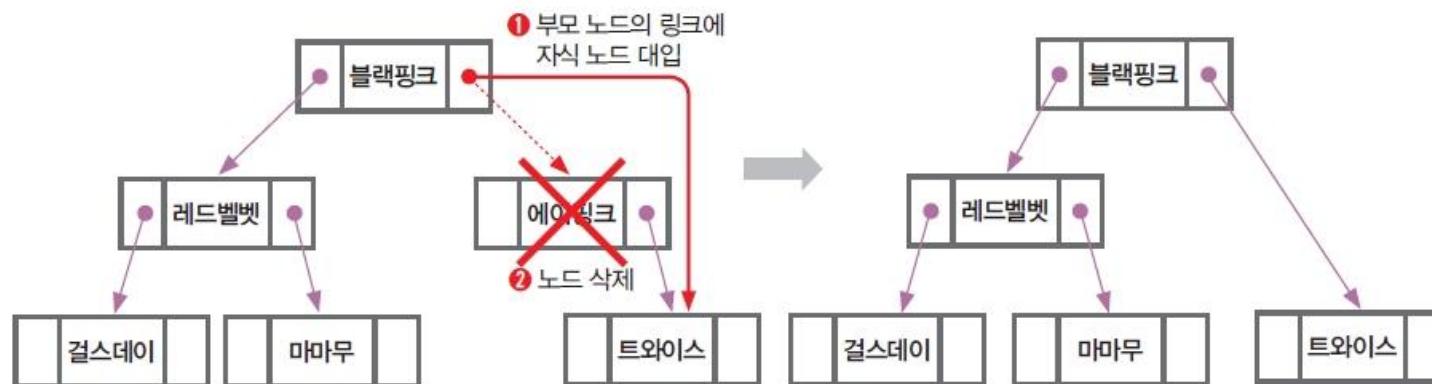
-> 현재 노드가 부모 노드의 왼쪽 링크인지, 오른쪽 링크인지 구분

```
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
```

## 이진 탐색 트리에서 데이터 삭제

-> 자식 노드가 하나인 노드를 삭제 - 에이핑크를 삭제하는 경우(에이핑크에 오른쪽 노드만 존재)



```
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)
```



## 이진 탐색 트리에서 데이터 삭제

-> 자식 노드가 하나인 노드를 삭제 - 만약 삭제되는 노드의 왼쪽 자식 노드만 있다면?

```
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)
```

-> 자식 노드가 둘 있는 노드를 삭제하는 경우 : 재귀를 사용해야 편리 (10장에서 재귀함수 배움)





## 이진 탐색 트리에서 데이터 삭제 코드 1 (32번 슬라이드와 동일)

```
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
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 :
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
37     memory.append(node)
38
```



## 이진 탐색 트리에서 데이터 삭제 코드 2

```
39 deleteName = '마마무'
40
41 current = root
42 parent = None
43 while True:
44     if deleteName == current.data :
45
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
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
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')
78             break
79         parent = current
80         current = current.right
```

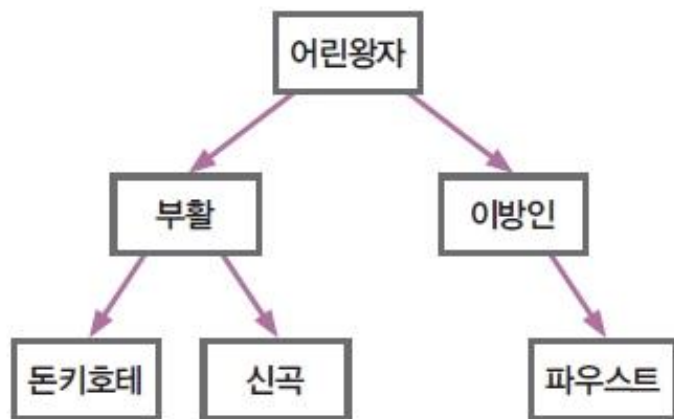
실행결과    마마무 이(가) 삭제됨.

# 이진 탐색 트리 응용

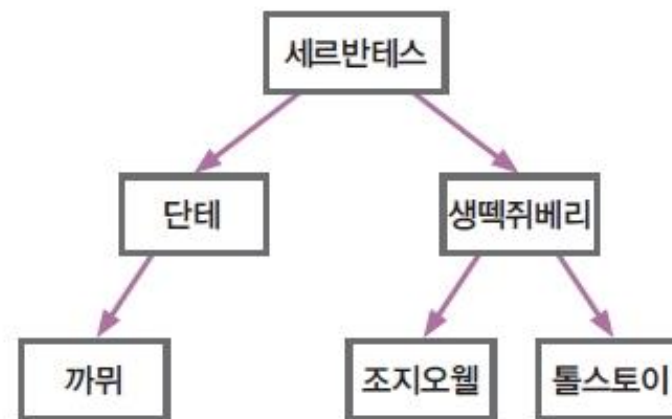
---



- 이진 탐색 트리는 데이터를 보관하고 검색할 때 효율적
  - > 도서관에 새로 입고된 책 정보를 이진 탐색 트리에 보관해서 검색하는 예
  - > 책 이름 트리와 작가 이름 트리를 따로 구현



(a) 책 이름 이진 탐색 트리



(b) 작가 이름 이진 탐색 트리



### ■ 도서관에 새로 입고된 책 정보를 이진 탐색 트리에 보관해서 검색하는 예 코드 1

```
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
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
```



## 도서관에 새로 입고된 책 정보를 이진 탐색 트리에 보관해서 검색하는 예 코드 2

```
19 ## 메인 코드 부분 ##
20
21 ### 책 이름 트리 ###
22 node = TreeNode()
23 node.data = bookAry[0][0]
24 rootBook = node
25 memory.append(node)
26
27 for book in bookAry[1:] :
28     name = book[0]
29     node = TreeNode()
30     node.data = name
31
32     current = rootBook
33     while True :
34         if name < current.data :
35             if current.left == None :
36                 current.left = node
37                 break
38             current = current.left
39         else :
40             if current.right == None :
41                 current.right = node
42                 break
43             current = current.right
44
45     memory.append(node)
46
47 print("책 이름 트리 구성 완료!")
```

```
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()
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 :
67                 current.right = node
68                 break
69             current = current.right
70
71     memory.append(node)
72
73 print("작가 이름 트리 구성 완료!")
74
```



## 도서관에 새로 입고된 책 정보를 이진 탐색 트리에 보관해서 검색하는 예 코드 3

```
75 ## 책 이름 및 작가 이름 검색 ##
76 bookOrAuth = int(input('책검색(1), 작가검색(2)-->'))
77 findName = input('검색할 책 또는 작가-->')
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
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
```

### 실행결과

작가 이름 트리 구성 완료!책검색(1), 작가검색(2)-->1  
검색할 책 또는 작가-->어린왕자  
어린왕자 을(를) 찾음.

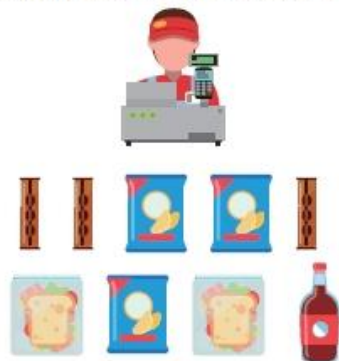
## 편의점에서 판매된 물건 목록 출력하기

난이도 ★★☆☆☆

### 예제 설명

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

편의점에서 하루 판매된 물건(중복○)



오늘 판매된 물건(중복×)



### 실행 결과

```
Python
File Edit Shell Debug Options Window Help
===== RESTART: C:\CookData\WEx08-01.py =====
오늘 판매된 물건(중복○) --> ['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
>>> |
```





## 편의점에서 판매된 물건 목록 출력하기 코드 1

```
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 root = None
12 dataAry = ['바나나맛우유', '레쓰비캔커피', '츄파춥스', '도시락', '삼다수', '코카콜라', '삼각김밥']
13 sellAry = [random.choice(dataAry) for _ in range(20)]
14
15 print('오늘 판매된 물건(중복O) -->', sellAry)
16
```




## 편의점에서 판매된 물건 목록 출력하기 코드 2

```
17 ## 메인 코드 부분 ##
18 node = TreeNode()
19 node.data = sellAry[0]
20 root = node
21 memory.append(node)
22
```

```
23 for name in sellAry[1:] :
24
25     node = TreeNode()
26     node.data = name
27
28     current = root
29     while True :
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45 print("이진 탐색 트리 구성 완료!")
46
```



## 편의점에서 판매된 물건 목록 출력하기 코드 3

```
47 def preorder(node) :  
48       
49  
50  
51  
52  
53  
54 print('오늘 판매된 종류(중복X)--> ', end = ' ' )  
55 preorder(root)
```

### 실행결과

오늘 판매된 물건(중복O) --> ['삼각김밥', '삼다수', '레쓰비캔커피', '코카콜라', '바나나맛우유', '츄파춥스', '도시락', '레쓰비캔커피', '삼다수', '코카콜라', '레쓰비캔커피', '츄파춥스', '레쓰비캔커피', '바나나맛우유', '삼다수', '삼각김밥', '츄파춥스', '레쓰비캔커피', '레쓰비캔커피', '레쓰비캔커피']  
이진 탐색 트리 구성 완료!  
오늘 판매된 종류(중복X)--> 삼각김밥 레쓰비캔커피 도시락 바나나맛우유 삼다수 코카콜라 츄파춥스



### 폴더 및 하위 폴더에 중복된 파일 이름 찾기

난이도 ★★☆☆☆

#### 예제 설명

특정 폴더를 지정해서 해당 폴더 및 그 하위 폴더에 모든 파일을 조회한다. 그리고 이름이 동일한 파일이 있으면 그 이름을 출력한다. 예로 C:/Program Files/Common Files/ 폴더 및 그 하위 폴더 아래에는 이름이 동일한 파일이 몇 개 있다. 단 여러 번 중복되더라도 한 번만 출력하자.

#### 실행 결과

```
Python
File Edit Shell Debug Options Window Help
===== RESTART: C:\CookData\WE08-02.py =====
C:/Program Files/Common Files/ 및 그 하위 디렉터리의 중복된 파일 목록 -->
['VSTOLoaderUI.dll', 'TFSTOfficeAdd-inUI.dll', 'tipresxc.dll.mui', 'TFSTOfficeAdd-in.dll', 'pdmui.dll', 'VSTOInstallerUI.dll', 'vmciver.dll']

>>>
```

Ln: 660 Col: 29



### 폴더 및 하위 폴더에 중복된 파일 이름 찾기 코드 1

```
1 import os
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 root = None
12 fnameAry = []
13
```

```
14 ## 메인 코드 부분 ##
15 folderName = 'C:/Program Files/Common Files/'
16 for dirName, subDirList, fnames in os.walk(folderName) :
17     for fname in fnames :
18         fnameAry.append(fname)
19
20 node = TreeNode()
21 node.data = fnameAry[0]
22 root = node
23 memory.append(node)
24
25 dupNameAry = []
26
```

os.walk() : 파일의 패스, 폴더들, 파일들을 튜플로 리턴



### 폴더 및 하위 폴더에 중복된 파일 이름 찾기 코드 2

```
27 for name in fnameAry[1:] :
28
29     node = TreeNode()
30     node.data = name
31
32     current = root
33     while True :
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50 dupNameAry = list(set(dupNameAry))
51
52 print(folderName, '및 그 하위 디렉터리의 중복된 파일 목록 -->')
53 print(dupNameAry)
```

set() : 다수 데이터에서 중복되지 않은 것 추출



# 다음 강의 예고

---

- 그래프
  - 그래프 기본
  - 그래프 구현
  - 그래프 응용



# Q & A

감사합니다.