10장. 색인과 이진 검색 트리

이진 검색 트리 실습

- 이진 검색 트리(BST)를 구현
- BST에 일련의 연산을 하고 난 후의 BST 데이터를 출력

이진 검색 트리 테스트 코드 및 실행결과

BST > binarySearchTreeDemo.py > ...

```python
from binarySearchTree import *

bst1 = BinarySearchTree()
bst1.insert(10)
bst1.insert(20)
bst1.insert(5)
bst1.insert(80)
bst1.insert(90)
bst1.insert(7550)
bst1.insert(30)
bst1.insert(77)
bst1.insert(15)
bst1.insert(40)
bst1.delete(7550)
bst1.delete(10)

print("preorder: ")
bst1.preorder(bst1.getRoot())
print("\ninorder: ")
bst1.inorder(bst1.getRoot())
print("\npostorder: ")
bst1.postorder(bst1.getRoot())
print("\n")
```

```
preorder:
15 5 20 80 30 77 40 90
inorder:
5 15 20 30 40 77 80 90
postorder:
5 40 77 30 90 80 20 15
```

이진 검색 트리의 구현

코드 10-1 binarySearchTree.py

```
1 class TreeNode:
2    def __init__(self, newItem, left, right):
3        self.item = newItem
4        self.left = left
5        self.right = right
6
7 class BinarySearchTree:
8    def __init__(self):
9        self.__root = None
10
```

```
11   # [알고리즘 10-1] 구현: 검색
12   def search(self, x) -> TreeNode:
13       return __searchItem(self.__root, x)
14
15   def __searchItem(self, tNode:TreeNode, x) -> TreeNode:
16       if (tNode == None):
17           return None
18       elif (x == tNode.item):
19           return tNode
20       elif (x < tNode.item):
21           return self.__searchItem(tNode.left, x)
22       else:
23           return self.__searchItem(tNode.right, x)
24
```

이진 검색 트리의 구현

```
25  # [알고리즘 10-3] 구현: 삽입
26  def insert(self, newItem):
27      self.__root = self.__insertItem(self.__root, newItem)
28
29  def __insertItem(self, tNode:TreeNode, newItem) -> TreeNode:
30      if (tNode == None):
31          tNode = TreeNode(newItem, None, None)
32      elif (newItem < tNode.item):  # branch left
33          tNode.left = self.__insertItem(tNode.left, newItem)
34      else:                         # branch right
35          tNode.right = self.__insertItem(tNode.right, newItem)
36      return tNode
37
```

이진 검색 트리의 구현

```
38  # [알고리즘 10-7] 구현: 삭제
39  def delete(self, x):
40      self.__root = self.__deleteItem(self.__root, x)
41
42  def __deleteItem(self, tNode:TreeNode, x) -> TreeNode:
43      if (tNode == None):
44          return None           # Error! Item not found
45      elif (x == tNode.item):   # item found!
46              tNode = self.__deleteNode(tNode)
47      elif (x < tNode.item):
48              tNode.left = self.__deleteItem(tNode.left, x)
49      else:
50              tNode.right = self.__deleteItem(tNode.right, x)
51      return tNode # tNode: parent에 매달리는 노드
52
```

이진 검색 트리의 구현

```
53  def __deleteNode(self, tNode:TreeNode) -> TreeNode:
54      # 3가지 case
55      #    1. tNode이 리프 노드
56      #    2. tNode이 자식이 하나만 있음
57      #    3. tNode이 자식이 둘 있음
58      if tNode.left == None and tNode.right == None: # case 1(자식이 없음)
59          return None
60      elif tNode.left == None:  # case 2(오른자식뿐)
61          return tNode.right
62      elif tNode.right == None: # case 2(왼자식뿐)
63          return tNode.left
64      else: # case 3(두 자식이 다 있음)
65          (rtnItem, rtnNode) = self.__deleteMinItem(tNode.right)
66          tNode.item = rtnItem
67          tNode.right = rtnNode
68          return tNode  # tNode survived
69
```

이진 검색 트리의 구현

```
70  def __deleteMinItem(self, tNode:TreeNode) -> tuple:
71      if tNode.left == None:
72          # found min at tNode
73          return (tNode.item, tNode.right)
74      else: # branch left
75          (rtnItem, rtnNode) = self.__deleteMinItem(tNode.left)
76          tNode.left = rtnNode
77          return (rtnItem, tNode)
78
79  # 기타
80  def isEmpty(self) -> bool:
81      return self.__root == self.NIL
82
83  def clear(self):
84      self.__root = self.NIL
```