2018년도 가을학기 SCSC 알고리즘
실습 9주차

문현수, munhyunsu@cs-cnu.org

Thursday November 08, 2018

1 개요

우리는 지난 실습시간에 컴퓨터공학에서 부모-자식 관계로 '연결되어있는' 데이터를 표현할 때 가장 널리 사용하는 트리(Tree)를 살펴보았다. 이러한 트리 중에서 정렬된 트리(Ordered Tree)라고도 불리는 이진 탐색 트리(Binary Search Tree)가 있다. BST(Binary Search Tree)는 노드(Node)의 키(Key)값을 기준으로 각 노드를 정렬하여 트리 형태로 저장한다.

BST는 대부분의 연산에 O(logn) 시간 복잡도를 가지기 때문에 성능을 필요로 하는 응용에서 자주 사용된다. Insert, Delete, Search에서 이미 정렬되어 있다는 특징으로 평균 O(logn) 최악의 경우 O(n)이 걸린다. 컴퓨터공학 다양한 분야에서 활용되는 BST지만 그대로 사용하기에는 결정적인 문제가 있다. 만약 Root의 키가 '최대값' 혹은 '최소값'으로 설정되면 어떻게 될까? 스스로 생각해보자.

2 이진 탐색 트리(Binary Search Tree)

BST에서 사용할 노드는 key, value와 함께 left, right, 그리고 parent도 멤버 변수로 가지고 있어야 한다. 이 parent는 Insert, Delete할 때 활용된다.

Listing 1: 에지 정보를 포함하는 Node 클래스 node.py

```
1 class Node(object):
2     def __init__(self, key=None, value=None,
3                  parent=None, left=None, right=None):
4         self._key = key
5         self._value = value
6         self._parent = parent
7         self._left = left
8         self._right = right
```

```
9
10      def set_key(self, key):
11          self._key = key
12
13      def get_key(self):
14          return self._key
15
16      def set_value(self, value):
17          self._value = value
18
19      def get_value(self):
20          return self._value
21
22      def set_parent(self, parent):
23          self._parent = parent
24
25      def get_parent(self):
26          return self._parent
27
28      def set_left(self, left):
29          self._left = left
30
31      def get_left(self):
32          return self._left
33
34      def set_right(self, right):
35          self._right = right
36
37      def get_right(self):
38          return self._right
39
40      def __str__(self):
41          return str({'key': self._key, 'value': self.
                _value,
42                      'left': self._left, 'right': self.
                           _right})
```

구현한 노드 클래스를 이용하여 Figure 1와 같은 트리를 구성해보도록 하자.

Listing 2: Binary Search Tree Class

```
1   class BinarySearchTree(object):
```

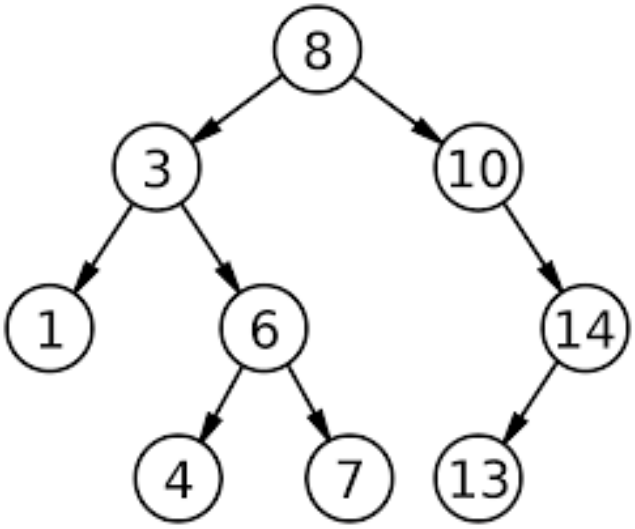

Figure 1: 이진 탐색 트리.png

```
2       def __init__(self):
3           self._root = None
4           self._size = 0
5
6       def get_size(self):
7           return self._size
8
9       def get_root(self):
10          return self._root
```

BST는 멤버 변수(필드)로 root 노드와 크기(size)를 가진다. 이는 Insert, Delete가 일어날 때마다 변화되어야 한다.

Listing 3: Binary Search Tree Class Interface

```
11      def search_by_key(self, key):
12          pass
13
14      def insert_node(self, key, value):
15          pass
16
17      def find_min(self, current_node=None):
18          pass
19
20      def delete_node(self, key):
```

```
21          pass
22
23      def print_bst(self):
24          pass
```

위의 멤버 함수(메소드)들은 BST가 가지고 있어야할 것들이다. 이번 실습 및 과제에서는 인터페이스라고 부르는 이러한 사양을 만족하도록 구현해볼 것이다.

Listing 4: Binary Search Tree Class print_method

```
23      def print_bst(self):
24          stack = list()
25          current_node = self._root
26          while True:
27              while current_node is not None:
28                  stack.append(current_node)
29                  current_node = current_node.get_left()
30              if len(stack) != 0:
31                  pop_node = stack.pop()
32                  print(pop_node.get_key(), pop_node.
                        get_value())
33                  current_node = pop_node.get_right()
34              else:
35                  break
```

위의 트리 출력 함수는 inorder 순회를 하고 있다. 결과적으로 BST에서는 key 값의 오름차순 출력이 이루어진다.

Listing 5: is_bst()

```
1   def is_bst(root):
2       isit = True
3       queue = list()
4       queue.append(root)
5       while len(queue) != 0:
6           current_node = queue.pop(0)
7           if current_node.get_left() is not None:
8               queue.append(current_node.get_left())
9               if current_node.get_key() < current_node.
                    get_left().get_key():
10                  isit = False
11          if current_node.get_right() is not None:
```

```
12                queue.append(current_node.get_right())
13                if current_node.get_key() > current_node.
                      get_right().get_key():
14                    isit = False
15
16        return isit
```

위의 함수는 BST를 검증하는 함수이다. 일정한 규칙이 있기 때문에 BST는 모든 노드를 순회하며 규칙에 따르고 있는지 확인하는 것으로 검증할 수 있다.

2.1 insert_node()

Listing 6: insert sudo code

```
1  void insert(Node*& root, int key, int value) {
2    if (!root)
3      root = new Node(key, value);
4    else if (key == root->key)
5      root->value = value;
6    else if (key < root->key)
7      insert(root->left, key, value);
8    else  // key > root->key
9      insert(root->right, key, value);
10 }
```

BST에 노드를 추가하는 것은 간단하다. 우리가 지난 시간에 트리를 순회했던 것을 이용하여 '조건에 맞는' 위치를 찾아가 연결해주면 된다. 위의 코드는 insert 멤버 함수를 재귀적으로 구현한 수도 코드다.

2.2 delete_node()

BST를 구현할 때 가장 어려운 것은 노드를 지우는 작업이다. 서로 '연결되어 있는' 형태이기 때문에 아무 고려없이 연결을 끊어버릴 수 없다. 노드를 지울 때에는 크게 4가지 상황이 있다.

- Left, Right Node가 모두 존재
- Left 노드가 존재
- Right 노드가 존재
- 자식 노드가 없음

지우려는 노드의 상황을 위의 조건에 맞게 따져본 후 삭제 작업을 해야한다.

Listing 7: insert sudo code

```
def binary_tree_delete(self, key):
    if key < self.key:
        self.left_child.binary_tree_delete(key)
        return
    if key > self.key:
        self.right_child.binary_tree_delete(key)
        return
    # delete the key here
    if self.left_child and self.right_child: # if both children are present
        successor = self.right_child.find_min()
        self.key = successor.key
        successor.binary_tree_delete(successor.key)
    elif self.left_child:   # if the node has only a *left* child
        self.replace_node_in_parent(self.left_child)
    elif self.right_child:  # if the node has only a *right* child
        self.replace_node_in_parent(self.right_child)
    else:
        self.replace_node_in_parent(None) # this node has no children
```

위는 delete_node()의 수도코드다. 하지만 그대로 구현하기 너무 어려울 경우 자신의 논리로 구현해도 무방하다. 자신만의 논리, 알고리즘을 세우기 위해선 아래 그림과 같이 직접 각 상황에 대하여 순서를 따져보아야 한다.

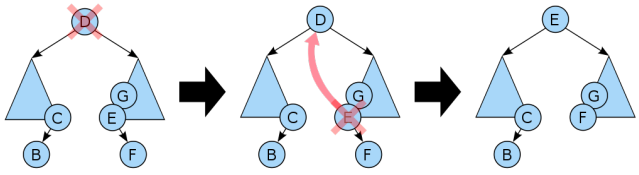

Figure 2: 이진 탐색 트리 제거

3 과제

3.1 과제 목표

- 이진 탐색 트리(Binary Search Tree) 구현 - 모든 인터페이스

3.2 제출 관련

- 마감 날짜: 2018. 11. 14. 23:59:59
- 딜레이: 1일당 10% 감점(처음 2일까지는 -2)
- 제출 방법: 과목 사이버캠퍼스
- 제출 형식: 과제 리포트 PDF(HWP, DOC 받지 않음!), 소스코드(구현한 .py 만 추가할 것)를 압축한 .zip 파일
- 리포트에 포함해야하는 내용: 목표, 목표를 위해 알아야하는 것, 해결 방법, 결과, (선택)느낀점 or전달할 말
- 제출 파일 제목: AL_201550320_문현수_09.zip(파일명 준수!)

3.3 조교 연락처

- 문현수
- munhyunsu@cs-cnu.org
- 공학5호관 633호 데이터네트워크연구실
- 이메일, 연구실 방문