201201871 서현택

1.목표

이진탐색트리 구현

2.과제를 해결하는 방법

1. 이진탐색트리에 대한 이해
2. insert, delete, search 인터페이스 구현

3.과제를 해결한 방법

1. search_by_tree

self._root에 key=key인 노드를 넣고, self._root가 없다면 False 반환하여 탐색 결과가 False 라는 것을 나타낸다. 찾으려는 key가 현재 노드의 key값과 일치하면 True를 반환하고, 적으면 현재 노드의 왼쪽 자식 노드를 search_by_key에 넣고 순환시켜 탐색한다. 찾으려는 key가 현재 노드의 key값보다 크다면 현재 노드의 오른쪽 자식 노드를 search_by_key에 넣고 순환시켜 탐색한다.

2. insert_node

현재 노드가 없다면, self._root에 Node를 입력하여 추가하고, size를 1 늘린다. 그 외의 경우엔 __insert_node를 통하여 삽입을 한다.

__insert_node에서는 삽입하려는 노드의 key값과 현재 노드의 key값을 비교하여 적거나 같으면 왼쪽 자식 노드가 있는지 판단한 후, 있다면 왼쪽 자식 노드에서 __insert_node를 넣어 재귀를 통해 삽입을 진행한다. 왼쪽 자식 노드가 없다면 왼쪽 노드에 추가하려는 노드를 삽입한다.

반대로 삽입하려는 노드의 key값이 현재 노드의 key값보다 크다면 오른쪽 자식 노드가 있는지 판단한 후, 있다면 오른쪽 자식 노드에서 __insert_node을 넣어 재귀를 통해 삽입을 진행한다. 없다면 오른쪽 노드에 추가하려는 노드를 삽입한다.

3. delete_node

self._root와 deleted 값에 _deleted_node의 값을 저장한 뒤 deleted 값을 반환하여 삭제 여부를 확인한다.

_deleted_node에서는 node가 없다면, node와 False를 반환하여 삭제가 되지 않았음을 나타낸다.

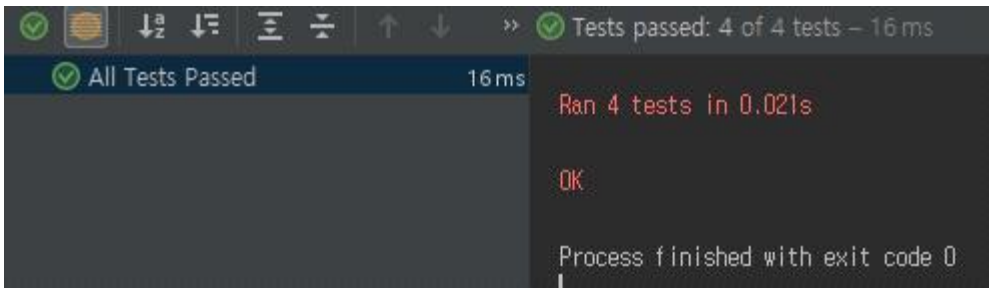
만약 삭제하려는 key값과 node의 key값이 같다면 deleted에 True를 저장한 뒤 node의 왼쪽 자식과 오른쪽 자식이 있는지를 판단하여 둘 다 있다면 parent의 현재 node 값, child에는 node의 오른쪽 노드 값을 저장하고 child의 왼쪽 자식 노드가 없어질 때까지 반복하여 가장 마지막 자식 값을 child에 저장하고, child의 left 값으로 node의 left값을 넣는다.

만약 parent 노드와 node가 같지 않다면 parent의 왼쪽 노드 값으로 child의 오른쪽 node 값을 저장하고, child의 오른쪽 노드 값으로 node의 오른쪽 노드 값을 저장한다. node에는 child의 값을 저장한다.

이 외에 node가 왼쪽 노드 또는 오른쪽 노드를 가진다면 node에 node의 왼쪽 노드 또는 오른쪽 노드를 저장한다.
만약 삭제하려는 key값이 node의 key값보다 적다면 node의 left와 deleted에 _deleted에 node의 왼쪽 노드를 넣어 순환시킨 값을 저장한다.
만약 삭제하려는 key값이 node의 key 값보다 크다면 node의 right와 deleted에 _deleted에 node의 오른쪽 노드를 넣어 순환시킨 값을 저장한다.
마지막으로 이런 과정을 통해 저장된 node와 deleted 값을 반환한다.

4.결과화면

(1) 테스트 코드



(2)BST_sample