



bti\_insert를 이용하여 lecture에 주어진 모양의 binary tree를 먼저 구성하였다.

search 함수는 조금 수정하여 search 함수를 통해 key값을 갖는 노드가 base로부터 어디에 위치해있는지 출력하도록 하였다.

처음 주어진 이진트리는 levelorder traverse를 통해 확인하였다.

그 뒤 bti\_sort함수, 즉 주어진 불균형한 이진트리의 값들을 모두 배열에 순서대로 (오름차순) 뽑아내는 작업을 수행하고 balance 함수를 통해 균형잡힌 트리를 다시 만들어 root의 주소를

base->left에 저장하였다. sort와 balance 작업은 모두 Inorder traverse의 재귀적인 동작과 유사하게 동작한다.

균형잡힌 이진트리를 level order traverse를 통해 제대로 형성되었음을 확인하였다. 마지막으로 똑같은 노드에 대해 search를 수행하여 제대로 바뀌었음을 한번 더 확인하였다.

이렇게 균형잡힌 이진트리를 만들면 탐색속도에 있어서 큰 이득을 볼 수 있다. 따라서 주어진 트리가 심한 불균형 이진트리여서 탐색속도가 느리다면 balance 작업을 고려해볼 수 있다.