* Binary Search Trees : Binary Search Algorithm과 유사하게 즉, 이진 탐색과 유사한 방법으로 탐색할 수 있도록 해주는 틀. 키 값의 관계가 key(u)<= key(v) <= key(w) 이다. (u: left subtree, v : parent , w : right subtree) 그리고 external nodes는 items를 저장하지 않는다. Search, Insertion 그리고 Deletion도 위의 관계로 탐색한다. Deletion에서는 internal node 라면 삭제 후, 키 값에 맞춰서 다시 정렬해줘야 한다. O(n)의 공간을 사용하고, get, put 그리고 erase는 O(h) 시간이 걸린다. 높이 h는 worst case에선 O(n)이, best case에서는 O(log n)이다.
* AVL Trees : subtree의 높이를 적절히 제어해 전체 tree가 어느 한쪽으로 늘어지지 않도록 한(균형 잡힌) binary Search Tree이다. 같은 parent 밑의 children 들의 높이가 최대 1만 차이 나야 한다. Insertion, Deletion 후에는 Restructuring이라는 balance를 맞춰주는 일을 해야 한다. Linked-structure binary tree를 사용하면 single restructure은 O(1) time이 걸린다. Find, put 그리고 erase는 O(log n) 시간이 걸린다.