Handson 1

Sofia Pisani
Mat:646301

October 14, 2023

1 Binary Search Tree

A binary tree is a binary search tree if for each of its nodes, the nodes key is greater or equal to all keys in its left subtree, and lesser than all keys in its right subtree.

We implemented the public function is_bst() which returns whether the given tree is a binary search tree. The function calls a private recursive function that given a node_id returns whether the subtree with it at its root is a bst, and the maximum and minimum values in the subtree.

Being given a tree, we know that it's a bst if all the following are true:

- All of its nodes are roots of bsts
- The maximum key of the left subtree is lesser or equal to the roots key
- The minimum key of the right subtree is greater than the roots key

This justifies the recursive implementation, which is of linear complexity as it visits each node once.

2 Balanced Tree

A binary tree is balanced if, for each of its nodes, the heights of its left and right subtrees differ at most by one.

We implemented the public function is_balanced() which returns whether a given tree is balanced.

The function calls a private recursive function that given a node_id, returns whether the subtree with it at its root is a balanced tree, and the height of the subtree.

Again the recursive implementation doesn't require much explanation. The solution is linear as it visits each node once.

3 Max Heap

A binary tree is a max heap if it is complete, and if every node's key is greater or equal than its children's. We implemented the public function is_max_heap() which returns whether the given tree is a max heap.

The function calls two private recursive functions. One for checking the completeness of the tree and one for checking the max-heap property.

3.1 Completeness

The function rec_is_complete() returns whether the subtree with given root is complete, whether it is perfect, and its height.

A tree is complete if all of its levels are full, except possibly the last level, which must be filled from the left. A tree is perfect if all of its levels are full.

We can check if a tree is complete by knowing whether its left and right subtrees are complete, perfect, and their heights. In fact, at least one of the following has to be true for the tree to be perfect:

- Both subtrees are perfect and of the same height. The tree will also be perfect in this case.
- The left subtree is perfect, and the right one is complete, with both having the same height.
- The left subtree is complete, and the right one is perfect, but of height one less than the left.

Which justifies our recursive approach. This check is linear in time as it visits each node once.

3.2 Max heap propriety

The check of the propriety is simple, and works by recursively returning both the current key, and whether the propriety has been maintained until now. This check is also linear.

As both checks are linear the whole function is linear.