Imagine a process P invoking `insert(key)`. If it suspends right after lock acquisition, it prevents other processes from using the same insertion point, they will wait. Thus, this data-structure is **not obstruction-free**, and so neither lock-free, nor wait-free.

But some progress guarantees can be provided for specific histories. Consider a concurrent history H and its complete prefix $*H$. Let $H*$ be $H \setminus *H$. Let $H*$ satisfies the following condition.

- For any $start.insert_i(k) \in H*$ there is $end.insert_j(k) \in H$ which precedes $start.insert_i(k)$, i.e. all insert operations in $H*$ insert keys which are already present in the tree prior to their invocation.

Then all operations in $H*$ complete at fixed number of steps bounded by the height of the tree at the end of $*H$.

Proof.

- `insert` operations in $H*$ find the same key being present in the tree and finish execution without locking anything.
- As tree remains constant after the end of $*H$, `contains` and `insert` operations traverse a path of a constant length, which is bound by the height of the tree.