CS 120: Intro to Algorithms and their Limitations

Lecture 5: — Tuesday September 19, 2023

Pset Due: September 20, 2023 **Denny Cao**

---

§1 Announcements

- SRE next Thursday
- Wednesday morning OH on Zoom
- Late Policy: 3 days max for each assignment. Late days $\in \mathbb{N}$

§2 Dynamic Predecessor Data Structure

For a sorted array:

- Insert$(k, v) = O(n)$
- Delete$(k) = O(n)$
- Search$(k) = O(\log n)$
- Next-smaller$(k) = O(\log n)$

§3 Binary Search Tree

Our goal is to solve Dynamic Predecessors more easily.

Definition 3.1 (Binary Search Tree (BST))**.** Data structure defined **recursively**. Base case: $\emptyset$ or has a root R and every vertex has:

- Key K
- Value V
- Left and Right Children V_{left} and V_{right}, each a BST (Could be a $\emptyset$)
- Satisfies the **BST Property**

Property 3.2 (BST Property) $\forall v$: if v has left child v_l :
the keys of v_l and all its descendants are $\leq k_v$.
If v has a right child, similar definition.

Algorithm 3.3 — `If $T = \emptyset$`
[Insert] Insert$(T(k, v))$:
if $T = \emptyset$: Return new BST w/key K, value V, no children
Let $v = \text{root}(T)$
if $k \leq K_V$:
T.Left = Insert$(T_L(K, V))$
else `T.right` = Insert$(T_R(K, v))$ return T

Definition 3.4 (Height h)**.** Length of the longest path from v to a leaf.

- `height(∅) = -1`

Proof that Insert Maintains BST Property. By induction on height `h` "`Insert`$(T(K,V))$ maintains BST if `height(T)` $\leq$ `h`"
Base Case: $h = -1$: No vertices → no vertices to check.
If true for $< h : Insert(T(K,V))$: `T` has a root `v`, so `v` $=$ `root(T)` is well defined.
If $k \leq k_v$: $Insert(T_L(K,V))$: T_L has height $\leq h-1$ □

- $Insert(k,v) = O(h)$
- $Delete(k) = O(h)$
- $Search(k) = O(h)$
- $Next - Smaller(k) = O(h)$

Property 3.5 (AVL Property)

- Every vertex is augmented with height
- Every pair of siblings have heights differing by ≤ 1
- Maintain balance by rotations: Left rotation, right rotation,