CS3000: Algorithms & Data — Summer '24 — Laney Strange

Binary Search Tree Procedures

We went over the Binary Search Trees in class. Below we give pseudocode for three of its procedures: in-order traversal, search, and insert. The first two procedures are recursive and the third is iterative.

We identify a tree T through its root, $T.root$. A given node in the tree, x, is itself a root of a small subtree. Every node has a value, which we'll call $x.key$. Because x is a binary search tree, all the values in its left subtree are smaller than $x.key$ and all the values in its right subtree are greater than key.

Every node x in the tree maintains pointers to its children $(x.left, x.right)$ and to its parent $x.p$.

```
TRAVERSE(x)
if x ≠ NIL
    TRAVERSE(x.left)
    print x.key
    TRAVERSE(x.right)
```

```
BST-SEARCH(x, k)
if x == NIL or k == x.key
    return x
if k < x.key
    return BST-SEARCH(x.left, k)
else
    return BST-SEARCH(x.right, k)
```

```
BST-INSERT(T, z)
x = T.root
y = NIL
while x ≠ NIL
    y = x
    if z.key < x.key
        x = x.left
    else
        x = x.right
z.p = y
if y == NIL
    T.root = z
else if z.key < y.key
    y.left = z
else
    y.right = z
```

Here's how we typeset the above function in CLRS style:

```
\begin{codebox}
\Procname{$\proc{BinarySearch}(x, \id{key})$}
\li \If $x == \const{Null}$
\Then
\li \Return \const{False}
\li \ElseIf  $\id{x.value} == \id{key}$
\Then
\li \Return \const{True}
\li \ElseIf $\id{key} < \id{x.value}$
\Then
\li \Return $\proc{BinarySearch}(\id{x.left}, \id{key})$
\li \Else
\li \Return $\proc{BinarySearch}(\id{x.right}, \id{key})$
\end{codebox}
```