CS3000: Algorithms & Data — Summer '23 — Laney Strange

Binary Search (with a BST)

We went over the Binary Search algorithm in class; this handout is so you can see the function typeset in the CLRS pseudocode style. In this version, the function takes in a Binary Search Tree x and looks for a *key*. It returns True if the key is found in the tree and False otherwise.

We identify a tree through its root, x. The root (and every node) has a value, which we'll call $x.value$. Because x is a binary search tree, all the values in its left subtree are smaller than $x.value$ and all the values in its right subtree are greater than $x.value$.

We won't worry about how the tree got created, just the algorithmic step of searching for a key in the tree. We'll make the following assumptions:

- The BST is balanced
- We access a tree x by its root.
- $x.left$ is x's left subtree
- $x.right$ is x's right subtree
- All the values are distinct

BinarySearch(x, key)

```
1  if x == Null
2       return False
3  elseif x.value == key
4       return True
5  elseif key < x.value
6       return BinarySearch(x.left, key)
7  else
8       return BinarySearch(x.right, key)
```

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}
```