Module 6a: Binary Search Trees

CPSC 110

Peyton Seigo

2018-10-12

Module 6a: Binary Search Trees

Learning goals

- Be able to reason informally about the time required to search data.
- Be able to identify problem domain information that should be represented using binary search trees.
- Be able to check whether a given tree conforms to the binary search tree invariants.
- Be able to use the design recipes to design with binary search trees.

Notes

Linear search:

- Best case: `1` operation, constant O(1)
- Worst case: `n` operations, linear O(n)
- Average case: `n/2` operations, linear O(n)

Binary Search Trees:

- A binary tree (every node has 0, 1, or 2 children) with the properties:
  - ALL nodes in left sub-tree have a value less than the parent
  - ALL nodes in right sub-tree have a value greater than the parent
- These rules are invariant; holds true ALL the way down a branch!

Functions using a BST

- Some function may produce `Type or false`.
  - `lookup-key`, a function that searches for a node with a particular key and returns the node's value, may not find the node.
  - Signature for `lookup-key` is `BST Natural -> String or false`

Testing a BST

- We have 2 self-reference cases, so our check-expects should check both cases (i.e. left and right)
- Just like for lists, tests should test "**2-deep**"
  - Test all four scenarios for traversing a BST two levels
  - Left Left, Left Right, Right Left, Right Right