Self-Practice Week 5 - Search (part 1)

Symbol Tables and BSTs

The goal of this assignment is to better understand the functioning of symbol tables and binary search trees.

Exercise 1 – Linked List vs Ordered Array Symbol Table

Consider two alternative implementations of a symbol table: one using an unordered linked list, and one using an ordered array. Experimentally compare the performance of the two implementations on `put()` and `get()` operations, as you vary the number of `n` of key/value pairs you insert and search for. At what value of `n` does the ST implemented via an ordered array become 10, 100, and 1000 faster than the one implemented via sequential search?

Exercise 2 – Is a Binary Tree a Binary Search Tree?

Consider the Binary Search Tree data structure seen at lectures to realise the Symbol Table ADT. Design and implement an algorithm that, given as argument a `Node` (with a value and a pair of left/right children nodes), determines if this is the root of a binary search tree and returns True if so, or False otherwise.

Hint: design a recursive function `isBST(node_x, min_key, max_key)` that determines whether `node_x` is the root of a binary search tree with all keys between `min_key` and `max_key`.

Exercise 3 – Interval Search in a BST

Design and implement an algorithm that, given a Binary Search Tree and two keys `x` and `y`, finds all keys between (and excluding) `x` and `y` in the BST. The runtime should be proportional to `lg N + M`, where `N` is the size of the BST and `M` is the number of keys returned.

Hint: combine ideas from BST search and in-order traversal.