HandsOn - 1

Simone Passèra

Oct 18, 2024

1 Introduction

The objective of this hands-on is to implement recursive traversals of a binary tree in Rust.

The exercises I have implemented follow the same strategies we have seen in class.

2 Code Implementation

2.1 Exercise 1

Exercise: Write a method to check if the binary tree is a Binary Search Tree.

The `is_bst` function checks if a binary tree satisfies the binary search tree (BST) property by calling the recursive helper `rec_is_bst`. This helper verifies that each node's key lies within a valid range.

For each node, it ensures the key is between the given min and max bounds, then recursively checks the left and right subtrees. The recursion stops when None is encountered (indicating a leaf node's child), returning true. If any node violates these constraints, the function returns false.

2.2 Exercise 2

Exercise: Write a method to solve the Maximum Path Sum problem. The method must return the sum of the maximum simple path connecting two leaves.

The `max_path_sum` function computes the maximum path sum in a binary tree by calling the recursive helper `rec_max_path_sum`. This helper returns two values: the maximum sum that can be obtained by including the current node and one of its subtrees, and the overall best path sum found so far in the subtree.

For each node, the function recursively calculates the maximum sums for the left and right subtrees. It then adds the node's value to the greater of the two sums to find the maximum sum involving the current node. The overall best path sum is determined by comparing the best sums from both subtrees and the sum that includes the current node and both subtrees.

The recursion stops when None is encountered (indicating a leaf node's child), and the final result is the maximum path sum across the entire tree.