Hands On 1

Competitive Programming and Contests

Leonardo Crociani (Mat. 615392)

Contents

1 Hands On 1

1.1 Introduction

The solutions for each problem have been implemented using the `Tree` struct provided on the course page. These functions are implemented as methods for this struct.

1.2 IsBST

To check if a tree is a BST, we need to verify that for each node, the maximum value from the left child is smaller than the node's key, and the minimum value from the right child is greater than the node's key. Creating external functions to compute the minimum and maximum is not efficient, so each function application on a subtree returns a triple: `<condition_state, min_in_its_subtree, max_in_its_subtree>`.

1.3 IsBalanced

To check if a tree is balanced, at every node, the height of the left subtree and the height of the right subtree must differ by at most 1. As above, the implemented function returns, for each node, both the condition and the height of the node.

1.4 IsMaxHeap

A tree can be considered a max heap if it is complete, and every node's key is greater than the keys of its children. To check the property, the recursive function has to keep track of the max_heap condition in the current node, the completeness condition for the current node, and its max value.

1.5 Testing

I've created 4/5 tests for each function (13 in total). I designed them on paper, outlining both the tree and the expected result. If a test failed, I thoroughly checked both the code and the test. I'm confident that the tests are now consistent. Additionally, some friends provided me with additional tests.

1.6 Source code

In the .zip file I've provided you can find both the algorithms and the tests in `/hands-on/1/lib.rs`. To test the code, just execute `cargo test --package hands-on-1 --lib -- tests --nocapture` in the code folder (initialized with `cargo new --lib hands-on-1`).