

# First Handson Report

Simone Stanganini mat. 616834

October 21, 2025

These are my implementations of the 2 problems asked in the handson:

- **fn check\_bst()**: calls the recursive function **rec\_check\_bst()**. This function recursively checks the left and right subtree. The calls return a struct that contains the max and min found in the subtree and if the bst rule was respected. The base case is, of course, true. The first check is for the bad case: when the current node's key is greater than the min from the right subtree or less than the max from the left subtree. If so, we return false and propagate the failure. If we pass this check, we propagate the results and update the min and the max with the current node's key.
- **fn max\_path\_sum()**: This function first create **max** variable, then calls the recursive function **rec\_max\_path\_sum()** and passes it as a mutable reference for tracking the maximum path found at every moment, then in the recursive function we get the best path sum from the left and right subtrees. Then we calculates two values:
  1. **current\_path\_sum**: the path sum through the current node ( $\text{key} + \text{left} + \text{right}$ ).
  2. **res**: the best path **to one leaf** to propagate up to the parent ( $\text{key} + \max(\text{left}, \text{right})$ ).

Then we updates the global **max** if **current\_path\_sum** is larger and we returns **res** for the parent call. The base case (an empty child) returns 0.