Exercise Session n. 10

Algorithms and Data Structures

Exercise 157 (m15)

Consider the following classic insertion algorithm for a binary search tree:

```
BST-Insert(t, k)
  if t == nil:
    return new-node(k)
  else if k ≤ t.key:
    t.left = BST-Insert(t.left, k)
  else t.right = BST-Insert(t.right, k)
return t
```

Write an algorithm Sort-For-Balanced-BST(A) that takes an array of numbers A, and prints the elements of A in a new order so that, if the printed sequence is passed to BST-Insert, the resulting BST would be of minimal height. Also, analyze the complexity of your solution.

Exercise 163 (f15)

Consider an algorithm BST-Find-Sum(T, v) that, given a binary search tree T containing n distinct numeric keys, and given a target value v, finds and returns two nodes in T whose keys add up to v. The algorithm returns null if no such keys exist in T. BST-Find-Sum may not modify the tree, and may only use a constant amount of memory.

Question 1
Write BST-Find-Sum. You may use the basic algorithms that operate on binary search trees (BST-Min, BST-Successor, BST-Search, etc.) without defining them explicitly.

Question 2
Write a variant of BST-Find-Sum(T, v) that works in O(n) time. If your solution to Exercise 1 already has this complexity bound, then simply say so.