**Advanced Algorithms**

**Exercise for Lecture 5**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name** |  | **Student ID** |  |
| **Problem 1** |  | | |
| **Problem 2** |  | | |
| **Problem 3** |  | | |
| **Problem 4** |  | | |
| **Total Score** |  | | |
| **Notes** | Deadline: **2023-09-26 24:00**  Submission Format: ‘**Lecture5\_Name\_Student ID.docx**’, and please send to: **[algorithms\_23fall@163.com](mailto:algorithms_23fall@163.com)**.  This assignment is meant to be an evaluation of your **individual** understanding coming into the course and should be completed **without collaboration** or outside help. | | |

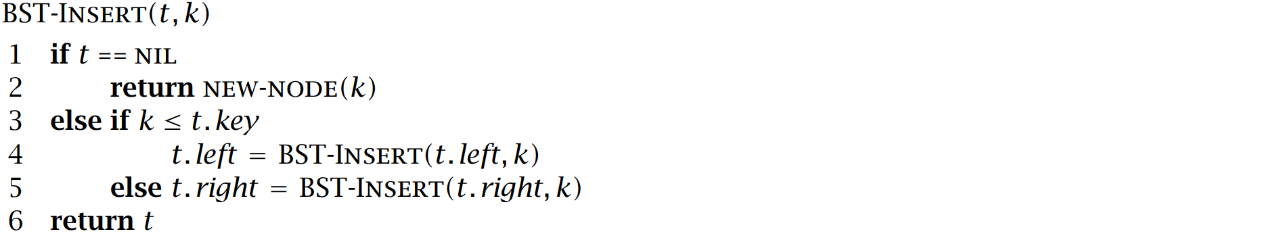
**Problem 1. [25 points]** Draw a binary search tree containing keys

, inserted in this order. Then, add keys , in this order, and again draw the tree. Then delete keys and , in this order, and again draw the tree.

**Solution:**

**Problem 2. [20 points]** Given a collection of numbers and a number , the upper bound of in is the minimal value such that , or NULL if no such value exists. For example, given , the upper bound of is , while the upper bound of is and the upper bound of is NULL. Write an algorithm UPPER-BOUND-BST() that returns the upper bound of in a binary search tree . Analyze the complexity of UPPER-BOUND-BST.

**Solution:**

**Problem 3. [25 points]** Consider the following classic insertion algorithm for a binary search tree:

Write an algorithm SORT-FOR-BALANCED-BST() that takes an array of numbers , and prints the elements of 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.

**Solution:**

**Problem 4. [30 points]** Consider an algorithm BST-FIND-SUM() that, given a binary search tree containing distinct numeric keys, and given a target value , finds and returns two nodes in whose keys add up to . The algorithm returns NULL if no such keys exist in . BST-FIND-SUM may not modify the tree, and may only use a constant amount of memory.

1. Write BST-FIND-SUM(). You may use the basic algorithms that operate on binary search trees (TREE-MINIMUM, TREE-SUCCESSOR, TREE-SEARCH, etc.) without defining them explicitly.
2. Write a variant of BST-FIND-SUM() that works in time. If your solution to *Exercise a* already has this complexity bound, then simply say so.

**Solution:**