**Dept. of Computer Science**
Wesley W. Terpstra
Dr.-Ing. Ilia Petrov
Prof. Alejandro Buchmann, Ph. D.
Summer Term 2009

TECHNISCHE
UNIVERSITÄT
DARMSTADT

# Introduction to Computer Science II

## 4. Lab

To execute all of the recently planned missions, the bureau has had to hire a lot of new agents. Unfortunately, it appears some double agents have slipped through the hiring process. These double agents might be foreign nationals, still serving their country of origin, or just spies committing industrial espionage. These double agents threaten not only the success of missions involving them, but also the confidentiality of the bureau's work. They must be eliminated.

On a few occasions, interference from double agents has been detected. Since the time-frame of these intelligence failures is known, it is possible to exonerate any agent logged as elsewhere at that time. Therefore, the bureau has begun spying on its employees en masse. Whenever an agent is known to be somewhere, this time-period is recorded.

You are to compile a database of when agents were legitimately occupied. Agents are guilty until proven innocent, so your task is only to clear agents you can prove were uninvolved in the breach. Any agents not cleared by you will be handled by internal affairs. As the actions of double agents continue to be detected, it must be possible to query the database at any time. Otherwise innocent agents might be subjected to undue ... scrutiny. Therefore, insertion and query operations must be processed interleaved; new breaches are constantly being detected (requiring a query) and agent activity logs continue to be processed (requiring an insert).

It is clear to you that an appropriate index structure is needed for efficiency. This data structure must store the time intervals agents were busy. It must also support finding all agents who were busy when a breach involving double agents was detected. Your experience suggests that you could augment an AVL tree to find overlapping intervals instead of just keys.

**You must implement:** CounterIntelligence.java to identify innocent agents!

The bureau needs this assignment completed by **23:59 June 21th, 2009**!

*Handle carefully as this assignment is prone to self destruct!*

**Step 0:** Download the skeleton code from `https://www.dvs.tu-darmstadt.de/teaching/inf2/2009-en/Problem4.zip`

**Step 1:** Implement simple binary tree key insertion in .add().

*Hint:* Use the start of the interval as the primary sort key and the agent name as the secondary sort key.

*Hint:* Write the function recursively to save pain later.

**Step 2a:** Modify the add() function to update height.

*Hint:* Use the helper function updateHeight.

**Step 2b:** Implement AVL right and left rotation.

*Hint:* Step2bTest does not use the add() function. It only checks rotateLeft() / rotateRight(). You can safely implement these independently.

**Step 2c:** Keep the tree balanced.

*Hint:* Use right and left rotations in rebalance() when balance() is $\geq 2$ or $\leq -2$.

**Step 3:** Modify the add() and rotation functions to update biggest_end.

*Hint:* A helper function like updateBiggest() might simplify your code.

**Step 4:** Implement remove() to remove an agent's erroneous alibi.

*Hint:* You will need to the helper method removeSmallest.

**Step 5:** Implement the query function to exonerate agents.

*Hint:* Use biggest_end to decide if you need to explore the left subtree.

Some additional information about AVL trees can be found under:

a) `http://en.wikipedia.org/w/index.php?title=AVL_tree&oldid=292056402`
b) Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. *Introduction to Algorithms*, 2nd edition. Chapter 13.5 (Problem 13-3).

Information about interval trees and search can be found under:

a) `http://en.wikipedia.org/w/index.php?title=Interval_tree&oldid=284625704`
b) Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. *Introduction to Algorithms*, 2nd edition. Chapter 14.3.