CodeToasters

Week #4

05/22/13

Bonus Problem Review – Evil Banks

We'll start the day with a brief review of last week's bonus problem. Rather than exhaustively solving it, we'll open up a general discussion of the optimizations people have attempted and their implementations.

The problem, for those who haven't seen it:

Banks can now treat every transaction that occurs in a one-hour period as simultaneous, and they can use this to their advantage by processing the transactions of any user in whatever order they want, including ways that make the user overdraft where they wouldn't otherwise (say by processing all the withdrawals first and leaving the deposits 'til the end).

To avoid being caught doing this, they decide they need a function that does the following: given a list of transactions (positive and negative integers), return the smallest amount below zero that can be hit (summing the numbers one by one in any order) while also only going below zero once. For the sake of the problem assume that the starting balance is always zero.

Warm-up Walkthrough – Quicksort – Peter Lyle-Dugas

Peter today will walk us through the implementation of quicksort, a popular nlog(n) in-place sorting algorithm.

For his implementation thereof, see: http://bit.ly/11cVQaU

Main Problem – (De)Serializing an Unbalanced Binary Tree – Evan Brynne

The goal of this function is to write two functions, one of which takes the head node of an unbalanced binary tree and serializes it, and another which takes the serialized form of that tree and converts it back to a traditional tree form (with a provided Node class). Minimize storage space as much as possible for all possible tree structures.

Bonus Problem – Number Integrity

Given a list of integers between one and nine inclusive, modify as few entries as possible to make possible the recovery of any other single number in the list. Basically, any one value in the list (that isn't one of the modified values) should be derivable from a minimal number of modified values and the rest of the unmodified list items. The modified data-integrity numbers must also be values between one and nine.

The goal of this problem is to have to change as few list items as possible to make the rest of the list derivable.