DUE SAT MAY 7, 2016 AT MIDNIGHT VIA COURSE WEBSITE

HE.

You may collaborate with other students on the homework but you must submit your own individually written solution, identify your collaborators, and acknowledge any external sources that you consult. Do no submit a solution that you cannot explain to me. These extra credit problems will augment your grade, but they need to be *perfectly correct*.

PROBLEM 1 *Arrays*

Many algorithms use arrays as a data structure; and good programming practice requires us to first initialize an array before using it. The time to initialize or "zero-out" the array is usually proportional to the size of the array. In some cases, however, the time it takes to zero-out the array will overwhelm the actual running time of the algorithm. This problem develops a way to overcome this problem.

Suppose your algorithm needs to access an array $A[1,\ldots,n]$ only k times during its execution. Design a method that "simulates" access to a zeroed array A using only $O(k)$ time (thus, you cannot spend $O(n)$ time to initialize A since k might be, say $\sqrt{n}$ and therefore much smaller). By "simulate access," we mean devise two functions $\textsc{read}(i)$ and $\textsc{write}(i,x)$ so that when you call read on an index i that has not already been set, it will read 0. When you call read on an index i that has been written to before, it returns the most recent value of x for which there has been a call to $\textsc{write}(i,x)$. The array A may initially contain garbage (assume it is adversarially chosen garbage). You can also use extra memory in order to run your simulation (but this extra memory may also initially contain garbage). Both READ and WRITE should run in constant time.

PROBLEM 2 *Fibonacci*

Recall that the n^{th} Fibonacci number is $F_n = F_{n-1} + F_{n-2}$. The naive recursive method for computing the n^{th} Fibonacci number takes exponential time. Through dynamic programming, one can compute F_n using n integer operations (i.e., additions and multiplications). Surprisingly, there is an even faster method to compute F_n using only integer operations. Devise and analyze and algorithm that computes F_n in $\Theta(\log n)$ integer operations. Hint, consider the matrix-vector product

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

PROBLEM 3 *Balance*

A binary tree is *adjusted* if the length of any two paths from root to leaves differs by at most 1. Write an algorithm $\textsc{check}(r)$, that on input the root of a tree r determines if tree r is adjusted.