Assignment 1, COMP251, Winter 2019. Jan 10, 2019.

Exercise 1. ALGORITHM DESIGN, ORACLES. We have an oracle that takes two inputs, x and y, and reports whether $x = y$. Complexity in this exercise is measured by the number of uses of this oracle. A set A of n (not necessarily different) elements is given. The number of occurrences of an element is called its frequency. We also know that there is an element with frequency $> n/2$. Show how you can find that element by using the oracle not more than $O(n)$ times.

Exercise 2. DESIGN OF A DIVIDE-AND-CONQUER ALGORITHM. Consider an $n \times n$ chessboard, where n is a power of two. Define a trimino as a 3-piece tile (i.e., a tile that would cover positions $(1,1)$, $(1,2)$ and $(2,1)$ on the chess board). Triminos can be rotated. Develop a divide-and-conquer algorithm for covering the chessboard with triminos such that exactly one of the n^2 spaces on the chessboard is not covered.

Exercise 3. THE FIBONACCI EXAMPLE GENERALIZED. How would you compute x_n in a uniform cost model, where we have $x_0 = 2$, $x_1 = 0$, $x_2 = 7$, and for $n > 3$, $x_n = 7x_{n-1} - 11x_{n-2} + 5x_{n-3}$?

Exercise 4. BIT COMPLEXITY. How would you compute x_n efficiently, where we have $x_0 = 7$, $x_1 = 13$, $x_n = x_{n-1}^3 x_{n-2}^5$, in a bit model of complexity? Only integer arithmetic is permitted. What is your complexity? If it is $O(n^c)$, give c, and if it is $O(c^n)$, give c.

Exercise 5. COMPUTING HAMMING DISTANCES. The Hamming distance between two binary vectors is the number of components that differ. Let $x_1, \ldots, x_n$ be vectors of $\{0,1\}^n$. We would like to compute the Hamming distances between x_i and x_j for all $i \neq j$. Using a RAM model of computation, how fast can you do this (in $O(.)$ notation)? The idea is to beat the obvious $\Theta(n^3)$ algorithm by a good margin, i.e., by achieving $O(n^\alpha)$ complexity for some $\alpha < 2.9$. Partial credit for only getting $o(n^3)$. Hint: You may want to recast this problem in terms of one or more other problems.