MA/CSSE 473 – Design and Analysis of Algorithms

Homework 9A (31 points total) Updated for Winter, 2017

Problems for enlightenment/practice/review (not to turn in, but you should think about them):

How many of them you need to do serious work on depends on you and your background. I do not want to make everyone do one of them for the sake of the (possibly) few who need it. You can hopefully figure out which ones you need to do.

4.4.11 [5.5.4] (multiplication à la Russe)
4.5.2 [5.6.2] (quickselect example and efficiency)

Problems to write up and turn in:

1. (8) (Nim strategy) In class (probably Day 18 or 19 in Winter 2017 schedule) we state(d) that

   in n-pile Nim, a player is guaranteed to be able to win if and only if the Nim sum (as defined in class) is nonzero at the beginning of that player's turn.

   We proved three lemmas (Slide 10) that can be used to prove this statement. Use the results of one or more of these lemmas to construct a proof by induction (perhaps induction on the total number of chips in all of the piles) that the above statement is correct for any nonnegative number of piles and any non-negative number of chips.

   These are the lemmas:

   – Let $x_1, \ldots, x_n$ be the sizes of the piles before a move, and $y_1, \ldots, y_n$ be the sizes of the piles after that move.
   – Let $s = x_1 \oplus \ldots \oplus x_n$, and $t = y_1 \oplus \ldots \oplus y_n$.
   - **Lemma 1:** $t = s \oplus x_k \oplus y_k$, where the removed stones are from pike k.
   - **Lemma 2:** If $s = 0$, then $t \neq 0$.
   - **Lemma 3:** If $s \neq 0$, it is possible to make a move such that $t=0$.

2. (5) Using the algorithm from class (and from Section 4.5 [5.6]and referenced in the previous problem) consider the following situation:

Pile #	Chips
1	77
2	46
3	27
4	74

   Which pile should the player take chips from and how many chips should be taken in order to guarantee a win? Show your work.

3. (6) 4.4.8 [5.5.2] (Ternary Search) Find exact solution to recurrence, not just big-O, so you can compare the two algorithms for efficiency.

4. (12) 4.4.10 [5.5.3] (fake coin divide-into-three) Levitin made me do it! Find exact solution to recurrence, not just big-O, so you can compare the two algorithms for efficiency.