473 Levitin problems and hints HW 09A

Problems 1 and 2 are not from the textbook. Both are about NIM.

1. (8) (Nim strategy) In class (Day 19 in the Fall, 2014 schedule) we stated 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 (see the ICQ solution from that day for details). Use one or more of these lemmas to construct a proof by 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.

Problem 3: 4.4.8 [5.5.2] Ternary Search

2. Consider ***ternary search***—the following algorithm for searching in a sorted array $A[0..n-1]$: if $n = 1$, simply compare the search key K with the single element of the array; otherwise, search recursively by comparing K with $A[\lfloor n/3 \rfloor]$, and if K is larger, compare it with $A[\lfloor 2n/3 \rfloor]$ to determine in which third of the array to continue the search.

   a. What design technique is this algorithm based on?

   b. Set up a recurrence relation for the number of key comparisons in the worst case. (You may assume that $n = 3^k$.)

   c. Solve the recurrence for $n = 3^k$.

   d. Compare this algorithm's efficiency with that of binary search.

Author's Hints:

2. The algorithm is quite similar to binary search, of course. In the worst case, how many key comparisons does it make on each iteration and what fraction of the array remains to be processed?

Problem 4: 4.4.10[5.5.3] fake coin divide into three

3. a. Write a pseudocode for the divide-into-three algorithm for the fake-coin problem. (Make sure that your algorithm handles properly all values of n, not only those that are multiples of 3.)

   b. Set up a recurrence relation for the number of weighings in the divide-into-three algorithm for the fake-coin problem and solve it for $n = 3^k$.

   c. For large values of n, about how many times faster is this algorithm than the one based on dividing coins into two piles? (Your answer should not depend on n.)

Author's Hints:

3. While it is obvious how one needs to proceed if $n \bmod 3 = 0$ or $n \bmod 3 = 1$, it is somewhat less so if $n \bmod 3 = 2$.