Homework Assignment #3

Due Time/Date

2:20PM Tuesday, September 26, 2023. Late submission will be penalized by 20% for each working day overdue.

How to Submit

Please write or type your answers on A4 (or similar size) paper. Drop your homework by the due time in Yih-Kuen Tsay's mail box on the first floor of Management College Building 2. You may discuss the problems with others, but copying answers is strictly forbidden.

Problems

There are five problems in this assignment, each accounting for 20 points. (Note: problems marked with "(X.XX)" are taken from [Manber 1989] with probable adaptation.)

1. (3.5) For each of the following pairs of functions, determine whether $f(n) = O(g(n))$ and/or $f(n) = \Omega(g(n))$. Justify your answers.

	$f(n)$	$g(n)$
(a)	$\sqrt{n}$	$(\log n)^2$
(b)	$n^4 2^n$	4^n

2. Suppose f is a strictly increasing function that maps every positive integer to another positive integer, i.e., if $1 \leq n_1 < n_2$, then $1 \leq f(n_1) < f(n_2)$, and $f(n) = O(g(n))$ for some other function g. Is it true that $\log f(n) = O(\log g(n))$? Please justify your answer. How about $2^{f(n)} = O(2^{g(n)})$? Is it true?

3. (3.18) Consider the recurrence relation

   $$T(n) = 2\ T(n/2) + 1,\ T(2) = 1.$$

   We try to prove that $T(n) = O(n)$ (we limit our attention to powers of 2). We guess that $T(n) \leq cn$ for some (as yet unknown) c, and substitute cn in the expression. We have to show that $cn \geq 2c(n/2) + 1$. But this is clearly not true. Find the correct solution of this recurrence (you can assume that n is a power of 2), and explain why this attempt failed.

4. Solve the following recurrence relation using *generating functions*. This is a very simple recurrence relation, but for the purpose of practicing you must use generating functions in your solution.

   $$\begin{cases} T(0) = 0 \\ T(1) = 1 \\ T(n) = T(n-1) + 2\ T(n-2), \quad n \geq 2 \end{cases}$$

5. The Fibonacci word sequence of bit strings is defined as follows:

$$FW(i) = \begin{cases} 0 & \text{if } i = 0 \\ 1 & \text{if } i = 1 \\ FW(i-1) \cdot FW(i-2) & \text{if } i \geq 2 \end{cases}$$

Here $\cdot$ denotes the operation of string concatenation. The first six Fibonacci words (from $FW(0)$ to $FW(5)$), for instance, are: 0, 1, 10, 101, 10110, 10110101. A given bit pattern may or may not occur in a Fibonacci word (as a substring). For instance, 10101 occurs in $FW(5)$ (but not $FW(4)$ or earlier), while 11101 never occurs in any Fibonacci word.

Suppose we are given a bit pattern p (that is not just a single bit) and asked to determine whether it occurs in some Fibonacci word. To be efficient, we certainly want to search as few Fibonacci words as possible for the pattern. Please give a lower bound m_l and an upper bound m_u in terms of $F^{-1}(|p|)$ ($|p|$ denotes the length of p), as tight as possible, such that, if p will ever occur in any Fibonacci word, it occurs already in some word between $FW(m_l)$ and $FW(m_u)$ inclusively.

Note: Let $F^{-1}(n)$ (for $n \geq 2$) denote the index value i ($i \geq 2$) such that $F(i-1) < n \leq F(i)$, i.e., n is between the two Fibonacci numbers $F(i-1)$ and $F(i)$, possibly equal to $F(i)$ but greater than $F(i-1)$; here, the Fibonacci numbers start from 1, which is indexed as the 0-th element in the sequence. For instance, $F^{-1}(2) = 2$, $F^{-1}(3) = 3$, $F^{-1}(5) = 4$, $F^{-1}(8) = 5$, and $F^{-1}(6) = F^{-1}(7) = 5$.