CS 5200 Homework - 1

Instructor Avah Banerjee

Feb 15, 2024, 12 Noon CST

Problem 1. In this problem we assume all functions are of the form $\mathbb{N} \to \mathbb{R}^+$. Let

$$f^r(n) = \underbrace{f(f(\cdots f(n)}_{r\text{-times}}$$

. The iterated logarithm, denoted as $\log^*(n)$, is defined as:

$$\log^*(n) = \begin{cases} 0 & \text{if } n \leq 1, \\ 1 + \log^*(\log(n)) & \text{if } n > 1. \end{cases}$$

Determine if the following statements holds (briefly justify your answers):

1. $n^{1+1/\log n} = O(n)$ (5 pts)
2. $2^{\log^*(n)} = \Theta(\log^*(n))$ (5 pts)
3. If $h(n)g(n) = \Omega(g(n))$ then $h(n) = \Omega(n)$. (5 pts)
4. Let $f(n) = \sqrt{n}$ and $g(n) = f^{\log\log(n)}(n)$. Then $g(n) = \omega(1)$. (10 pts)

Problem 2 (5+5+5+20 pts). Let $f(n)$ be the nth Fibonacci number with $f(0) = 1, f(1) = 1, f(2) = 2, f(3) = 3, f(4) = 5 \ldots$ and so on. We define an inverse function $f^{-1}(n)$ for all $n \geq 1$ as follows:

$$f^{-1}(n) = m, \text{ if } m \text{ is the least positive integer such that } f(m) \geq n.$$

For example we have: $f^{-1}(1) = 1, f^{-1}(6) = 5, f^{-1}(15) = 7, f^{-1}(40) = 9$ etc.

(a) Show that $f^{-1}(f(n) + n) = n + 1$ whenever $n \geq 5$

(b) Show that $f^{-1}(m(f(n))) \leq n + m$ where m is also a positive integer.

(c) Is it true that $f^{-1}(n) = O(\log n)$?.

(d) Devise an algorithm that computes $f^{-1}(n)$ for a given input n. Prove its correctness and analyze its running time.

Problem 3 (40 pts) Implement the algorithm (in Python) to test primality as described in the section labeled "Carmichael Numbers" in the textbook (page 28), and verify the claim that the algorithm succeeds with at least a 3/4 probability. You also need to implement a deterministic primality tester, which can be used to verify if the output of the randomized primality tester is correct. This tester could be as simple as the trivial primality tester we discussed in class. You are obviously not allowed to use off-the-shelf APIs or functions except for basic arithmetic operations on integers, such as addition and multiplication. However, you should implement the modular arithmetic operations yourself. To determine the success probability, run the algorithm at least 10,000 times on numbers chosen uniformly at random between $\{1, 2^{32} - 1\}$. During your experiments, if you encounter any Carmichael numbers, you should output them. For this, you need to create another function that checks if a number is a Carmichael number.