Homework 1

Vancouver Summer Program 2017
Algorithms and the Internet
Due: July 25, 2017, by 11:59 p.m.

---

1. (Enter Fibonacci.) The Fibonacci sequence is defined as follows: $F_0 = F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for all integers $n \geq 2$.

   (a) Use mathematical induction to show that $F_n \geq 2^{n/2}$ for all $n \geq 6$. There is more than one base case; identify all base cases!

   (b) Similarly, show that $F_n \leq 2^n$ for all n.

   (c) (**Bonus**) Find a if a and b are integers such that $x^2 - x - 1$ is a factor of $ax^{17} + bx^{16} + 1$. **Hint**: The answer is F_n for some $n \geq 1$. It is enough to show this and find n explicitly; you do not need to compute F_n.

2. (Testing i*Phones.) You are testing the physical durability of the new i*Phones to determine the (maximum) height from which they can fall and still not break. You are performing this testing in increments of half a feet. You want to find the largest height, X, from which an i*Phone can be safely dropped. If you wanted to minimize the number of devices used for this test, you could start at 0.5 feet, and keep increasing the height to a maximum of $H \times 0.5$ feet, until a drop breaks the device. This sequential test would require only one device for testing but would require (potentially) a large number of tests. There is a natural tradeoff between the number of devices you break during the testing and number of tests performed. (For example, a binary search between 0 and H would require $\log H$ devices in the worst case.)

   Suppose your manager at a*Pple Corp. limits you to a testing budget of n devices, determine an efficient strategy for identifying X. Let $g_n(H)$ be the number of tests needed by a strategy given a maximum height H and n devices. A strategy is efficient if $\lim_{H\to\infty} g_n(H)/g_{n-1}(H) = 0$.

   Clearly describe your testing strategy and prove that it is indeed efficient.

3. (Studyitis) During VSP 2017, the students are sitting in an $n \times n$ grid. A sudden outbreak of studyitis (a rare condition that lasts forever; symptoms include yearning for homework) causes some students to get infected. The infection begins to spread every minute (in discrete time-steps). Two students are considered adjacent if they share an edge (i.e., front, back, left or right, but NOT diagonal); thus, each student is adjacent to 2, 3 or 4 others. A student is infected in the next time step if either

   - the student was previously infected (there is no cure for this condition), or
   - the student is adjacent to *at least two* students who are affected by studyitis.

Prove that if less than n students are initially infected then the whole class will not be completely infected.

Hint: Can you identify some property at the initial stage and prove, by induction, that this property is preserved at each time step? Think of *invariants*.

4. How Complex is Complex?

   (a) Suppose that you are given an integer $n \geq 0$, and you are asked to compute F_n, the nth Fibonacci number. Devise an algorithm for computing F_n and analyze its worst-case time complexity. Express your solution as a $O\big(T(n)\big)$, where $T(n)$ is the number of steps (either additions or multiplications, depending on your algorithm) required to compute F_n. If $T(n)$ now includes the time required to write the output to memory, can you compute F_n in time $T(n)$ that is a *polynomial* in the size of the input? Justify your answer.

   (b) Suppose you are choosing between three algorithms:

      i. Algorithm A solves the problem at hand by dividing it into five subproblems of half the original size, recursively solving each subproblem, and then combining the solutions in linear time.

      ii. Algorithm B solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in linear time.

      iii. Algorithm C solves problems of size n by dividing them into 9 subproblems of size $n/3$ and recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

      What are the running times of the algorithms and which algorithm would you choose? Hint: Write down the running-time recurrences for each and use the Master Theorem for some algorithms.

      The Master Theorem. The recurrence $T(n) = aT(n/b) + f(n)$ can be solved as follows:

      - If $af(n/b) = \kappa f(n)$ for some constant $\kappa < 1$, then $T(n) \in \Theta\big(f(n)\big)$
      - If $af(n/b) = Kf(n)$ for some constant $K > 1$, then $T(n) \in \Theta\big(n^{\log_b a}\big)$
      - If $af(n/b) = f(n)$, then $T(n) = \Theta\big(f(n)\log_b n\big)$
      - If none of these three cases apply, you're on your own.

   (c) If the statement is true, provide a proof, and if not, provide a counterexample:

      i. Let f, g be functions from $\mathbb{N}$ to $[0, \infty)$. If $f(n) \in O\big(g(n)\big)$, is $2^{f(n)} \in O\big(2^{g(n)}\big)$?

      ii. $O(f+g) = O(f) + O(g)$

   (d) Show that $5n + 8n^2 + 100n^3 \in O\big(n^3 \log(n)\big)$. Is $5n + 8n^2 + 100n^3 \in \Theta\big(n^3 \log(n)\big)$?