Algorithms Homework - Day 3

Instructor: Pat Devlin — prd41@math.rutgers.edu

Summer, 2016

(1) You have a very important question that you need a computer to figure out, and you have three algorithms to solve it, which have running times $T_1(n)$, $T_2(n)$, and $T_3(n)$. Suppose $T_1(n) = \Theta(n \lg(n))$, $T_2(n) = \Theta(n^2)$, and $T_3(n) = \Theta(2^n)$. You run algorithm 1 on your smart phone, algorithm 2 on your personal computer, and algorithm 3 on the fastest supercomputer in the world[1]. Suppose when $n = 30$, each of the algorithms finishes in one minute.

- (a) About how long would it take each algorithm to run when $n = 31$? How long when $n = 35$? What about $n = 45$?
- (b) If you gave each algorithm an hour to run, then what's the largest n you could get each to solve? What if you gave each algorithm a day? A year?
- (c) In about 2×10^{15} minutes [five billion years], the sun will have expanded to the point where it encompasses the earth, at which point something atomically strange will probably happen to what used to be our planet. What's the largest n these algorithms could solve before this happens?

(2) Pick at least one algorithm that we mentioned in class (e.g., from homework 1), and find its running time. (Don't pick algorithms we already explicitly found the running times for.)

(3) Think of your own sorting algorithm (or ask Pat to describe one for you). Write it down and analyze its running time.

(4) Come up with an algorithm that tests whether or not a number, n, is prime. Assume that each arithmetical operation only takes one step (i.e., multiplying, adding, dividing, et cetera). Analyze its running time.

(5) You have a machine that lets you multiply two numbers, but every time you use it, you need to pay \$1. For example, you can compute a number x^2 by paying just one dollar (give the machine x and x), and you can compute x^3 for 2 dollars (give the machine x and x. Then give the machine x^2 and x.).

- (a) Try to compute x^{100} as cheaply as possible.
- (b) Generalize this.

(6) Suppose G is a graph with n vertices.

- (a) Come up with an algorithm that checks whether or not G has any triangles (three vertices all of which are adjacent). Try to write this down and analyze its running time [as a function of n].
- (b) Try to generalize this result.

(7) Suppose G is a graph with n vertices and m edges.

[1]Currently China's TaihuLight, which can do an astounding 10^{17} operations every second. For comparison, the newest smart phones can do about 10^9 operations a second, and my old laptop can do about 10^{10}.

(a) Suppose I hand you a coloring of the vertices of G. Come up with an algorithm that determines whether or not this coloring is a *proper* coloring. Try to write this down and analyze its running time [as a function of m and/or n].

(b) Try to come up with an algorithm that tries to find a proper coloring of G using only 2 colors, and analyze its running time.

(c) Try to generalize this.

(8) Recall the Fibonacci numbers are given by $F_n = \begin{cases} 1, & \text{if } n = 1 \text{ or } n = 2 \\ F_{n-1} + F_{n-2}, & \text{if } n > 2. \end{cases}$, and consider the following algorithm to compute the n^{th} Fibonacci number.

```
fib1(n):
begin
  if $n < 3$ then return 1
  else return fib1($n-1$) + fib1($n-2$)
end
```

(a) Does this algorithm actually work?

(b) If $T(n)$ is the running time for this algorithm, argue that $T(n) = T(n-1) + T(n-2) + C$ for some constant $C > 0$.

(c) Show that this algorithm takes an exponential amount of time to run. [Hint: compare $T(n)$ to F_n and use a result from homework 2, problem 7]

(d) If you wanted to find F_{30}, what would you do? Try to make your strategy into an algorithm that determines F_n, and analyze its running time. (Challenge: find an algorithm that runs in polynomial time)