CMPS 2200: Homework 2

Monday, Sep 28

Complete the problems below and turn in your written answers on Canvas or in person. I encourage you to verbally discuss approaches to solving individual problems, but your written submission must be your work, and only your work.

1. Suppose that for a given task you are choosing between the following three algorithms:

   - Algorithm $\mathcal{A}$ solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
   - Algorithm $\mathcal{B}$ solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.
   - Algorithm $\mathcal{C}$ solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

   What are the asymptotic running times of each of these algorithms? Which algorithm would you choose?

2. For each recurrence below, use either the brick method, substitution, or induction to derive an asymptotic upper bound.

   (a) $T(n) = 2T(n/3) + 1$

   (b) $T(n) = 5T(n/4) + n$

   (c) $T(n) = 7T(n/7) + n$

   (d) $T(n) = 9T(n/3) + n^2$

   (e) $T(n) = 8T(n/2) + n^3$

   (f) $T(n) = 49T(n/25) + n^{3/2}\log n$

   (g) $T(n) = T(n-1) + 2$

   (h) $T(n) = T(n-1) + n^c$, with $c \geq 1$

   (i) $T(n) = T(\sqrt{n}) + 1$

3. Now that you have some practice solving recurrences, let's work on implementing some algorithms. In lecture we discussed a divide and conquer algorithm for integer multiplication. This algorithm takes as input two n-bit strings $x = \langle x_L, x_R \rangle$ and $y = \langle y_L, y_R \rangle$ and computes the product xy by using the fact that $xy = 2^{n/2} x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$. Write the algorithm specification in SPARC. Then, use the stub functions in `main.py` to implement two algorithms for integer multiplication: a divide and conquer algorithm that runs in quadratic time, and the Karatsaba-Ofman algorithm running in subquadratic time. Then test the empirical running times across a variety of inputs to test whether your code scales in the manner described by the asymptotic runtime?