Assignment 17: Fast Matrix Multiplication And More...

January 24, 2019
CS: DS&A
Proof School

Due: We'll see...

Problem 1.
a) Describe how to multiply two $n \times n$ matrices together in a "divide and conquer" algorithm. You may assume n is a power of 2.

Suggestion: Suppose you knew how to multiply pairs of 2×2 matrices. How would you go about multiplying two 4×4 matrices? Imagine breaking up a 4×4 into four 2×2's, as shown below. Carry out the process of multiplying the 4×4 matrices. Which 2×2's do you wind up multiplying together?

$$\left[\begin{array}{cc|cc} a & b & c & d \\ e & f & g & h \\ \hline i & j & k & l \\ m & n & o & p \end{array}\right] \cdot \left[\begin{array}{cc|cc} q & r & s & t \\ u & v & w & x \\ \hline y & z & a' & b' \\ c' & d' & e' & f' \end{array}\right]$$

b) What is the recurrence relation for the running time $T(n)$ of your matrix multiplication algorithm? Now do a recursion tree analysis, and compute the actual running time $T(n)$. How does that compare with the normal algorithm for multiplying matrices together?

c) Wouldn't it be nice if we could pull a Karatsuba-like move and *reduce the number of multiplications by one*? What would the running time $T(n)$ be if you could? (Again, do a tree analysis.) Spend some time trying to figure out how to get one less recursive multiplication. If you can't figure it out, that's fine–I'm happy to tell you. But try it yourself first!

Problem 2.
We've seen a bunch of $T(n)$ recurrence relations, and the corresponding runtimes (and in the previous problem, you've found two more):

$$\begin{aligned} T(n) = 2T(n/2) + \Theta(n): &\quad \Theta(n \log n) \\ T(n) = 2T(n/2) + \Theta(1): &\quad \Theta(n) \\ T(n) = T(n/2) + \Theta(1): &\quad \Theta(\log n) \\ T(n) = 4T(n/2) + \Theta(n): &\quad \Theta(n^2) \\ T(n) = 3T(n/2) + \Theta(n): &\quad \Theta(n^{\log_2 3}) \end{aligned}$$

It's time to try to figure out what's going on. How can we generalize this table? Try to make a conjecture, and prove it.