1. Assume your machine has 64 bit words. Assume you can multiply two n word numbers in time $2n^2$ with a standard algorithm. Assume you can multiply two n word numbers in time $12n^{\lg 3}$ with a "fancy" algorithm.

   (a) Approximately, how large does n have to be for the fancy algorithm to be better?

   (b) How many bits is that?

   (c) How many decimal digits is that?

2. Use the same assumptions as for problem (1), except assume you can multiply two n word numbers in time only $4n^{\lg 3}$ with a "fancy" algorithm.

   (a) Approximately, how large does n have to be for the fancy algorithm to be better?

   (b) How many bits is that?

   (c) How many decimal digits is that?

3. We can multiply large integers recursively by splitting each integer into thirds (rather than halves as done in class). In order to multiply the two three-digit numbers abc and def the standard algorithm would do nine atomic multiplications.

   (a) Explain how you can do fewer atomic multiplications by forming the product $(a + b + c)(d + e + f)$. How many atomic multipications do you use?

   (b) Write a recurrence for how fast this version of integer multiplication is. In order to keep this simple, just use $a\alpha n$, where a is a constant, for the time to do all of the additions at each invocation of the routine.

   (c) Solve the recurrence using the tree method. Assume n is a power of three. Show your work.

   (d) Solve the recurrence using the "master theorem" (posted). Assume n is a power of three. Show your work.

   (e) How does the asymptotic running time of this algorithm compare to the previous version with time $\Theta(n^{\lg 3})$?