In lecture, we described an algorithm of Karatsuba that multiplies two \( n \)-digit integers using \( O(n^{\log_2 3}) \) single-digit additions, subtractions, and multiplications. In this lab we’ll look at some extensions and applications of this algorithm.

1. Describe an algorithm to compute the product of an \( n \)-digit number and an \( m \)-digit number, where \( m < n \), in \( O(m^{\log_2 3} - 1) \) time. *Hint:* Break up the bigger number into chunks with \( m \) bits each.

2. Describe an algorithm to compute the decimal representation of \( 2^n \) in \( O(n^{\log_2 3}) \) time. (The standard algorithm that computes one digit at a time requires \( \Theta(n^2) \) time.)

3. Describe a divide-and-conquer algorithm to compute the decimal representation of an arbitrary \( n \)-bit binary number in \( O(n^{\log_2 3}) \) time. [*Hint:* \( x = a \cdot 2^{n/2} + b \). *Watch out for an extra log factor in the running time.*]

**Other Divide and Conquer Problems:**

4. Given an arbitrary array \( A[1..n] \), describe an algorithm to determine in \( O(n) \) time whether \( A \) contains more than \( n/4 \) copies of any value. *Do not use hashing, or radix sort, or any other method that depends on the precise input values.*

**Think about later:**

5. Suppose we can multiply two \( n \)-digit numbers in \( O(M(n)) \) time. Describe an algorithm to compute the decimal representation of an arbitrary \( n \)-bit binary number in \( O(M(n)\log n) \) time.