

1. How many calls (as a function of  $q$ ) to the function  $f$  will the following program use? We are mainly interested in the case where  $q$  is small but positive.
  - Step 1. Set  $x = 0.0, y = 1.0$ .
  - Step 2. Set  $z = (x + y)/2, r = f(z)$ .
  - Step 3. If  $r > 0$  set  $y = z$ ; otherwise set  $x = z$ .
  - Step 4. If  $y - x > q$  then go to Step 2; otherwise return.
  
2. Suppose you run Insertion Sort (Algorithm 1.12) on an array of  $n$  distinct integers. Suppose the array results in Insertion Sort running as fast as it is able to run on any such array.
  - 2a. For each of the 9 steps of the algorithm give a formula for the best case number of times the step is executed.
  
  - 2b. What property must the input array have to obtain these best case running times?
  
3. Suppose you have developed an integer multiply algorithm similar to Algorithm 5.2, except that you divide each input number into three equal size parts (instead of two parts). Suppose your algorithm has  $k$  multiplies of numbers with one third of the digits of the inputs and some additions. (Consider only the case where the two inputs have the same number of digits.)
  - 3a. How small must  $k$  be for your algorithm to be faster (for large inputs) than the classical algorithm (Algorithm 1.8)
  
  - 3b. How small must  $k$  be for your algorithm to be faster (for large inputs) than the Karatsuba algorithm (Algorithm 5.2)
  
4. You have three algorithms that solve the same problem. You should figure out how fast each algorithm is and then order them from fastest to slowest (for large problems). All algorithms use constant time for small problems (the time depends on the problem size). Ignore any overhead time used to divide a problem into subproblems or to put the answers for subproblems together to form an answer to the overall problem.
  - 4a. For a problem of size  $n$  Algorithm A generates three subproblems of size  $n - 1$  and four subproblems of size  $n - 2$ . How does the time for algorithm A depend on  $n$ ?
  
  - 4b. For a problem of size  $n$  Algorithm B generates one subproblem of size  $n - 1$  and six subproblems of size  $n - 2$ . How does the time for algorithm B depend on  $n$ ?
  
  - 4c. For a problem of size  $n$  Algorithm C generates eight subproblems of size  $n - 2$ . How does the time for algorithm C depend on  $n$ ?
  
  - 4d. Which algorithm is fastest, second fastest, and slowest (for large  $n$ )?
  
5. Suppose  $x^6 = x + t$ . Find an approximation to that solution to this equation with the property that  $x$  is large and real when  $t$  is large and real. In other words, we want  $x$  as a function of  $t$ . Write your approximation as the leading terms of a power series in some (possibly fractional) power of  $t$ . Carry your calculations out to at least three terms plus a big O term.