

CS 310 Homework 1

Fall 2009

Due date: (in class) 5:15 pm on Wed., Sep. 23, on paper. I strongly encourage you to submit a printed solution. Handwritten solution will only be accepted if clearly legible!

Goal of this assignment – review of runtime analysis, recursion and Java.

1. You should learn to recognize and sum a *geometric series*. Try these (see your calculus book or do a Web search on geometric series if you can't remember the formulas)
 - a. Sum from $i=1$ to $i=10$ of 2^i .
 - b. Sum from $i=1$ to infinity of $2/3^i$.

2. How many binary digits are there in the numbers 2^{100} , 5^{100} and 10^{100} ? How are the answers to these three questions related? (Hint: this is a question about logarithms.)

3. A. Show that

$$\log_a(x) = c * \log_b(x)$$

for some constant c (depending only on the constants a and b).

C. Calculate the ratio between the number of digits required to write a number in base 10 and the number of digits required to write the same number in base 2.

4. A. Weiss, problem 5.14. Except in the obvious cases, give reasons for your ranking.

B. Rank the following three functions: $\log N$, $\log(N^2)$, $\log^2 N$. Explain.

You should find all the mathematics you need in Weiss, Chapter 5. You may find it useful to remember that one way to compare the relative growth rates of $f(n)$ and $g(n)$ is to look at the ratio $f(n)/g(n)$ as $n \rightarrow$ infinity. If that ratio approaches 0, then g grows faster than f : $f(n) = O(g(n))$. If it approaches infinity then f grows faster than g . If the ratio approaches a constant different from both 0 and infinity then f and g grow at the same rate.

5. A. Find a big-O estimate for the running time (in terms of n) of the following function (with explanation)

```
int mysterySum( int n )
{
    int i, j, s=0;
    for(i=0; i < n; i++) {
        for(j=0; j < i; j++) {
            s += i*i;
        }
    }
}
```

B. Is this version of mysterySum faster? Is the big-O analysis different?

```
int mysterySum1( int n )
{
    int i, j, s=0;
    for(i=0; i < n; i++) {
        int i2 = i*i;
        for(j=0; j < i; j++) {
            s += i2;
        }
    }
}
```

C. Replace the inner loop in mysterySum by an $O(1)$ expression and compute the running time of the new program.

D. Find a single $O(1)$ expression giving the same result.

Hint: Evaluate the function by hand (or compile and run the code) for a few values of n and try to see the pattern.

6. **Bonus question** – propose a logarithmic run-time program for the power2 function discussed in class.
7. The following program computes the Fibonacci series, where each number in the series is the sum of the previous two numbers: 0 1 1 2 3 5 8 13 ... The following recursive program calculates the n th term in the Fibonacci series (assume n is non-negative and the first term is the zero-th):

```
int fib(int n)
{
    if(n == 0) return 0;
    if(n == 1) return 1;
    return fib(n-2)+fib(n-1);
}
```

A. What is the approximate complexity of this function? You don't have to derive the exact solution, just give an estimate (like – constant, linear, quadratic, exponential etc.) and explain.

B. Write a more efficient function (recursive or otherwise) that calculates the n th term in the Fibonacci series.

Review of Java. You don't need to try make these programs work online, but it would be a good idea to compose and run some test Java programs to make sure you have the right programming environment set up. Use “java -version” to make sure it's “build 1.6.x” for some x , or at least 1.5.x, to match the book and our default Java environment on the UNIX systems.

8. Weiss, Problem 3.10. What are the answers after we make names private in the Person object?
9. Weiss, Problem 3.13.
10. Weiss, Problem 4.6
11. Weiss, Problems 4.7, 4.9