Instructions

1. **Goal of this assignment – review of runtime analysis, recursion and Java.** I strongly encourage you to submit a printed solution. Handwritten solution will only be accepted if clearly legible!

2. Review of Java includes q. 7-9. You don’t need to try make the program in question 7 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 our default Java environment on the UNIX systems.

Questions

1. You should learn to recognize and sum a geometric series. Try these:

   **Note:** You don’t necessarily have to know exactly how to solve these equations. It’s enough if you look it up in your calculus text or online. The important thing for me is that you know how to look these things up and understand how they generalize.

   (a) \[ \sum_{i=1}^{10} 2^i. \]

   (b) \[ \sum_{i=1}^{\infty} \left(\frac{2}{3}\right)^i. \]

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 and change of base.)

3. (a) Show that \(\log_a(x) = c \ast \log_b(x)\) for some constant \(c\) (expressed only in terms of the constants \(a\) and \(b\)). **Hint:** This is probably easier than you think... It follows quite directly from the log properties. You should, though, prove that it’s true for any \(a\) and \(b\) and not prove by an example.

   (b) 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. Notice that this question relates to question 2 and 3a above.

4. (a) Order the following functions by growth rates:

   \[ N, \sqrt{N}, N^{1.5}, N^2, N \log N, N \log \log N, N \log^2 N, N \log(N^2), 2/N, 2^N, 2^{N/2}, 37, N^3, N^2 \log N. \]

   Indicate which functions grow at the same rate.

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

   You should find all the mathematics you need in the class notes and in Kleinberg and Tardos, chapter 2. 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 \(n \rightarrow \infty\). 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 \(\infty\) 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):

```c
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?

```c
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. The following program computes $2^n$:

```c
int power2(int n)
{
    if (n==0) return 1;
    return power2(n-1)+power2(n-1);
}
```

(a) Find a recurrence formula as we learned in class. Find the runtime. What is the big problem with this function? (hint: We discussed something similar in class).

(b) Introduce a small modification that makes the function run in linear time. Show why the runtime is linear.

(c) (bonus) The following function also calculates $2^n$:

```c
int power2New(int n)
{
    if (n==0) return 1;
    if (n % 2 == 0) {
        int result = power2New(n/2);
        return result*result;
    } else
        return 2*power2New(n-1);
}
```

Explanation: If n is even, then $2^n = (2^{n/2})^2$, so we can calculate $2^{n/2}$ recursively and square, cutting half of n in one move. Otherwise, we resort to the previous method. Show that the runtime of power2New is logarithmic in n. **Hint:** It's easy to show it when n is even, but sometimes n is odd... The trick is to show that the entire function is logarithmic nonetheless.
7. A combination lock has the following basic properties: the combination (a sequence of three numbers) is hidden; the lock can be opened by providing the combination; and the combination can be changed, but only by someone who knows the current combination. Design a class with public methods `open` and `changeCombo` and private data fields that store the combination. The combination should be set in the constructor. Do not compile and run your code, just provide a paper copy.

8. What is the difference between a final class and other classes? Why are final classes used?

9. What is an interface? How does the interface differ from an abstract class? What members may be in an interface?

10. (a) Suppose a `List<String>` `list1` has elements “A”, “B”, “C”, and “D”. What is returned by:
    1. `list1.iterator().next();`
    2. `list1.listIterator().next();`
    3. `list1.listIterator(2).next();`
    4. `list1.listIterator(4).previous();`

(b) Say what is deleted (or what happens) if next/previous is replaced by remove in all of the above operations. Explain.

(c) If we had the following sequence of commands:
    `list1.listIterator(2).next(); list1.listIterator(2).remove(); list1.listIterator(4).previous();`
    what would be returned? What would the list look like following these operations?