CSE 373 Data Structures 12wi, Homework 2

Due at the BEGINNING of class, Friday, 1/20/2012

Here are some questions on complexity and algorithm analysis. You only need to turn in written solutions, although you will need to run some code for one of the problems.

Problems

1. Prove (using Induction) that:

$$\sum_{i=1}^{N} i^3 = (\sum_{i=1}^{N} i)^2$$

Hints: Start with N=1 as the base case, then show how $\sum_{i=1}^{N+1} i^3$ ends up being equal to $(\sum_{i=1}^{N+1} i)^2$.

More hints: You already know what the sum of $\sum_{i=1}^{N} i$ is, and you should use the induction hypothesis $\sum_{i=1}^{N} i^3 = (\sum_{i=1}^{N} i)^2$ to come up with your answer. Referring to the induction examples on pages 6 and 7 and the examples from the slides may be helpful.

2. Order the following functions from slowest growth rate to fastest growth rate.
   N^2, N logN, 2/N, $\log^2 N$, 2^N, $\sqrt{N}$, 56, $N^2 \log N$, $N^{1.5}$, N^3,
   $2^{N/2}$, log N, N log (N^2), N log log N, N $\log^2$ N, N.
   If any of the functions grow at the same rate, be sure to indicate this.

3. Suppose $T_1(N) = O(f(N))$ and $T_2(N) = O(f(N))$. Which of the following are true?

   a. $T_1(N) + T_2(N) = O(f(N))$
   b. $T_1(N) - T_2(N) = o(f(N))$
   c. $T_1(N) / T_2(N) = O(1)$
   d. $T_1(N) = O(T_2(N))$

You do not need to prove an item is true (just saying true is enough for full credit), but you must give a counter example in order to demonstrate an item is false if you want full credit. To give a counter example, give values for $T_1(N)$, $T_2(N)$ and $f(N)$ for which the statement is false (e.g. "The statement is false if $T_1(N)=100N$, $T_2(N)=2N^2$ and $f(N)=N^3$"). Hints: Think about the definitions of big O and little o.

4. For each of the following **seven** program fragments:

    a. Give an analysis of the running time
    b. Implement the code in Java, and give the actual running time for several (at least 4) values of N.
    c. Compare your analysis with the actual running times.

For part (b), please turn in a printout of your Java code, (no electronic submission required). Hints: you will want to use assorted (at least 4) large values of *n* to get meaningful experimental results. You may find the library function `System.nanoTime()` to be useful in timing code fragments. A link to some Java code showing an example of timing can be found here:

`http://www.cs.washington.edu/education/courses/cse373/12wi/homework/hw02/Timing.java`

1)
```
sum = 0;
for (i = 0; i < n; i++) {
    sum++;
}
```

2)
```
sum = 0;
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
        sum++;
    }
}
```

3)
```
sum = 0;
for (i = 0; i < n; i++) {
    for (j = 0; j < n * n; j++) {
        sum++;
    }
}
```

4)
```
sum = 0;
for (i = 0; i < n; i++) {
    for (j = 0; j < i; j++) {
        sum++;
    }
}
```

5)
```
sum = 0;
for (i = 0; i < n; i++) {
    for (j = 0; j < i; j++) {
        sum++;
    }
    for (k = 0; k < 8000; k++) {
        sum++;
    }
}
```

6)
```
sum = 0;
for (i = 0; i < n; i++) {
    for (j = 0; j < i * i; j++) {
            sum++;
    }
}
```

7)
```
sum = 0;
for (i = 1; i < n; i++) {
    for (j = 1; j < i*i; j++) {
        if (j % i == 0) {
            for (k = 0; k < j; k++) {
                sum++;
            }
        }
    }
}
```

Note that there are THREE parts to this question, do all 3. a) calculate big-O, b) run the code *for several values of N* (4 or more) and time it, c) talk about what you see. For part c, be sure to say something about what you saw in your run-times, are they what you expected based on your big-O calculations? If not, any ideas why not? Graphing the values you got from part b might be useful for your discussion. Remember that when giving the big-O running time for a piece of code we always want the tightest bound we can get.

5. Show that the function $130n + 1140 + 5n^3$ is $O(n^3)$. (You will need to use the definition of $O(f(n))$ to do this. In other words, find values for c and n_0 such that the definition of big-O holds true as we did with the examples in lecture.