COMS 3134 Homework 1

Submission instructions

All programs must compile and run on CUNIX to receive credit. Submit your electronic files via `http://courseworks.columbia.edu`. We prefer electronic submission of theory though you will not be penalized for paper submissions (Do **not** print out your programs). I recommend using LaTeX for typesetting math). Include a README file that explains exactly what each file in your submission is. Place all the files you want to submit into a submission directory with the following naming scheme.

`<your_uni>_hw<number>`

So if my UNI is `uni1234` am submitting homework 6, my directory would be `uni1234_hw6`. Archive your submission directory using

```
tar -czvf uni1234_hw1.tar.gz uni1234_homework1
```

and upload `uni1234_hw1.tar.gz` to courseworks.

Multiple Submissions: You can submit multiple times, but we will only consider the latest submission based on the timestamp in courseworks. Please give at least 1-2 minutes between two submissions.

I recommend that you keep a pristine copy of your submission folder in case there is any submission error.

Theory Problems

1. **6 points.** (Weiss 2.1) Order the following functions by growth rate: $N, \sqrt{N}, N^{1.5}, N^2, N \log N, N \log \log N, N \log^2 N, N \log(N^2), 2/N, 2^N, 2^{N/2}, 37, N^2 \log N, N^3$. Indicate which functions grow at the same rate.

2. **8 points.** (Weiss 2.2) 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) $\frac{T_1(N)}{T_2(N)} = O(1)$
   (d) $T_1(N) = O(T_2(N))$

3. **8 points.** (Weiss 2.11) An algorithm takes 0.5 ms for input size 100. How long will it take for input size 500 if the running time is the following (assume low-order terms are negligible):

(a) linear

(b) $O(N \log N)$

(c) quadratic

(d) cubic

4. **8 points.** (Weiss 2.25) Programs A and B are analyzed and found to have worst-case running times no greater than $150N \log_2 N$ and N^2, respectively. Answer the following questions, if possible [and justify your answers]:

   (a) Which program has the better guarantee on the running time for large values of N (i.e., $N > 10,000$)?

   (b) Which program has the better guarantee on the running time for small values of N (i.e., $N < 100$)?

   (c) Which program will run faster on average for $N = 1,000$?

   (d) Is it possible that program B will run faster than porgram A on all possible inputs?

Programming

1. **15 points.** (Weiss 1.13) Design a generic class, `Collection`, that stores a collection of `Objects` (in an array), along with the current size of the collection. Provide public methods `isEmpty`, `makeEmpty`, `insert`, `remove`, and `isPresent`. `isPresent(x)` returns `true` if and only if an `Object` that is equal to `x` (as defined by `equals`) is present in the collection.

   CHANGE: use the main function provided on the homepage for the assignment. This will standardize the tests that we run on your code and save you some time writing this test function.

   Hint: most of the methods for this class will be easy to implement, except possibly the ttremove implementation, which is a bit tricky. You might want to read ahead to Array Lists for some ideas on how to do this cleanly.

2. **15 points.** (Weiss 2.7) For each of the following six program fragments:

   (a) Give an analysis of the running time (Big-Oh will do).

   (b) Implement the code in Java, and give the running time for several values of N.

   (c) Compare your analysis with the actual running times.

```
(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*i; j++)
            for (k=0; k<j; k++)
                sum++;
```

```
(6) 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++
```

Hint: to measure **true** running times instead of just the incremented sum variable, use a simple timer object such as the following (feel free to use this one exactly).

```
public class TimeInterval {
    private long startTime, endTime;
    private long elapsedTime; // Time Interval in milliseconds
// Commands
    public void startTiming() {
        elapsedTime = 0;
        startTime = System.currentTimeMillis();
    }

    public void endTiming() {
        endTime = System.currentTimeMillis();
        elapsedTime = endTime - startTime;
    }

    //Queries
    public double getElapsedTime() {
        return (double) elapsedTime / 1000.0;
    }
}
```

Some of these code fragments will be extremely fast for even large values of N and some will be extremely slow for even small values of N, so find a reasonable range of N to try with each fragment. Report your running times in your writeup, and include charts and figures.