Computer Science 501
Data Structures & Algorithms
The College of Saint Rose
Fall 2015

Lab 4: Analysis

Due: 11:59 PM, Sunday, October 4, 2015

This lab approaches analysis of algorithms from both the theoretical and empirical perspectives. The theoretical questions in the lecture assignment and problem set portions of the lab, will give you a chance to practice with the analysis tools we have considered in class. In the empirical study, you will use the timing code you developed last week to generate tabular and graphical format data, and see if you can match your results to the theoretical expectations.

You may work individually or in a group of 2 or 3 on this lab, but it is essential that each individual understands how to do the lecture assignment and problem set questions, as exam questions will be very similar. As you know, exams are not group efforts.

Getting Set Up

Create a document where you will record your answers to the lecture assignment and lab questions. If you use plain text, call it "`lab4.txt`". If it's a Word document, you can call it whatever you'd like, but when you submit, be sure you convert it to a PDF document "`lab4.pdf`" before you submit it.

Lecture Assignment Questions

We will usually discuss these questions at the start of class on the lab due date, so no credit can be earned for late submissions of lecture assignment questions.

?LA Question 1:
Bailey Problem 5.6, p. 112-3 (3 points)

?LA Question 2:
Bailey Problem 5.14, p. 113 (3 points)

?LA Question 3:
For each of the following, indicate a natural size metric and the basic operation: (3 points total)

1. computing the product of n numbers
2. computing $n!$
3. finding the largest element in an array

Problem Set Problems

Question 1:
Compare the asymptotic growth rates of the following pairs of functions and decide if one grows faster than the other or if they grow at the same rate. Prove your answers using limits. All logarithms are base 2, unless otherwise noted. (4 points)

1. $5n^2 + 2n + 5$ and n^2
2. $n \log n$ and n^2
3. $2^{\log n}$ and n
4. 2^n and 2^{2n}

Question 2:
Prove the following using limits (4 points):

1. $n \in \Omega(\log n)$
2. $5n + 6 \in \Theta(n)$
3. $n \log n \in \Omega(n)$
4. $n \in \Theta(n - n^{\frac{1}{2}})$

Question 3:
For each of the following functions, indicate the class $O(g(n))$ to which the function belongs. Use the "smallest" $g(n)$ possible to obtain the tightest bound. You need not prove your assertions in this case, but justify your choice informally. (2 points)

1. $\frac{1}{n} + 12$
2. $\frac{\sin n}{n}$

Question 4:
For each of the following functions, indicate the class $O(g(n))$ to which the function belongs. Use the "smallest" $g(n)$ possible to obtain the tightest bound, unless otherwise specified. Prove your assertions using the definition of $O(g(n))$ (*i.e.*, produce constants). (6 points)

1. $762n^2 - 56n + 37$
2. $(n^3 + 100)^5$
3. $n \log n$, use $g(n) = n^2$

? Question 5:
For each of the following functions, indicate the class $\Omega(g(n))$ to which the function belongs. Use the "largest" $g(n)$ possible to obtain the tightest bound unless otherwise specified. Prove your assertions using the definition of $\Omega(g(n))$ (*i.e.*, produce constants). (6 points)

1. $762n^2 - 56n + 37$
2. $2^n + n^2$
3. $n \log n$, use $g(n) = n$

? Question 6:
For each of the following functions, indicate the class $\Theta(g(n))$ to which the function belongs. Prove your assertions using the definition of $\Theta(g(n))$ (*i.e.*, produce constants). (5 points, 1 for the first, 4 for the second)

1. $762n^2 - 56n + 37$ (hint: you've already done this, just bring it all together)
2. $n^4 + 50n^2 - 23$

? Question 7:
Analyze the following code segment and give an exact formula involving summations for the number of times the word "hello" is printed. Then simplify your formula to obtain an exact formula for the number of times the word "hello" is printed that does not involve summations. (In the code segment, `Math.pow(2,j)` is a function that returns 2 raised to the j^{th} power.) (5 points)

```
for ( i = 1; i <= n; i++ )
    for ( j = 1; j <= n; j++ ) {
        for ( k = 1; k <= Math.pow(2,j); k++ ) {
             System.out.println("hello");
        }
        System.out.println("hello");
    }
```

Note: you may test your formula by implementing this code and then running it for different values of n.

Empirical Analysis Study

Last week, you developed one or more Java classes capable of measuring the time taken for a variety of experiments. Please refer back to that lab description for the list of experiments.

Your first task is to run your program(s) repeatedly (likely thousands of times) to generate a set of timing data.

- For each experiment, you will need to run instances for a range of problem sizes. Powers of 2 are often a good choice. Be sure to include large problem sizes where running times do not make this prohibitively expensive.
- Choose the number of times to run the for loop within a given instance of an experiment so that the minimum time between starting and stopping the timer is at least, say, 10 milliseconds, but that the maximum time is not more than a few seconds.
- Run each problem size several times (at least 10) and use the minimum time you observe in each case for the results you present.
- To minimize the chance that compiler optimizations, especially Java's "just in time" runtime optimizer, do not taint your results, run the Java Virtual Machine with flags that disable this. For Java/Oracle JVMs, adding the `-Xint` to your Java command line will disable the JIT compiler.

Your second task is to gather your results into both tabular and graphical formats. You might use familiar tools such as a spreadsheet program for this, but I encourage you to look into *gnuplot*, which is a free data plotting program available for most platforms. It is installed on mogul, and an example of how to use it is available there in `/home/shared/gnuplotexample`.

Your final task is to present your results for each experiment. Before addressing the individual experiments, write an introductory paragraph describing the overall approach, including as many details about your test environment as you can (include Java version and any options used when running, operating system version, CPU configuration (number of cores, chips), architecture, and speed, and memory size at a minimum, but also things like cache sizes if you can find them).

Then for each experiment, include the following:

- A restatement the theoretical expectation for the efficiency (*e.g.*, $O(n^2)$) that you gave for last week.
- A presentation of your timing results in tabular and graph formats.
- A comparison of your timing results with the theoretical expectation, explaining any discrepencies to the best of your ability.

Submitting

Before 11:59 PM, Sunday, October 4, 2015, submit your lab for grading. There are two things you need to do to complete the submission: (i) Copy your file with the answers to the lecture assignment and lab questions into your project directory. Be sure to use the correct file name. If you prepared your answers in Word, export to a PDF file and submit that. (ii) Email a copy of your lab (a `.7z` or `.zip` file containing your project directory) to *terescoj@strose.edu*. Please use a meaningful subject line such as "Joe Student Lab4 Submission".

Grading

This assignment is worth 85 points, which are distributed as follows:

Feature	Value	Score
LA Question 1 (5.6)	3	
LA Question 2 (5.14)	3	
LA Question 3 (size metrics)	3	
Question 1: compare growth rates	4	
Question 2: limit proofs	4	
Question 3: Big O informal	2	
Question 4: prove Big O class	6	
Question 5: prove Big Omega class	6	
Question 6: prove Big Theta class	5	
Question 7: analyze code segment	5	
Empirical analysis study introduction	4	
Empirical analysis study choice of problem sizes, reps	4	
Empirical analysis study timing data	12	
Empirical analysis study tables and graphs	12	
Empirical analysis study comparison with theory	12	
Total	85	