CIS 121—Data Structures and Algorithms with Java - Fall 2015

Big-Oh Lab — TBD

Lab Problem Solutions

Problem 1

Order the following functions such that if f precedes g, then $f(n)$ is $O(g(n))$.

$$\sqrt{n},\ n,\ n^{1.5},\ n^2,\ n\lg n,\ n\lg\lg n,\ n\lg n^2,\ 2^{n/2},\ 2^n,\ \lg(n!),\ n^2\lg n,\ n^3,\ 2^{2^n}$$

Solution.

$$\sqrt{n},\ n,\ n\lg\lg n,\ n\lg n,\ \lg(n!),\ n\lg n^2,\ n^{1.5},\ n^2,\ n^2\lg n,\ n^3,\ 2^{n/2},\ 2^n,\ 2^{2^n}$$

Problem 2

Provide a runtime analysis of the following loop. That is, find both Big-Oh and Big$-\Omega$:

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

Solution.

Observe that for fixed values of i, j, the innermost loop runs $\max\{1, j-i+1\} \leq n$ times. For instance, when $i = j = 0$, the innermost loop evaluates once. When $i = 0, j = n$, the innermost loop evaluates $n+1$ times. The middle loop runs a total number of $O((n-i)^2)$ times. Therefore, the entire block of code runs in $O(n^3)$.

To find a lower bound on the running time, we consider smaller subsets of values for i, j and lower-bound the running time for the algorithm on these subsets. (A lower bound there would also be a lower bound for the original code's runtime!) Consider the values of i such that $0 \leq i \leq n/4$ and values of j such that $3n/4 \leq j \leq n$. For each of the $n^2/16$ possible combinations of these values of i and j, the innermost loop runs at least $n/2$ times. Therefore, the running time is at least

$$(n^2/16)(n/2) = \Omega(n^3)$$

Alternate Solution for Big-Oh.

One could solve this using exact sums, but we will leverage some Big-Oh notation. We know that the innermost loop runs in at most $(j-i+1)$ time for fixed i, j (see other solution). Therefore the body of the middle loop runs at most $c(j-i+1)$ times. Therefore, we can express the runtime of the code shown as

$$\sum_{i=0}^{n}\sum_{j=i}^{n} c(j-i+1) = O(n^3)$$

.

Problem 3

In this problem, you are **not** allowed to use the theorems about Big-Oh stated in the lecture notes. Your proof should follow exclusively from the definition of Big-Oh.

Prove or disprove the following statement:

$$f(n) + g(n) \text{ is } \Theta(\max\{f(n), g(n)\}), \text{ where } f, g : R \to R^+.$$

Solution.

We show both Big-Oh and Big-Ω separately.

Let $c = 2, n_0 = 1$. $f(n)+g(n) \leq 2\max\{f(n), g(n)\},\ \forall n \geq 1$. Therefore, $f(n)+g(n) = O(\max\{f(n), g(n)\})$.

Let $c = 1, n_0 = 0$. $f(n)+g(n) \geq \max\{f(n), g(n)\},\ \forall n \geq 0$. Therefore, $f(n)+g(n) = \Omega(\max\{f(n), g(n)\})$.

Therefore, as we have shown Big-Oh and Big-Ω, we have shown Big-Θ.

Problem 4

Prove or disprove the following statement:

$$2^n \text{ is } O(n!).$$

Solution.

We want to show that $2^n \leq c(n!)\ \forall n \geq n_0$, for a positive real-valued c and integer n_0.

Proof.

$$\begin{aligned}
2^n &\leq c(n!) \\
\lg(2^n) &\leq \lg c + \lg(n!) \\
n &\leq \lg c + \lg(n \cdot (n-1) \cdot (n-2) \ldots 2) \\
n &\leq \lg c + \sum_{i=0}^{n} \lg i \\
n \leq \sum_{i=n/2}^{n} \lg i &\leq \lg c + \sum_{i=0}^{n} \lg i \\
n \leq n/2 \lg(n/2) &\leq \sum_{i=n/2}^{n} \lg i \\
2 &\leq \lg(n/2)
\end{aligned}$$

It is clear that choosing any $n_0 \geq 8$ and $c > 0$ will suffice! To check our work, let $n_0 = 8$ and $c = 1$.

$$2^8 \leq 8! \rightarrow 256 < 40320$$

A particularly obvious result, but it doesn't hurt to check.

□

Problem 5

Provide a runtime analysis of the following loop:

```
for (int i = 2; i < n; i = i*i)
    for (int j = 1; j < Math.sqrt(i); j = j+j)
        System.out.println("*");
```

Problem 6

Prove or disprove the following statement:

$$\lg(n!) \text{ is } \Theta(n \lg n).$$