Homework 2 Solutions

CSC 140 – Advanced Algorithm Design and Analysis

If you find any errors in the solution writeup, please let me know. Ask in class or come to office hours if you need any further help understanding the problems and their solutions.

1) Your plot should look undeniably like the right side of a parabola.

2) The Orders of Power Functions tells us $5n < 5n^2$ and $3n^0 < 3n^2$ for all $n > 1$. This means $10n^2 + 5n + 3 < 18n^2$ for all $n > 1$. By definition, this means $10n^2 + 5n + 3$ is in $O(n^2)$.

Note that what I did here was find values $c = 18$ and $n_0 = 1$ that made $10n^2 + 5n + 3 < cn^2$ for all $n > n_0$. Since this is the form used in the definition of big-O, it allowed me to make my conclusion.

3) For all $n > 0$, $9n^2 + 5n + 3 > 0$ (because each term is either positive or the product of positive values). Adding n^2 to both sides gives us $10n^2 + 5n + 3 > n^2$ for all $n > 0$. By definition, this means $10n^2 + 5n + 3$ is in $\Omega(n^2)$.

Note that what I did here was find values $c = 1$ and $n_0 = 0$ that made $10n^2 + 5n + 3 > cn^2$ for all $n > n_0$. Since this is the form used in the definition of big-Ω, it allowed me to make my conclusion.

4) One definition of O requires we find positive c and n_0 such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$. When $c = 2$ and $n_0 = 1$, it is true that $2^{n+1} \leq c \cdot 2^n$ for all $n \geq n_0$ (in fact, $2^{n+1} = c \cdot 2^n$, so it's true for all n). Therefore $2^{n+1} = O(2^n)$. The ratio $2^{n+1}/2^n$, as n goes to infinity, is 2, so $2^{n+1} = \Theta(2^n)$.

On the other hand, $2^{2n} \not\leq c \cdot 2^n$ for all $n \geq n_0$ for any positive c and n_0 because (if we factor out a 2^n from both sides) $2^n \not\leq c$ for all $n \geq n_0$ for any n_0. Also, as n goes to infinity, $2^{2n}/2^n = 2^n = \infty$, which means $2^{2n} = \omega(2^n)$.

5) The limit column gives $\lim_{n\to\infty} \frac{f(n)}{g(n)}$.

f	g	limit	o	O	ω	Ω	Θ	Notes
$\log n^2$	$\log n + 5$	2	no	yes	no	yes	yes	
$\sqrt{n}$	$\log n^2$	∞	no	no	yes	yes	no	poly dominates log
$(\log n)^2$	$\log n$	∞	no	no	yes	yes	no	$f/g = \log n$
n	$(\log n)^2$	∞	no	no	yes	yes	no	poly dominates log
$n \log n + n$	$\log n$	∞	no	no	yes	yes	no	$f/g = n + n/\log n$
$\log n^2$	$(\log n)^2$	0	yes	yes	no	no	no	$f/g = 2/\log n$
10	$\log 10$	c	no	yes	no	yes	yes	f/g is a constant
2^n	$10n^2$	∞	no	no	yes	yes	no	exp dominates poly
2^n	$n \log n$	∞	no	no	yes	yes	no	exp dominates poly
2^n	3^n	0	yes	yes	no	no	no	$f/g = (2/3)^n$
2^n	n^n	0	yes	yes	no	no	no	$f/g = (2/n)^n$

6) I'll do two. If you have questions about the others, please ask. (b) The statement `j<n` is executed the most. Each time the inner loop is activated, this comparison occurs $n+1$ times (it's true n times and false once). The inner loop is activated 3 times, so `j<n` is executed $3n + 3$ times. This is $\Theta(n)$.

Note that I asked you to find the statement that is executed the most. In this problem, there is only one: `j<n`. You get the same end result, however, if you choose to analyze any *statement that contributes to the dominant term of the algorithm's work polynomial. In this case that means `j++` or `sum++`, which are each executed exactly $3n$ times, would also yeild $\Theta(n)$.*

(e) To simplify, let's assume that n is a power of 2. The `j<=n` gets executed the most. Each time the inner loop is activated, j goes through the sequence $1, 2, 4, 8, \ldots, n, 2n$, and it's when the value is $2n$ that the condition fails. This means, for example, that when $n = 8$ and the inner loop is activated, the `j<=n` comparison occurs 5 times (1, 2, 4, 8, 16). This is $\log_2 8 + 2$, or more generally $\log_2 n + 2$. Since the inner loop is activated n times, the `j<=n` comparison occurs exactly $n \log_2 n + 2n$ times, which is $\Theta(n \log n)$. When n is not a power of 2, it is still the least power-of-two greater than n that terminates the inner loop. In this case the exact number of times `j<=n` is executed is $n\lfloor \log_2 n \rfloor + 2n$