CS:3330, Algorithms

HW05 Order Problems

Logan Zweifel September 19, 2021

1 Theta

Show that $6n^2 + 12n \in \Theta(n^2)$.

The following is the definition for Theta. For a given complexity function $f(n)$,

$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$$

This means that $\Theta(f(n))$ is the set of complexity functions $g(n)$ for which there exists some positive real constants c_1 and c_2 and some nonnegative integer N such that for all $n \geq N$,

$$c_1 * f(n) \leq g(n) \leq c_2 * f(n)$$

I show that $6n^2 + 12n \in \Theta(n^2)$ by using the equations from my work on questions 3 and 6 from HW04, a picture my work for those problems from the previous homework is attached at the end of this question, as they ask to find the Big-O and Ω of this same function, and because, for $n \geq 1$,

$$1 * n^2 \leq 6n^2 + 12n \leq 18n^2$$

Where $c_1 = 1$, $c_2 = 18$ and $N = 1$ were used to obtain the result.

I show that $6n^2 + 12n \in O(n^2)$. Because, for $n \geq 1$,

$$\begin{aligned} 6n^2 + 12n &\leq 6n^2 + 12n^2 \\ &\leq 18n^2 \end{aligned}$$

Where $c = 18$ and $N = 1$ were used to obtain the result.

I show that $6n^2 + 12n \in \Omega(n^2)$. Because, for $n \geq 0$,

$$6n^2 + 12n \geq 1 * n^2$$

Where $c = 1$ and $N = 0$ were used to obtain the result.

Figure 1: Previous Homework work

2 Little-o

Show directly, using the definition of "Little-o", that $4n \in o(n^2)$.

The following is the definition for "Little-o". For a given complexity function $f(n)$, $o(f(n))$ is the set of complexity functions $g(n)$ satisfying the following: For every positive real constant c, there exists a nonnegative integer N such that for all $n \geq N$,

$$g(n) \leq c * f(n)$$

I show that $4n \in o(n^2)$ by first assuming that $c > 0$ which is given by the definition of "Little-o". Now, I need to find the valid N values.

$$\begin{aligned} 4n &\leq c * n^2 \\ \frac{4}{c} &\leq n \end{aligned}$$

Therefore, you can choose any $N \geq \frac{4}{c}$. For this problem, the valid values for N are dependent on the value of the constant c.

3 Little-w

Show directly, using the definition of "Little-ω", that $4n^3 \in \omega(n)$.

The following is the definition for "Little-ω". For a given complexity function $f(n)$, $\omega(f(n))$ is the set of complexity functions $g(n)$ satisfying the following: For every positive real constant c, there exists a nonnegative integer N such that for all $n \geq N$,

$$g(n) \geq c * f(n)$$

I show that $4n^3 \in \omega(n)$ by first assuming that $c > 0$ which is given by the definition of "Little-ω". Now, I need to find a valid N value.

$$\begin{aligned} 4n^3 &\geq c * n \\ 4n^2 &\geq c \\ 2n &\geq \sqrt{c} \\ n &\geq \frac{\sqrt{c}}{2} \end{aligned}$$

Therefore, you can choose any $N \geq \frac{\sqrt{c}}{2}$. For this problem, the valid values for N are dependent on the value of the constant c.

4 Big-O

Show directly, using the definition of Big-O, that for all $a, b, > 1$, $log_a(n) \in O(log_b(n))$.

The following is the definition for Big-O. For a given complexity function $f(n)$, $O(f(n))$ is the set of complexity functions $g(n)$ for which there exists some positive real constant c and some nonnegative integer N such that for all $n \geq N$,

$$g(n) \leq c * f(n)$$

In addition, Both this question and the next one will use the logarithm change of base formula which is the following:

$$log_b(x) = \frac{log_c(x)}{log_c(b)}$$

The following was done using the definition of Big-O and the logarithm change of base formula and for $n \geq 1$:

$$\begin{aligned}
log_a(n) &\leq c * log_b(n) \\
log_a(n) &\leq c * \frac{log_a(n)}{log_a(b)} \\
\frac{log_a(n)}{log_a(n)} &\leq c * \frac{1}{log_a(b)} \\
1 &\leq \frac{c}{log_a(b)} \\
log_a(b) &\leq c
\end{aligned}$$

Essentially, this shows that for N=1, the valid constant values that can satisfy the definition of Big-O is directly related to the integers a and b.

5 Omega

Show directly, using the definition of Ω, that for all $a, b, > 1$, $log_a(n) \in \Omega(log_b(n))$.

The following is the definition for Omega. For a given complexity function $f(n)$, $\Omega(f(n))$ is the set of complexity functions $g(n)$ for which there exists some positive real constant c and some nonnegative integer N such that for all $n \geq N$,

$$g(n) \geq c * f(n)$$

The following was done using the definition of Omega and the logarithm change of base formula defined in the previous problem. For $n \geq 1$:

$$\begin{aligned}
log_a(n) &\geq c * log_b(n) \\
log_a(n) &\geq c * \frac{log_a(n)}{log_a(b)} \\
\frac{log_a(n)}{log_a(n)} &\geq c * \frac{1}{log_a(b)} \\
1 &\geq \frac{c}{log_a(b)} \\
log_a(b) &\geq c
\end{aligned}$$

Just as the previous problem did, the valid values for the constant c to satisfy the definition of Omega are dependent on the values of a and b.

6 Theta

Show that for all $a, b, > 1$, $log_a(n) \in \Theta(log_b(n))$.

The following is the definition for Theta. For a given complexity function $f(n)$,

$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$$

This means that $\Theta(f(n))$ is the set of complexity functions $g(n)$ for which there exists some positive real constants c_1 and c_2 and some nonnegative integer N such that for all $n \geq N$,

$$c_1 * f(n) \leq g(n) \leq c_2 * f(n)$$

I show that for all $a, b, > 1$, $log_a(n) \in \Theta(log_b(n))$ by using the work done in problems 1 and 2 to satisfy the conditions of Theta. For $N = 1$ and $c_1 \leq log_a(b) \leq c_2$,

$$c_1 * log_b(n) \leq log_a(n) \leq c_2 * log_b(n)$$

The constants in this solution can not be explicitly defined as they are dependent on the values of a and b. However, given the relationship stated earlier between both constant and $log_a(b)$ there is a defined range of valid constant values for any given N, therefore showing that $log_a(n) \in \Theta(log_b(n))$.

7 Theta using limits

Use limits to show $13n^2 + 7n + \sqrt{2} \in \Theta(n^2)$

This problem uses Theorem 1.3 from the book which is property 13 on the propertiesoforder.pdf. That property/theorem is the following:

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \begin{cases} c & \text{implies } g(n) \in \Theta(f(n)), \text{ if } c > 0 \\ 0 & \text{implies } g(n) \in o(f(n)) \\ \infty & \text{implies } f(n) \in o(g(n)) \end{cases}$$

Applying this limit rule to this specific problem results in the following:

$$\begin{aligned} \lim_{n\to\infty} \frac{g(n)}{f(n)} &= \lim_{n\to\infty} \frac{13n^2 + 7n + \sqrt{2}}{n^2} \\ &= \lim_{n\to\infty} \frac{13n^2}{n^2} \\ &= 13 * \lim_{n\to\infty} \frac{n^2}{n^2} \\ &= 13 \end{aligned}$$

In the second line of these equations, I am allowed to drop the lower degree terms, and only have $13n^2$ on top because as a function moves towards infinity, a higher degree term will dominate the lower terms and then I can pull 13 from within the limit as it is a constant. The result of this limit is a constant greater than 0, 13, and therefore I have shown that $13n^2 + 7n + \sqrt{2} \in \Theta(n^2)$.

8 $t(n)$

Consider the function

$$t(n) = \begin{cases} 2n & \text{if } n \text{ is even} \\ 1 & \text{if } n \text{ is odd} \end{cases}$$

a) Show that $t(n) \in O(n) - \Omega(n)$

A function that is $t(n) \in O(n) - \Omega(n)$ is a function that is $O(f(n)$, but is not $\Omega(f(n))$. First, I will show that $t(n)$ is $O(n)$. Referring to the definition of Big-O stated in problem 1, for $n \geq 0$ and all subsequent even n values exclusively (first case of $t(n)$),

$$2n \leq 3 * n \equiv 2n \leq 3n$$

Where $c = 3$ and $N = 0$ were used to obtain the first case of $t(n)$. For the second case where values of n are odd, the same values for c and N will be used to prove both cases work for these numbers.

$$1 \leq 3n$$

As I have proved both cases of $t(n)$ are in $O(n)$, $t(n)$ as a complete function must be $O(n)$ as well.

Referring to the definion of Omega stated in problem 2, $t(n)$ can't be $\Omega(n)$ as for any value of c no matter how small, $c * n$ will eventually be greater than 1 and not satisfy the definition of Omega when n is an odd number.

I have shown that $t(n)$ is $O(n)$ and is not $\Omega(n)$ and have therefore proved that $t(n) \in O(n) - \Omega(n)$.

b) Show that $t(n) \notin o(n)$

I will show that $t(n) \notin o(n)$ by using a proof by contradiction. Let c=1. If $t(n) \in o(n)$, then there must exist some N such that, for $n \geq N$,

$$t(n) \leq 1 * n \equiv t(n) \leq n$$

This equality is not true for all even n values greater than 0. This contradiction proves that $t(n) \notin o(n)$.