UWI

Design and Analysis of Algorithms

Tutorial 2 - Sample Solutions

Asymptotic Notations

Understanding asymptotic notations and their properties is essential when performing time efficiency analysis.

1. Explain the formal definitions of the following notations.
   (a) O (Big Oh)
   A function $t(n)$ is said to be in $O(g(n))$, denoted as $t(n) \in O(g(n))$, if $t(n)$ is bounded above by some constant multiple of $g(n)$ for all large n, i.e., if there exist some positive constant c and some non-negative integer n_0 such that:
   $$t(n) \leq cg(n) \text{ for all } n \geq n_0 \tag{1}$$
   (b) Ω (Big Omega)
   A function $t(n)$ is said to be in $\Omega(g(n))$, denoted as $t(n) \in \Omega(g(n))$, if $t(n)$ is bounded below by some positive constant multiple of $g(n)$ for all large n, i.e., if there exist some positive constant c and some non-negative integer n_0 such that:
   $$t(n) \geq cg(n) \text{ for all } n \geq n_0 \tag{2}$$
   (c) Θ (Big Theta)
   A function $t(n)$ is said to be in $\Theta(g(n))$, denoted as $t(n) \in \Theta(g(n))$, if $t(n)$ is bounded both above and below by some positive constant multiples of $g(n)$ for all large n, i.e., if there exist some positive constant c_1 and c_2 and some non-negative integer n_0 such that:
   $$c_2 g(n) \leq t(n) \leq c_1 g(n) \text{ for all } n \geq n_0 \tag{3}$$

2. Explain what is meant by the following asymptotic notation properties.
   (a) $f(n) \in O(f(n))$
   $f(n)$ grows no faster than itself.
   (b) $f(n) \in O(g(n)) \iff g(n) \in \Omega(f(n))$
   $f(n)$ grows no faster than $g(n)$ if and only if $g(n)$ grows at least as fast as $f(n)$.
   (c) If $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$, then $f(n) \in O(h(n))$
   If $f(n)$ grows no faster than $g(n)$ and $g(n)$ grows no faster than $h(n)$, then $f(n)$ grows no faster than $h(n)$.
   (d) If $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, then $t_1(n) + t_2(n) \in O(max\{g_1(n), g_2(n)\})$
   If $t_1(n)$ grows no faster than $g_1(n)$ and $t_2(n)$ grows no faster than $g_2(n)$, then $t_1(n) + t_2(n)$ will grow no faster than the larger between $g_1(n)$ and $g_2(n)$.

Comparing Orders of Growth

We can compare orders of growth by using limits to derive whether two functions grow at the same, lower or higher rates.

1. State the implications if the below values result from the following limit:

$$\lim_{n\to\infty} \frac{t(n)}{g(n)} \tag{4}$$

   (a) 0 (Zero)
   Implies that $t(n)$ has a smaller order of growth than $g(n)$ i.e. $t(n) \in O(g(n))$

   (b) c (Some Positive Constant)
   Implies that $t(n)$ has the same order of growth as $g(n)$ i.e. $t(n) \in \Theta(g(n))$

   (c) ∞ (Infinity)
   Implies that $t(n)$ has a larger order of growth than $g(n)$ i.e. $t(n) \in \Omega(g(n))$

2. State L'Hopital's rule and discuss when it should be used.

$$\lim_{n\to\infty} \frac{t(n)}{g(n)} = \lim_{n\to\infty} \frac{t'(n)}{g'(n)} \tag{5}$$

L'Hopital's rule is a technique used to evaluate limits an indeterminate forms like in equations (6) or (7). If this occurs, we differentiate the numerator and denominator then take the limit.

$$\lim_{n\to\infty} \frac{t(n)}{g(n)} = \frac{0}{0} \tag{6}$$

$$\lim_{n\to\infty} \frac{t(n)}{g(n)} = \frac{\pm\infty}{\pm\infty} \tag{7}$$

3. Compare the orders of growth of $\frac{n^2(n-4)}{4}$ and n^3 using limits.

$$\begin{aligned}
\lim_{n\to\infty} \frac{t(n)}{g(n)} &= \lim_{n\to\infty} \frac{\frac{n^2(n-4)}{4}}{n^3} && (8)\\
&= \frac{1}{4} \lim_{n\to\infty} \frac{n^2(n-4)}{n^3} && (9)\\
&= \frac{1}{4} \lim_{n\to\infty} \frac{n^3 - 4n^2}{n^3} && (10)\\
&= \frac{1}{4} \lim_{n\to\infty} (1 - \frac{4n^2}{n^3}) && (11)\\
&= \frac{1}{4} \lim_{n\to\infty} (1 - \frac{4}{n}) && (12)\\
&= \frac{1}{4}(1) && (13)\\
& && (14)
\end{aligned}$$

Since the limit is equal to a positive number, the functions have the same orders of growth i.e. $t(n) \in \Theta(g(n))$.

Time Efficiency Calculations

Calculating the time efficiency of algorithms can involve the use of summation rules and formulas. It is important we learn to calculate the time efficiencies for simple brute force algorithms.

1. Analyse the following algorithms and calculate their time efficiencies using summation rules.
   (a) Algorithm 1: Sequential Search

```
for $i \leftarrow 0$ to $n$–1 do
    if $A[i] = K$ then
        return i
    end if
end for
return –1
```

$$C_w(n) = \sum_{i=0}^{n-1} 1 \tag{15}$$
$$= [(n-1) - (0) + 1] \tag{16}$$
$$= n \tag{17}$$

Since $C_w(n) = n$, we disregard constant (if any), which implies that $C_w(n) \in \Theta(n)$.

   (b) Algorithm 2: Matrix Multiplication

```
for $i \leftarrow 0$ to $n$–1 do
    for $j \leftarrow 0$ to $n$–1 do
        $C[i,j] \leftarrow 0.0$
        for $k \leftarrow 0$ to $n$–1 do
            $C[i,j] \leftarrow C[i,j] + A[i,k] \times B[k,j]$
        end for
    end for
end for
return C
```

$$C(n) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1} 1 = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}[(n-1) - (0) + 1] \tag{18}$$
$$= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} n = \sum_{i=0}^{n-1}[(n-1) - (0) + 1]n \tag{19}$$
$$= \sum_{i=0}^{n-1} n^2 = [(n-1) - (0) + 1]n^2 \tag{20}$$
$$= n^3 \tag{21}$$

Since $C(n) = n^3$, this implies that $C(n) \in \Theta(n^3)$.