COMP3711: Design and Analysis of Algorithms

Tutorial 1

Asymptotic notations

Asymptotic upper bound

Definition (big-Oh)

$f(n) = O(g(n))$: There exists constant $c > 0$ and n_0 such that $f(n) \leq c \cdot g(n)$ for $n \geq n_0$

Asymptotic lower bound

Definition (big-Omega)

$f(n) = \Omega(g(n))$: There exists constant $c > 0$ and n_0 such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

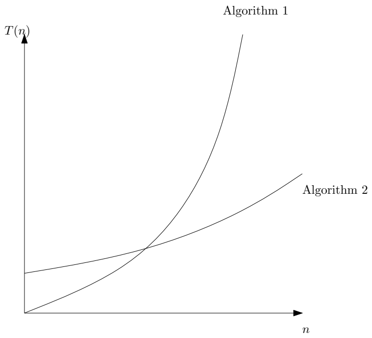
Asymptotic tight bound

Definition (big-Theta)

$f(n) = \Theta(g(n))$: $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

Comparing time complexity

Example:



Algorithm 2 is clearly superior

- $T(n)$ for Algorithm 1 is $O(n^3)$
- $T(n)$ for Algorithm 2 is $O(n^2)$
- Since n^3 grows much more rapidly, we expect Algorithm 1 to take much more time than Algorithm 2 when n increases

Some Basic mathematic background on exponentials

For all real $a \neq 0$, m and n, we have the following identities:

$$\begin{aligned} a^0 &= 1 \\ a^1 &= a \\ a^{-1} &= 1/a \\ (a^m)^n &= (a^n)^m = a^{mn} \\ a^m a^n &= a^{m+n} \\ a^{1/n} &= \sqrt[n]{a} \end{aligned}$$

Some Basic mathematatic background on logarithms

For all real $a > 0, b > 0, c > 0$, and n:

$$
\begin{aligned}
a &= b^{\log_b a} \\
\log_c(ab) &= \log_c a + \log_c b \\
\log_b a^n &= n \log_b a \\
\log_b a &= \frac{\log_c a}{\log_c b} \\
\log_b(1/a) &= -\log_b a \\
\log_b a &= \frac{1}{\log_a b} \\
a^{\log_b n} &= n^{\log_b a}
\end{aligned}
$$

Question 1

For each of the following statement, answer whether the statement is true or false.

(a) $1000n + n \log n = O(n \log n)$.

(b) $n^2 + n \log(n^3) = O(n \log(n^3))$.

(c) $n^3 = \Omega(n)$.

(d) $n^2 + n = \Omega(n^3)$.

(e) $n^3 = O(n^{10})$.

(f) $n^3 + 1000n^{2.9} = \Theta(n^3)$

(g) $n^3 - n^2 = \Theta(n)$

Solution 1

(a) True.
(b) False.
(c) True.
(d) False.
(e) True.
(f) True.
(g) False.

Question 2

For each pair of expressions (A, B) below, indicate whether A is O, Ω, or Θ of B. Note that zero, one, or more of these relations may hold for a given pair; list all correct ones. Justify your answers.

(a) $A = n^3 + n \log n; \quad B = n^3 + n^2 \log n.$

(b) $A = \log \sqrt{n}; \quad B = \sqrt{\log n}.$

(c) $A = n \log_3 n; \quad B = n \log_4 n.$

(d) $A = 2^n; \quad B = 2^{n/2}.$

(e) $A = \log(2^n); \quad B = \log(3^n).$

Solution 2

	A	Relation:	B
(a)	$n^3 + n \log n$	Ω, Θ, O	$n^3 + n^2 \log n$
(b)	$\log \sqrt{n}$	Ω	$\sqrt{\log n}$
(c)	$n \log_3 n$	Ω, Θ, O	$n \log_4 n$
(d)	2^n	Ω	$2^{n/2}$
(e)	$\log(2^n)$	Ω, Θ, O	$\log(3^n)$

Solution 2: Step by step

Notes:

(a) Both are $\Theta(n^3)$, the lower order terms can be ignored. Note that if $A(n) = \Theta(B(n))$, then automatically $A(n) = O(B(n))$ and $A(n) = \Omega(B(n))$.

(b) After simplifying, A is $(1/2)\log n$, and B is $\sqrt{\log n}$. Substituting $m = \log n$, we can see ratio A/B grows as $m/2\sqrt{m} = \sqrt{m}/2$ which tends to infinity as n (and hence m) tends to infinity, i.e., $A(n) = \Omega(B(n))$.

(c) Log base conversion only introduces a constant factor.

(d) The ratio is $2^n/2^{n/2} = (2)^{n/2}$ which goes to infinity in the limit.

(e) After simplifying these are $n\log 2$ and $n\log 3$, both of which are $\Theta(n)$.

Question 3

Suppose $T_1(n) = O(f(n))$ and $T_2(n) = O(f(n))$. Which of the following are true? Justify your answers.

(a) $T_1(n) + T_2(n) = O(f(n))$

(b) $\frac{T_1(n)}{T_2(n)} = O(1)$

(c) $T_1(n) = O(T_2(n))$

Solution 3

(a) True. Since $T_1(n) = O(f(n))$ and $T_2(n) = O(f(n))$, it follows from the definition that there exist constants $c_1, c_2 > 0$ and positive integers n_1, n_2 such that $T_1(n) \le c_1 f(n)$ for $n \ge n_1$ and $T_2(n) \le c_2 f(n)$ for $n \ge n_2$. This implies that, $T_1(n) + T_2(n) \le (c_1 + c_2) f(n)$ for $n \ge \max(n_1, n_2)$. Thus, $T_1(n) + T_2(n) = O(f(n))$.

(b) False. For a counterexample to the claim, let $T_1(n) = n^2, T_2(n) = n, f(n) = n^2$. Then $T_1(n) = O(f(n))$ and $T_2(n) = O(f(n))$ but $\frac{T_1(n)}{T_2(n)} = n \ne O(1)$.

(c) False. We can use the same counterexample as in part (b). Note that $T_1(n) \ne O(T_2(n))$

Question 4

Let $f(n)$ and $g(n)$ be non-negative functions. Using the basic definition of Θ-notation, prove that $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.

Solution 4

For any value of n, $\max(f(n), g(n))$ is either equal to $f(n)$ or equal to $g(n)$. Therefore, for all n,

$$\max(f(n), g(n)) \leq f(n) + g(n).$$

Using $c = 1$ and $n_0 = 1$ in the big-oh definition, it follows that

$$\max(f(n), g(n)) = O(f(n) + g(n)).$$

Also, for all n,

$$\max(f(n), g(n)) \geq f(n)$$

and

$$\max(f(n), g(n)) \geq g(n).$$

Adding we have

$$2 \times \max(f(n), g(n)) \geq f(n) + g(n).$$

Therefore,

$$\max(f(n), g(n)) \geq \frac{1}{2}(f(n) + g(n)).$$

Solution 4

Using $c = 1/2$ and $n_0 = 1$ in the Omega definition, it follows that

$$\max(f(n), g(n)) = \Omega(f(n) + g(n)).$$

Since $\max(f(n), g(n)) = O(f(n) + g(n))$ and $\max(f(n), g(n)) = \Omega(f(n) + g(n))$, it implies that $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.