COMP3711: Design and Analysis of Algorithms

Tutorial 1

Asymptotic notation

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 $n \geq n_0$.
Equivalent definition: $\lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty$

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 $n \geq n_0$.
Equivalent definition: $\lim_{n \to \infty} \frac{f(n)}{g(n)} > 0$

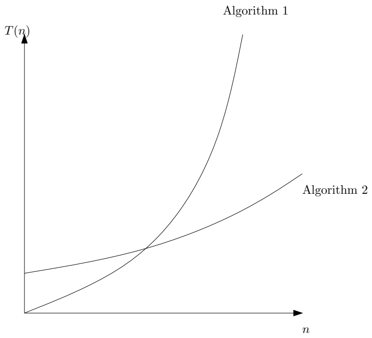
Asymptotic tight bound

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 more rapidly than n^2, we expect Algorithm 1 to take much more time than Algorithm 2 for large n.

Review: Basic facts on exponents

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

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

Review: Basic Facts on logarithms

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

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

$$
\begin{aligned}
\log_b a &= \frac{\log_c a}{\log_c b} \\
\log_b a &= \frac{\log_a a}{\log_a b} = \frac{1}{\log_a b}
\end{aligned}
$$

$$
\begin{aligned}
a^{\log_b n} &= (b^{\log_b a})^{\log_b n} = (b^{\log_b n})^{\log_b a} = n^{\log_b a} \\
4^{\log_2 n} &= n^{\log_2 4} = n^2
\end{aligned}
$$

Question 1

For each of the following statements, 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

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

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

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

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

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

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

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

(g) $n^3 - n^2 = \Theta(n)$ 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 $A(n) = \Theta(B(n))$,
$\Rightarrow A(n) = O(B(n))$ and $A(n) = \Omega(B(n))$.

(b) After simplifying, $A = \frac{1}{2}\log n$, and $B = \sqrt{\log n}$.
Set $m = \log n$. The ratio $\frac{A}{B} = \frac{m}{2\sqrt{m}} = \frac{\sqrt{m}}{2}$.
Then $\lim_{n\to\infty} \frac{A}{B} = \lim_{m\to\infty} \frac{A}{B} = \infty$,
i.e., $A(n) = \Omega(B(n))$.

(c) Log base conversion only introduces a constant factor, i.e.,
$n \log_3 n = n \cdot \log_3 4 \cdot \log_4 n = \Theta(n \log_4 n)$.

(d) $2^n/2^{n/2} = (2)^{n/2} \to \infty$ as $n \to \infty$.
Alternatively, notice that $2^{n/2} = \sqrt{2^n}$.

(e) After simplifying notice that $A = n\log 2$ and $B = n\log 3$, 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. From the definition of $T_1(n) = O(f(n))$ and $T_2(n) = O(f(n))$, there exist constants $c_1, c_2 > 0$ and positive integers n_1, n_2 such that

$$\forall n \geq n_1,\ T_1(n) \leq c_1 f(n) \quad \text{and} \quad \forall n \geq n_2,\ T_2(n) \leq c_2 f(n).$$

This implies that,

$$\forall n \geq \max(n_1, n_2),\ T_1(n) + T_2(n) \leq (c_1 + c_2) f(n).$$

Thus, $T_1(n) + T_2(n) = O(f(n))$.

(b) False. Counterexample: $T_1(n) = n^2,\ T_2(n) = n,\ f(n) = n^2$.

Then $T_1(n) = O(f(n)),\ T_2(n) = O(f(n))$ but

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

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

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,

$$\max(f(n), g(n)) \geq f(n) \quad \text{and} \quad \max(f(n), g(n)) \geq g(n).$$

Then

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

Then, Using $c = 1/2$ and $n_0 = 1$ in the definition of Ω,we get

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

Solution 4

We have now seen both

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

and

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

proving

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

Question 5

In the analysis of the max-subarray algorithms we (will) see nested loops of the form

for $i = 1$ *to* n
  for $j = i$ *to* n
    for $k = i$ *to* j
      do one unit of work

In class we give an intuitive explanation as to why this code performs $\Theta(n^3)$ units of work.

Prove this fact rigorously.

Solution 5

The code

for i = 1 to n
for j = i to n
for k = i to j
do one unit of work

does

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=i}^{j} 1 = \sum_{i=1}^{n}\sum_{j=i}^{n}(j-i+1) = \sum_{i=1}^{n}\sum_{k=1}^{n-i+1} k$$

units of work (the last equality is from changing indices). Now note that $\sum_{k=1}^{t} k = \Theta(t^2)$ so

$$\sum_{i=1}^{n}\sum_{k=1}^{n-i+1} k = \sum_{i=1}^{n}\Theta((n-i+1)^2) = \sum_{j=1}^{n}\Theta(j^2) = \Theta\left(\sum_{j=1}^{n} j^2\right) = \Theta\left(\Theta(n^3)\right) = \Theta(n^3)$$