MAT2440, Classwork33, Spring2025

ID:________________________ Name:________________________

1. **Big-Omega** and **Big-Theta** Notation.

$f(x) \le C g(x)$, $x > k$.

$C' g(x) \le f(x)$, $x > k'$

Big-O : $f(x)$ is $O(g(x))$ $\Longleftrightarrow$ $g(x)$ is an upper bound of $f(x)$.

Big-Ω : $f(x)$ is $\Omega(g(x))$ $\Longleftrightarrow$ $g(x)$ is a lower bound of $f(x)$.

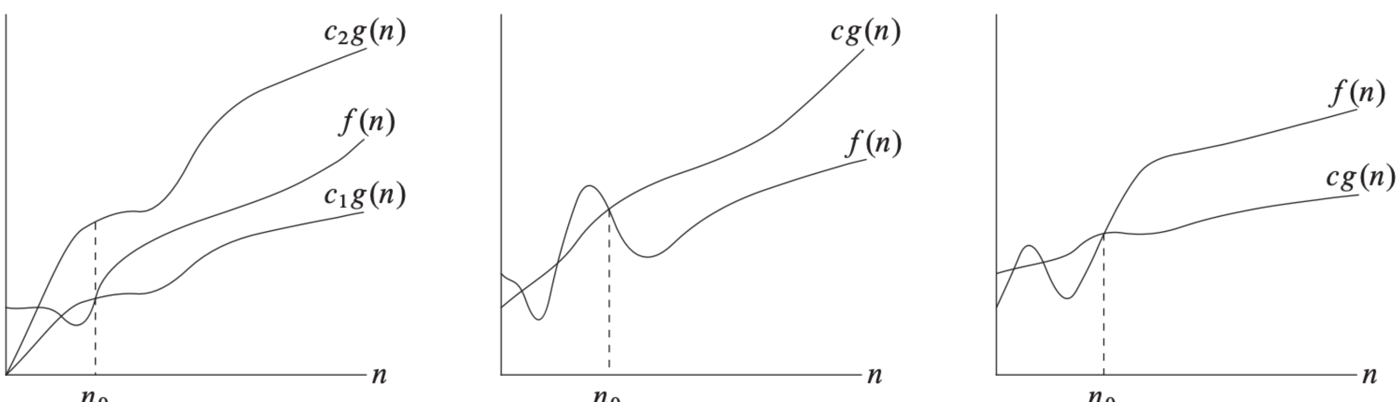
Big-Θ : $f(x)$ is $\Theta(g(x))$ $\Longleftrightarrow$ $g(x)$ is both an upper bound and a lower bound of $f(x)$.

Therefore, when $f(x)$ is $\Theta(g(x))$ (big-Theta of $g(x)$), that $f(x)$ is of order $g(x)$, and that $f(x)$ and $g(x)$ are of the same order.

Big-Θ notation provides the exact growth rate of a function

2. Mathematically, "f is $\Theta(g(n))$ " means

$$c_1 g(n) \le f(n) \le c_2 g(n) \text{ for } n > k.$$



$f(n) = \Theta(g(n))$ $\quad$ $f(n) = O(g(n))$ $\quad$ $f(n) = \Omega(g(n))$

3. Show that $f(x) = 3x^2 + 2x + 3$ is $\Theta(x^2)$ by directly finding c_1, c_2, and n.

$$3x^2 \le 3x^2 + 2x + 3 \le 3x^2 + 2\cdot x^2 + 3x^2 \le 8x^2$$

$$\Rightarrow 3x^2 \le 3x^2+2x+3 \le 8x^2 , \quad \text{for } x > 1$$

$\Rightarrow f(x)$ is $\Theta(x^2)$ with witnesses $C_1 = 3$

$C_2 = 8$

$n = 1$

(or k)

($f(x)$ is $O(x^2)$ and $f(x)$ is $\Omega(x^2)$)

§ 3.3. Complexity of Algorithms

1. Accuracy and Efficiency of an Algorithm and the Complexity.

A good algorithm needs to be accurate and efficient.

Efficiency can be analyzed by the computational complexity:

- Time complexity: the time required for the algorithm.
- Space complexity: the computer memory required for the algorithm.

2. Time Complexity

Time required for the algorithm

= the number of operators needed × the time needed for one operation.

The operations used to measure time complexity:

the Comparison, addition (subtraction), multiplication, division of integers, and etc.

3. The time complexity analysis of maximum finding algorithm:

> **procedure** $max(a_1, a_2, \cdots, a_n$: a list of n numbers)
> $n :=$ the length of $\{a_i\}$
> $tempMax := a_1$
> **for** $i := 2$ **to** n
>   **if** $tempMax < a_i$ **then** $tempMax := a_i$ ← 1 comparison
> **return** $tempMax$ { $tempMax$ is the largest element}

(1) The number of operations in the loop:

$$\left.\begin{array}{ll} i = 2, & 1 \\ i = 3, & 1 \\ i = 4, & 1 \\ \vdots & \\ i = \_\_, & 1 \end{array}\right\} \text{ n−1 comparison}$$

(2) $f(n) =$ n−1 $= O($n$)$ which has linear complexity.

(3) The time complexity for this algorithm is O(n).

(4) The complexity is measured by the growth rate of f(n) where $f(n)$ is the number of operations for the input size n.