

# I. Algorithm Analysis

- Algorithm analysis is to identify the time category (or order of growth) of an algorithm
- Eight most common categories
  - $1, \log n, n, n(\log n), n^2, n^3, 2^n, n!$

Most Popular

## Three Notations

$O$  (Big Oh)

$O$

$\Theta$  (Big Theta)

$\Theta$

$\Omega$  (Big Omega)

$\Omega$

$O$ : Best & Worst Case differ  
 $\Theta$ : Always same  $t(n)$   
 $\Omega$ : Lower Bound

## II. Big-Oh ( $O$ ) Notation

- Ex: Algorithm SequentialAdd( $A[0\dots n-1]$ )

sum  $\leftarrow 0$

$i \leftarrow 0$

while ( $i < n$ ) do

    sum  $\leftarrow$  sum +  $A[i]$

$i \leftarrow i + 1$

return sum

- Big O notation denotes UPPER BOUND

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

$O(n \log n)$

$n \log n$  or faster

## Rules To Simplify $T(n)$

1. Eliminate low order terms

$$T(n) = 4*n + 5 \rightarrow 4*n$$

2. Eliminate Constant Coefficients

$$T(n) = 4*n \rightarrow n \rightarrow O(n)$$

- Big - O notation with Basic Operation

• It's difficult to calculate the total number of all executions of all operations

∴ To simplify the calculation, we consider only the number of executions of the basic operation

- Big - O formal definition:

• An algorithm's running time  $t(n)$  is said to be in  $O(g(n))$ , denoted  $t(n) \in O(g(n))$ , if there exist constants  $c$  &  $n_0$  that satisfy the following condition

$$t(n) \leq c * g(n) \text{ for all } n \geq n_0$$

$g(n)$   
↓  
One of  
the  
categories  
in  
regression

### III. Big Omega Notation ( $\omega$ ) ( $\Omega$ )

- An algorithm's running time  $t(n)$  is said to be in  $\Omega(g(n))$ , denoted  $t(n) \in \Omega(g(n))$ . If there exist constants  $c$  &  $n_0$  that satisfy the following condition

$$t(n) \geq c * g(n) \text{ for all } n \geq n_0$$

- Denotes LOWER BOUND

$$\Omega(n * \log n)$$

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

$n \log n$  or slower

## IV. Big Theta Notation ( $\Theta$ )

- An algorithm's running time  $t(n)$  is said to be in  $\Theta(g(n))$ , denoted  $t(n) \in \Theta(g(n))$  if there exist constants  $c_1, c_2$ , &  $n_0$  that satisfy the following condition

$$c_1 * g(n) \leq t(n) \leq c_2 * g(n) \text{ for all } n \geq n_0$$

Scratch:

O

$\Omega$

$$t(n) = n^2 + 3n + 1 \in O(n)?$$

if  $t(n) = 2xn + 5$  can

$$t(n) = n^2 + 3n + 1 \leq C * n \text{ for all } n \geq n_0$$

we say  $t(n) \in \Omega(n)$ ? Yes

Impossible, because  $n^2$  will increase  
faster than  $n$

Can we say  $t(n) \in \Omega(n^2)$

$$t(n) = 2 \cdot n + 5 \geq C * n^2 \text{ for all } n \geq n_0$$

if  $t(n) = n$ , can we say  $t(n) \in O(n)$ ? Yes ↵

if  $t(n) = n$ , can we say  $t(n) \in O(n^2)$ ? Yes ↵

Meets upper bound  
(or faster)

$O =$  or faster

$\Omega =$  or slower

$$t(n) \geq C * (\Omega \cdot n \log n) \text{ for all } n \geq n_0$$

$$2n + 5 \geq C * (\Omega \cdot n \log n) \text{ for all } n \geq n_0$$

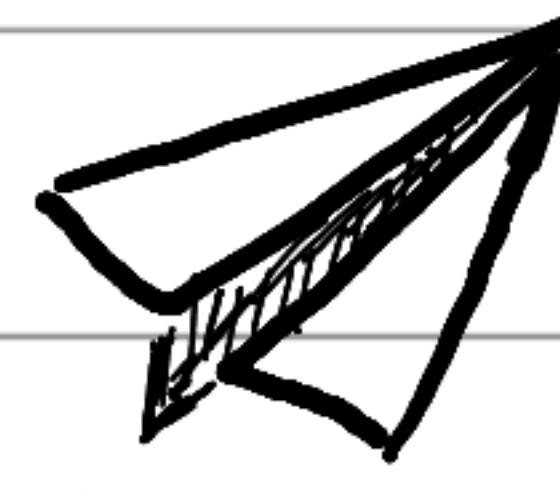
not possible ↵

Scratch:

$$c_1 \cdot n \leq 2^n + 3 \leq c_2 \cdot n$$

$$c_1: 1 \quad n \leq 2^n + 3 \leq 5n$$

$$c_2: 5 \quad \text{True}$$



5 minutes

00000

1 1 1

2 2 1

3 2 2

4 1 1

5 2 2