

# **CS 401: Computer Algorithm I**

## **Running Time Analysis**

Xiaorui Sun

# Time Complexity

**Problem:** An algorithm can have different running time on different inputs

**Solution:** The complexity of an algorithm associates a number  $T(N)$ , the “time” the algorithm takes on problem size  $N$ .

On **which** inputs of size  $N$ ?

Mathematically,

$T$  is a function that maps positive integers giving problem size to positive integers giving number of simple operations

# Time Complexity (N)

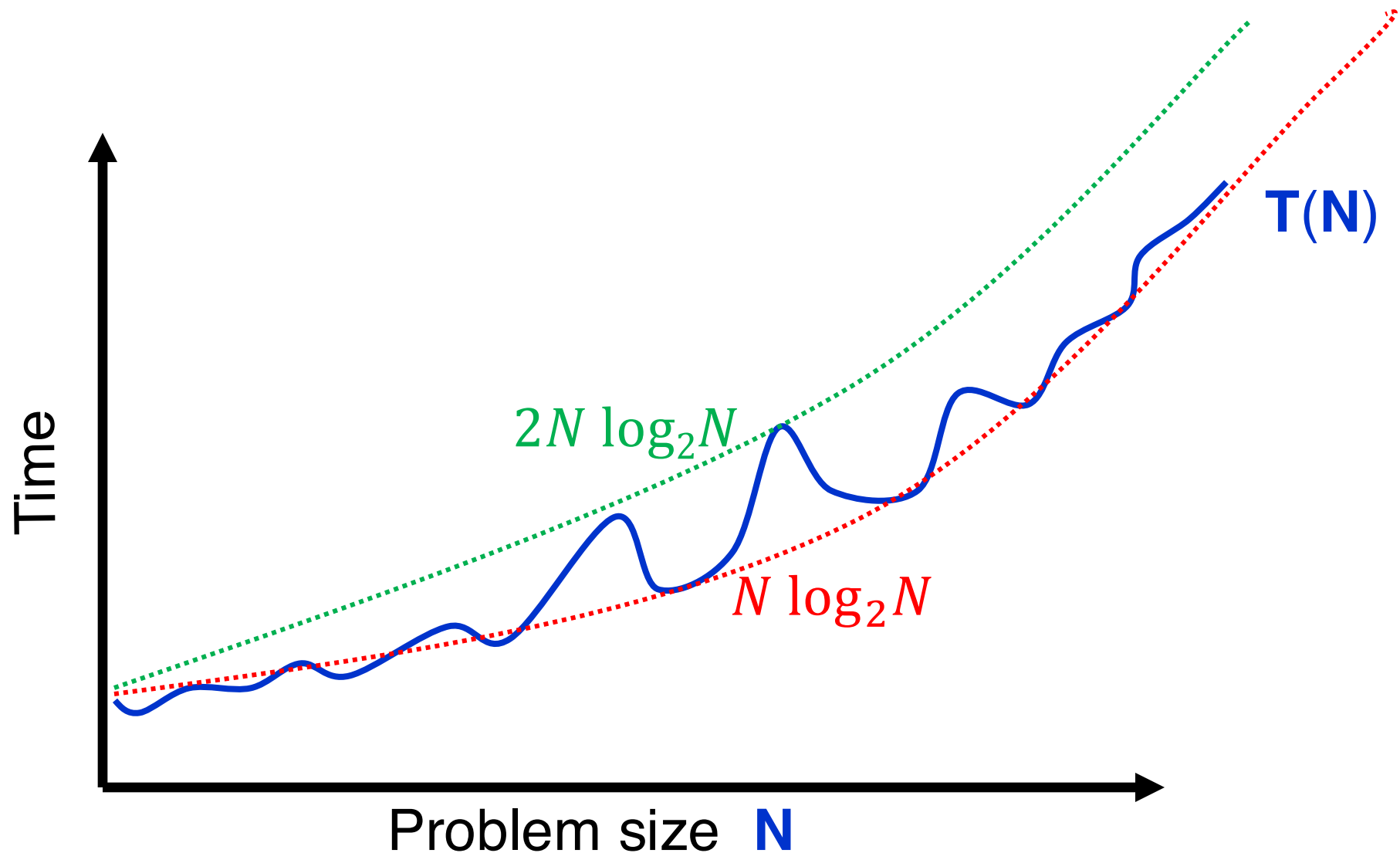
**Worst Case Complexity:** **max** # simple operations algorithm takes on any input of size **N**

This Course

**Average Case Complexity:** **avg** # simple operations algorithm takes on inputs of size **N**

**Best Case Complexity:** **min** # simple operations algorithm takes on any input of size **N**

# Time Complexity on Worst Case Inputs




# O-Notation

Given two positive functions **f** and **g**

- **f(N)** is **O(g(N))** iff there is a constant **c > 0** and **N<sub>0</sub> ≥ 0** s.t.,  
**0 ≤ f(N) ≤ c · g(N)** for all **N ≥ N<sub>0</sub>**

E.g. **f(N) = 32N<sup>2</sup> + 17N + 1**

- **f(N) = O(N<sup>2</sup>)**.  Choose **c = 50**, **N<sub>0</sub> = 1**
- **f(N)** is neither **O(N)** nor **O(N log N)**.

Typical usage: Gale-Sharpley makes **O(n<sup>2</sup>)** proposals.

# Quiz

Let  $f(N) = 32N^2 + 17N \log_2 N + 1000$ . Which of the following are true?

A.  $f(N)$  is  $O(N^2)$ .

B.  $f(N)$  is  $O(N^3)$ .

**Answer: C**

C. Both A and B.

Answer at:

[pollev.com/xiaoruisun673](https://pollev.com/xiaoruisun673)

D. Neither A nor B.

All the quizzes in this course are to encourage you to think and participate the course.

If you participate the quiz, then you get the participation credit.

# Properties

Reflexivity.  $f$  is  $O(f)$ .

Constants. If  $f$  is  $O(g)$  and  $c > 0$ , then  $c \cdot f$  is  $O(g)$ .

Products. If  $f_1$  is  $O(g_1)$  and  $f_2$  is  $O(g_2)$ , then  $f_1 \cdot f_2$  is  $O(g_1 \cdot g_2)$ .

Sums. If  $f_1$  is  $O(g_1)$  and  $f_2$  is  $O(g_2)$ , then  $f_1 + f_2$  is  $O(\max \{g_1, g_2\})$ .



Fastest growing term dominates

Transitivity. If  $f$  is  $O(g)$  and  $g$  is  $O(h)$ , then  $f$  is  $O(h)$ .

# Asymptotic Bounds for common fns

- **Polynomials:**

$a_0 + a_1n + \cdots + a_dn^d$  is  $O(n^d)$

- **Logarithms:**

$\log_a n = O(\log_b n)$  for all constants  $a, b > 0$

- **Logarithms:** log grows slower than every polynomial

For all  $k > 0$ ,  $\log n = O(n^k)$

- $n \log n = O(n^{1.01})$



# Asymptotic Bounds for common fns

**exponential function**

- **Exponential:** For all  $k, l > 0$ ,  $n^k = O(\exp(n^l))$
- Practice: when  $n^k = O(\exp((\log_2 n)^l))$ ?
  - $k = 1, l = 10$ ?
  - $k = 100, l = 10$ ?
  - $k = 0.01, l = 10$ ?
  - $k = 1, l = 0.1$ ?
  - $k = 100, l = 0.1$ ?
  - $k = 0.01, l = 0.1$ ?

**This is not an exponential function**

# $\Omega$ -Notation

Given two positive functions **f** and **g**

- **f(N)** is  $\Omega(g(N))$  iff there is a constant **c > 0** and **N<sub>0</sub> ≥ 0** s.t.,  
**f(N) ≥ c · g(N) ≥ 0** for all **N ≥ N<sub>0</sub>**

E.g. **f(N) = 32N<sup>2</sup> + 17N + 1**

- **f(N)** is both  $\Omega(N^2)$  and  $\Omega(N)$ .  Choose **c=32, N<sub>0</sub>=1**
- **f(N)** is not  $\Omega(N^3)$ .

**Fastest growing term dominates**

Typical usage: Gale-Sharpley makes  $\Omega(n^2)$  proposals in the worst case.

# Question

Which is an equivalent definition of big Omega notation?

- A.  $f(N)$  is  $\Omega(g(N))$  iff  $g(N)$  is  $O(f(N))$ .
- B.  $f(N)$  is  $\Omega(g(N))$  iff there is a constant  $c > 0$  s.t.  
 $f(N) \geq c \cdot g(N) \geq 0$  for infinitely many  $N$ .
- C. Both A and B.
- D. Neither A nor B.

**Answer: A**