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)

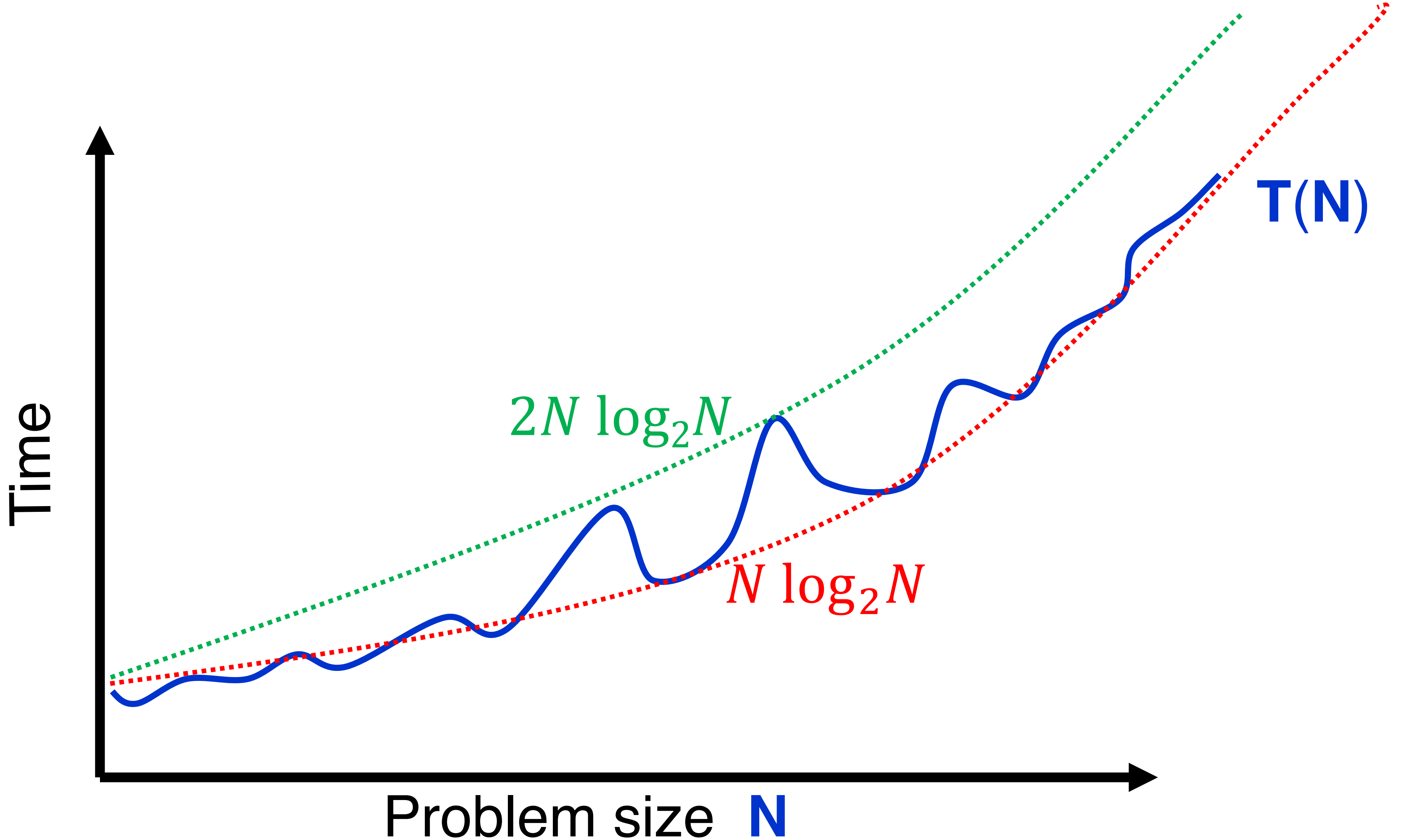
Worst Case Complexity: **max** # simple operations algorithm takes on any input of size **N**

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 **$N_0 \geq 0$** s.t., **$0 \leq f(N) \leq c \cdot g(N)$** for all **$N \geq N_0$**

E.g. **$f(N)=32N^2+17N+1$**

- **$f(N)=O(N^2)$.** ← Choose $c=50$, $N_0=1$
- **f(N)** is neither **O(N)** nor **O(N log N).**

Typical usage: Gale-Sharpley makes $O(n^2)$ proposals.

Question

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.

D. Neither A nor B.

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 **$O(g1 \cdot g2)$.**

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:

  $a_0 + a_1 n + \cdots + a_d n^d$ is $O\left(n^d\right)$

- 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

Ω-Notation

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

- **f(N)** is **Ω(g(N))** iff there is a constant **c>0** and **$N_0 \geq 0$** s.t., **$f(N) \geq c \cdot g(N) \geq 0$** for all **$N \geq N_0$**

E.g. **$f(N)=32N^2+17N+1$**

- **f(N)** is both **$\Omega(N^2)$** and **$\Omega(N)$.** Choose $c=32$, $N_0=1$
- **f(N)** is not **$\Omega(N^3)$.**

Fastest growing term dominates

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

Θ-Notation

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

- **f(N)** is **Θ(g(N))** iff there are **$c_0>0$, $c_1>0$** and **$N_0 \geq 0$** s.t. **$c_0 \cdot g(N) \leq f(N) \leq c_1 \cdot g(N)$** for all **$N \geq N_0$**

E.g. **$f(N)=32N^2+17N+1$**

- **f(N)** is **$\Theta(N^2)$.** ← Choose $c_0=32$, $c_1=50$, $N_0=1$
- **f(N)** is neither **Θ(N)** nor **$\Theta(N^3)$.**

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

Question

Which is an equivalent definition of big Theta notation?

A. **f(N)** is **Θ(g(N))** iff **f(N)** is both **O(g(N)) and Ω(g(N))**.

B. **f(N)** is **Θ(g(N))** iff $\lim_{N\to\infty}\frac{f(N)}{g(N)} = c$ for some constant $0 < c < \infty$.

C. Both A and B.

Answer: A

D. Neither A nor B.

Summary

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

- **f(N)** is **O(g(N))** iff there is a constant $c>0$ s.t.,
  f(N) is eventually always $\leq c\, g(N)$

- **f(N)** is **Ω(g(N))** iff there is a constant $\varepsilon>0$ s.t.,
  f(N) is $\geq \varepsilon\, g(N)$ for infinitely

- **f(N)** is **Θ(g(N))** iff there are constants c_1, $c_2>0$ so that
  eventually always $c_1 g(N) \leq f(N) \leq c_2 g(N)$