

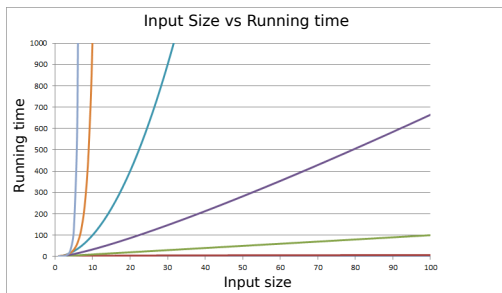
CS311H: Discrete Mathematics

Asymptotic Analysis

Instructor: Işıl Dillig

Complexity Theory Intro

- ▶ **Algorithmic complexity theory:** How fast does the running time of an algorithm grow with respect to input size?
- ▶ Some algorithms scale better as input size grows



Complexity Theory

- ▶ Complexity theory is concerned with expressing the running time of an algorithm with respect to input size
- ▶ Idea 1: Express running time in terms of the number of operations – abstract away choice of hardware and PL
- ▶ Idea 2: Not interested in exact number of operations, just an asymptotic estimate
 - ▶ Number of operations proportional to some function $f(n)$ s.t. they become equal as $n \rightarrow \infty$
 - ▶ e.g., instead of $2 \cdot n^3 + \frac{1}{2}n^2 + \frac{11}{9}n + 4\log(n)$, we just say running time is cubic

Big-O Notation

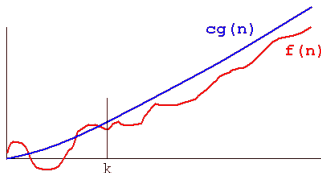
- ▶ Useful tool for asymptotic analysis is **Big-O notation**
- ▶ **Intuition:** If algorithm is $O(f(n))$, then running time is bounded by a function proportional to $f(n)$ for sufficiently large n
- ▶ **Example:** If algorithm is $O(1)$, this means it's constant-time, i.e., running time of algorithm does not depend on input size
- ▶ **Example:** If algorithm is $O(n)$, running time of algorithm grows linearly in input size

Formal Definition of Big-O

A function $f(n)$ is $O(g(n))$ if there exist positive constants C, k such that:

$$\forall n > k. |f(n)| \leq C|g(n)|$$

- ▶ i.e., for sufficiently large n , $f(n)$ is bounded from above by a function proportional to $g(n)$



An Assumption in this Course

A function $f(n)$ is $O(g(n))$ if there exist positive constants C, k such that:

$$\forall n > k. |f(n)| \leq C|g(n)|$$

- ▶ In the context of algorithmic complexity, $f(n)$ represents the number of operations performed
- ▶ Hence, both $f(n)$ and $g(n)$ will always be positive – you can assume this for this class, so we'll omit the absolute value

Example I

Prove that $4n + 2$ is $O(n)$



Example II

Prove that $\frac{1}{2}n^2$ is **not** $O(n)$

- ▶ Proof by contradiction – suppose there was some C, k s.t.:

$$\forall n > k. \frac{1}{2}n^2 \leq Cn$$

- ▶ Implies $\forall n > k. n \leq 2C$
- ▶ But this is not true because k, C are constants and we can make n arbitrarily large

Example III

Prove that $\log_2(3n^2 + 1)$ is $O(\log_2 n)$



Useful Theorem

Let $f(n)$ be a polynomial of degree d . Then $f(n)$ is $O(n^d)$.

- ▶ $f(n)$ must be $a_d n^d + a_{d-1} n^{d-1} + \dots + a_0$
- ▶ **Observe:** $f(n) = n^d (a_d + \frac{a_{d-1}}{n} + \dots + \frac{a_0}{n^d})$
- ▶ Since $n > 1$, $f(n) \leq n^d (a_d + a_{d-1} + \dots + a_0)$
- ▶ Now, pick $C = a_d + a_{d-1} + \dots + a_0$, and $k = 1$
- ▶ Hence, $\forall n > 1. f(n) \leq C \cdot n^d$

Growth of Combination of Functions

- ▶ We are often interested in understanding the **combined growth** of multiple functions
- ▶ Consider a procedure $P(n) = \text{foo}(n); \text{bar}(n);$
- ▶ If we know foo is $O(f(n))$ and bar is $O(g(n))$, what can we say about P ?

Growth of Sum of Two Functions

Suppose $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$. Then,
 $(f_1 + f_2)(x)$ is $O(\max(g_1(x), g_2(x)))$



Example

► Find a Big-O estimate for $f(n) = \log(3n^2 + 1) + \frac{11}{6}n^2$

►

►

►

►

Another Example

- ▶ Consider a procedure $P(n) = \text{foo}(n); \text{bar}(n);$
- ▶ If complexity of `foo` is $O(n)$ and that of `bar` is $O(n \log n)$, what can we say about complexity of P ?
- ▶

Another Useful Theorem

Suppose $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$. Then, $(f_1 \cdot f_2)(x)$ is $O((g_1(x) \cdot g_2(x)))$

- ▶ We know $\exists C_1, C_2, k_1, k_2$ such that:

$$\forall n > k_1. f_1(x) \leq C_1 \cdot g_1(x)$$

$$\forall n > k_2. f_2(x) \leq C_2 \cdot g_2(x)$$

- ▶ Let $k = \max(k_1, k_2)$ and $C = C_1 C_2$
- ▶ Hence, $\forall n > k. (f_1 \cdot f_2)(x) \leq C \cdot g_1(x) \cdot g_2(x)$
- ▶ Thus, $(f_1 \cdot f_2)(x)$ is $O(g_1(x) \cdot g_2(x))$

Example

► Find a Big-O estimate for $f(n) = (\frac{1}{2}n^2 + 5)(\log(3n^2 + 1))$



Another example

- ▶ Assuming complexity of g is $O(\log n)$, find a Big-O estimate for the following procedure f :

$f(n) = \{ \text{for}(i = 0; i < n; i+=2) \{ g(i, n) \} \}$

- ▶ Total # of operations = # of times loop executes * # of operations performed by g
- ▶ What is big-O estimate for the loop?
- ▶ Big-O estimate for f :

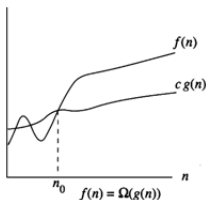
Big-Omega Notation

- ▶ Big-O notation is useful for giving an upper bound for $f(n)$ for large values of n
- ▶ But sometimes we are also interested in a **lower bound**!
- ▶ For this purpose, we use the **Big-Omega** Ω notation, which represents asymptotic lower bounds

Formal Definition of Big-Omega

A function $f(n)$ is $\Omega(g(n))$ if there exist positive constants C, k such that:

$$\forall n > k. |f(n)| \geq C|g(n)|$$



- ▶ i.e., for sufficiently large n , $f(n)$ is bounded from **below** by a function proportional to $g(n)$
- ▶ As before, we will assume $f(n)$ and $g(n)$ are positive

Example

1. Prove that $2n^2 + n$ is $\Omega(n)$

► Find a C and k :

►

2. Prove that $5n + 1$ is not $\Omega(n^2)$

► Suppose there exists constants such that:

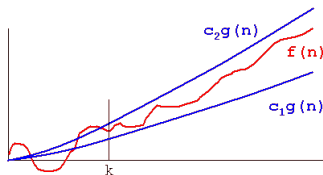
$$\forall n > k. 5n + 1 \geq C \cdot n^2$$

► Implies $\forall n > k. C \cdot n^2 - 5n - 1 \leq 0$

► But since $C > 0$, $C \cdot n^2 - 5n - 1$ is an upward-looking parabola $\Rightarrow$ for n large enough, $C \cdot n^2 - 5n - 1$ is positive

Big Theta

- ▶ **So far:** Big-O gives asymptotic upper bounds, and Big-Omega gives asymptotic lower bounds
- ▶ But sometimes we are interested in a function that serves both as an asymptotic lower **and** upper bound



- ▶ This is expressed using **Big-Theta** notation

Formal Definition of Big-Theta

- ▶ A function $f(x)$ is said to be $\Theta(g(x))$ if $f(x)$ is both $\Omega(g(x))$ and $O(g(x))$
- ▶ Another way of saying this:

$$\exists C_1, C_2, k > 0. \forall n > k. C_1|g(x)| \leq |f(x)| \leq C_2|g(x)|$$

Example I

Prove that $10n + 4^{\log_2 n}$ is $\Theta(n^2)$



Example II

Prove that $\log n! = \Theta(n \log n)$

- First show $\log n!$ is $O(n \log n)$:

Example II, cont.

- Now show that $\log n!$ is $\Omega(n \log n)$:

A Useful Theorem

1. $f(x)$ is $O(g(x))$ if:

$$\lim_{x \rightarrow \infty} \left| \frac{f(x)}{g(x)} \right| < \infty$$

2. $f(x)$ is $\Omega(g(x))$ if:

$$\lim_{x \rightarrow \infty} \left| \frac{f(x)}{g(x)} \right| > 0$$

3. $f(x)$ is $\Theta(g(x))$ if:

$$0 < \lim_{x \rightarrow \infty} \left| \frac{f(x)}{g(x)} \right| < \infty$$

Example I

- ▶ Show that $\frac{x^2+1}{x+1}$ is $\Theta(x)$
- ▶ Use previous theorem:

$$\lim_{x \rightarrow \infty} \frac{x^2 + 1}{x + 1} \cdot \frac{1}{x} = 1$$

- ▶ Hence, it's case (3) $\Rightarrow \Theta(x)$