طراحی الگوریتم

(نمادهای مجانبی)

دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان

بهار ۱۴۰۰

نمادهای مجانبی

Asymptotic Notation in Seven Words

suppress *constant factors* *and* *lower-order terms*

too system-dependent *irrelevant for large inputs*

نمادهای مجانبی

Big-O Notation (English Version)

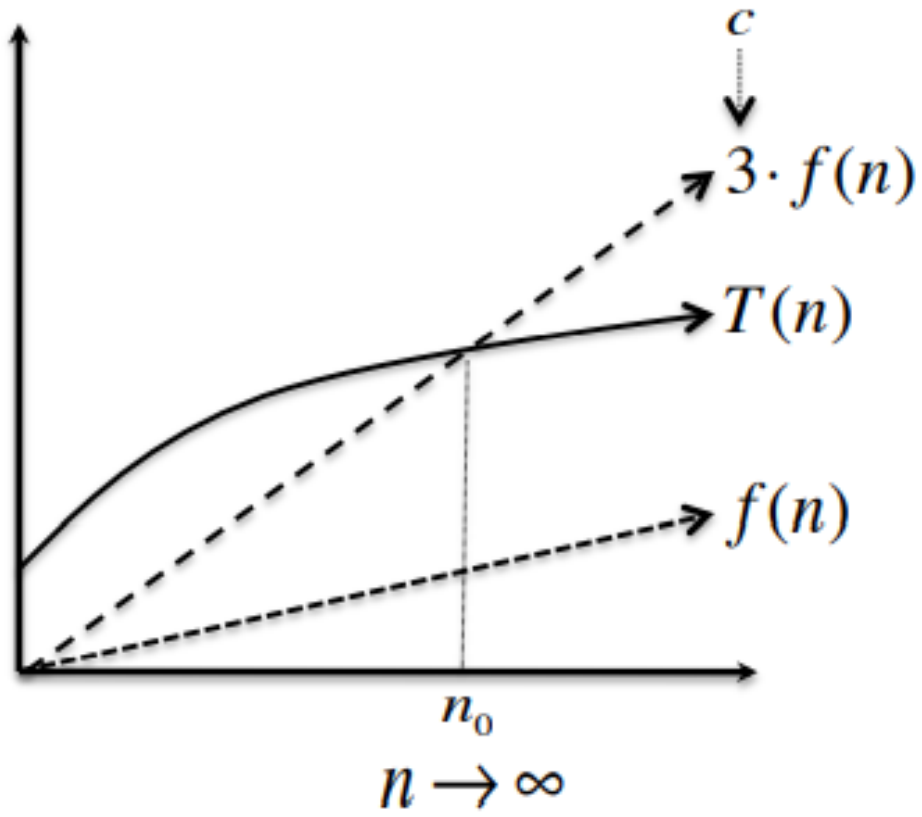
$T(n) = O(f(n))$ if and only if $T(n)$ is eventually bounded above by a constant multiple of $f(n)$.

Big-O Notation (Mathematical Version)

$T(n) = O(f(n))$ if and only if there exist positive constants c and n_0 such that

$$T(n) \leq c \cdot f(n) \qquad (2.1)$$

for all $n \geq n_0$.




دانشگاه صنعتی اصفهان

نمادهای مجانبی

Big-Omega Notation (Mathematical Version)

$T(n) = \Omega(f(n))$ if and only if there exist positive constants c and n_0 such that

$$T(n) \geq c \cdot f(n)$$

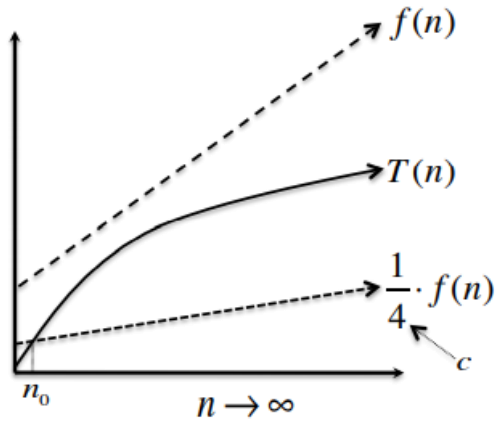
for all $n \geq n_0$.

Big-O Notation (Mathematical Version)

$T(n) = O(f(n))$ if and only if there exist positive constants c and n_0 such that

$$T(n) \leq c \cdot f(n) \qquad (2.1)$$

for all $n \geq n_0$.

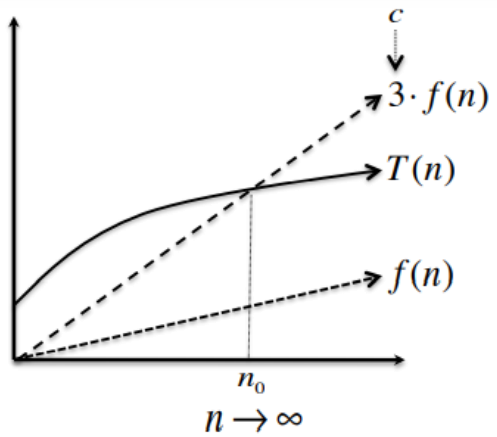


Big-Theta Notation (Mathematical Version)

$T(n) = \Theta(f(n))$ if and only if there exist positive constants c_1, c_2, and n_0 such that

$$c_1 \cdot f(n) \leq T(n) \leq c_2 \cdot f(n)$$

for all $n \geq n_0$.



نمادهای مجانبی

A Word of Caution

Algorithm designers often use big-O notation even when big-theta notation would be more accurate. This book will follow that tradition. For example, consider a subroutine that scans an array of length n, performing a constant number of operations per entry (like the `Merge` subroutine in Section 1.4.5). The running time of such a subroutine is obviously $\Theta(n)$, but it's common to only mention that it is $O(n)$. This is because, as algorithm designers, we generally focus on upper bounds—guarantees about how long our algorithms could possibly run.

A Word of Caution

When we say that c and n_0 are constants, we mean they *cannot depend on* n. For example, in Figure 2.1, c and n_0 were fixed numbers (like 3 or 1000), and we then considered the inequality (2.1) as n grows arbitrarily large (looking rightward on the graph toward infinity). If you ever find yourself saying "take $n_0 = n$" or "take $c = \log_2 n$" in an alleged big-O proof, you need to start over with choices of c and n_0 that are independent of n.

نمادهای مجانبی

Little-O Notation (Mathematical Version)

$T(n) = o(f(n))$ if and only if for every positive constant $c > 0$, there exists a choice of n_0 such that

$$T(n) \leq c \cdot f(n) \qquad (2.2)$$

for all $n \geq n_0$.

2.4.4 Where Does Notation Come From?

Asymptotic notation was not invented by computer scientists—it has been used in number theory since around the turn of the 20th century. Donald E. Knuth, the grandfather of the formal analysis of algorithms, proposed using it as the standard language for discussing rates of growth, and in particular for algorithm running times.

> "On the basis of the issues discussed here, I propose that members of SIGACT,[8] and editors of computer science and mathematics journals, adopt the O, Ω, and Θ notations as defined above, unless a better alternative can be found reasonably soon."[9]


دانشگاه صنعتی اصفهان


دانشگاه صنعتی اصفهان