

بسم الله الرحمن الرحيم

دانشگاه صنعتی اصفهان - دانشکده مهندسی برق و کامپیوتر
(نیم سال تحصیلی ۴۰۰۱)

طراحی الگوريتمها

حسين فلسفين

Asymptotic Notations (نمادهای مجانبی)

$O, \Omega, \Theta, o, \omega$

* $f(n) \in O(g(n))$ **is like** $a \leq b$ ($a, b \in \mathbb{R}$)

* $f(n) \in \Omega(g(n))$ **is like** $a \geq b$

* $f(n) \in \Theta(g(n))$ **is like** $a = b$

* $f(n) \in o(g(n))$ **is like** $a < b$

* $f(n) \in \omega(g(n))$ **is like** $a > b$


نمادهای مجانبی (O ، Θ ، Ω و ω) و معرفی رسمی مفهوم مرتبه

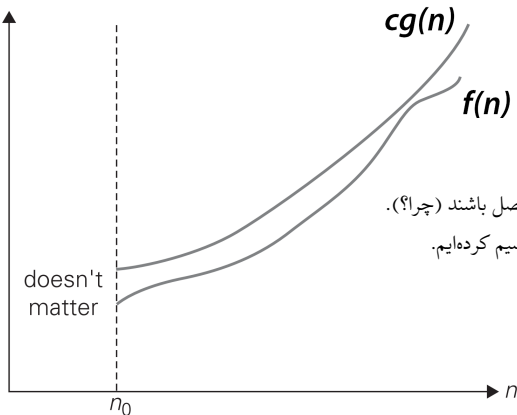
The use of big- O notation in computer science was popularized by Donald Knuth, who also introduced the big- Ω and big- Θ notations.

*Asymptotic Notations
Asymptotic Behavior
Eventual Behavior*



Definition: A complexity function $f(n)$ is said to be in $O(g(n))$, denoted $f(n) \in O(g(n))$, if $f(n)$ is bounded above by some constant multiple of $g(n)$ **for all large n** , i.e., if there exist some positive constant c and some nonnegative integer n_0 such that $f(n) \leq cg(n)$ for all $n \geq n_0$.

"big-O"  Asymptotic upper bound



$$100n+200 \in O(n^2), n \log_2(n) \in O(n^2), 50n^2 \in O(n^2), 0.001n^3 \notin O(n^2).$$

.....

Example: Show that $7n^2 \in O(n^3)$.

Solution: Note that when $n \geq 7$, we have $7n^2 \leq n^3$ (we can obtain this inequality by multiplying both sides of $n \geq 7$ by n^2). Consequently, we can take $c = 1$ and $n_0 = 7$ as witnesses to establish the relationship $7n^2 \in O(n^3)$. **Alternatively**, when $n \geq 1$, we have $7n^2 \leq 7n^3$, so that $c = 7$ and $n_0 = 1$ are also witnesses to the relationship $7n^2 \in O(n^3)$.


Example: Is it also true that $n^3 \in O(7n^2)$?

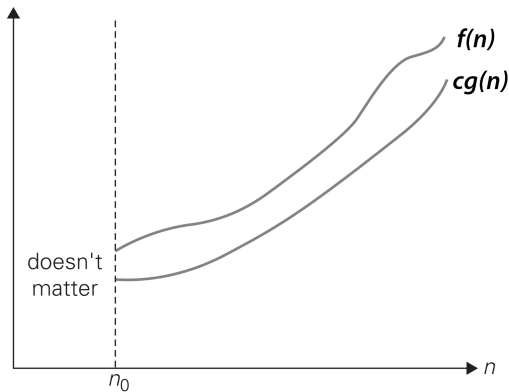
Solution: To determine whether $n^3 \in O(7n^2)$, we need to determine whether witnesses c and n_0 exist, so that $n^3 \leq c(7n^2)$ whenever $n \geq n_0$. We will show that no such witnesses exist using a proof by **contradiction**. If c and n_0 are witnesses, the inequality $n^3 \leq c(7n^2)$ holds for all $n \geq n_0$. Observe that the inequality $n^3 \leq c(7n^2)$ is equivalent to the inequality $n \leq 7c$, which follows by dividing both sides by the positive quantity n^2 . However, no matter what c is, it is not the case that $n \leq 7c$ for all $n \geq n_0$ no matter what n_0 is, because n can be made arbitrarily large. It follows that no witnesses c and n_0 exist for this proposed big- O relationship. Hence, $n^3 \notin O(7n^2)$.

Example: Show that $n^2 \notin O(n)$.

Solution: To show that $n^2 \notin O(n)$, we must show that no pair of witnesses c and n_0 exist such that $n^2 \leq cn$ whenever $n \geq n_0$. We will use a proof by **contradiction** to show this. Suppose that there are constants c and n_0 for which $n^2 \leq cn$ whenever $n \geq n_0$. Observe that when $n > 0$ we can divide both sides of the inequality $n^2 \leq cn$ by n to obtain the equivalent inequality $n \leq c$. However, no matter what c and n_0 are, the inequality $n \leq c$ cannot hold for all n with $n \geq n_0$. In particular, once we set a value of n_0 , we see that when n is larger than the maximum of n_0 and c , it is not true that $n \leq c$ even though $n \geq n_0$. This contradiction shows that $n^2 \notin O(n)$.

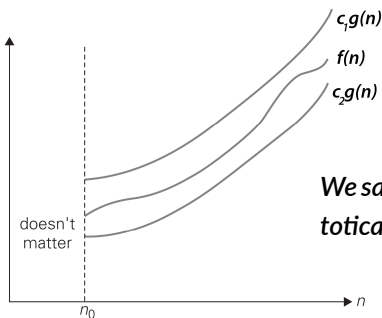
Definition: A complexity function $f(n)$ is said to be in $\Omega(g(n))$, denoted $f(n) \in \Omega(g(n))$, if $f(n)$ is bounded below by some positive constant multiple of $g(n)$ **for all large n** , i.e., if there exist some positive constant c and some nonnegative integer n_0 such that $f(n) \geq cg(n)$ for all $n \geq n_0$.

"big- Ω "  Asymptotic lower bound



Definition: A complexity function $f(n)$ is said to be in $\Theta(g(n))$, denoted $f(n) \in \Theta(g(n))$, if $f(n)$ is bounded both above and below by some positive constant multiples of $g(n)$ for all large n , i.e., if there exist some positive constants c_1 and c_2 and some nonnegative integer n_0 such that

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



We say that $g(n)$ is an asymptotically tight bound for $f(n)$.

به راحتی با استفاده از تعاریف می‌توان نشان داد که:

$$\Theta(g(n)) = \Omega(g(n)) \cap O(g(n))$$

به عبارت دیگر:

For any two functions $f(n)$ and $g(n)$, we have $f(n) \in \Theta(g(n))$ if and only if $f(n) \in \Omega(g(n))$ and $f(n) \in O(g(n))$.

Definition: For a given complexity function $g(n)$, $o(g(n))$ is the set of all complexity functions $f(n)$ satisfying the following: For **every** positive real constant c there exists a nonnegative integer n_0 such that, for all $n \geq n_0$,

$$f(n) \leq cg(n).$$

The definitions of O -notation and o -notation are similar. The main difference is that in $f(n) \in O(g(n))$, the bound $f(n) \leq cg(n)$ holds for **some** constant $c > 0$, but in $f(n) \in o(g(n))$, the bound $f(n) \leq cg(n)$ holds for **all** constants $c > 0$.

حواسمان باشد که اختلاف در این است (و نه در چیز دیگر) که در تعریف \mathcal{O} کوچک ما صور عمومی (\forall) داریم، اما در تعریف \mathcal{o} بزرگ، صور ما وجودی (\exists) است.

Because the bound holds for **every** real positive constant c , it holds for **arbitrarily small** c . For example, if $f(n) \in o(g(n))$, there is an n_0 such that, for $n \geq n_0$, $f(n) \leq 0.0000001g(n)$. We see that $f(n)$ becomes **insignificant** relative to $g(n)$ as n becomes large. For the purposes of analysis, if $f(n)$ is in $o(g(n))$, then $f(n)$ is **eventually much better** than functions such as $g(n)$.

به زبان دیگر داریم:

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0.$$

مثال: نشان می‌دهیم $n \in o(n^2)$. ثابت حقیقی $c > 0$ را در نظر بگیرید. به دنبال یک عدد صحیح ناصفر همچون n_0 هستیم که برای $n \geq n_0$,

$$n \leq cn^2.$$

اگر دو طرف این نامساوی را بر cn تقسیم کنیم داریم $n \geq \frac{1}{c}$. پس $n_0 = \frac{1}{c}$. توجه کنید که مقدار n_0 به مقدار c وابسته است، پس برای آنکه داشته باشیم $n \leq 0.01n^2$ باید n_0 را (حداقل) برابر با ۱۰۰، و برای آنکه داشته باشیم $n \leq 0.0001n^2$ باید n_0 را (حداقل) برابر با ۱۰۰۰۰ قرار دهیم.

.....
مثال: $n \notin o(100n)$. چرا؟ چون اگر چنین باشد و قرار دهیم $c = 0.001$ ، آنگاه n_0 وجود دارد که به ازای هر $n \geq n_0$ داریم $n \leq 0.001 \times 100n$ ، که محال است.

Definition: For a given complexity function $g(n)$, $\omega(g(n))$ is the set of all complexity functions $f(n)$ satisfying the following: For **every** positive real constant c there exists a nonnegative integer n_0 such that, for all $n \geq n_0$,

$$cg(n) \leq f(n).$$

.....
For example, $\frac{n^2}{2} \in \omega(n)$, but $\frac{n^2}{2} \notin \omega(n^2)$. The relation $f(n) \in \omega(g(n))$ implies that

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty,$$

if the limit exists. That is, $f(n)$ becomes **arbitrarily large** relative to $g(n)$ as n approaches infinity.

Theorem: If $g(n) \in o(f(n))$, then $g(n) \in O(f(n)) \setminus \Omega(f(n))$. That is, $g(n)$ is in $O(f(n))$ but is not in $\Omega(f(n))$.

Proof: Because $g(n) \in o(f(n))$, for **every** positive real constant c there exists an n_0 such that, for all $n \geq n_0$ we have $g(n) \leq cf(n)$, which means that the bound certainly holds for **some** c . Therefore, $g(n) \in O(f(n))$.

We will show that $g(n)$ is not in $\Omega(f(n))$ using proof by **contradiction**. If $g(n) \in \Omega(f(n))$, then there exists some real constant $c > 0$ and some n_1 such that, for all $n \geq n_1$ we have $g(n) \geq cf(n)$. But, because $g(n) \in o(f(n))$, there exists some n_2 such that, for all $n \geq n_2$ we have $g(n) \leq \frac{c}{2}f(n)$. Both inequalities would have to hold for all n greater than or equal to $\max\{n_1, n_2\}$. This contradiction (i.e., $cf(n) \leq \frac{c}{2}f(n)$) proves that $g(n)$ cannot be in $\Omega(f(n))$.

□

Notice that $O(f(n)) \setminus \Omega(f(n)) = O(f(n)) \setminus \Theta(f(n))$.

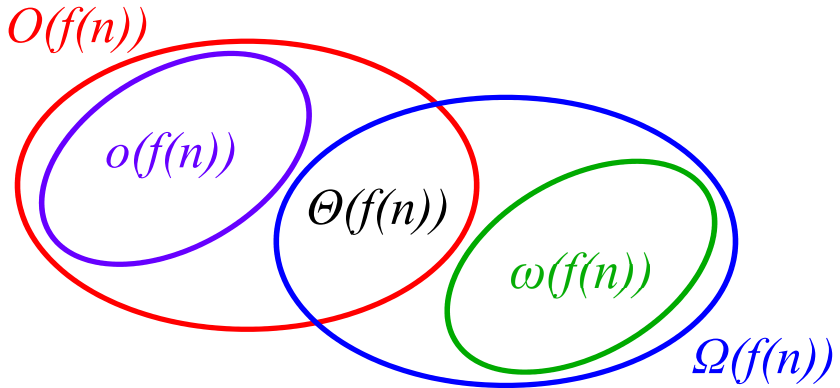
You may think that $o(f(n))$ and $O(f(n)) \setminus \Omega(f(n))$ must be the same set. This is **not** true. There are **unusual** functions that are in $O(f(n)) \setminus \Omega(f(n))$ but that are not in $o(f(n))$:

$$g(n) = \begin{cases} n, & \text{if } n \text{ is even,} \\ 1, & \text{if } n \text{ is odd.} \end{cases}$$

It can be shown that $g(n) \in O(n) \setminus \Omega(n)$ but $g(n) \notin o(n)$.

When complexity functions represent the time complexities of actual algorithms, **ordinarily** the functions in $O(f(n)) \setminus \Omega(f(n))$ are the same ones that are in $o(f(n))$. (Although esoteric functions arise in advanced algorithm analysis, a small set of time complexities suffice for most algorithms we will see in this course.)

An important diagram:



گزاره‌های زیر را بدون اثبات بیان می‌کنیم. توصیه می‌شود به عنوان تمرین همه یا بعضی از آنها را اثبات کنید.

Transitivity:

$f(n) \in \Theta(g(n))$ and $g(n) \in \Theta(h(n))$ imply $f(n) \in \Theta(h(n))$,
 $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$ imply $f(n) \in O(h(n))$,
 $f(n) \in \Omega(g(n))$ and $g(n) \in \Omega(h(n))$ imply $f(n) \in \Omega(h(n))$,
 $f(n) \in o(g(n))$ and $g(n) \in o(h(n))$ imply $f(n) \in o(h(n))$,
 $f(n) \in \omega(g(n))$ and $g(n) \in \omega(h(n))$ imply $f(n) \in \omega(h(n))$.

Reflexivity:

$f(n) \in \Theta(f(n))$, $f(n) \in O(f(n))$, and $f(n) \in \Omega(f(n))$.

Symmetry:

$f(n) \in \Theta(g(n))$ if and only if $g(n) \in \Theta(f(n))$.

Transpose symmetry:

$f(n) \in O(g(n))$ if and only if $g(n) \in \Omega(f(n))$,

$f(n) \in o(g(n))$ if and only if $g(n) \in \omega(f(n))$.

- * A relation on a set A is called an **equivalence relation** if it is reflexive, symmetric, and transitive. Equivalence relations are important throughout mathematics and computer science. One reason for this is that in an equivalence relation, when two elements are related it makes sense to say they are **equivalent**.
- * Let R be an equivalence relation on a set A . The set of all elements that are related to an element a of A is called the **equivalence class** of a . The equivalence classes of R form a **partition** of A .
- * Θ defines an equivalence relation on the set of complexity functions.

ما در ابتدای بحث نمادهای مجانبی $O, \Omega, \Theta, o, \omega$ را به نمادهای $\leq, \geq, =, <, >$ نظیر کردیم. اما تفاوت‌هایی نیز وجود دارد:

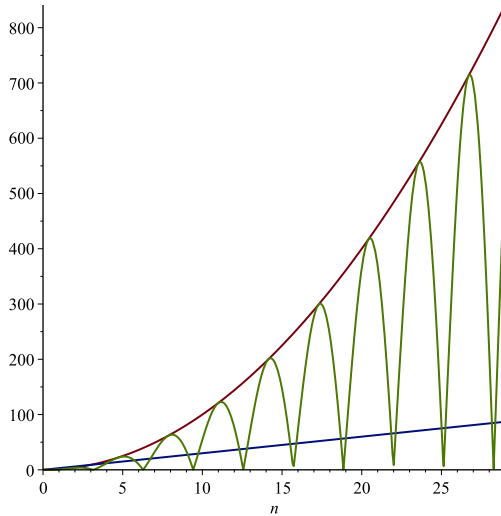
خاصیت *Trichotomy* (به معنای به سه دسته تقسیم کردن):

For any two real numbers a and b , exactly one of the following must hold: $a < b$, $a = b$, or $a > b$.

اما **نمی‌توان** همواره دو توابع پیچیدگی را با یکدیگر مقایسه کرد. این به معنای آن است که برای دو تابع پیچیدگی $f(n)$ و $g(n)$ ممکن است نه $f(n) \in O(g(n))$ برقرار باشد و نه $f(n) \in \Omega(g(n))$. دو مثال ذیل را ببینید:

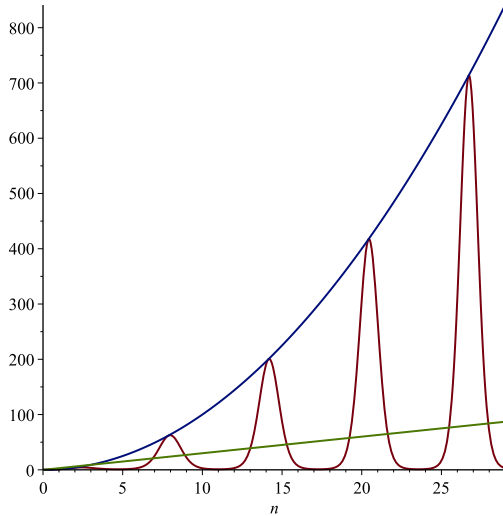
$$f(n) = 3n \text{ and } g(n) = |n^2 \sin(n)|$$

$g(n)$ بین 0 و n^2 نوسان می‌کند



$$f(n) = 3n \text{ and } g(n) = n^{1+\sin(n)}$$

$g(n)$ بین ۱ و n^2 نوسان می‌کند



Using Limits for Comparing Orders of Growth

به‌ندرت از تعاریف رسمی نمادهای مجانبی برای مقایسه مرتبه رشد دو تابع معین استفاده می‌شود. یک روش بسیار ساده‌تر برای انجام چنین کاری بهره‌گیری از حد نسبت دو تابع است (وقتی n به سمت $+\infty$ میل می‌کند):

$$\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = \begin{cases} c & \text{implies } g(n) \in \Theta(f(n)) \text{ if } c > 0 \\ 0 & \text{implies } g(n) \in o(f(n)) \\ \infty & \text{implies } f(n) \in o(g(n)) \end{cases}$$

Note that the first two cases mean that $g(n) \in O(f(n))$ and the first and third cases mean that $g(n) \in \Omega(f(n))$. The limit-based approach is **often more convenient than the one based on the definitions because it can take advantage of the powerful calculus techniques developed for computing limits, such as L'Hôpital's rule and Stirling's formula.**

L'Hôpital's Rule: If $f(x)$ and $g(x)$ are both differentiable with derivatives $f'(x)$ and $g'(x)$, respectively, and if

$$\lim_{x \rightarrow \infty} f(x) = \lim_{x \rightarrow \infty} g(x) = \infty,$$

then

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \lim_{x \rightarrow \infty} \frac{f'(x)}{g'(x)}$$

whenever the limit on the right exists.

Stirling's formula: $n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$ for large values of n .

Example: Compare the orders of growth of $\ln^3(n)$ and n .

$$\lim_{n \rightarrow \infty} \frac{\ln^3 n}{n} = \lim_{n \rightarrow \infty} \frac{\ln^2 n}{n} = \lim_{n \rightarrow \infty} \frac{\ln n}{n} = \lim_{n \rightarrow \infty} \frac{1}{n} = 0.$$

Example: Compare the orders of growth of $n!$ and 2^n .

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{n!}{2^n} &= \lim_{n \rightarrow \infty} \frac{\sqrt{2\pi n} \left(\frac{n}{e}\right)^n}{2^n} = \\ &= \lim_{n \rightarrow \infty} \sqrt{2\pi n} \frac{n^n}{2^n e^n} = \lim_{n \rightarrow \infty} \sqrt{2\pi n} \left(\frac{n}{2e}\right)^n = \infty. \end{aligned}$$

محاسبه حد با نرم افزار میپل

$$\lim_{n \rightarrow \infty} \left(\frac{(\log_2(n))^{\log_2(n)}}{(1.000000000000000001)^n} \right);$$

0.

$$\lim_{n \rightarrow \infty} \left(\frac{n^{10000000000000000000000000000}}{(\log_2(n))^{\log_2(n)}} \right);$$

0

$$\log_b a = \frac{\log_c a}{\log_c b} \quad (a, b, c \in \mathbb{R}^+)$$

For $b > 1$ and $a > 1$, $\log_a n \in \Theta(\log_b n)$ **because**

$$\log_a n = \frac{1}{\log_b a} \log_b n = \log_a b \log_b n.$$

We say that a function $f(n)$ is **polylogarithmically bounded** if $f(n) \in O(\log^k(n))$ for some constant k . It can be shown that $\log^b(n) \in o(n^a)$ for any constant $a > 0$. Thus, any positive polynomial function grows faster than any polylogarithmic function.

We say that a function $f(n)$ is **polynomially bounded** if $f(n) \in O(n^k)$ for some constant k . It can be shown that for all real constants a and b such that $a > 1$, $n^b \in o(a^n)$. Thus, any exponential function with a base strictly greater than 1 grows faster than any polynomial function.

فی المجموع:

$$\begin{aligned} \Theta(1) \rightsquigarrow \Theta(\log(n)) \rightsquigarrow \Theta(\log^2(n)) \rightsquigarrow \Theta(\log^j(n)) \rightsquigarrow \Theta(\log^k(n)) \rightsquigarrow \\ \Theta(n) \rightsquigarrow \Theta(n \log(n)) \rightsquigarrow \Theta(n^2) \rightsquigarrow \Theta(n^j) \rightsquigarrow \Theta(n^k) \rightsquigarrow \\ \Theta(a^n) \rightsquigarrow \Theta(b^n) \rightsquigarrow \Theta(n!) \end{aligned}$$

where $k > j > 2$ and $b > a > 1$. If a complexity function $g(n)$ is in a category that is to the left of the category containing $f(n)$, then

$$g(n) \in o(f(n)).$$

The iterated logarithm function

We use the notation $\log^* n$ (read “log star of n ”) to denote the **iterated logarithm**, defined as

$$\log^* n = \min\{i \geq 0 : \log^{(i)} n \leq 1\}$$

or

$$\log^* n = \begin{cases} 1 + \log^*(\log n), & \text{if } n > 1, \\ 0, & \text{otherwise.} \end{cases}$$

The iterated logarithm is **a very slowly growing function**: $\log^* 2 = 1$, $\log^* 4 = 2$, $\log^* 16 = 3$, $\log^* 65536 = 4$, $\log^* 2^{65536} = 5$. Since the number of atoms in the observable universe is estimated to be about 10^{80} , which is much less than 2^{65536} , we rarely encounter an input size n such that $\lg^* n > 5$.

Solved Exercise: Is $2^{n+1} \in O(2^n)$? Is $2^{2n} \in O(2^n)$?

Solution:

از آنجا که $\lim_{n \rightarrow \infty} \frac{2^{n+1}}{2^n} = 2$ پس $2^{n+1} \in \Theta(2^n)$ پس $2^{n+1} \in O(2^n)$. حال فرض کنید که $2^{2n} \in O(2^n)$ ، در این صورت یک عدد حقیقی مثبت همچون c و یک عدد صحیح نامنفی همچون n_0 وجود دارد که به‌ازای هر $n \geq n_0$ داریم $2^{2n} \leq c2^n$. پس با این حساب برای هر $n \geq n_0$ داریم $2^n \leq c$ که ناممکن است. پس فرض خلف باطل است و داریم $2^{2n} \notin O(2^n)$.

Solved Exercise: Prove that $\omega(f(n)) \cap o(f(n))$ is the empty set.

Solution:

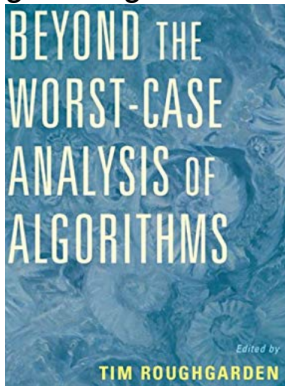
قبلاً ثابت کردیم که $o(f(n)) \subseteq O(f(n)) \setminus \Omega(f(n))$. از طرفی به‌راحتی می‌توان نشان داد که $\omega(f(n)) \subseteq \Omega(f(n))$. پس بدیهی است که

$$\omega(f(n)) \cap o(f(n)) = \emptyset.$$

Exercise: Prove that if $c \geq 0$, $d > 0$, $g(n) \in O(f(n))$, and $h(n) \in \Theta(f(n))$, then $c \times g(n) + d \times h(n) \in \Theta(f(n))$.

(با استفاده از تعریف اثبات کنید.)

Typical algorithms courses rely almost entirely on a single analysis framework, that of worst-case analysis, wherein an algorithm is assessed by its worst performance on any input of a given size. The purpose of this book is to popularize **several alternatives to worst-case analysis** and their most notable algorithmic applications, from clustering to linear programming to neural network training.



Amortized Analysis

In an **amortized analysis**, we average the time required to perform a sequence of data-structure operations over all the operations performed. With amortized analysis, we can show that the average cost of an operation is small, if we average over a sequence of operations, even though a single operation within the sequence might be expensive.

- * Aggregate analysis
- * The accounting method
- * The potential method

👉 **CLRS, Chapter 17**