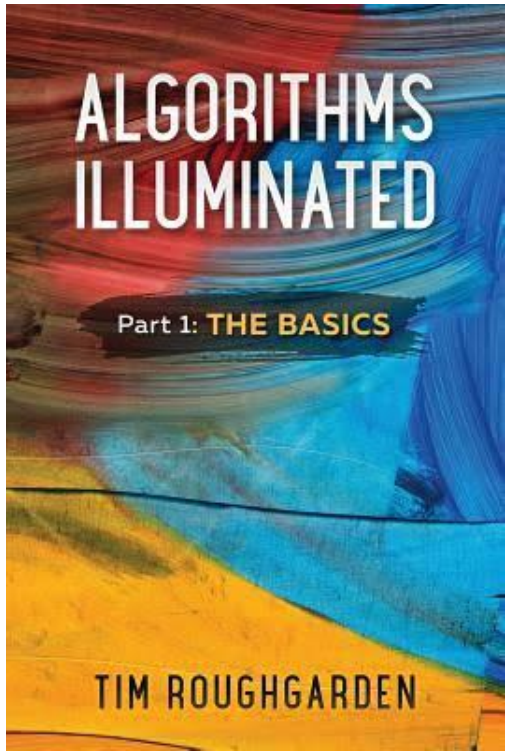
طراحی الگوریتم

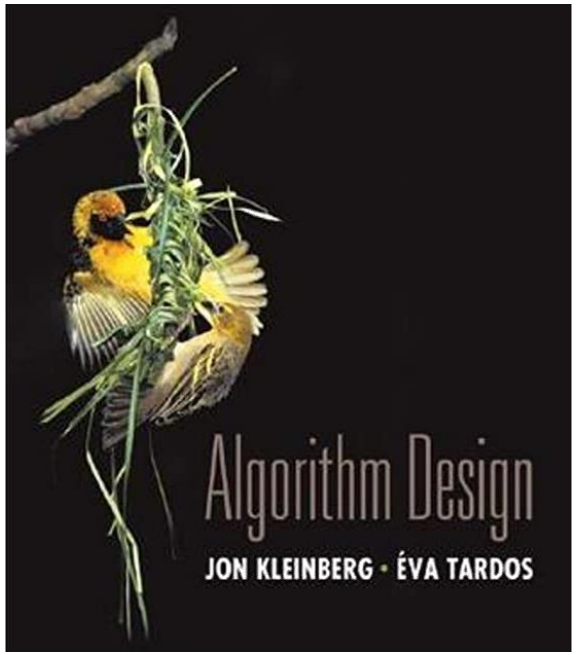
(ضرب، مرتب‌سازی، نمادهای مجانبی)

دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان

بهار ۱۴۰۰

ضرب اعداد صحیح



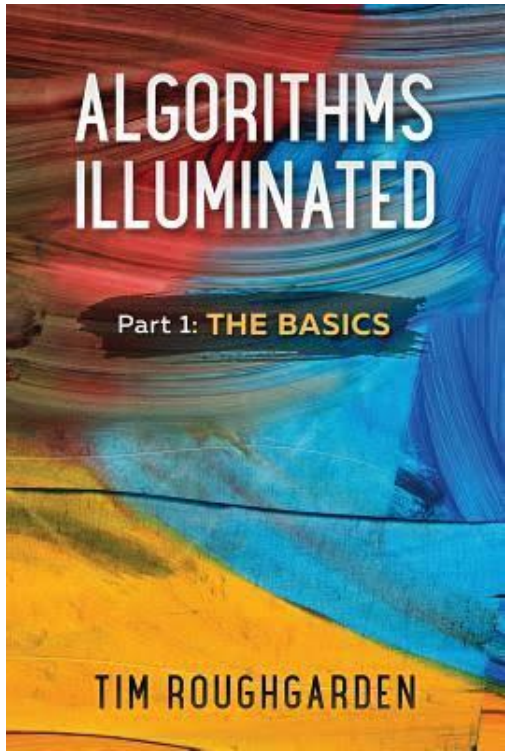


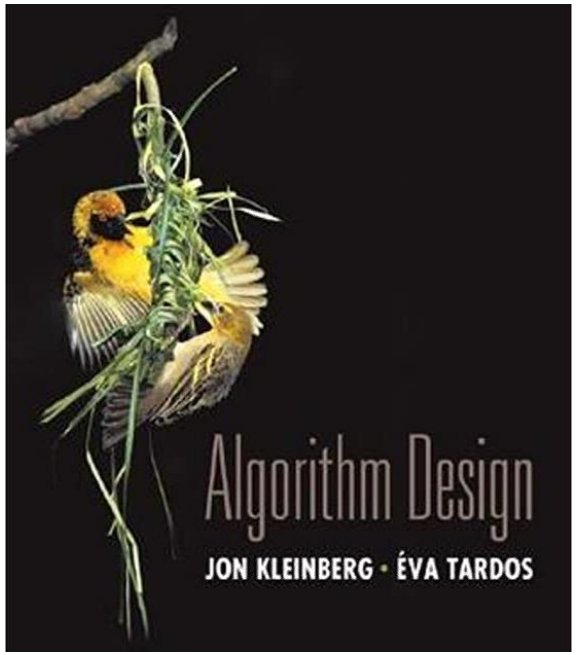
فصل اول، صفحه ۳

ضرب اعداد صحیح

ورودی: دو عدد صحیح نامنفی

هدف: حاصل‌ضرب اعداد ورودی





فصل اول، صفحه ۳

آیا می‌توان بهتر عمل کرد؟

Mantra (Tim Roughgarden):

Can we do better?

“Perhaps the most important principle for the good algorithm designer is to refuse to be content.”

Aho, Hopcroft, Ullman

روش ضرب Karatsub (۱۹۶۰)

$$\begin{array}{r} 5678 \\ \times\ 1234 \\ \hline \end{array}$$

روش ضرب Karatsub (۱۹۶۰)

Karatsuba

Input: two n-digit positive integers x and y.
Output: the product $x \cdot y$.
Assumption: n is a power of 2.

---

if $n = 1$ **then** // base case
  compute $x \cdot y$ in one step and return the result
else // recursive case
  $a, b :=$ first and second halves of x
  $c, d :=$ first and second halves of y
  compute $p := a + b$ and $q := c + d$ using grade-school addition
  recursively compute $ac := a \cdot c$, $bd := b \cdot d$, and $pq := p \cdot q$
  compute $adbc := pq - ac - bd$ using grade-school addition
  compute $10^n \cdot ac + 10^{n/2} \cdot adbc + bd$ using grade-school addition and return the result

سودوکد(Pseudocode)

On Pseudocode

This book explains algorithms using a mixture of high-level pseudocode and English (as in this section). I'm assuming that you have the skills to translate such high-level descriptions into working code in your favorite programming language. Several other books

and resources on the Web offer concrete implementations of various algorithms in specific programming languages.

The first benefit of emphasizing high-level descriptions over language-specific implementations is flexibility: while I assume familiarity with *some* programming language, I don't care which one. Second, this approach promotes the understanding of algorithms at a deep and conceptual level, unencumbered by low-level details. Seasoned programmers and computer scientists generally think and communicate about algorithms at a similarly high level.

Still, there is no substitute for the detailed understanding of an algorithm that comes from providing your own working implementation of it. I strongly encourage you to implement as many of the algorithms in this book as you have time for. (It's also a great excuse to pick up a new programming language!) For guidance, see the end-of-chapter Programming Problems and supporting test cases.