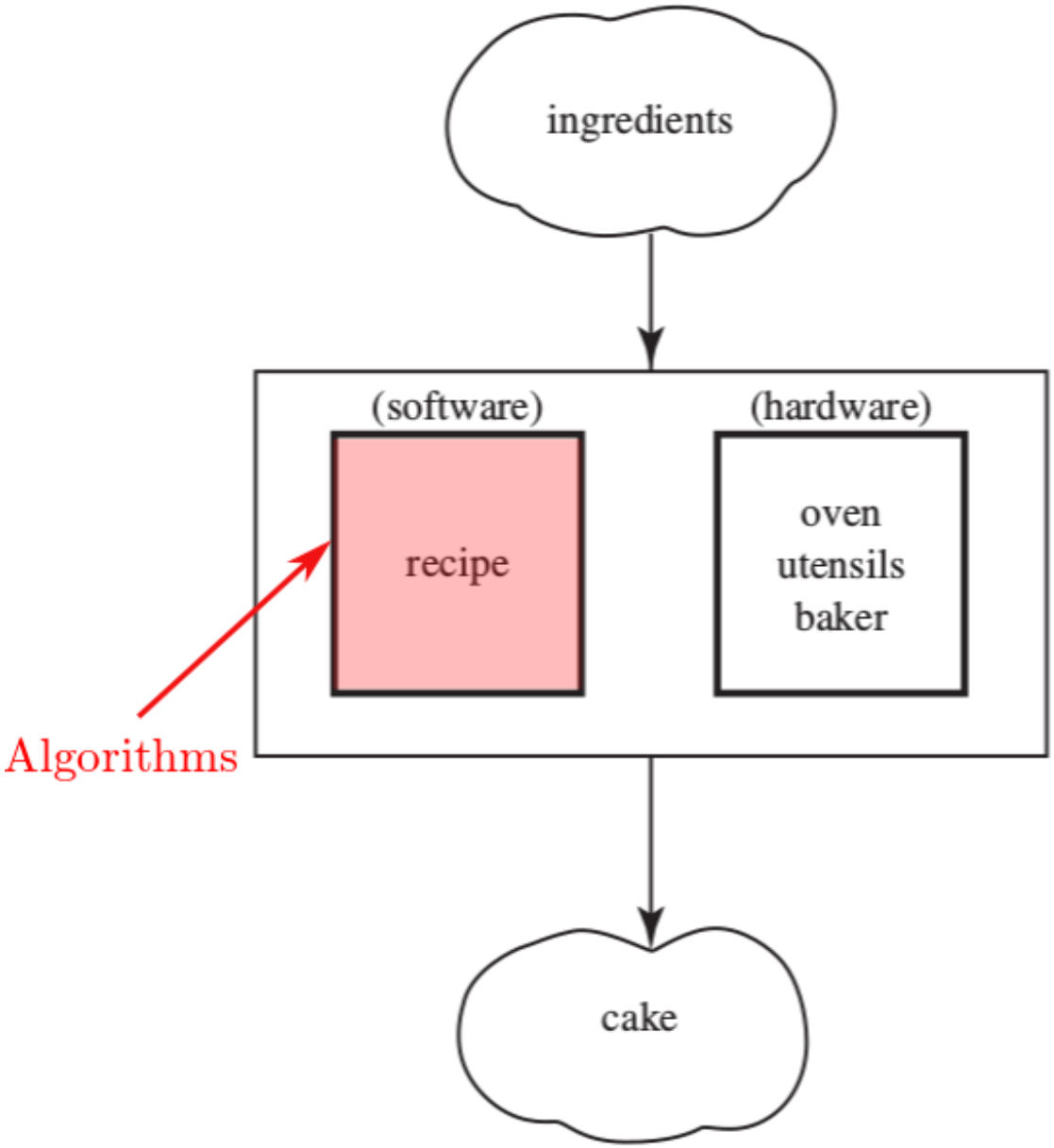
باسمه تعالی

Introduction

Introduction

What is an algorithm?

- A **step-by-step procedure** to accomplish a specific task.
- The **idea** behind any reasonable computer program.


ingredients
(software)
(hardware)
recipe
oven
utensils
baker
Algorithms
cake

- Somewhere between 400 and 300 B.C. – **Euclid** invented an algorithm for finding the greatest common divisor (**gcd**) of two positive integers. It is the **first non-trivial algorithm** ever devised.

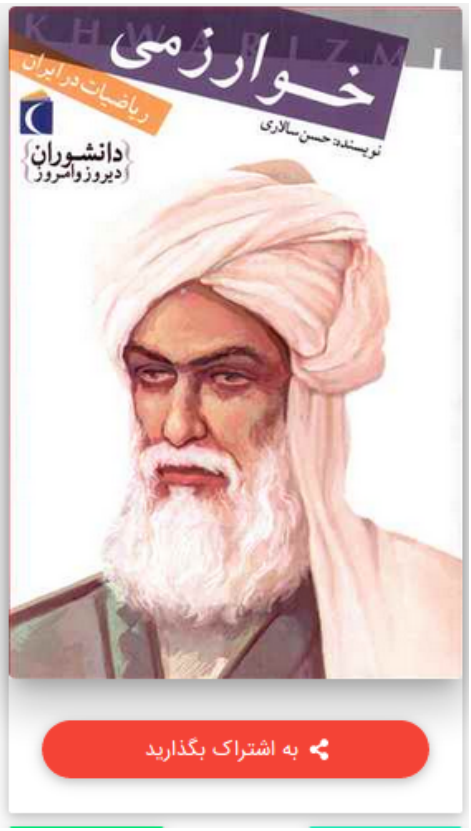
  Euclid's algorithm is based on applying repeatedly the equality

  $$gcd(m, n) = gcd(n, m \bmod n)$$

  until $m \bmod n$ is equal to 0

  Example: $gcd(60, 24) = gcd(24, 12) = gcd(12, 0) = 12$

- The word algorithm is derived from the name of the Persian mathematician **Mohammed al-Khowarizmi**, who lived during the 9^{th} century, and who is credited with providing the **step-by-step rules for adding, subtracting, multiplying, and dividing ordinary decimal numbers**. When written in Latin, the name became Algorismus, from which algorithm is but a small step.



خوارزمی (ریاضیات در ایران)

محراب قلم

بدون امتیاز

قیمت اصلی: ~~۷۰,۰۰۰ ریال~~

مقدار تخفیف: ۷,۰۰۰ ریال

قیمت نهایی خرید شما: ۶۳,۰۰۰ ریال

۱ + -

افزودن به سبد خرید

توضیحات:

انتشارات محراب قلم منتشر کرد:

خوارزمی با نگارش کتاب جبر و مقابله شاخه ی جدیدی را در ریاضیات بنیان نهاد که امروزه به نام جبر شناخته می شود. همچنین کتابی درباره ی حساب هندی نوشت و شیوه ی عددنویسی هندی (با رقم های ۱ تا ۹ و صفر) را به جهانیان آموزش داد. این شیوه که تا پیش از روزگار خوارزمی فقط در هند رواج داشت، اکنون در سراسر جهان به دانش آموزان آموزش داده می شود. خوارزمی با نوشتن این کتاب ها راهی را آغاز کرد که به تولید سامانه های دیجیتال (رقومی) انجامید. افزون بر این، او راه را برای برنامه ریزی سرهای فضایی با کمک معامله های جبری هموار کرد.

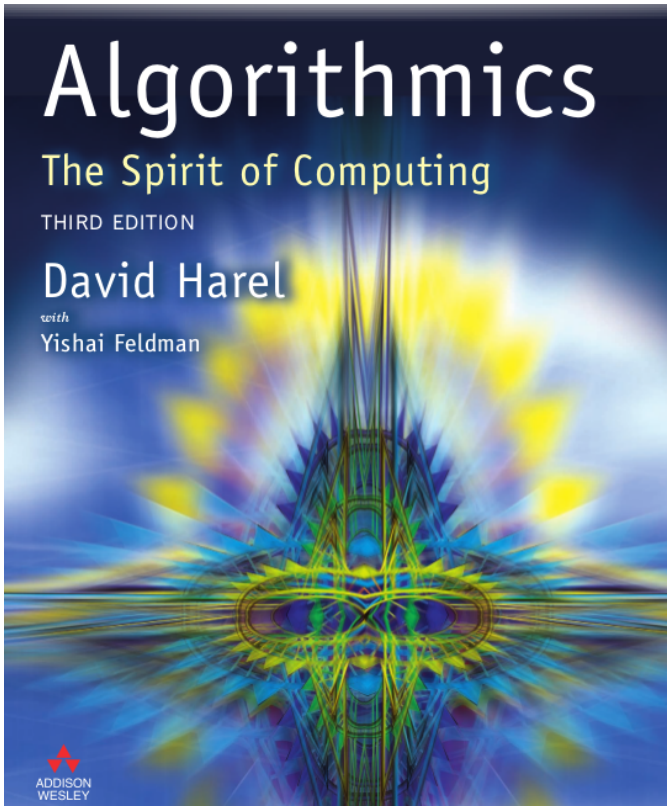
Let's be Motivated ...

What is science and what is art?

Science is what we understand well enough to explain to a computer; art is everything else. — Donald Ervin Knuth

> *Algorithmics is more than a branch of computer science. It is* **the core of computer science**, *and, in all fairness, can be said to be* **relevant to most of science, business, and technology**.

Is computer science is a good name for algorithmics?



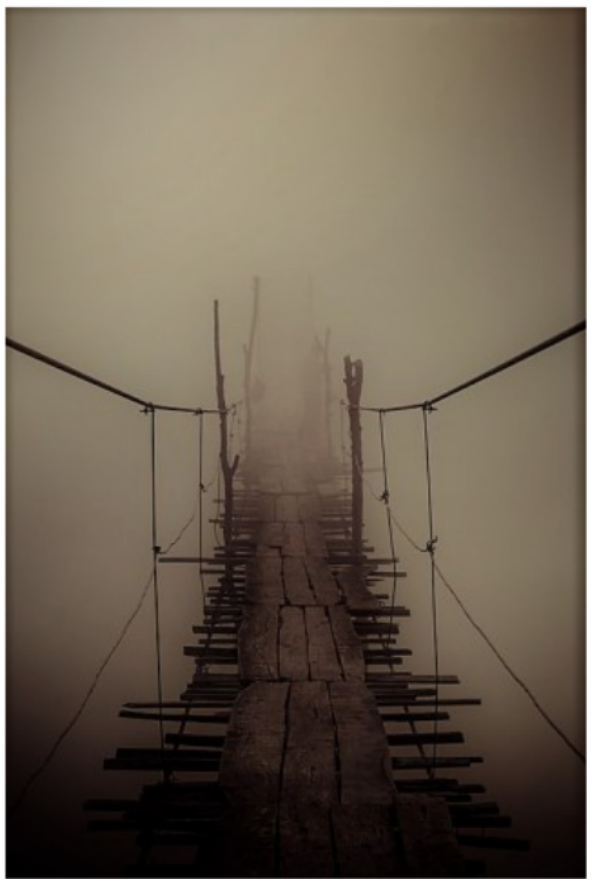
Top 10 Reasons to Major in Computing

1. Computing is part of everything we do!
2. Expertise in computing enables you to solve complex, challenging problems.
3. Computing enables you to make a positive difference in the world.
4. Computing offers many types of lucrative careers.
5. Computing jobs are here to stay, regardless of where you are located.
6. Expertise in computing helps you even if your primary career choice is something else.
7. Computing offers great opportunities for true creativity and innovativeness.
8. Computing has space for both collaborative work and individual effort.
9. Computing is an essential part of well rounded academic preparation.
10. Future opportunities in computing are without boundaries.

ACM Says–
https://www.acm.org/binaries/content/assets/education/top-10-reasons-to-major-in-computing.pdf

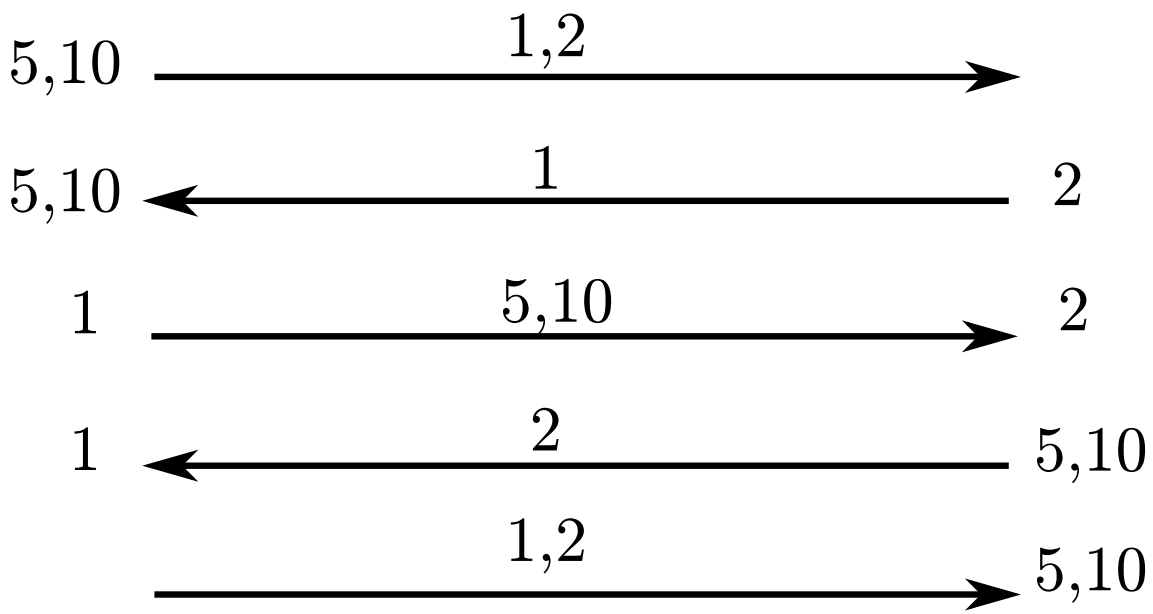
You would be accredited for your problem solving capability

According to a rumor on the Internet, interviewers at a well-known software company located near Seattle have given this problem to interviewees:



There are four people who want to cross a rickety bridge;
they all begin on the same side. You have 17 minutes to get them all across to the other side. It is night, and they have one flashlight. A maximum of two people can cross the bridge at one time. Any party that crosses, either one or two people, must have the flashlight with them. The flashlight must be walked back and forth; it cannot be thrown, for example. Person 1 takes 1 minute to cross the bridge, person 2 takes 2 minutes, person 3 takes 5 minutes, and person 4 takes 10 minutes. A pair must walk together at the rate of the slowerperson's pace.

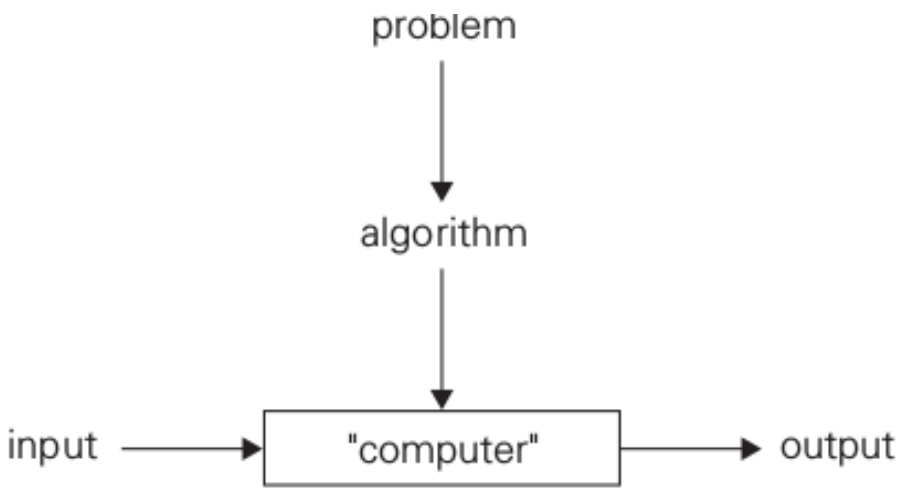
You would be accredited for your problem solving capability



Problem & Instance

An algorithm must solve a general, well-specified *problem*.

- An algorithmic problem is specified by describing the complete set of *instances* it must work on and of its output after running on one of these instances.



Example (Sorting Algorithm)

Problem: Sorting
Input: A sequence of n keys $a_1, ..., a_n$.
Output: The permutation (reordering) of the input sequence such that $a'_1 \leq a'_2 \leq \ \leq a'_{n1} \leq a'_n$.

An instance of sorting might be an array of names, like $\{Mike, Bob, Sally, Jill, Jan\}$, or a list of numbers like $\{154, 245, 568, 324, 654, 324\}$.

Methods of Specifying an Algorithm

- Flowchart: Dominant in earlier days of computing
- ✓ Natural language: Must be careful about ambiguity
- ✓ Pseudo codes: Mixture of a natural language and programming language

What about computer languages?

In our main textbook, pseudo codes are C++ like with some modifications to make them easier to read. For example:

`temp=x; x=y; y=temp;` → `exchane x and y`

`if (low<=x && x<=high)` → `if (low` $\leq$ `x` $\leq$ `high)`

new types:

- `index`: for index variables
- `number`: int or float
- `keytype`: data belongs to an ordered set

Algorithm Design Techniques

An algorithm design technique (or “strategy" or “paradigm") is **a general approach** to solving problems algorithmically that is **applicable to a variety of problems** from different areas of computing.

Divide & conquer
Dynamic Programming
Greedy **Backtracking**
Reduction
Branch & bound

- Learning such techniques is akin to

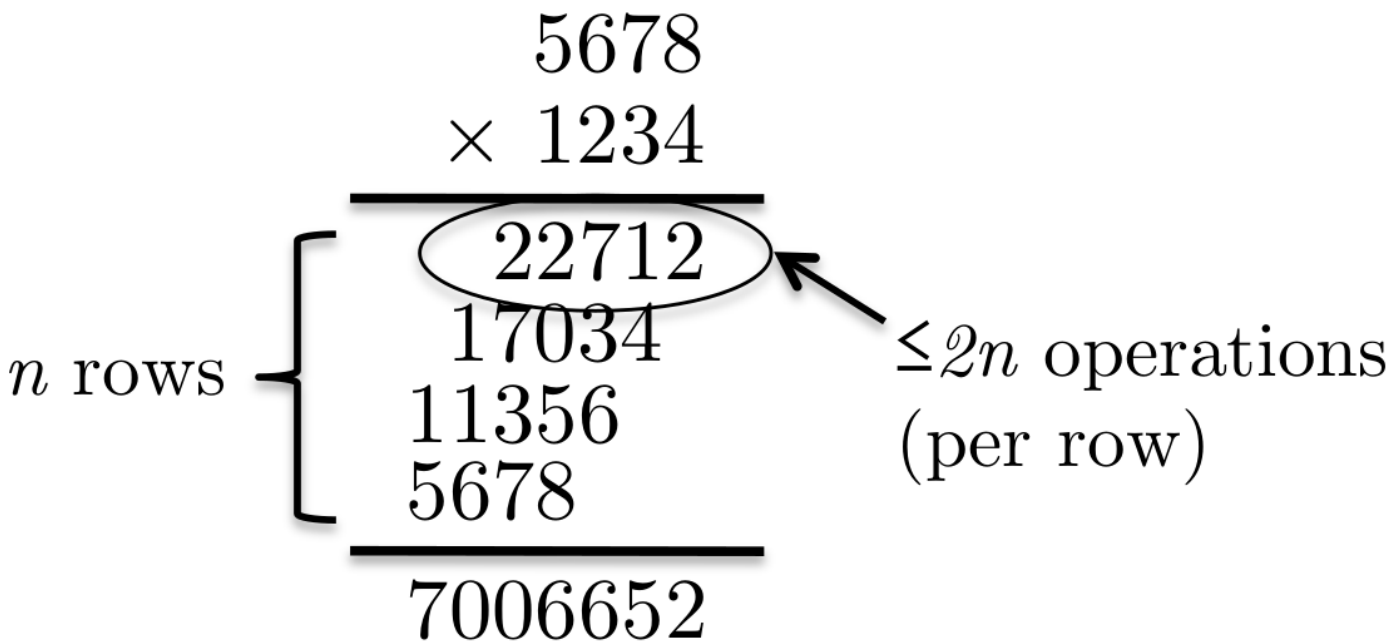
$$learning\ to\ fish\ as\ opposed\ to\ being\ given\ a\ fish$$

- It is not true, of course, that each of these general techniques will be necessarily **applicable to every problem** you may encounter.

Algorithm Analysis

Complexity
Correctness

Integer Multiplication



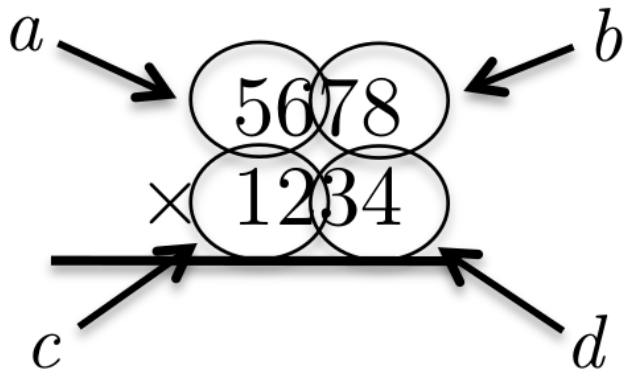
total number of operations $\leq$ constant.n^2

Integer Multiplication

Can We Do Better?

Integer Multiplication

Karatsuba Multiplication



Step1: Compute $a \cdot c = 56 \cdot 12 = 672$

Step2: Compute $b \cdot d = 78 \cdot 34 = 2652$.

Step3: Compute$(a + b) \cdot (c + d) = 134 \cdot 46 = 6164$.

Step4: calculate “Step3 - Step1 - step2"; $6164 - 672 - 2652 = 2840$.

Step5: Compute

$10^4 \cdot 672 + 10^2 \cdot 2840 + 2652 = 6720000 + 284000 + 2652 = 70066552$

Integer Multiplication

Is this procedure correct?

Integer Multiplication

$$
\begin{aligned}
X \cdot Y &= (a \cdot 10^{n/2} + b) \cdot (c \cdot 10^{n/2} + d) \\
&= a \cdot c \cdot 10^{n} + \underbrace{(a \cdot d + c \cdot b)}_{(a+b)\cdot(c+d)-a\cdot c-b\cdot d} \cdot 10^{n/2} + b \cdot d
\end{aligned}
$$

Karatsuba

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

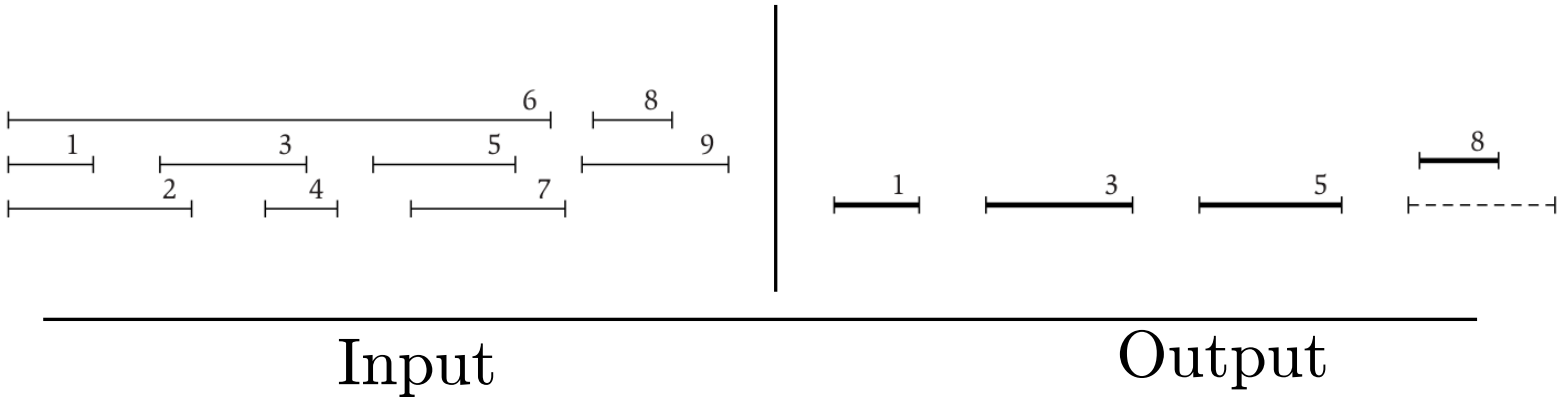
---

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

Integer Multiplication

Is Karatsuba algorithm less complex?

A superstar movie player problem



Input: A set I of n intervals on the line.
Output: What is the largest subset of mutually non-overlapping intervals which can be selected from I?

What is the OPTIMUM ALGORITHM to solve this problem?

Earliest Job First?

Shortest Job First?

Earliest-Finish Job First?

Exhaustive Scheduling?

What is the OPTIMUM ALGORITHM to solve this problem?

Earliest Job First?

Shortest Job First?

Earliest-Finish Job First? ☑ But we have to **PROVE**

Exhaustive Scheduling?

Text book

- No required textbook.
- However, you need to solve many problems, and many good books are available to *challenge yourself* with their examples and problems:
  1. Richard Neapolitan, Foundations Of Algorithms, Jones & Bartlett Learning, 4th /5th edition, 2009/2014
  2. Anany Levitin, Introduction to the Design and Analysis of Algorithms, Pearson, 3rd edition , 2011
  3. Jon Kleinberg, Algorithm Design, Pearson, 1 edition, 2005
  4. Steven S Skiena, The Algorithm Design Manual, Springer, 2nd edition, 2010
  5. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, Introduction to Algorithms, MIT Press, 3rd/4th edition, 2010/2022
  6. T. Roughgarden, The Algorithms Illuminated Book Series, Part 1-4, 2017-2020

Grading Policy

- Homeworks: 20%
- In class questions: 15%
- Midterm: 30%
- Final: 35%
- Presence: 5%