

CS 7200: Algorithm Design and Analysis

T. K. Prasad

Department of Computer Science and Engineering
Wright State University

<http://cecs.wright.edu/~tkprasad>

Algorithms: What, Why, How?

Algorithm: A step-by-step procedure for calculation.

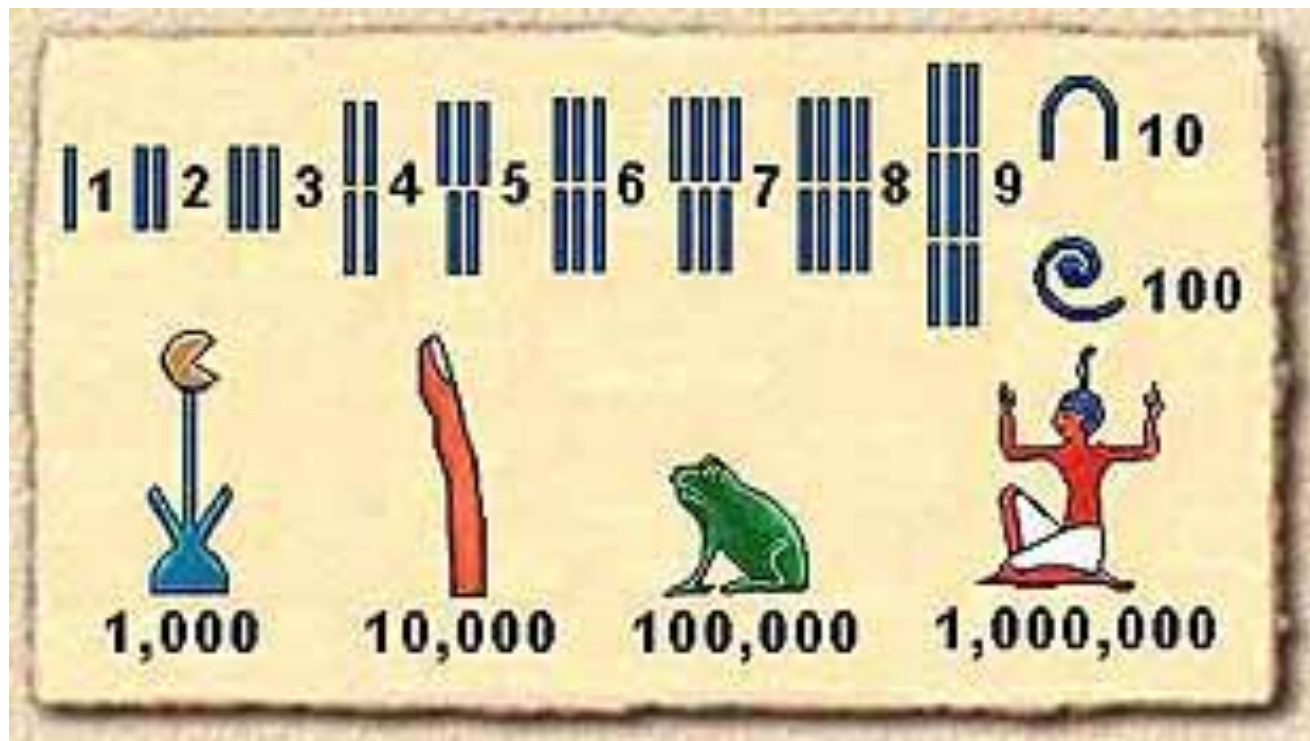
Word Origin: Named after the Persian, Islamic mathematician, astronomer, and geographer, Abu Abdallah Muhammad ibn Musa al-Khwarizmi (c. 780-850 AD).

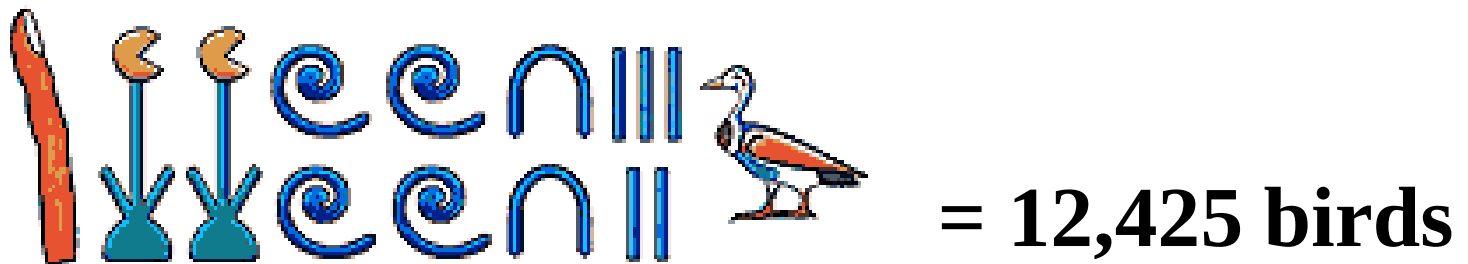
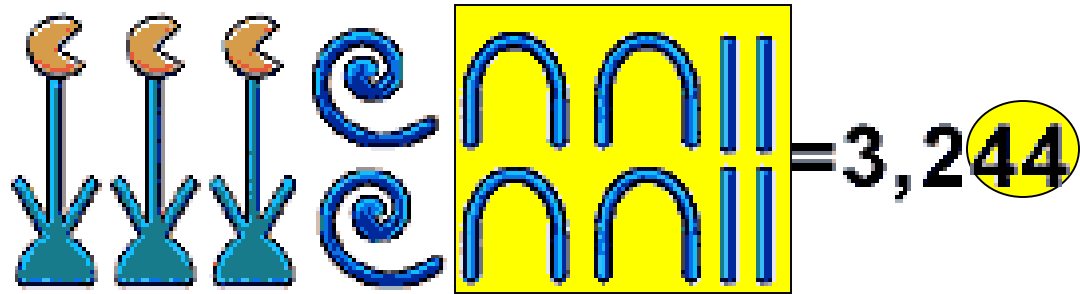
History: Originally used for performing arithmetic using Hindu-Arabic numerals and solving linear and quadratic equations.

Modern study of algorithm began with formalization of mechanizable/effective computations.

Symbols for numbers : Egyptian Numerals

Hieroglyphs : 3500 BC





Symbols for numbers : Roman Numerals



Symbols for numbers : Roman Numerals

- Alphabet: { I, V, X, L, C, D, M, ... }

- Numerals:

I, II, III, IV, V, VI, VII, VIII, IX, X

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

XI, XII, XIII, XIV, XV, XVI, XVII, XVIII, XIX, XX

11, 12, 13, 14, 15, 16, 17, 18, 19, 20

L, C, D, M, LXIII, XL, MCMLXIX

50, 100, 500, 1000, 63, 40, 1969

(cont' d)

- **XIX**
 - Is this $11 + 10 = 21$? Or, is this $10 + 9 = 19$?
 - **19**
- If smaller digit follows larger digit, the digits are added. If the smaller digit precedes larger, the smaller is subtracted from the larger.
- A symbol representing 10^x may not precede any symbol larger than 10^{x+1} . For example,
 - C cannot be preceded by I or V, but only by X (or, by a symbol equal to or larger than C).
 - The number “ninety-nine” is XCIX, not IC or ILL.

Summary: Two Lessons

- *Fixed Alphabet for numerals*: Use a fixed set of symbols to represent all numbers.
- *Unambiguous representation for number*: Each numeral must have a unique meaning (that is, a unique number associated with it).

Symbols for numbers: Hindu Arabic System

1	1	1 .	1	1 . .
2	11	11	1 1	1 . 1
3	12	12	1 2	1 . 2
4	13	13	1 3	1 . 3
5	14	14	1 4	1 . 4
6	15	15	1 5	1 . 5
7	16	16	1 6	1 . 6
8	17	17	1 7	1 . 7
9	18	18	1 8	1 . 8
1	19	19	1 9	1 . 9
	2	2 .	1 1	1 1 .

Read . as blank.

Need care
when writing
(aligning)
numbers
with blanks.

Invention of *Zero*

Attributed to Indian Mathematicians.

In fact, *zero* thrust itself upon us in the context of positional number system!

- 1,2,3,4,5,6,7,8,9
- 1 ,11,12,...
- ..
- 1 , 1 1, 1 2, ...

- 0,1,2,3,4,5,6,7,8,9
- 10,11,12,...
- ..
- 100, 101, 102, ...

Positions and Weights

TEN- THOUSANDS	THOUSANDS	HUNDREDS	TENS	UNITS
-------------------	-----------	----------	------	-------

43210

=

$$\begin{aligned} & 4 * 10,000 \\ & + 3 * 1,000 \\ & + 2 * 100 \\ & + 1 * 10 \\ & + 0 \end{aligned}$$

Algorithms: What, Why, How?

Example 1: `Mystery(int m, n) {
 $m := n - m;$ $n := n - m;$ $m := m + n;$ }`

Example 2: Euclid's algorithm for determining **Greatest Common Divisor** or **Highest Common Factor** for two non-negative integers.

Example?: Recipes and directions for making coffee, pasta, cakes, curries, etc.

Example?: Summarizing a story.

Example-Not: **Laugh; Feel Good;**
Determine whether the two given programs compute the same function?

Algorithms: What, Why, How?

Normally focus on techniques for *Efficient computation* in terms of time and space.

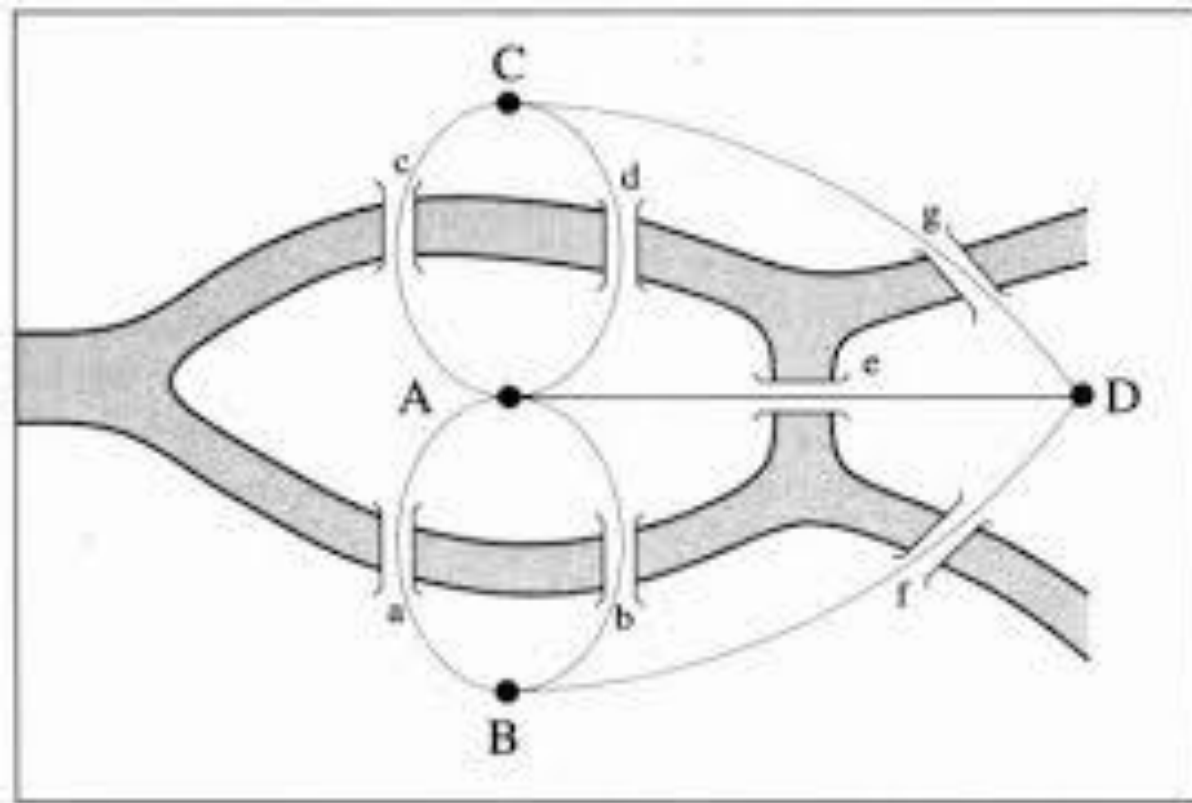
Ironically, even though *Efficiency* is “less important” than orthogonal features such as correctness, ease of use, cost, and security.

in practice, *efficiency* (is like a currency that) can be traded with additional desirable features to maintain acceptable performance.

Algorithms: What, Why, How?

Real World Problem

E.g., Königsberg problem



Algorithms: What, Why, How?

Real World Problem

E.g., Königsberg problem

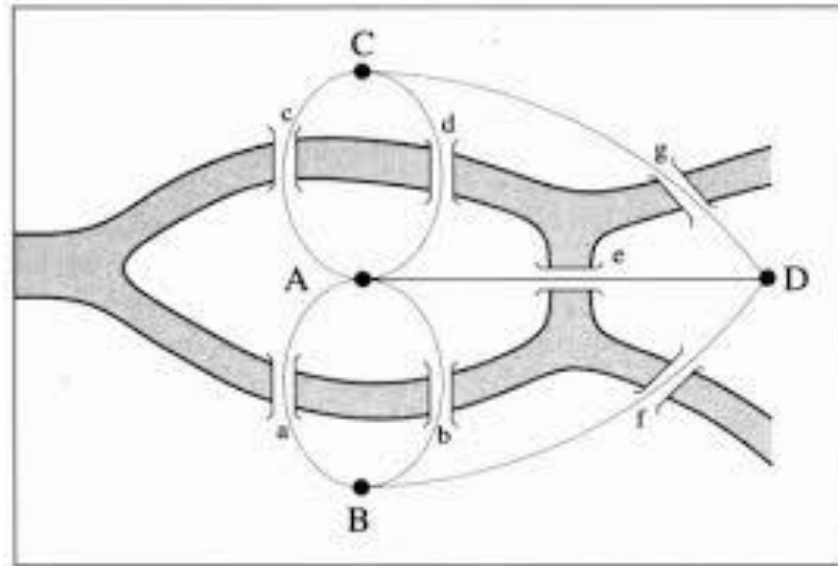
*Abstract via Mathematical Formulation
(Declarative Specification of the Solution)*

Does a solution always exist?

Sufficient Condition for its existence

*E.g., Graph-theoretic formulation
and Euler's characterization*

Königsberg Bridge problem



Every bridge cannot be crossed once and only once because there are four land masses with odd number of bridges connecting them.

(For an "Eulerian" path to exist, it is necessary and sufficient to have zero or two nodes of odd degree in the corresponding graph.
- [LEONHARD EULER, 1735])

Algorithms: What, Why, How?

Efficient Problem Solving Strategy

E.g., $2^8 = (2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2)$

E.g., Recursive doubling $2^8 = ((2^2)^2)^2$

Detailed description of the solution steps

Algorithms: What, Why, How?

Design: Naive exponential algorithms (e.g., generate and test approaches based on declarative specification) to efficient polynomial-time algorithms (using various techniques such as divide and conquer (recursion), and dynamic programming)

Analysis: Time and space complexity

Algorithms: What, Why, How?

Proof of Correctness: Using Mathematical

Induction, Contradiction, Loop Invariants, etc.

Theory: Complexity classes, Lower bounds,
Reduction.

Practice:

Overcome intractability via approximation;

Improve performance through randomization

Recent Personal Experience

Cory A. Henson, Krishnaprasad Thirunarayan, Amit P. Sheth: An Efficient Bit Vector Approach to Semantics-Based Machine Perception in Resource-Constrained Devices. International Semantic Web Conference (1) 2012: 149-164

ISSUE: Explanation and Discrimination operators for machine perception have clear declarative specification in logic. But general purpose reasoner fails to run to completion even for small problems on resource-constrained mobile devices.

Recent Personal Experience : Asymptotic Improvement Matters

Proposed bit vector encodings and algorithms that yield necessary and significant computational enhancements - including asymptotic order of magnitude improvement from cubic ($O(n^3)$) to linear ($O(n)$), with running times reduced from minutes to milliseconds, and problem size increased from 10's to 1000's.

Proof of correctness crucial for justifying that the “simple” algorithm works.

Recent Personal Experience : Even Constant Speedup Matters

Pramod Anantharam's work on Modelling and Analyzing
Bay Area Road Network Traffic

Total processing time for one year data for each road
link ~ 40+ min

Using 2.66 GHz, Intel Core 2 Duo with 8 GB memory

For 2534 links ~ 2 months

Using Apache Spark cluster with 864 cores and 17TB
memory

For 2534 links < 1 day

Algorithmic Design Paradigms

Design and analysis of computer algorithms.

-  Greedy.
-  Divide-and-conquer.
-  Dynamic programming.
-  Network flow.
-  Randomized algorithms.
-  Intractability.
-  Coping with intractability.

Critical thinking and problem-solving.

Applications

Wide range of applications.

- n Caching.
- n Compilers.
- n Databases.
- n Scheduling.
- n Networking.
- n Big Data analysis : Sensor, Social, Medical, ...
- n Signal processing.
- n Computer graphics.
- n Scientific computing.
- n Operations research.
- n Artificial intelligence.
- n Computational biology.
- n . . .

We focus on algorithms and techniques that are **useful in practice**.