Problems, Algorithms, Complexity, Cost

Problem

- Problem: Specification of the Relationship between a given Input and a desired Output

- Numerical problems: In- and Output are numbers

- Examples
  - Compute the discrete Fourier transform of a given vector $x$ of length $n$
  - Compute the product of two given matrices of compatible size
  - Compress an $n \times n$ image with a ratio ...
  - Sort a given list of integers
  - Multiply by 5, $y = 5x$, using only additions and shifts
  - Prepare a cheeseburger
Algorithm

- Algorithm: A precise description of a sequence of steps to solve a given problem.

- Examples:
  - Cooley-Tukey fast Fourier transform
  - A description of mat-mat multiplication by definition
  - JPEG encoding
  - Mergesort
  - $y = x << 2 + x$
  - Algorithms for “food problems:” www.epicurious.com

For writing/publications:
When you state an algorithm, start always with “Input: <description of input including all conditions>. Output: <description of output>.” This specifies which problem it solves.
Origin of the Word “Algorithm”

- Mathematician, astronomer and geographer; founder of Algebra (his book: Al'Jabr wa'al'Muqabilah)

- Khowârizm is today the small Soviet city of Khiva

- Earlier word Algorism: The process of doing arithmetic using Arabic numerals

- Algorithm: since 1957 in Webster Dictionary

Source:

Abu Ja'far Mohammed ibn Mûsâ al'Khowârizmî (c. 825)
Standard Analysis of Algorithms & Problems

- Analysis of Algorithms for
  - Runtime
  - Memory requirement (memory footprint)

- Runtime analysis of an algorithm:
  - Count “elementary” steps (e.g., floating point operations) dependent, typically, on the input size n
  - State result in asymptotic O-notation

- Runtime complexity of a problem = Minimum of the runtimes of all possible algorithms
  - Result also stated in asymptotic O-notation

Note: complexity is a property of a problem, not of an algorithm
Asymptotic Notation

- Goal: capture the asymptotic growth of function over $\mathbb{N}$ (or $\mathbb{R}$)
- Definition of $O$ ("upper bound")

$$O(g(n)) = \{f(n) \mid \text{there is a constant } c > 0, \ n_0 \in \mathbb{N} \text{ such that } 0 \leq f(n) \leq c g(n) \text{ for } n \geq n_0\}$$

- Usually written as (abuse of notation):

$$f(n) = O(g(n)) \text{ instead of } f(n) \in O(g(n))$$

- Give examples (blackboard)
Asymptotic Notation (contd.)

- Definition of $\Omega$ (“lower bound”)

$$\Omega(g(n)) = \{ f(n) \mid \text{there is a constant } c > 0, \ n_0 \in \mathbb{N} \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for } n \geq n_0 \}$$

- Give examples (blackboard)

- Definition of $\Theta$ (“exact asymptotic class”)

$$\Theta(g(n)) = \{ f(n) \mid \text{there are constants } c_1, c_2 > 0, \ n_0 \in \mathbb{N} \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for } n \geq n_0 \}$$

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

- Give examples (blackboard)
Other Examples and Pitfalls (Blackboard)

- General Properties
- Abuse: Computing with $O$ notation
Asymptotic Runtime Analysis of Divide-and-Conquer Algorithms

Recurrence

\[ T(n) = aT(n/b) + f(n), \quad a \geq 1, b > 1 \]

subproblem size = \( n/b \)

\( a \) subproblems

cost of conquer step = \( f(n) \)

Solution

\[
T(n) = \begin{cases} 
\Theta(n^{\log_b a}), & f(n) = O(n^{\log_b a - \varepsilon}), \text{ for some } \varepsilon > 0 \\
\Theta(n^{\log_b a} \log(n)), & f(n) = \Theta(n^{\log_b a}) \\
\Theta(f(n)), & f(n) = \Omega(n^{\log_b a + \varepsilon}), \text{ for some } \varepsilon > 0 
\end{cases}
\]

Stays valid if \( n/b \) is replaced by its floor or ceiling

Yeah, we need to look at some examples (blackboard): mat-mat-mult, sorting, searching in sorted list, polynomial mult.