Design and Analysis of Algorithm

M. Sakthi Balan

Department of Computer Science & Engineering
LNMIIT, Jaipur

Contents

1 Algorithms

2 Analysis of Algorithms

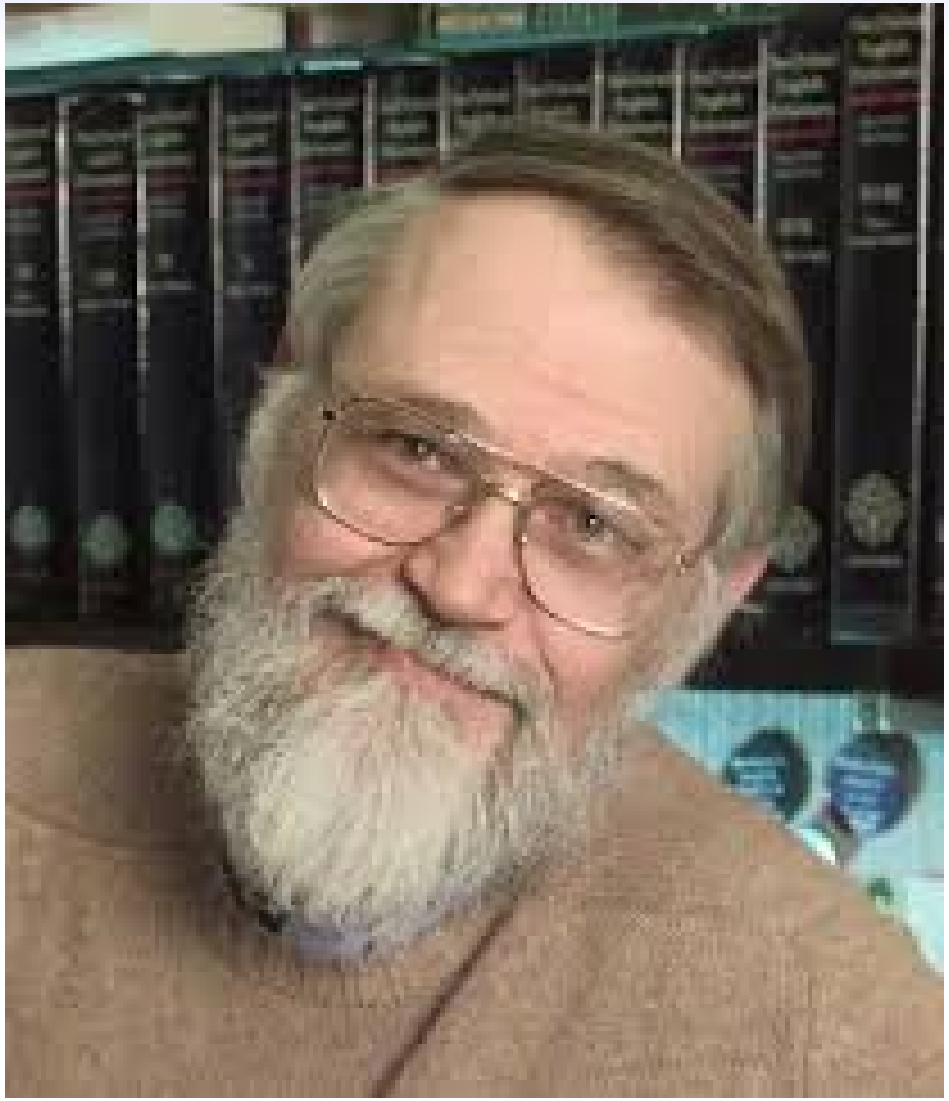
LNMIIT

LNMIIT

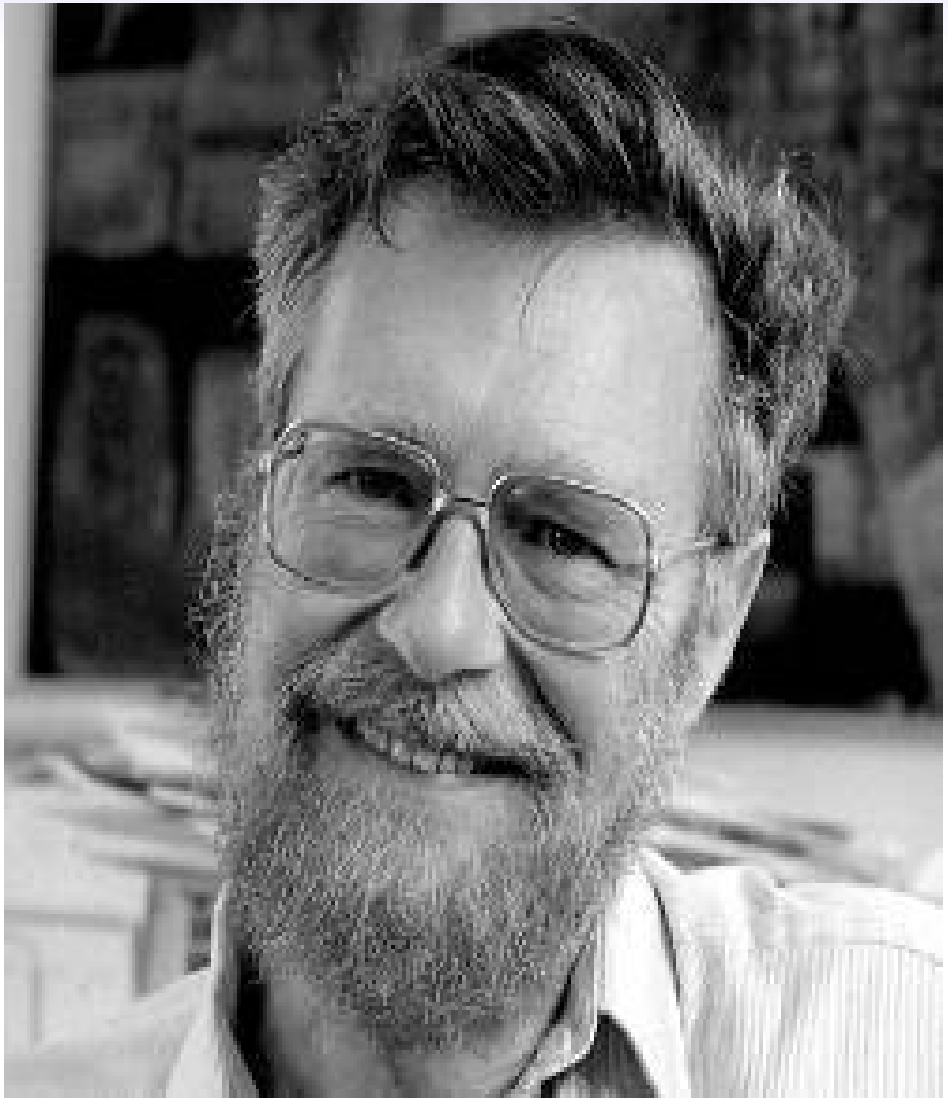
LNMIIT

LNMIIT

LNMIIT



LNMIIT



LNMIIT

LNMIIT

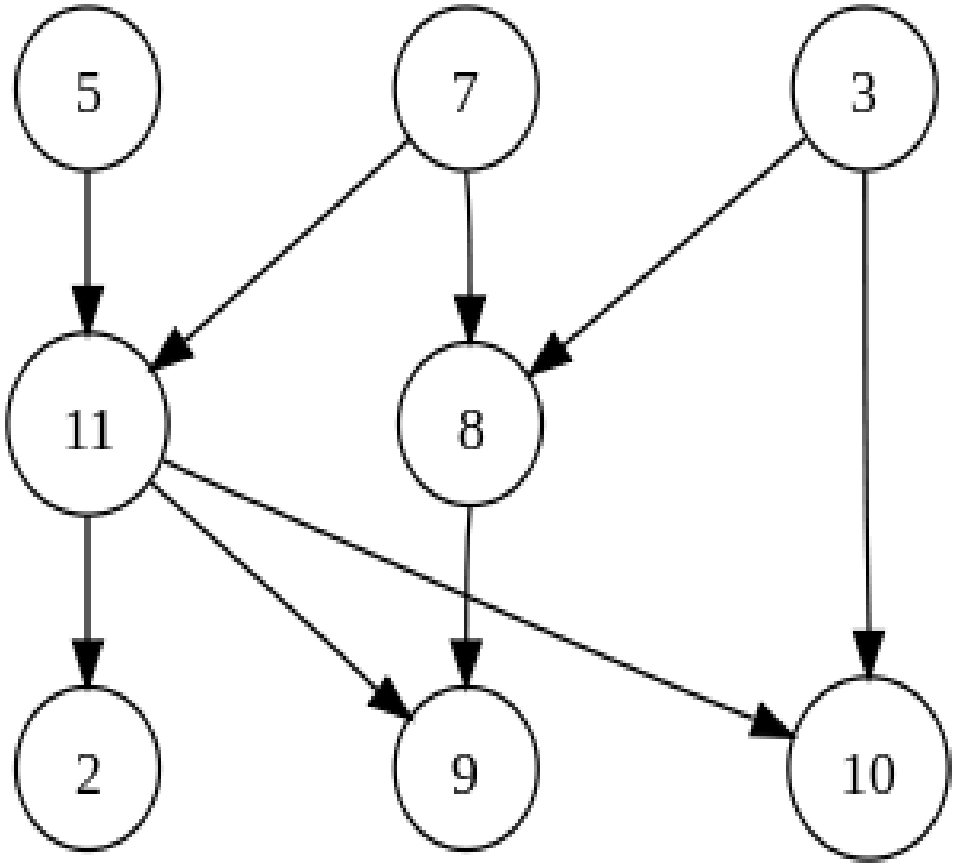
LNMIIT

Knuth's dictum:

Computer Science is algorithms. That's it. Not just theory. All. Everything. If there is not an algorithm lurking around, then it is not Computer Science.

Example: Train Travel

Example: Train Travel



LNMIIT

Example: Train Travel

Possible questions we can ask:

1. Is cities A and B are connected either directly or indirectly?
2. Can we travel from city A to city B and on the way visiting the cities $\{C, D, E\}$?
3. Can we find a route with minimum distance traveled for the above?
4. Can we find a route with minimum cost for the above?

Example: Train Travel

Possible variations / constraints we can have:

1. Can I travel only in the morning or only in the night?
2. Use only broad gauge lines
3. Travel only on specific days and only on some specific trains

Example: Train Travel

1. Representation of graph
2. Data structures to store the graph
3. How to manipulate the input to arrive at the result that we need

Example: Print Neatly

1. Put the line breaks in such a way that the distance between the words are almost equally spaced
2. Can we get a optimized solution for this?
3. What if some figures are to be put in the document. How to position the line breaks in an optimal way?

Dynamic Programming

Example: Print Neatly

1. Put the line breaks in such a way that the distance between the words are almost equally spaced
2. Can we get a optimized solution for this?
3. What if some figures are to be put in the document. How to position the line breaks in an optimal way?

Dynamic Programming

Example: Document Similarity

1. Given set of documents check how much similar they are:
   - Plagiarism
   - Checking the similarity in the different versions of the document
   - For classifying the documents

Example: Document Similarity

Edit Distance

1. How many changes to make to get one document from another?
2. Types of changes allowed: Add, remove or replace?
3. Find the way where there is minimum number of changes required

Recursive algorithm: But naive recursion is not efficient
Reason: Subproblems are solved again and again

LNMIIT

Example: Document Similarity

Edit Distance

1. How many changes to make to get one document from another?
2. Types of changes allowed: Add, remove or replace?
3. Find the way where there is minimum number of changes required

Recursive algorithm: But naive recursion is not efficient
Reason: Subproblems are solved again and again

LNMIIT

Example: Stable Matching Problem

Given a set of *n* men and a set of *n* women, find a "suitable" matching.

1. Participants rank members of opposite sex
2. Each man lists women in order of preference from best to worst
3. Each woman lists men in order of preference from best to worst

Example: Stable Matching Problem

	1st	2nd	3rd
Ram	Sita	Sharda	Parvati
Sharma	Sharda	Sita	Parvati
Gupta	Sita	Sharda	Parvati

	1st	2nd	3rd
Sita	Sharma	Ram	Gupta
Sharda	Ram	Sharma	Gupta
Parvati	Ram	Sharma	Gupta

Example: Stable Matching Problem

A matching S is a set of ordered pairs (m, w) with $m \in M$ and $w \in W$ such that

1. Each man $m \in M$ appears in at most one pair of S
2. Each woman $w \in W$ appears in at most one pair of S

A matching S is **perfect** if $|S| = |M| = |W| = n$

Example: Stable Matching Problem

A matching S is a set of ordered pairs (m, w) with $m \in M$ and $w \in W$ such that

1. Each man $m \in M$ appears in at most one pair of S
2. Each woman $w \in W$ appears in at most one pair of S

A matching S is **perfect** if $|S| = |M| = |W| = n$

Example: Stable Matching Problem

	1st	2nd	3rd
Ram	Sita	Sharda	Parvati
Sharma	Sharda	Sita	Parvati
Gupta	Sita	Sharda	Parvati

	1st	2nd	3rd
Sita	Sharma	Ram	Gupta
Sharda	Ram	Sharma	Gupta
Parvati	Ram	Sharma	Gupta

Example: Stable Matching Problem

	1st	2nd	3rd
Ram	Sita	Sharda	Parvati
Sharma	Sharda	Sita	Parvati
Gupta	Sita	Sharda	Parvati

	1st	2nd	3rd
Sita	Sharma	Ram	Gupta
Sharda	Ram	Sharma	Gupta
Parvati	Ram	Sharma	Gupta

LNMIIT

Example: Stable Matching Problem

A stable matching is a perfect matching with no unstable pairs.
Stable matching problem: Given the preference lists of n men and n women, find a stable matching (if one exists)

Example: Stable Matching Problem

	1st	2nd	3rd
Ram	Sita	Sharda	Parvati
Sharma	Sharda	Sita	Parvati
Gupta	Sita	Sharda	Parvati

	1st	2nd	3rd
Sita	Sharma	Ram	Gupta
Sharda	Ram	Sharma	Gupta
Parvati	Ram	Sharma	Gupta

LNMIIT

Algorithm

1. Named after *Al-Khwarizmi* a Persian Mathematician
2. On the Calculation with Hindu Numerals written in the year around 825
3. Spreads the Hindu-Arabic numeral system throughout the Middle East and Europe
4. Translated into Latin as *Algoritmi de numero Indorum*
5. *Algoritmi* is a latin form the name *Al-Khwarizmi* and that lead to the term *Algorithm*

Source: Taken from Wikipedia on Algorithms

LNMIIT

Algorithm

1. A sequence of well-defined computational steps that takes some input values and produces output
2. A tool for solving a computational problem

Properties of Algorithm

1. Finiteness – should end after finite number of steps
2. Definiteness – unambiguous
3. Effectiveness – feasible steps in feasible time
4. Input – zero(?) or more inputs
5. Output – at least one output

Problem Statement – Example

Sorting problem

Input: A sequence of n numbers $a_1, a_2, \ldots, a_n$.

Output: Find the reordering of the numbers $a'_1, a'_2, \ldots, a'_n$ such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

Instance and Correctness of a Problem

Instance

Given a problem statement for sorting, a specific input sequence like

$$25, 41, 12, 8, 0, 21, 78$$

is an instance of the sorting problem.

Correctness

An algorithm is said to be correct if for all input instances it halts with the correct output.

LNMIIT

Problem Statement

1. Write a problem statement for stable matching problem
2. Write a problem statement for Hamiltonian path problem

Analyzing an Algorithm

Predicting the resources:

- (Execution) Running Time
- (Space) Memory
- (Network) Communication bandwidth
- Energy

Types of Analysis

1. Priori Analysis – analysis done before implementing the algorithm in a machine
2. Posteriori analysis – empirical statistical data taken after executing the program implementing that algorithm in a machine

Efficiency of Algorithm

An algorithm is efficient if, when implemented, it runs quickly on real input instances

Efficiency of Algorithm

1. Bad algorithms may run *fast* on small instances
2. Good algorithm may run slow when it is coded in a sloppy way
3. Same algorithm run in different systems may take different execution time
4. Same algorithm run in even one system on different times may execute with different execution time
5. Some algorithms scale up very well but some not

LNMIIT

Efficiency of Algorithm

Efficiency should be

- Platform independent
- Instance independent
- Predictive value with respect to input size

Random-Access Machine Model

- Analysis should depend on the machine? – No
- Analysis should depend on the computing model? – No
- We need a generic machine where we can do our analysis – getting a level playing field
- RAM model is used – sequential way of executing statements and assuming for executing each statement the cost is a unit factor.

Input Size

- It refers to the number of items in the input.
- It depends on the problem
  - Sorting, FFT – number of items
  - Multiplication of two numbers, primality testing – number of bits needed to represent the number

Running Time

Worst-Case: Bound on the largest possible running time the algorithm could have over all inputs of a given size n.
Average-Case: Average over random instances

LNMIIT

Brute-Force Solution

- Exhaustive Search
- Trial and Error
- *Naive* Method

Efficiency of Algorithm (2)

An algorithm is efficient if it achieves qualitatively better worst-case performance, at an analytical level, than brute-force search.

Efficiency of Algorithm (2)

What do we mean by *qualitatively better worst-case*?

Order of Functions

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

Picture taken from book Algorithm Design by J. Kleinberg and E. Tardos

LNMIIT

Efficiency of Algorithm (3)

An algorithm is *efficient* if it has a polynomial running time

Asymptotic Order of Growth

- $\mathcal{O}$
- Ω
- Θ