Design and Analysis of Algorithms

Vandana M L
Department of Computer Science & Engineering

DESIGN AND ANALYSIS OF ALGORITHMS

Introduction to Algorithms, Design Techniques and Analysis

Slides courtesy of **Anany Levitin**

Vandana M L
Department of Computer Science & Engineering

Design and Analysis of Algorithms

Syllabus

- **UNIT I (12 Hours)**
  - Introduction
  - Analysis of Algorithm Efficiency,
  - Algebric structures
- **UNIT II (12 Hours)**
  - Brute Force,
  - Divide-and-Conquer
- **UNIT III (10 Hours)**
  - Decrease-and-Conquer
  - Transform-and-Conquer
- **UNIT IV (10 Hours)**
  - Space and Time Tradeoffs
  - Greedy Technique
- **UNIT V (12 Hours)**
  - Limitations of Algorithm Power
  - Coping with the Limitations of Algorithm Power
  - Dynamic Programming

Design and Analysis of Algorithms

Text Books

Book Type	Code	Title & Author	Publication Information		
			Edition	Publisher	Year
Text Book	T1	Introduction to The Design and Analysis of Algorithms Anany Levitin	2	Pearson	2012
Reference Book	R1	Introduction to Algorithms Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein	3	Prentice-Hall India	2009
Reference Book	R2	Fundamentals of Computer Algorithms Horowitz, Sahni, Rajasekaran,	2	Universities Press	2007
Reference Book	R3	Algorithm Design Jon Kleinberg, Eva Tardos,	1	Pearson Education	2006

Algorithm

What is an algorithm?

An algorithm is a sequence of unambiguous instructions for solving a problem,

i.e., for obtaining a required output for any legitimate input in a finite amount of time

Important Points about Algorithms

- The non-ambiguity requirement for each step of an algorithm cannot be compromised
- The range of inputs for which an algorithm works has to be specified carefully.
- The same algorithm can be implemented in several different ways
- There may exist several algorithms for solving the same problem.

Characteristics of Algorithm

Input: Zero or more quantities are externally supplied

Definiteness: Each instruction is clear and unambiguous

Finiteness: The algorithm terminates in a finite number of steps.

Effectiveness: Each instruction must be primitive and feasible

Output: At least one quantity is produced

Algorithm

Why do we need Algorithms?

- It is a tool for solving well-specified Computational Problem.
- Problem statement specifies in general terms relation between input and output
- Algorithm describes computational procedure for achieving input/output relationship
  This Procedure is irrespective of implementation details

Why do we need to study algorithms?

Exposure to different algorithms for solving various problems helps develop skills to design algorithms for the problems for which there are no published algorithms to solve it

Design and Analysis of Algorithms

Basic Issues related to Algorithms

- How to design algorithms
- How to express algorithms
- Proving correctness of designed algorithm
- Efficiency
  - Theoretical analysis
  - Empirical analysis

Design and Analysis of Algorithms

Basic Issues related to Algorithms

- How to design algorithms
- How to express algorithms
- Proving correctness of designed algorithm
- Efficiency
  - Theoretical analysis
  - Empirical analysis

What do you mean by Algorithm Design Techniques?

General Approach to solving problems algorithmically .
Applicable to a variety of problems from different areas of computing

Various Algorithm Design Techniques

- Brute Force
- Divide and Conquer
- Decrease and Conquer
- Transform and Conquer
- Dynamic Programming
- Greedy Technique
- Branch and Bound
- Backtracking

Importance Framework for designing and analyzing algorithms for new problems

Design and Analysis of Algorithms

Basic Issues related to Algorithms

- How to design algorithms
- How to express algorithms
- Proving correctness of designed algorithm
- Efficiency
  - Theoretical analysis
  - Empirical analysis

Methods of Specifying an Algorithm

- **Natural language**
  - Ambiguous
- **Pseudocode**
  - A mixture of a natural language and programming language-like structures
  - Precise and succinct.
  - Pseudocode in this course
    - omits declarations of variables
    - use indentation to show the scope of such statements as for, if, and while.
    - use ← for assignment
- **Flowchart**
  - Method of expressing algorithm by collection of connected geometric shapes

Euclid's Algorithm

Problem: Find gcd(m,n), the greatest common divisor of two nonnegative, not both zero integers m and n

Examples: gcd(60,24) = 12, gcd(60,0) = 60, gcd(0,0) = ?

Euclid's algorithm is based on repeated application of equality

gcd(m,n) = gcd(n, m mod n)

until the second number becomes 0, which makes the problem trivial.

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

Methods of Specifying an Algorithm

Two descriptions of Euclid's algorithm

Euclid's algorithm for computing gcd(m,n)

Step 1 If n = 0, return m and stop; otherwise go to Step 2

Step 2 Divide m by n and assign the value of the remainder to r

Step 3 Assign the value of n to m and the value of r to n. Go to step 1.

ALGORITHM Euclid(m,n)

//computes gcd(m,n) by Euclid's method

//Input: Two nonnegative,not both zero integers

//Output:Greatest common divisor of m and n

while n ≠ 0 do

    r ← m mod n

    m← n

    n ← r

return m

Algorithm for Sequential Search

ALGORITHM *SequentialSearch*$(A[0..n-1], K)$

```
//Searches for a given value in a given array by sequential search
//Input: An array A[0..n − 1] and a search key K
//Output: The index of the first element of A that matches K
//        or −1 if there are no matching elements
i ← 0
while i < n and A[i] ≠ K do
    i ← i + 1
if i < n return i
else return −1
```

Design and Analysis of Algorithms

Basic Issues related to Algorithms

- How to design algorithms
- How to express algorithms
- Proving correctness of designed algorithm
- Efficiency
  - Theoretical analysis
  - Empirical analysis

Design and Analysis of Algorithms

Basic Issues related to Algorithms

- How to design algorithms
- How to express algorithms
- Proving correctness of designed algorithm
- Efficiency
  - Theoretical analysis
  - Empirical analysis

Need of Analysis

- To determine resource consumption
  - CPU time
  - Memory space
- Compare different methods for solving the same problem before actually implementing them and running the programs.
- To find an efficient algorithm for solving the problem

Complexity of an algorithm

- A measure of the performance of an algorithm
- An algorithm's performance is characterized by
  - Time complexity

    How fast an algorithm maps input to output as a function of input
  - Space complexity

    amount of memory units required by the algorithm in addition to the memory needed for its input and output

PES UNIVERSITY ONLINE

How to determine complexity of an algorithm?

- Experimental study(Performance Measurement)
- Theoretical Analysis (Performance Analysis)

Limitations of Performance Measurement

- It is necessary to implement the algorithm, which may be difficult
- Results may not be indicative of the running time on other inputs not included in the experiment.
- In order to compare two algorithms, the same hardware and software environments must be used
- Experimental data though important is not sufficient

Performance Analysis

- Uses a high-level description of the algorithm instead of an implementation
- Characterizes running time as a function of the input size, n.
- Takes into account all possible inputs
- Allows us to evaluate the speed of an algorithm independent of the hardware/software environment

THANK YOU

Vandana M L
Department of Computer Science & Engineering

vandanamd@pes.edu