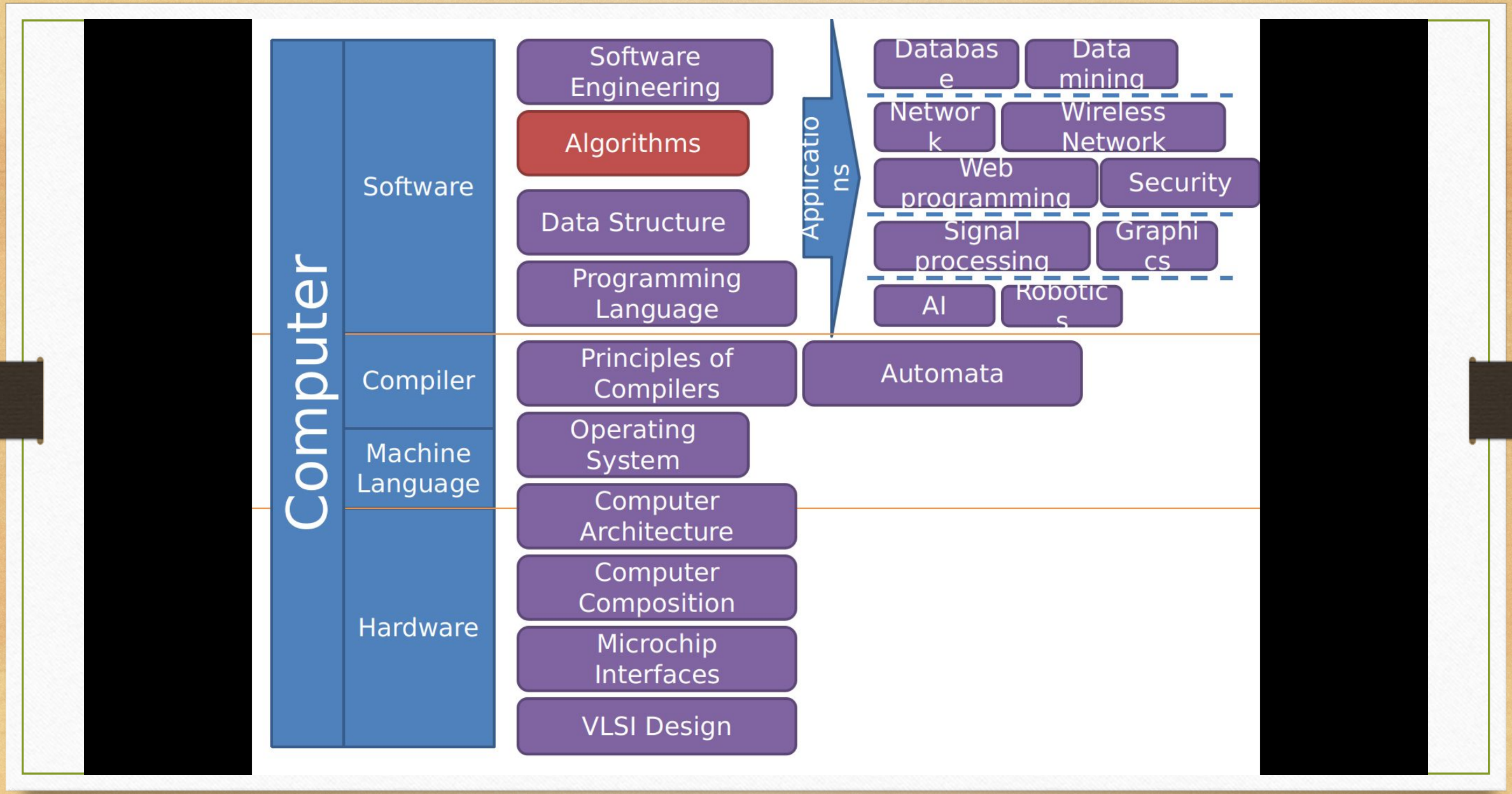
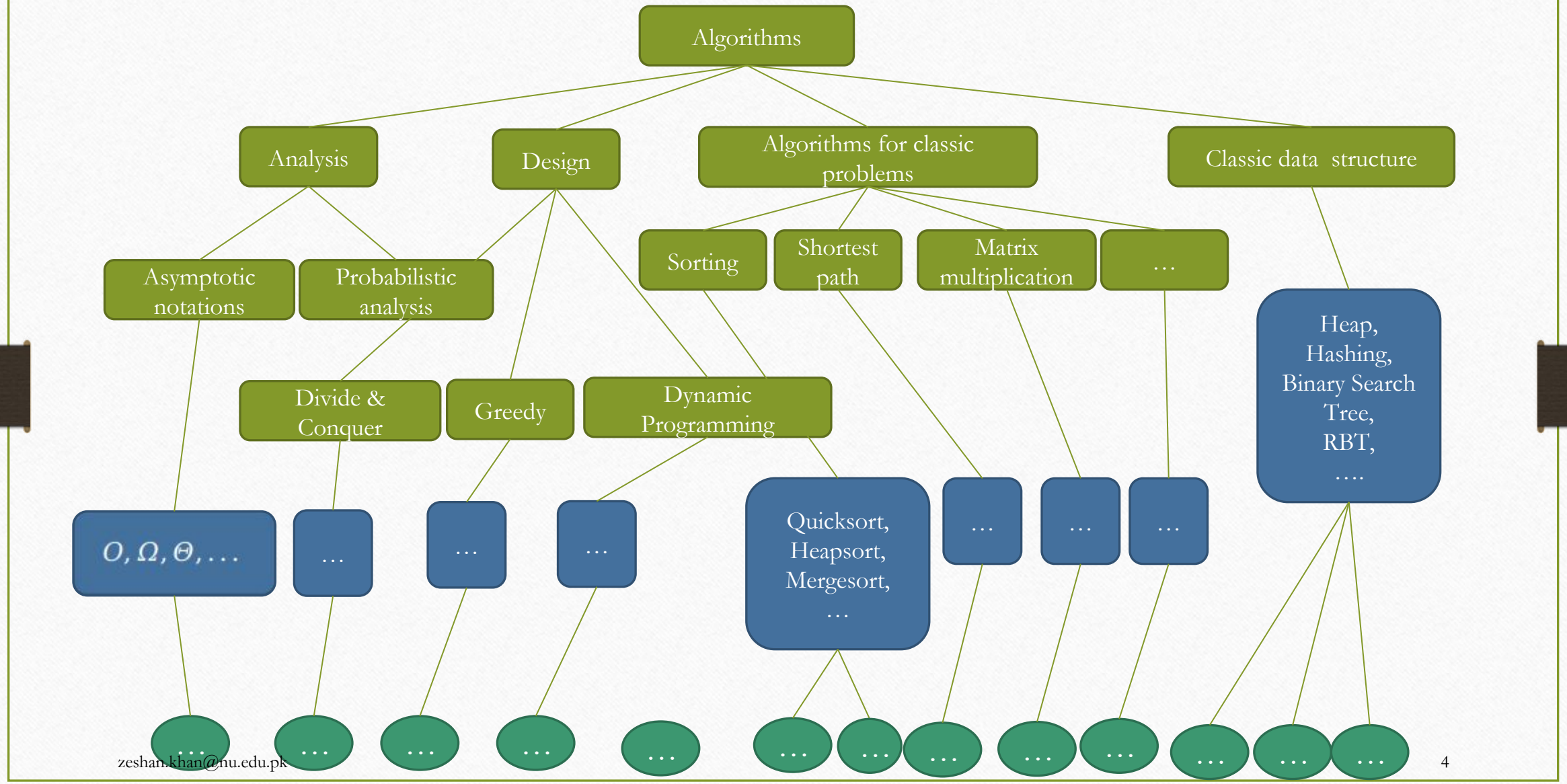
Design and Analysis of Algorithms

Introduction

Algorithm

- An ***algorithm*** is a (finite) sequence of **unambiguous** instructions for solving a problem.
- It has been the algorithm that has made possible the modern world
  - By David Berlinski, The Advent of the Algorithm, 2000
- 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. [David Harel]


Computer
Software
Compiler
Machine Language
Hardware
Software Engineering
Algorithms
Data Structure
Programming Language
Principles of Compilers
Operating System
Computer Architecture
Computer Composition
Microchip Interfaces
VLSI Design
Applications
Database
Data mining
Network
Wireless Network
Web programming
Security
Signal processing
Graphics
AI
Robotics
Automata


Algorithms
Analysis
Design
Algorithms for classic problems
Classic data structure
Asymptotic notations
Probabilistic analysis
Sorting
Shortest path
Matrix multiplication
…
Heap, Hashing, Binary Search Tree, RBT, ….
Divide & Conquer
Greedy
Dynamic Programming
$O, \Omega, \Theta, \ldots$
Quicksort, Heapsort, Mergesort, …

CS302

- Design and Analysis of Algorithms
- Pre: Data Structures

Assignments / Quizzes	15
Mid-Terms	25 (12.5 each)
Project	10
Final	50

CLOs

1. Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms
2. Analyse the time and space complexity of different algorithms by using standard analysis techniques for recursive and non-recursive algorithms.
3. Discussion on Asymptotic notations, standard complexity classes and representation of time complexities in asymptotic notations of standard complexity functions
4. Describe, compare, analyse, and solve general algorithmic problem types: Sorting, Searching, String Processing, Graph.
5. Implement the algorithms, compare the implementations empirically, and apply fundamental algorithms knowledge to solve real-world problems.
6. Understanding of NP-Completeness and Approximate Problems.

Contents

- Basics of Algorithms, Mathematical Foundation, Growth of Function, Asymptotic Notations.
- Divide and Conquer, Substitution Method, Recurrence-Tree Method, Master's Method.
- Sorting (Merge, Insertion, Quick, Heap, Counting, Radix),Data Structures (Stack, Queue, Linked List, Hash Table, Binary Tree).
- Dynamic Programming
- Greedy Algorithms, Graph Theory (Graph Categorization, Graph Terminology, Representation of Graphs, BFS & DFS, Strongly Connected Components, Greedy Algorithms: Kruskal's Algorithm, Prim's Algorithms, Bellman-Ford Algorithms, Dijkstra's Algorithm)
- Geometric Algorithms (Introduction, Graham Scan, Close Points). String Matching
- NP Complete Problems and Solutions using Approximation Algorithm

History of Algorithm

- GCD by the Euclid (Ancient era)
- Multiplication Arabic Numerals by Al-Khwarizmi (Muslim Golden era)
- Robert of Chester (12^{th} Century)
- Multiplication Arabic Numerals by Fibnacci (12^{th} Century)

Algorithm Analysis

- What is an algorithm?
- What are we interested in an algorithm?
- How to measure an algorithm?
- How to code divide-and-conquer algorithm?
  - Recursion
- How to calculate the running time of divide-and-conquer algorithm?
  - Recurrence equation

What is an algorithm?

- "a sequence of operations" (informal)
- E.g.
  - The algorithm to walk
  - The algorithm to cook instant noodle
  - The algorithm to sort N integers

What is an algorithm?

- Algorithm: walk to a destination

```
while (have not arrived at the destination)
{
    put the back foot in front of the front foot;
}
```

What is an algorithm?

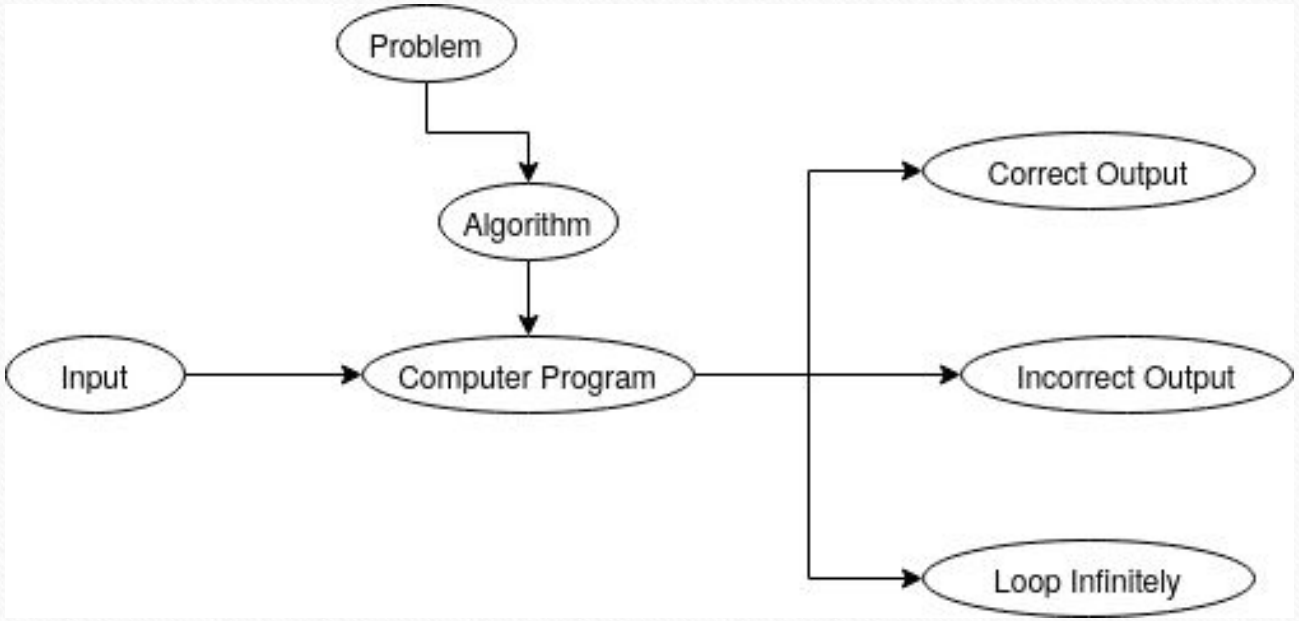
- Algorithm: cook a cup of instant noodles
  1. Pull back lid to the dotted line.
  2. Fill the cup to the inside line with boiling water from a kettle or from the microwave
  3. Close lid and let stand for 3 minutes.
  4. Stir well and add a pinch of salt and pepper to taste.

What are we interested in an algorithm?

- Correctness
- Efficiency
  - Time complexity – measure the execution time?
  - Space complexity

Correct Algorithm

- An algorithm is said to be ***correct*** if, for every input instance, it halts with the correct output.



How to measure an algorithm?

- The number of key operations
- The number of space units needed
- What if the input is uncertain?

How to measure an algorithm?

- E.g. Search a book in a box of books
  - Key operation: check the title of a book
  - Space unit: the space for one book

Why?

- Theoretical Reasons
  - To prove the requirements of a project.
- Practice Reasons
  - Analysis of human DNA 100,000 genes
  - Relevant Information from internet (google)
  - Adaptability (Facebook over Myspace)
  - News analysis by machine
  - Cryptography and digital signatures
  - Optimal resource allocation for a firm

Algorithm

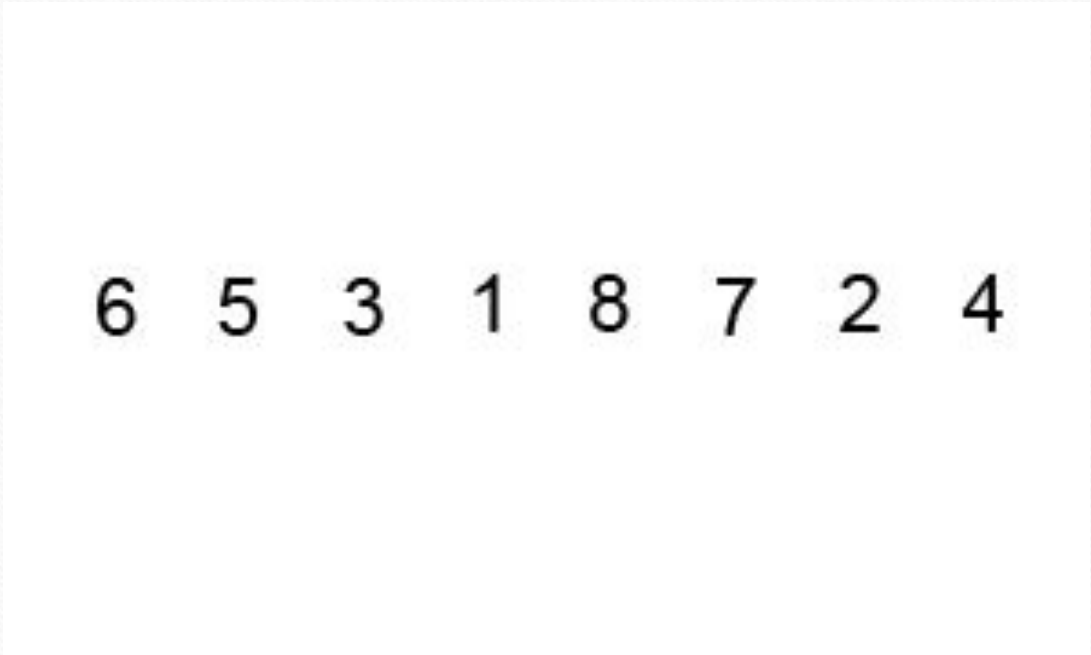
- *Algorithm SumOfSquares*
  - *INPUT: a; b; where a and b are integers*
  - *OUTPUT: c; where c is a sum of the squares of input numbers.*
  - *start;*
  - *c := a*a + b*b;*
  - *return c;*
  - *end;*

Bubble Sort

procedure bubbleSort(A : list of sortable items)

- n := length(A)
- repeat
  - swapped := false
  - for i := 1 to n-1 inclusive do
    - if A[i-1] ≥ A[i] then
      - swap(A[i-1], A[i])
      - swapped := true
    - end if
  - end for
- until not swapped

end procedure



Algorithmic Problem Solving

Trivial Approach

- Analyze the algorithm
  - Design a program
  - Implement the program
  - Execute the code
  - Measure the time
- See if the solution is ok
  - End The procedure

Algorithmic Approach

- Analyze the algorithm
  - Methods of Specifying an Algorithm
  - Proving an Algorithm's Correctness
- See if the solution is ok
  - Coding an Algorithm
  - End The procedure

Multiplication of Arabic Numerals

- Among many other contributions in mathematics, astronomy, and geography, he wrote a book about how to multiply with Arabic numerals.

Multiplication

1234567890

X 4578963210

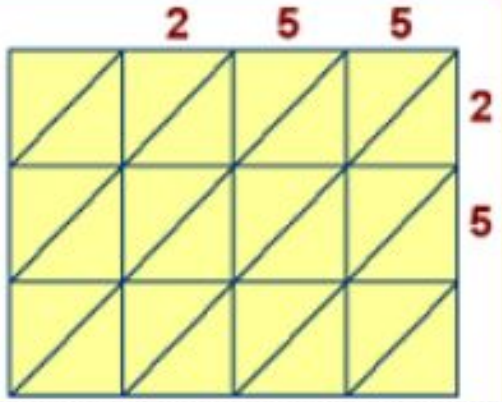
- 10*10=100 single digit multiplications

Multiplication of Arabic Numerals

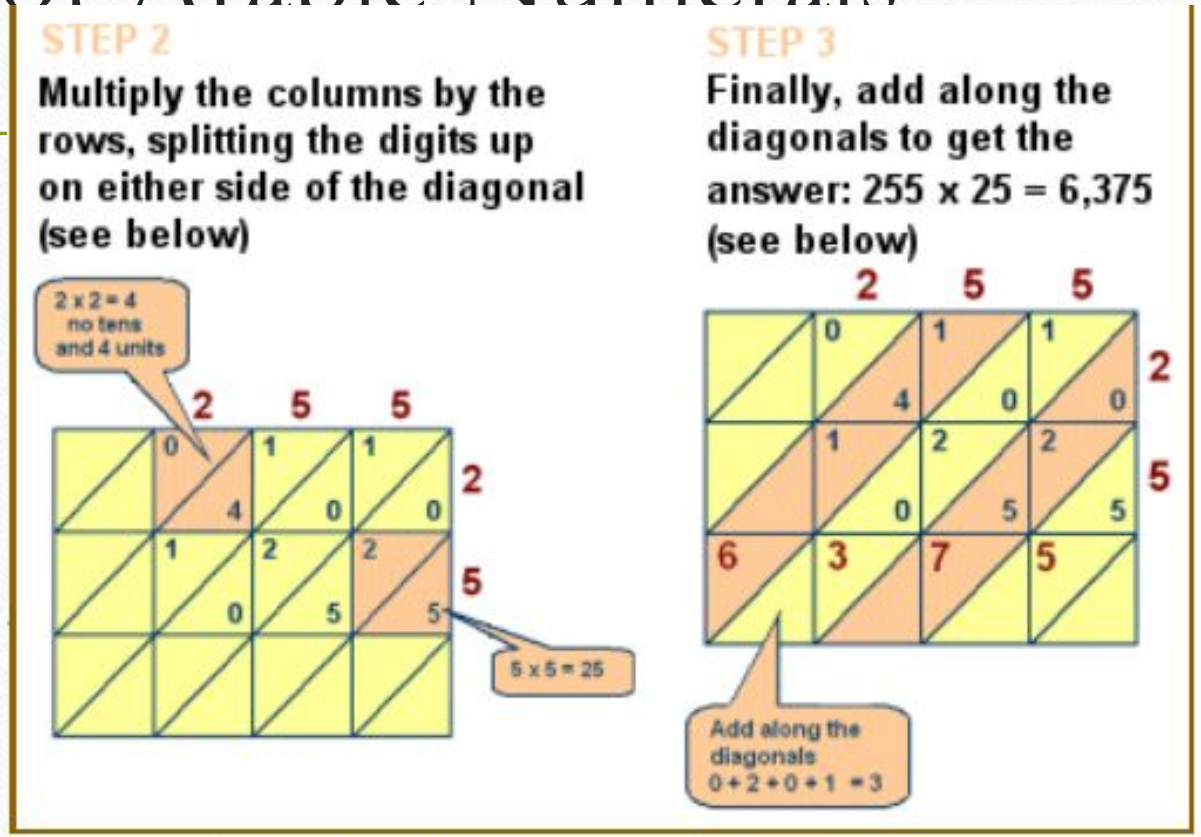
Lattice multiplication

STEP 1

To multiply using the lattice method, create a grid with diagonal lines, and split the numbers to be multiplied into their place values, e.g. 255 x 25 (see right)



Multiplication of Arabic Numerals



Fibonacci introduced lattice multiplication to Europe