Unit 1
Analysis of Algorithms

Prof. Purvi Tandel
Prof. Kinjal Mistree
Prof. Fenil Khatiwala
Department of Computer Engineering
CGPIT, UTU

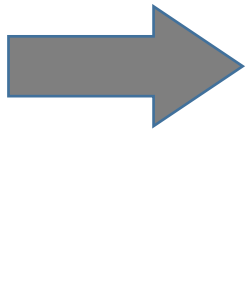
Outline

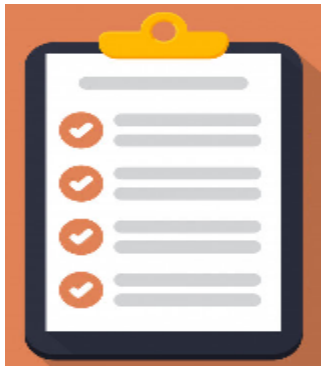
- **Algorithm**
- **Efficiency of algorithms**
- Performance analysis of algorithm
- Average and worst case analysis Elementary operation
- Asymptotic Notation
- Analyzing control statements
- Solving recurrences

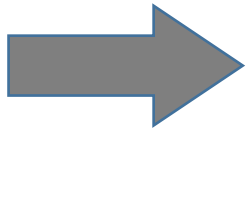
Algorithm

What is an algorithm?

- A step-by-step procedure, to solve the different kinds of problems.
- Suppose, we want to make a Chocolate Cake.
- An unambiguous sequence of computational steps that transform the input into the output.







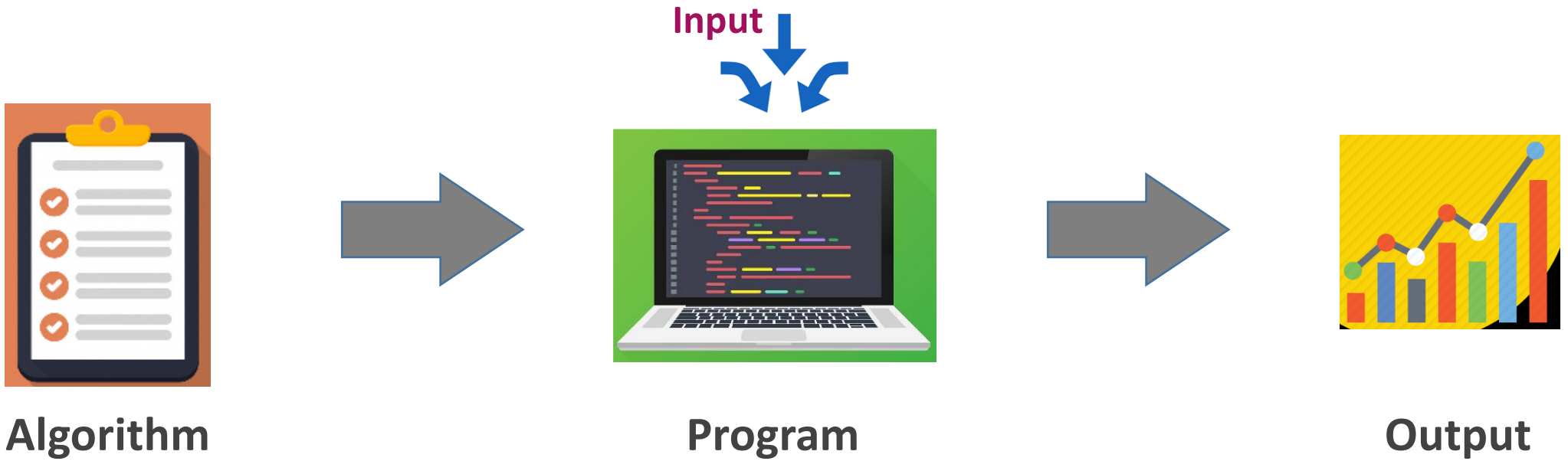
Input
Ingredients

Process
Recipe

Output
Cake

What is an algorithm?

- An algorithm is well defined computational procedure that takes some value, or set of values as input and produces some value, or set of values as output.



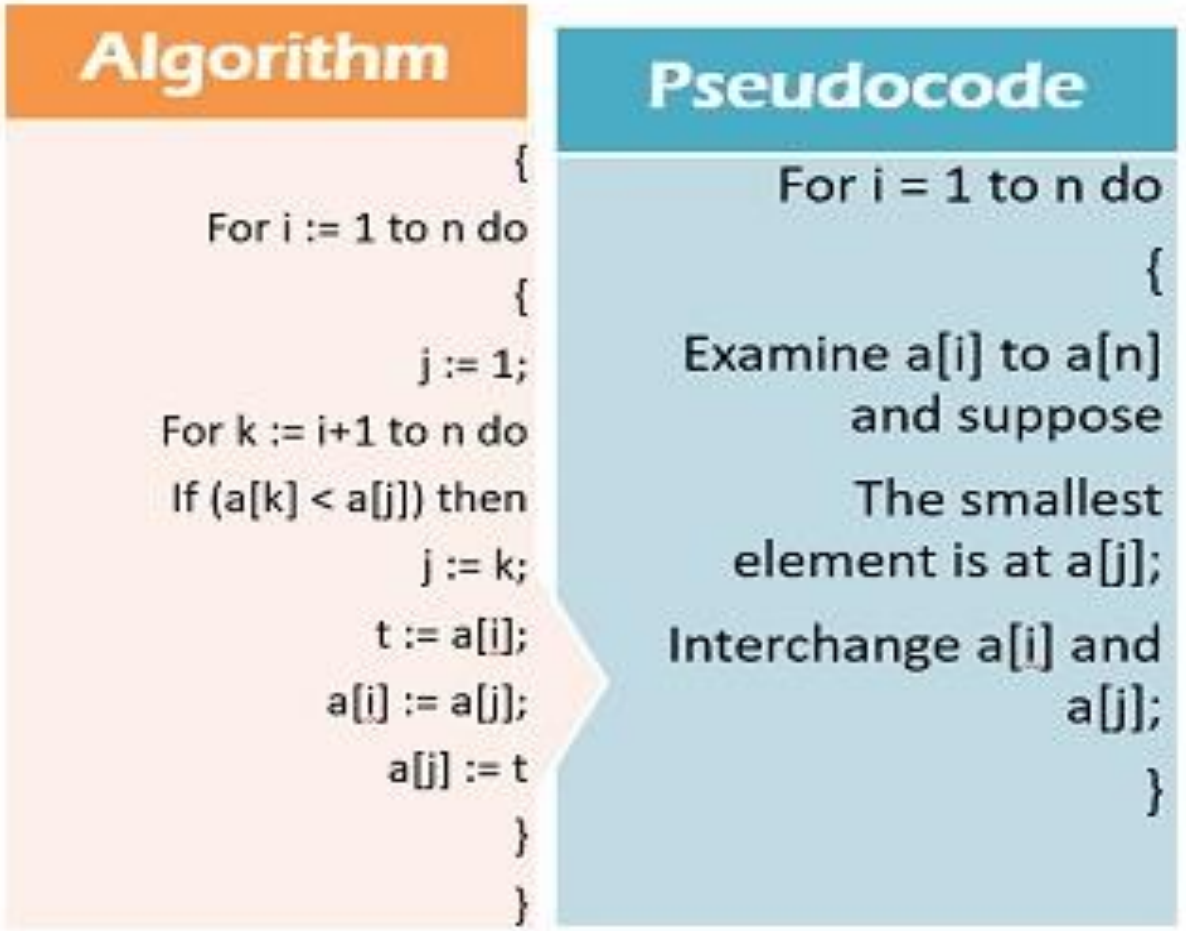
Characteristics of an algorithm

- **Finiteness**: An algorithm must always terminate after a finite number of steps.
- **Definiteness**: Each step of an algorithm must be precisely defined.
- **Input**: An algorithm has zero or more inputs.
- **Output**: An algorithm must have at least one desirable output.
- **Effectiveness**: All the operations to be performed in the algorithm must be sufficiently basic so that they can, in principle be done exactly and in a finite length of time.

Algorithm vs Pseudocode

Algorithm	Pseudocode
An algorithm is a sequence of steps which is utilized in order to solve a computational problem.	Pseudocode is a detailed description of an algorithm which is easier to read and is expressed in an English-like language.
The algorithm uses high-level constructs like snippet of code.	Pseudocode involves natural language with high-level programming builds.
Cooking recipe can be considered to be an algorithm if it describes precisely how to make certain dish, giving exact quantities to use and detailed instructions for how long to cook it.	Cooking recipe can be considered to be an pseudocode if it include vague notations like “add salt to taste” or “cook until tender”.

Algorithm vs Pseudocode



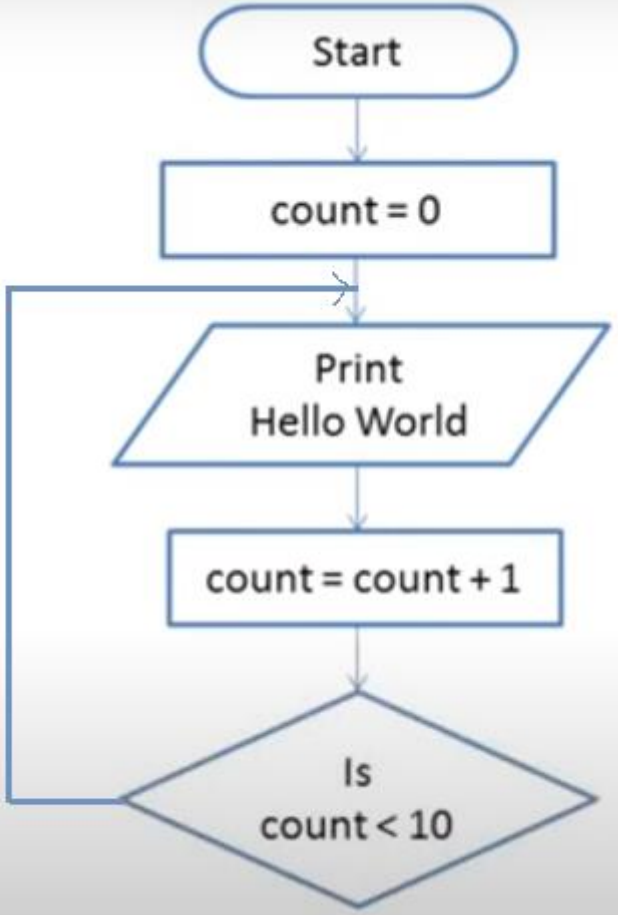
Algorithm vs Flowchart

Algorithm	Flowchart
An algorithm is step wise analysis of the work to be done.	Flowchart is a pictorial representation of an algorithm.
An algorithm is a precise or set of rules specifying how to solve some problem.	A flowchart is a diagram of the sequence of operations in a computer program.

Algorithm vs Flowchart

Algorithm in simple English

1. Initialize count = 0 (PROCESS)
2. Print Hello World (I/O)
3. Increment count by 1 (PROCESS)
4. Is count < 10 (DECISION)
   if YES go to step 2
   else
   Stop

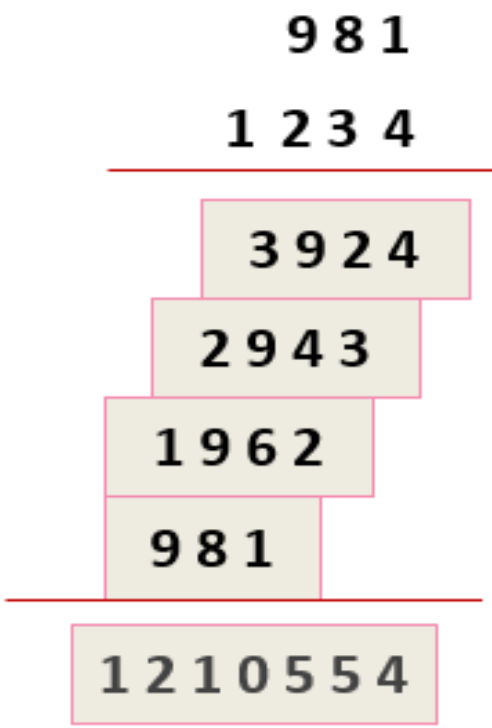


Study of an algorithm includes..

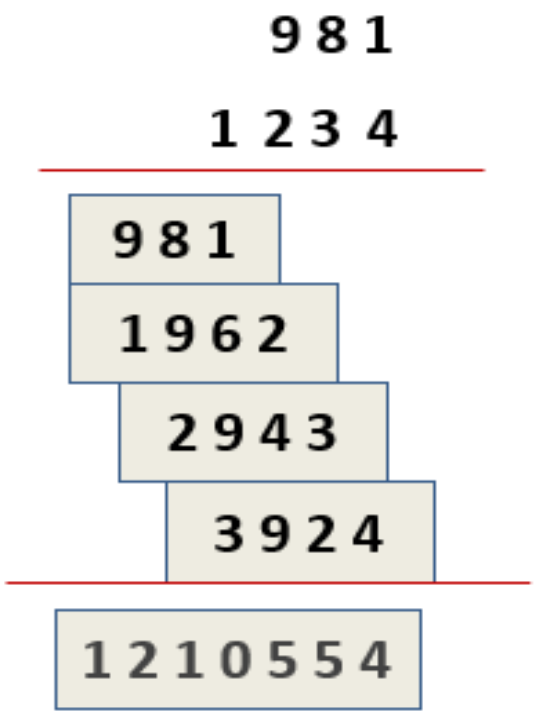
- How to device an algorithm
- How to validate an algorithm
- How to analyze an algorithm
- Testing a program
- How to evaluate performance of an algorithm

Importance of choosing a right algorithm
(Simple Multiplication Methods)

1. American approach



2. English approach



Importance of choosing a right algorithm (Simple Multiplication Methods)

3. à *la russe* multiplication

i. Write the multiplicand and multiplier side by side.
ii. Make two columns, one under each operand.
iii. Repeat step iv and v until the number in the left column is 1.
iv. Divide the number in the left hand column by 2, ignoring any fractions.
v. Double the number in the right hand column by adding it to itself.
vi. Next cross out each row where the number in the left hand column is even.
vii. Finally add up the numbers that remain in the right hand column.

981	1234	1234
490	~~2468~~	
245	4936	4936
122	~~9872~~	
61	19744	19744
30	~~39488~~	
15	78976	78976
7	157952	157952
3	315904	315904
1	631808	631808
		1210554

Importance of choosing a right algorithm (Simple Multiplication Methods)

4. Multiplication by divide and conquer

Both the multiplicand and the multiplier must have the same number of digits and this number be a power of 2. If not then it can be done by adding zeros on the left if necessary.

i. Multiply left half of the multiplicand by left half of multiplier and shift the result by no. of digits of multiplier **i.e. 4.**

ii. Multiply left half of the multiplicand by right half of the multiplier, shift the result by half the number of digits of multiplier **i.e. 2.**

iii. Multiply right half of the multiplicand by left half of the multiplier, shift the result by half the number of digits of multiplier **i.e. 2.**

iv. Multiply right half of the multiplicand by right half of the multiplier the result is **not** shifted at all.

Multiplicand	0 9 | 8 1
Multiplier	1 2 | 3 4

Multiply	Shift	Result
(09) * (12)	4	1 0 8 . . . .
(09) * (34)	2	3 0 6 . .
(81) * (12)	2	9 7 2 . .
(81) * (34)	0	2 7 5 4
		1 2 1 0 5 5 4

Efficiency of Algorithm

Analysis of Algorithm

What is the analysis of an algorithm?

- Analyzing an algorithm means calculating/predicting the resources that the algorithm requires.
- Two most important resources are computing time (time complexity) and storage space (space complexity).

Why analysis is required?

- By analyzing some of the candidate algorithms for a problem, the most efficient one can be easily identified.

Approaches for selecting efficient algorithm

There are two essential approaches to measuring algorithm efficiency:

Empirical analysis:

Program the algorithm and measure its running time on example instances

Theoretical analysis:

Derive a function which relates the running time to the size of instance

In this course our focus will be on Theoretical analysis.

Approaches for selecting efficient algorithm

Empirical (posteriori) approach	Theoretical (priori) approach
Programming different competing techniques & running them on various inputs using computer.	Determining mathematically the resources needed by each algorithm.
Implementation of different techniques may be difficult.	Uses the algorithm instead of an implementation.
The same hardware and software environments must be used for comparing two algorithms.	The speed of an algorithm can be determined independent of the hardware/software environment.
We have to test our algorithm only on small no. of instances.	We can consider larger no. of instances to test our algorithm.

Problem & Instance

Instance: An Instance of a problem consists of the input needed to compute the solution to the problem.

Example:

Problem: to multiply two positive numbers

Instance: 981 X 1234

Instance size: Any integer (generally $\boldsymbol{n}$) that in some way measures the number of components in an instance.

Examples:

Sorting problem: Instance size is number of elements to be sorted.

Graph problem: Instance size is number of nodes or edges or both.

Efficiency of Algorithm

- The efficiency of an algorithm is a measure of the amount of **resources consumed** in solving a **problem of size n**.
- The important resource is time, i.e., **time complexity.**

- To measure the efficiency of an algorithm it is required to measure its time complexity using any of the following approaches:
  1. To run it and measure how much processor time is needed.
  2. Mathematically computing how much time is needed as a function of input size.

Empirical Approach

Theoretical Approach

Efficiency of Algorithm

- Running time of an algorithm depends upon,
  1. **Input Size**
  2. **Nature of Input**

- Generally time **grows with the size of input**, for example, sorting 100 numbers will take less time than sorting of 10,000 numbers.
- So, running time of an algorithm is usually measured as a function of input size.

- In theoretical computation of time complexity, Running time is measured in terms of **number of steps/primitive operations performed**.

Thank You!