💧

Introduction to Complexity Analysis and Mathematics for Programming

Introduction to Complexity Analysis

Algorithm:

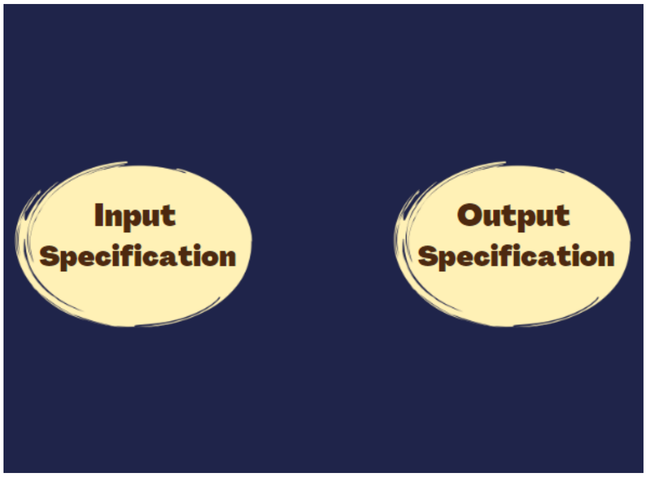
Step-by-step procedure to solve a ‘**computational problem**’.

Data Structure:

The storage and organization of the data are needed to solve the problem.

Algorithmic Problem:

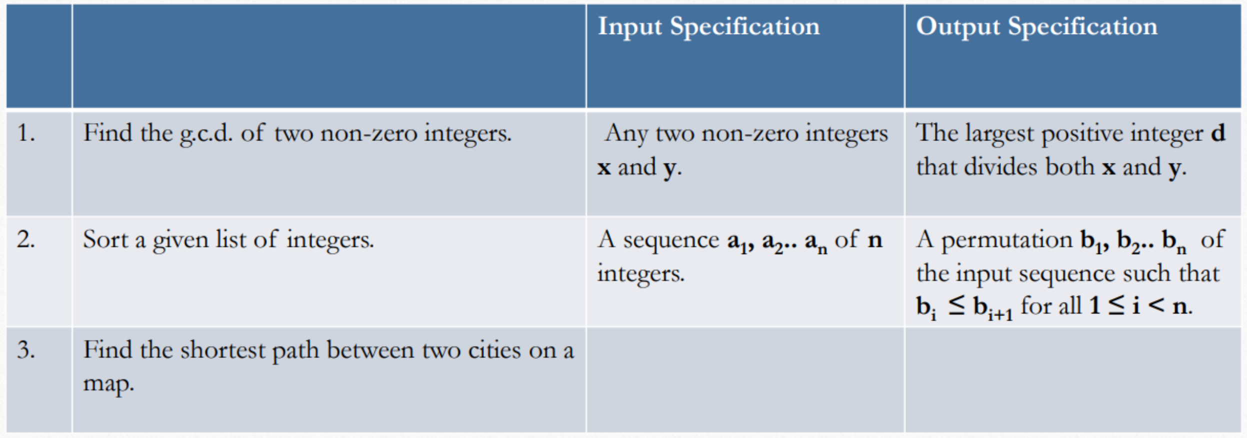
A specification of valid inputs to a problem and the valid outputs for each valid input.



Examples➖

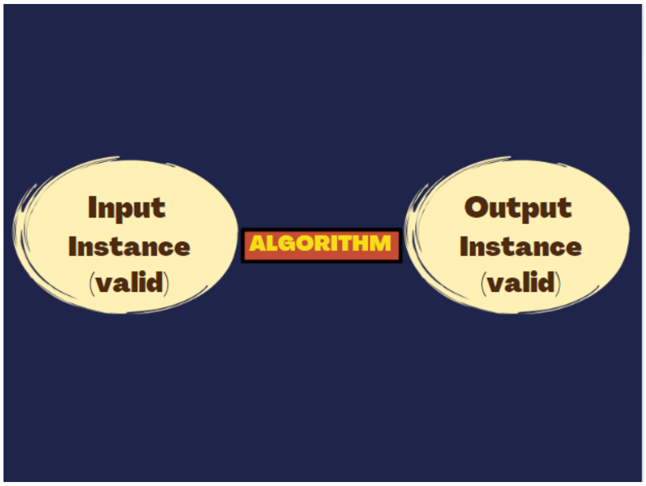
- Find the GCD of two non-zero integers.
- Sort the given list of integers.
- Find the shortest path between two cities on the map.

Algorithmic Problem: Specifications



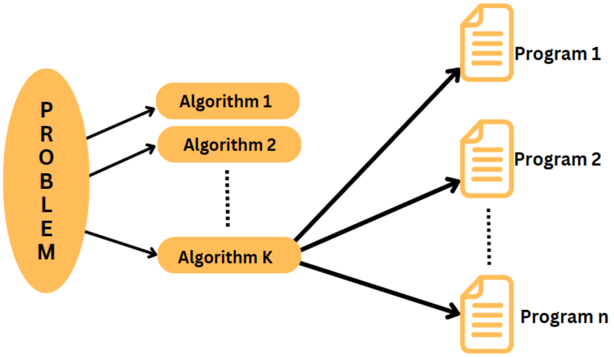
		Input Specification	Output Specification
1.	Find the g.c.d. of two non-zero integers.	Any two non-zero integers **x** and **y**.	The largest positive integer **d** that divides both **x** and **y**.
2.	Sort a given list of integers.	A sequence $\mathbf{a_1, a_2.. a_n}$ of **n** integers.	A permutation $\mathbf{b_1, b_2.. b_n}$ of the input sequence such that $\mathbf{b_i \le b_{i+1}}$ for all $\mathbf{1 \le i < n}$.
3.	Find the shortest path between two cities on a map.		

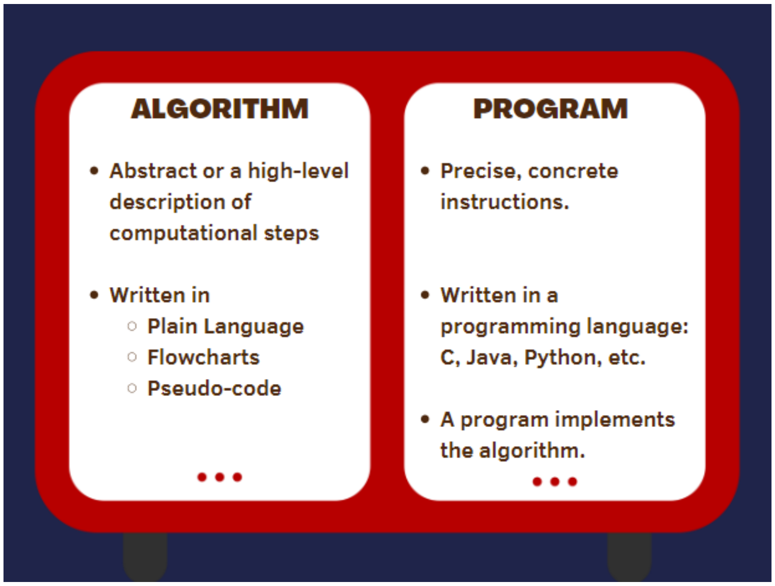
Note▬ Algorithm transforms the Input Instance into a correct Output Instance.



There may be many correct algorithms for a given algorithmic problem.

Algorithm vs Program





ALGORITHM

- Abstract or a high-level description of computational steps
- Written in
  - Plain Language
  - Flowcharts
  - Pseudo-code

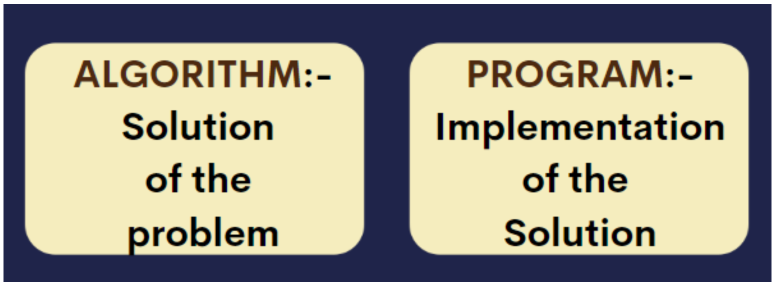
•••

PROGRAM

- Precise, concrete instructions.
- Written in a programming language: C, Java, Python, etc.
- A program implements the algorithm.

•••

In general terms, we can say



ALGORITHM:- Solution of the problem

PROGRAM:- Implementation of the Solution

Algorithm Analysis: Evaluating Algorithms

Qualities, and metrics to evaluate an algorithm:

- Runtime or execution time.
- Resources consumed (e.g. memory usage)
- Ease of understanding
- Ease of implementation

Often these qualities work against each other: a fast algorithm may consume too much memory, or maybe too hard to implement (or vice-versa)

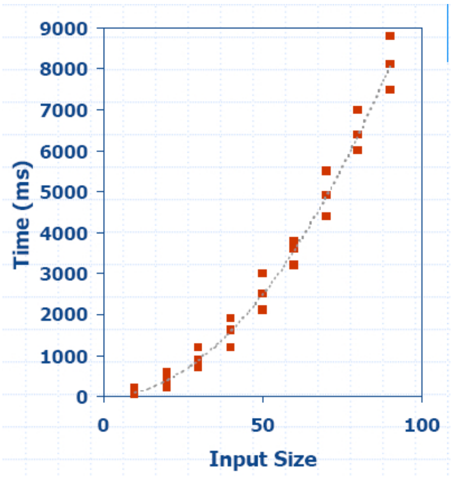
NOTE➖ The main important metric is Runtime or execution time. The second important metric is Resources consumed (e.g. memory usage).

How to measure the runtime of an algorithm?

One way is➖ Experimental Evaluation

Experimental Evaluation

- Write a program that implements the algorithm.
- Run the program on different inputs.
- Measure the time taken for execution.



Drawbacks of Experimental Evaluation

- Experimental evaluation measures the program's runtime. We want to measure the algorithm's runtime.
- Experiments can be done only with a restricted set of inputs. Does not tell anything about the algorithm's performance on other inputs.
- Have to actually implement the algorithm !
- A Program's runtime depends on many factors:
    - **The Algorithm**
    - **Actual inputs to the program**
    - Programming Language (Compiled vs. Interpreted, Compiler optimizations,..)
    - Operating System (Linux, Windows, Debian,..)
    - Underlying Hardware

  Only the first two factors are relevant for the algorithm's runtime!

Towards evaluating algorithms…

- We want to evaluate an algorithm, independent of the underlying hardware and software environments.
- We will study a formal, mathematical framework for evaluating and comparing algorithms.
- We will use an abstract or high-level description of an algorithm: **Pseudo-code**.
- **Main idea:** count the number of **'basic operations'** in a pseudo-code.

Pseudo Code

A mixture of natural (English) language & high-level programming constructs that describe the main idea behind an algorithm.

Try some pseudo codes

```
# Sum of even numbers of an array

Algo 1: iterate from i=0 to n-1, find the sum

Algo 2: Iterate from i=n-1 to 0, find the sum
```

Mathematics For Programming

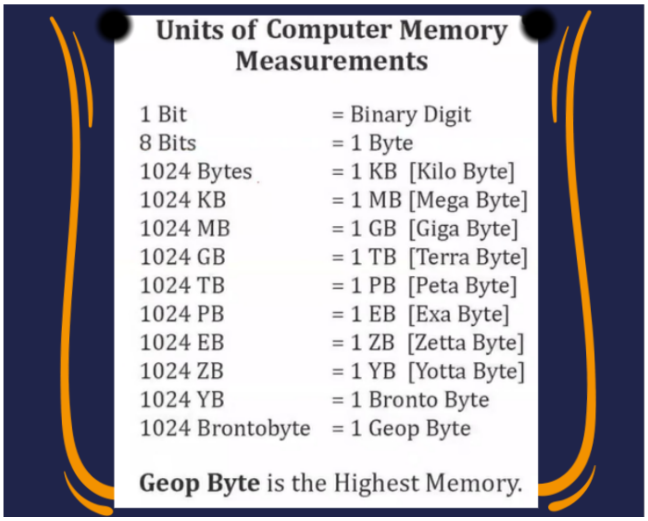
Units of Measurement

- **Liquids** like milk, juice, water, oil, etc. are measured in **liters and kilolitres**.

- Fruits, food items, grains, and vegetables are measured in **grams and kilograms.**

- Length and Distance are measured in **meters and kilometers.**

- Computer memory also is measured in some units. These units are **bits and bytes**



Units of Computer Memory Measurements

1 Bit	= Binary Digit
8 Bits	= 1 Byte
1024 Bytes	= 1 KB [Kilo Byte]
1024 KB	= 1 MB [Mega Byte]
1024 MB	= 1 GB [Giga Byte]
1024 GB	= 1 TB [Terra Byte]
1024 TB	= 1 PB [Peta Byte]
1024 PB	= 1 EB [Exa Byte]
1024 EB	= 1 ZB [Zetta Byte]
1024 ZB	= 1 YB [Yotta Byte]
1024 YB	= 1 Bronto Byte
1024 Brontobyte	= 1 Geop Byte

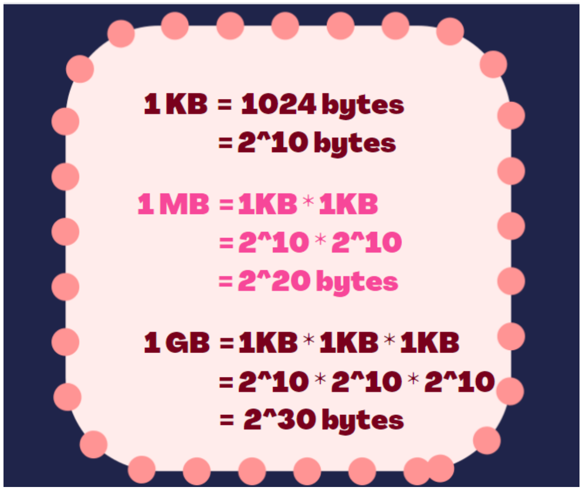
Geop Byte is the Highest Memory.

Bit

A bit (binary digit) is **the smallest unit of data that a computer can process and store**. A bit is always in one of two physical states, similar to an on/off light switch. The state is represented by a single binary value, usually a **0 or 1**.

Everything in the computer is stored in the form of 0s and 1s only. Examples➖ Songs, videos, games, etc.

When we purchase a laptop, we look for the **processor**'s size, **RAM (Main Memory)** size, and **HD (Secondary Memory)** size. The sizes are in **GB(Gigabyte).**



1 KB = 1024 bytes
= 2^10 bytes

1 MB = 1KB * 1KB
= 2^10 * 2^10
= 2^20 bytes

1 GB = 1KB * 1KB * 1KB
= 2^10 * 2^10 * 2^10
= 2^30 bytes

Important Series

- **Sum of n natural numbers.**

$$1+2+3+4+...+n=\frac{n*(n+1)}{2}$$

- **Sum of squares of n natural numbers.**

$$1^2+2^2+3^2+...+n^2=\frac{n*(n+1)*(2n+1)}{6}$$

- **Sum of cubes of n natural numbers.**

$$1^3+2^3+3^3+...+n^3=\frac{(n*(n+1))^2}{4}$$

- **Factorial of a natural number n.**

$$n!=n*(n-1)*(n-2)*.....*3*2*1$$

- **General Logarithmic Series**

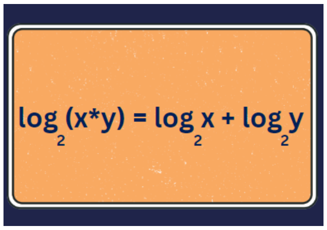
$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+...+\frac{1}{n}=\log_{10} n$$

Properties of logarithms

Generally in algorithms, we will use base-2 logarithms known as **binary logarithms.**

$$\log_2 x^n = n \log_2 x$$

$$\log_x x = 1$$



$\log_2 (x*y) = \log_2 x + \log_2 y$

Learn more about logarithmic general properties.

Questions➖

1. **log2(1024)=?**

$$log2(1024) = log2(2^{10}) = 10 * log2(2) = 10 * 1 = 10$$

2. **n=2^(1024) then log2(log2(n))=?**

$$log2(log2(2^{1024})) = log2(log2(2^{1024})) = log2(1024 * log2(2)) = log2(1024) = log2(2^{10}) = 10 * log2(2) = 10 * 1$$

3. **What is the answer to the given series?**

$$1 + 2 + 3 + 4 + ... + (n - 1) = ?$$

Answer➖ n(n-1)/2

4. **What is the answer to the given series?**

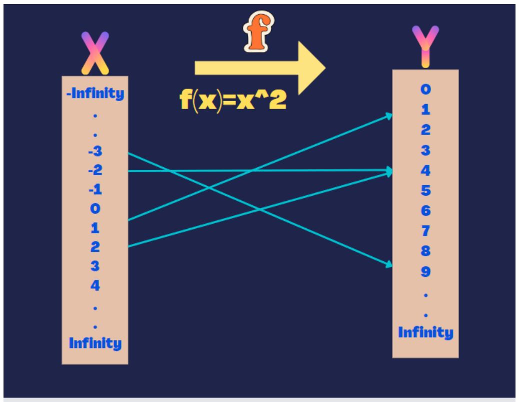
$$log(1) + log(2) + log(3) + log(4) + ... + log(n) = ?$$

Answer➖ log(1*2*3*4*….*n)=log(n!)

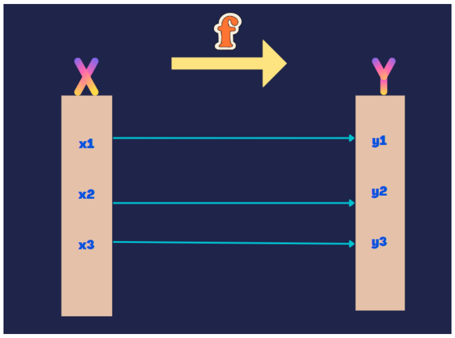
Functions

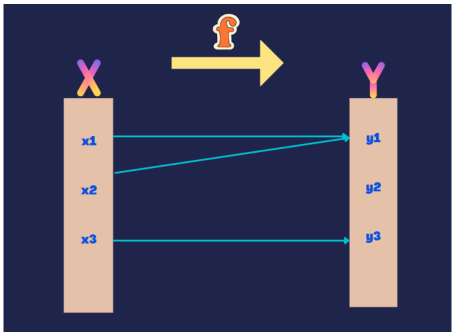
A function is defined as **a relation between a set of inputs having one output each**. In simple words, a function is a relationship between inputs where each input is related to exactly one output.

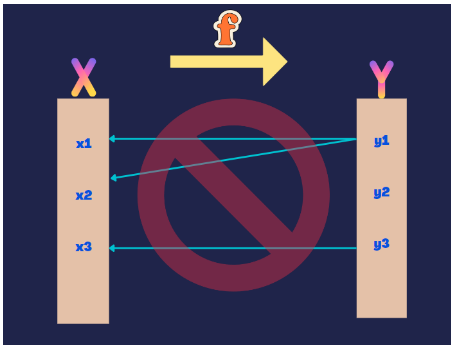
Set X to Set Y, and assign each element of X to exactly one element of Y.



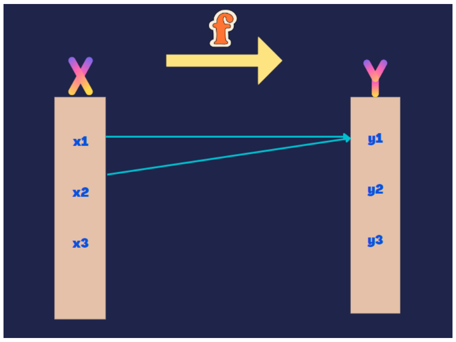
Examples of functions➖





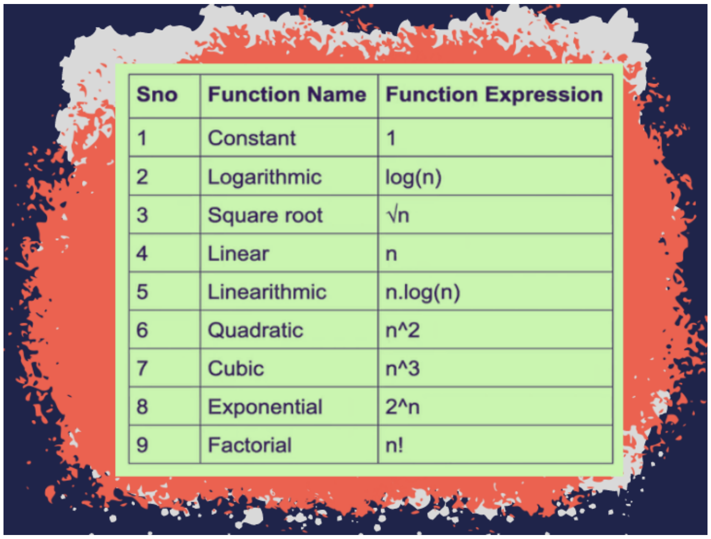


Not Possible



Use of functions in Programming

- **In Algorithms, the time complexity is very much related to functions in Math**.
- The following functions are commonly used in Algorithms:



Sno	Function Name	Function Expression
1	Constant	1
2	Logarithmic	log(n)
3	Square root	√n
4	Linear	n
5	Linearithmic	n.log(n)
6	Quadratic	n^2
7	Cubic	n^3
8	Exponential	2^n
9	Factorial	n!

Happy Coding!