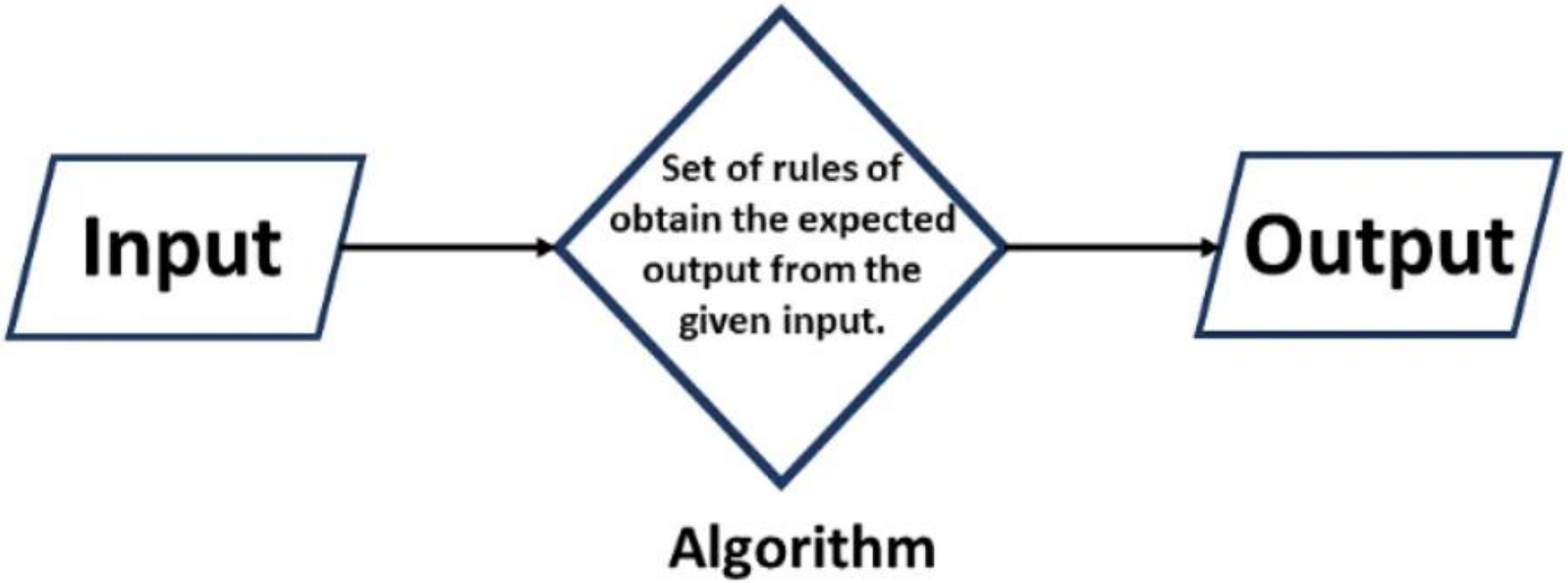
INTRODUCTION TO ALGORITHMS

Overview of Algorithms

-By Dr. S. Sridhar

What is an algorithm?

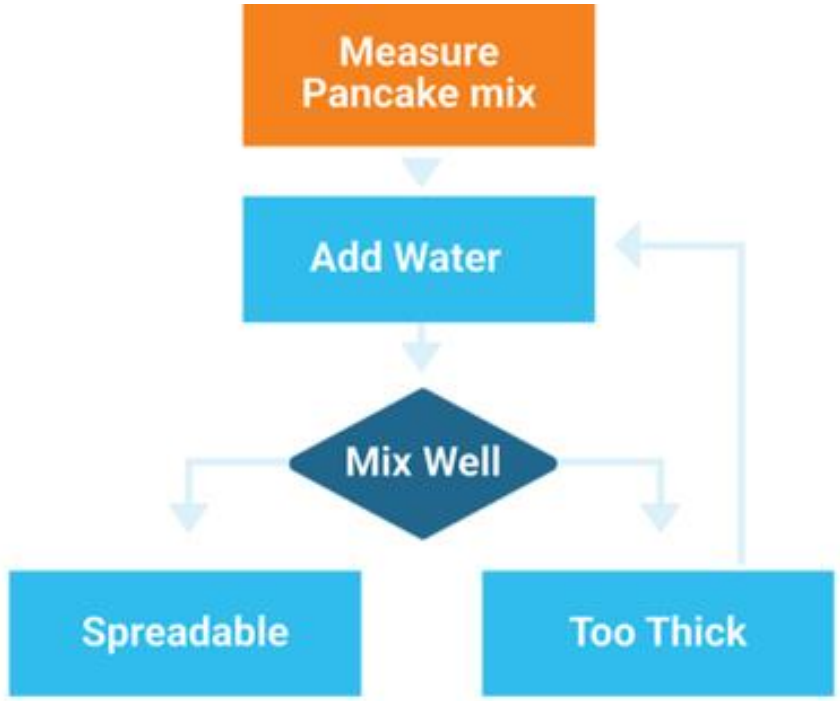
- It is a procedure adapted to execute a task
- It is a step-by-step instruction



Algorithms in Daily Life

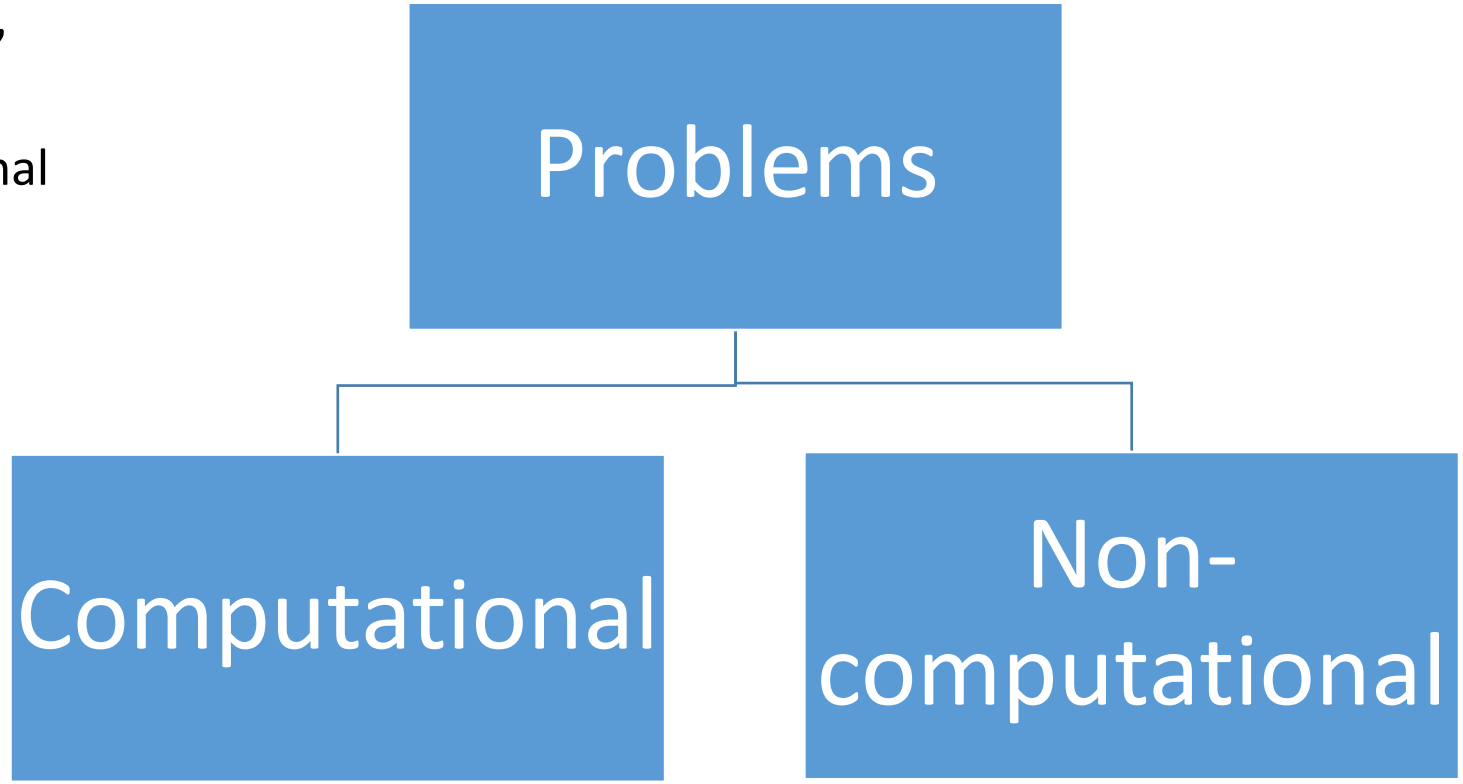
We use many algorithms in daily life without knowing, some of them are:

- Packing items in a suitcase
- Searching for a specific book
- Making a pancake mix
- Scheduling daily activities
- Sending messages via email/SMS



TYPES OF PROBLEMS

- Problems can be classified into 2 types in context with computer algorithms namely,
  - Computational
  - Non-computational



Computational problems

- These are problems which can be solved by the computer
- They can be characterized by the following,
    - Formalization of all legal inputs and expected outputs of a given problem
    - Relationship between the problem output and input

Non-Computational Problems

- These are problems that cannot be solved by computers
- Computers can solve calculations and crunch numbers easily but humans outrun computers in recognition
- For computers to recognize an object it requires much programming and much training for the model, but humans can easily recognise objects
- problems where an opinion is needed are considered non-computational as computers can not offer suggestions or opinions

Types of computational Problems

Structuring problems

- The input Is restructured based on certain conditions or properties
- Example:
    - Sorting a list in ascending or descending order

Search Problem

- Searching problem involves searching for a target in a list of all possibilities.
- All potential solutions may be generated and represented in the form of a list or graph
- Based on the condition, the best solution is fetched.
- Puzzles are a good example of search problem

Construction problems

- These involve the construction based on the constraints associated with the problem

Decision Problem

- These are yes or no problems
- The output is either yes or no

Optimization problem

- These problems involve a certain objective function that is typically of the following form:
  - maximize
  - minimize

  based on a set of constraints that are associated with the problems
- For example: Finding the shortest route to Hyderabad from Chennai, this problem involves finding the shortest path, thus an optimization problem.
- All possible inputs of an Algorithm are called the domain of the input data
- The number of binary bits used to represent the input is called the input size

Characteristics of a Algorithm

- They can have zero or more inputs
- They must have at least 1 output
- It must be definite. The instruction should be clear and unambiguous without any confusion.
- It must have a well-defined and ordered procedure that contains the specific instructions in a specific order
- The order is significant as a change in the order can lead to a totally different result.
- Most importantly an Algorithm must be correct
- An algorithm should be effective, it should be traceable manually
- The algorithm must have a finite number of steps and must terminate after those steps

Additional characteristics of an algorithm

- The algorithm must be simple that is it must be easily implemented.
- It must be generic independent of any programming language

Algorithms Vs program

Algorithm

- These are like blueprints of a building
- These are like the user manual for the program

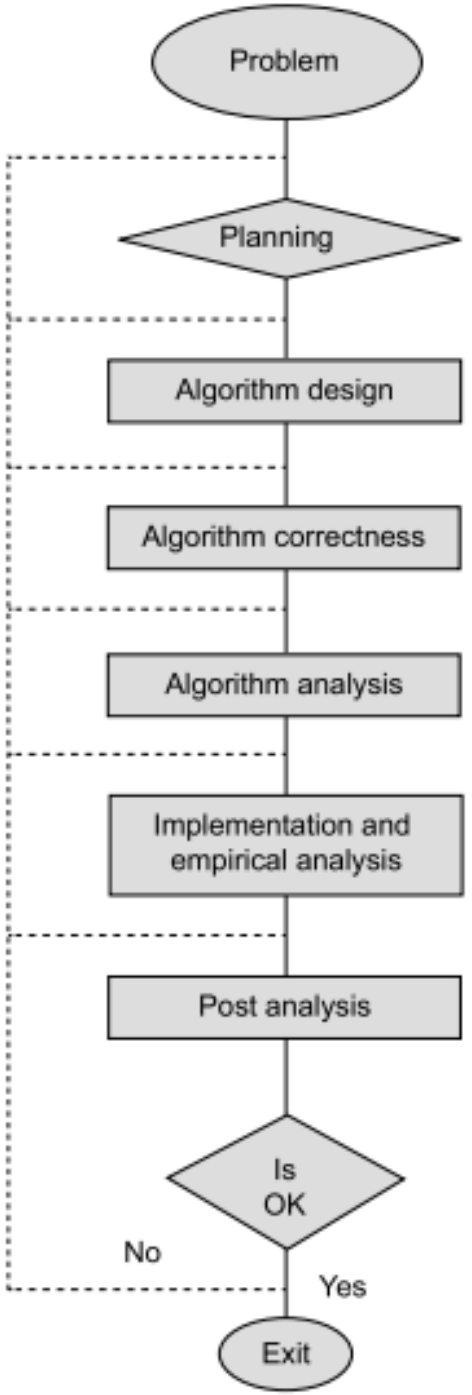
Program

- These are like the finished building itself
- These are built based on the algorithm

Stages of Solving a Problem

The following are the stages of solving a problem

- Understanding the problem
- Planning an algorithm
- Designing an algorithm
- Validating and verifying an algorithm
- Analyzing an Algorithm
- Implementing an algorithm
- Performing empirical Analysis(if necessary)



Understanding the problem

- The study of an algorithm starts with the computability theory
- The primary question in this is “Is the problem solvable?”
- To overcome that, a guideline(algorithm) is necessary
- Computers cannot solve problems that are not well-explained

Planning an algorithm

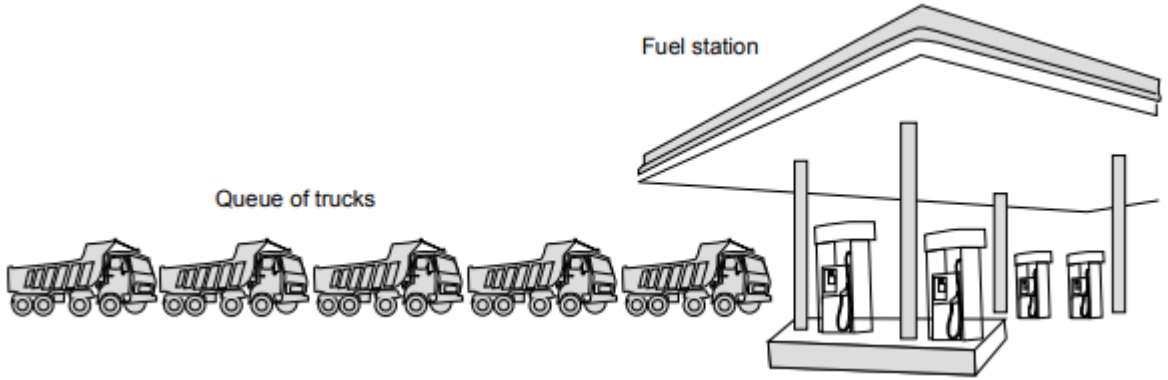
- Now that we have realized the importance of algorithms, we have to focus on planning them.
- These are a few constraints you must take into consideration while planning the algorithm
    - Model of Computation
    - Data organization

Model of Computation

- It is an abstraction of real-world computers
- Model is a theoretical model that does not exist in real life
- It is a Virtual Machine in which a program can be executed
- It is useless to talk about the speed of an algorithm as it varies from machine to machine.
- Therefore it is necessary to have a computational model which will give a proper reading.

Data organization

- It is the way the data is organized
- All algorithms need data to process information so it is necessary to organize data



- In the above-given picture, a few trucks are shown waiting in a “QUEUE” to fill gas.
- Queue is a data structure that follows FIFO[First In First Out], that is the truck that comes first(First Come) gets the gas first(First Serve).

Designing an Algorithm

- Now, that we have planned the algorithm next we have to design it.
- Here the following are our primary concerns
    - How to design the algorithm
    - How to express the algorithm
- Algorithm design is a way of developing the required algorithm using the appropriate design strategy
- These strategies differ from problem to problem
- Let us take an example, where you want to find a person's number in a telephone directory
- If we start searching for his name from page 1, that is called the brute force method, it is very inefficient and only considered if no other possibilities are there.
- Other way is to use the index of the book to find the result in a faster and more effective manner.

- Important design paradigms
  - Divide and conquer
  - Dynamic Programming
- Divide & and conquer method divides the problem into sub-problems and combines the result of the sub-program to get the final solution
- Thus, the important skill required for problem-solving is the selection and application of suitable design paradigms
- A skilled algorithm designer is called an ‘algorist’

Algorithm Specification

- After the algorithm is designed, it must be passed on to a programmer who can implement the algorithm
- This stage is called ‘algorithm specification’
- There are only 3 possibilities
- One is to use natural languages such as English to communicate the algorithm
- The drawback of using Natural language is that it is ambiguous and lacks precision.
- Hence a pseudocode is preferred
- Pseudocode is a mix of natural and mathematics.
- Programming language can also be used, but while using it the reader gets blogged with the programming code details.

Validation of an algorithm

- These are methods used to check the algorithm’s correctness
- Sometimes, an algorithm may not give the correct output as expected due to some logical errors
- Thus verifying an algorithm is necessary
- Algorithm validation is a process of checking whether the given algorithm gives correct outputs for valid inputs or not.
- This is done by comparing the output of the algorithm with the expected output.

Verification of an algorithm

- This is done after the validation process.
- It is a process of providing mathematical proof that the algorithm is working properly for all instances of data.
- One way of doing it is by creating a set of assertions that are expressed using mathematical logic
- Assertions are statements which indicate the condition of algorithm variables at variable point.
- . Preconditions indicate the conditions or variables before the execution of an algorithm, and postconditions indicate the status of the variables at the end of the execution of an algorithm
- All these executions are carried out theoretically on a paper.
- A proof for the algorithm is said to exist if the preconditions can be shown to imply the postconditions logically.
- Mathematical induction is one of the famous method used here.

Analysis of an Algorthm

- In the analysis of an algorithm, the complexity is taken into consideration
- It is because we are mainly focused on finding the optimal algorithm with fewer computer resources.
- Complexity theory is a field of algorithm that deals with the analysis of a solution in terms of computational resources and their optimization
- Complexity is the degree of difficulty associated with a problem and the algorithm
- For example, an algorithm for sorting 10 inputs is easy but when it comes to 1 million inputs it becomes difficult.
- This shows the connection between complexity and the size of the input
- Time complexity means the time taken by an algorithm to execute for different increasing inputs
- When it comes to the time, it can be run time or compile time, but the execution time(compile time) does not depend on problem instances
- Another type of complexity is space complexity, it refers to the amount of space it consumes when the program is run.

- In algorithm study, time complexity is not measured in absolute terms. For example, one cannot say the algorithm takes 3.67 seconds. It is wrong as time complexity is always denoted as a complexity function t(n T(N t(n n’ and is not an absolute value
- The variables n and N are used interchangeably and always reserved to represent the input size of the algorithm

Implementing the algorithm and performing empirical tasks.

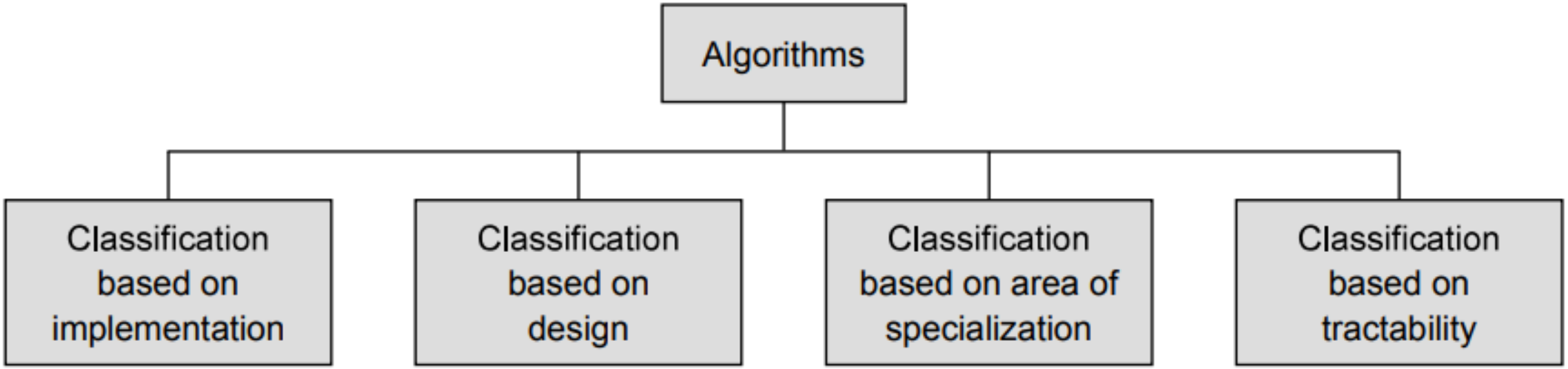
- After the design process, it is then implemented in the required programming language.
- Even though the algorithm is correct there might still be some errors that may occur, these might be syntax errors in the programming language or a logical error itself
- In the case of the 2nd event, one must revisit the designing process and correct the algorithm
- The analysis that is carried out after the program is developed is called empirical analysis
- A new area called experimental algorithmics, where analysis is performed for larger programs using a dataset, is emerging
- A dataset is a huge collection of valid input data of an algorithm; the standard dataset that is often used for testing algorithms is called a benchmark dataset
- This kind of analysis is called empirical analysis, these are also called posteriori analysis

Post (or postmortem) Analysis

- Any analysis should end with valuable insight.
- The following are the questions that are often raised in this problem-solving process
  - Is the problem solvable?
  - Are there any limits for the algorithm? Is there any theoretical limit to the efficiency of the algorithm
  - Is the algorithm efficient? Are there any other algorithms that are faster than the current algorithm
- The best possible outcome is the lower-bound
- The worst-case estimate of resources required by the algorithm is called an upper-bound

- The difference between the lower and upper bounds should be minimal for a better solution
- The difference between the upper and the lower bounds is called an algorithmic gap. Technically, this should be zero. However, in practice, there may be a vast difference between an upper and a lower bound
- a problem-solving process tries to reduce this difference by focusing on better planning, design, and implementation on a continuous base

Types of Algorithms



Based on implementation

- Based on implementation they can be classified as follows
  - Recursive and non recursive
    - Using recursion, the problem is reduced to another problem with a decrease in input instances.
    - The transformed problem is the same as the original one but with a different input, which is less than that of the original problem
    - The problem reduction process is continued till the given problem is reduced to a smaller problem that can be solved directly
    - Then the results of the sub-problems are combined to get the result of the given problem
    - This strategy is called recursion
  - Nature and Number of Processors
    - An algorithm that is designed for a single processor is called a sequential algorithm.
    - A parallel algorithm is designed for systems that use a set of processors. The concept of parallel processing and distributed processing are interrelated.
    - Parallel systems have mul- tiple processors that are located closely. Distributed systems, on the contrary, have multiple processors that are located at different places separated by a vast distance geographically.
    - Hence, distributed systems are called loosely coupled systems, while parallel systems are called tightly coupled systems.

- Exact and Approximation algorithm
  - An exact algorithm finds the exact solution for the problem
  - But sometimes it is difficult to find the exact solution, here approximation algorithms come in handy
- Deterministic vs Non-deterministic Algorithms
  - Deterministic algorithms always provide fixed predictable results for a given input. In contrast, non-deterministic algorithms or randomized algorithms take a different approach.
  - For deterministic algorithms, the output should always be true. On the other hand, randomized algorithms relax this condition.
  - It is argued that outputs based on random decisions may not often result in correct answers or the algorithm may not terminate at all. Thus, the accuracy of an output is associated with probability.
- Based on design
  - Based on design algorithms can be classified into brute force, dynamic programming, greedy approach, backtracking, and brach and bound algorithms

- Based on area of specialization
  - General algorithms such as searching and sorting
  - order statistics such as finding mean, median, and rank, These algorithms can be considered as building blocks of any area of specialization.
  - In graph algorithms, graphs are used for modeling complex systems by representing relationships among the subsystems
  - A graph represents a set of nodes[also called vertices] and edges
  - These nodes are connected by edges these are also called arcs
- Based on tracability
  - Tractability means solvability of a given problem within a reasonable amount of time and space.
  - An intractable problem is difficult to solve within the reasonable amount of computer resources. Based on tractability, the following categories are possible:

- **Easily solvable problems**
  - These problems have polynomial time complexity or their upper bounds are characterized by a polynomial. These problems are solvable
  - For example, sorting is an example of solvable or polynomial problems.

- **Unsolvable problems**
  - Problems such as halting problems cannot be solved at all.
  - These are called unsolvable or non-computable problems
- **Intractable problems**
  - These problems are of two categories.
    - One category comprises a set of problems that have algorithmic solutions but require more computer resources.
    - Hence, these solutions are practically non-implementable. In addition, these have been proved to be computationally hard.
    - The other category consists of a set of problems that have been proved to be computationally hard. For example, a TSP is a proven computationally hard problem.