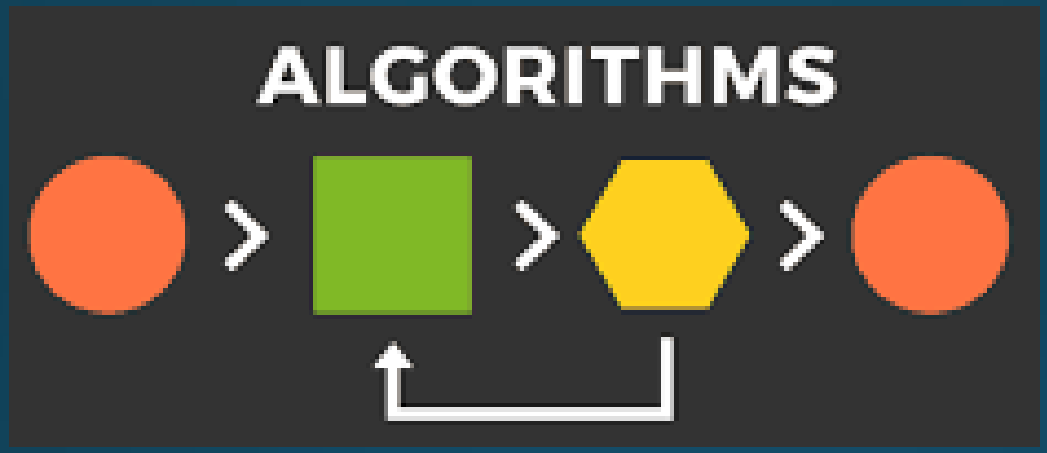


Module 3

Introduction to Algorithmic Thinking

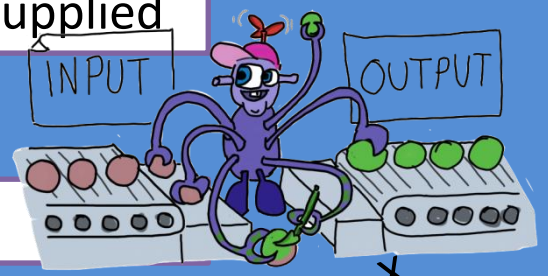
What is an Algorithm? And what are its characteristics?

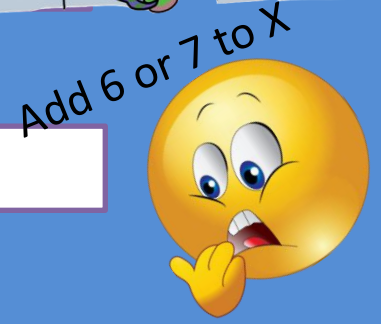
- algorithms are the foundation of computing

Def : An algorithm is a finite set of instructions that, if followed , accomplishes a particular task.

Criteria every algorithm must satisfy

Input	Zero or more quantities are externally supplied
Output	At least one quantity is produced.
Definiteness	Each instruction is clear and unambiguous
Finiteness	terminates after a finite number of steps
Effectiveness	Every instruction must be very basic





Algorithms are everywhere

- Search Engines
- GPS navigation
- Self-Driving Cars
- E-commerce
- Banking
- Medical diagnosis
- Robotics
- Algorithmic trading
- and so on …

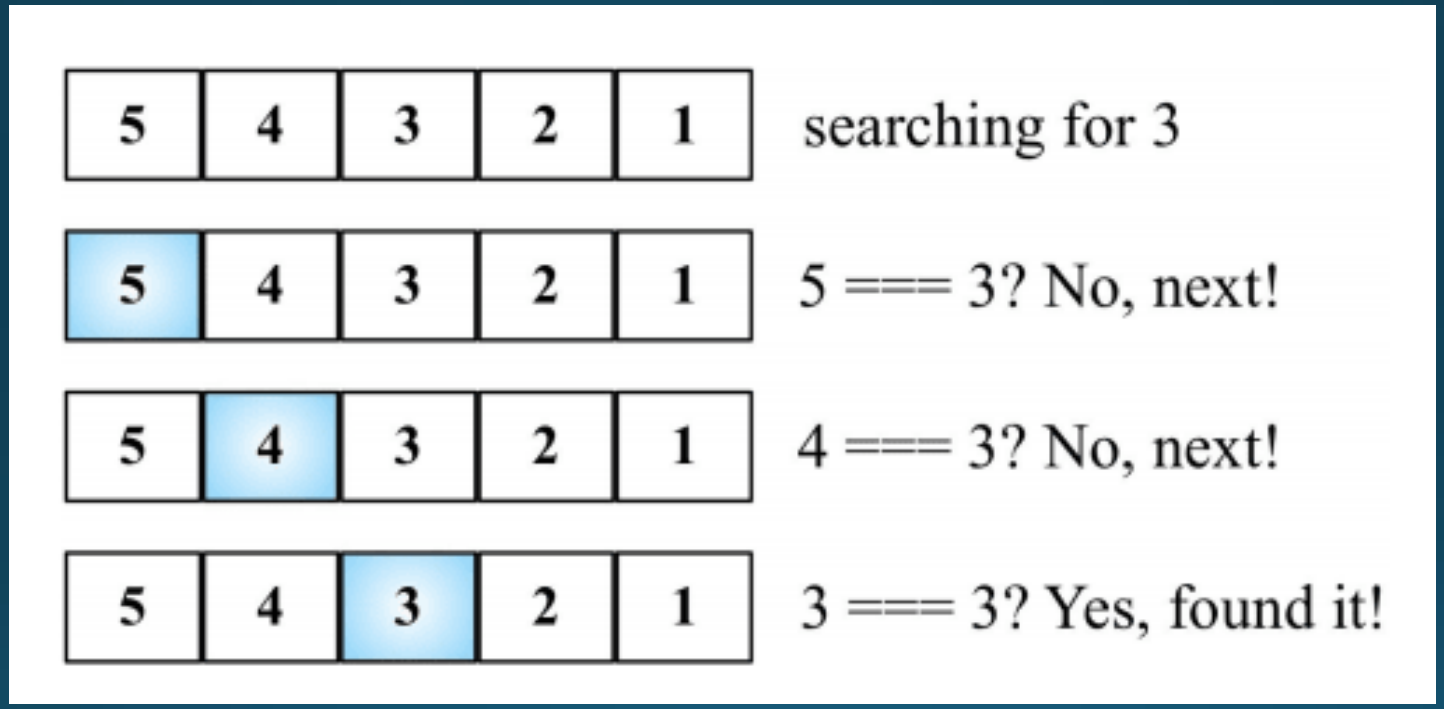
What is Algorithmic Thinking?

- Algorithmic thinking is the use of algorithms, or step-by-step sets of instructions, to complete a task.
- algorithmic thinking is this idea of coming up with steps to solve a problem.
- Algorithmic thinking has close ties to computer science and mathematics, as algorithms are the key to completing sequences of code or chunking big problems into smaller, more solvable parts.

Brute-force approach

- It is an intuitive, direct, and straightforward technique of problem-solving in which all the possible ways or all the possible solutions to a given problem are enumerated.
- Many problems solved in day-to-day life using the brute force strategy,
  - for example exploring all the paths to a nearby market to find the minimum shortest path.
  - Arranging the books in a rack using all the possibilities to optimize the rack spaces, etc.
- In fact, daily life activities use a brute force nature, even though optimal algorithms are also possible.
- Pros:
  - The brute force approach is a guaranteed way to find the correct solution by listing all the possible candidate solutions for the problem.
  - It is a generic method and not limited to any specific domain of problems.
  - The brute force method is ideal for solving small and simpler problems.
  - It is known for its simplicity and can serve as a comparison benchmark.
- Cons:
  - The brute force approach is inefficient for real-time problems.
  - This method relies more on compromising the power of a computer system for solving a problem than on a good algorithm design.
  - Brute force algorithms are slow.
  - Brute force algorithms are not constructive or creative compared to algorithms that are constructed using some other design paradigms.

Linear Search



Algorithm Efficiency

- In computer science, algorithmic efficiency is a property of an algorithm which relates to the amount of computational resources used by the algorithm. An algorithm must be analyzed to determine its resource usage, and the efficiency of an algorithm can be measured based on the usage of different resources.
- Computing time and storage requirement are the criteria for judging algorithms that have direct relationship to performance.
- *Space Complexity of an algorithm is the amount of memory it needs to run to completion.*
- *The time complexity of an algorithm is the amount of computer time it needs to run to completion*

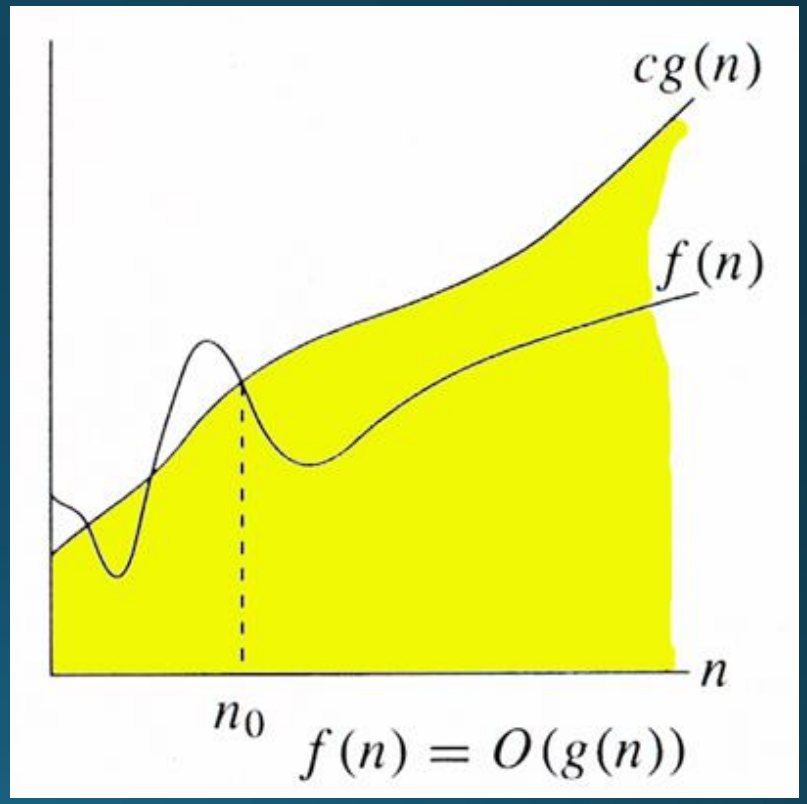
Time Complexity

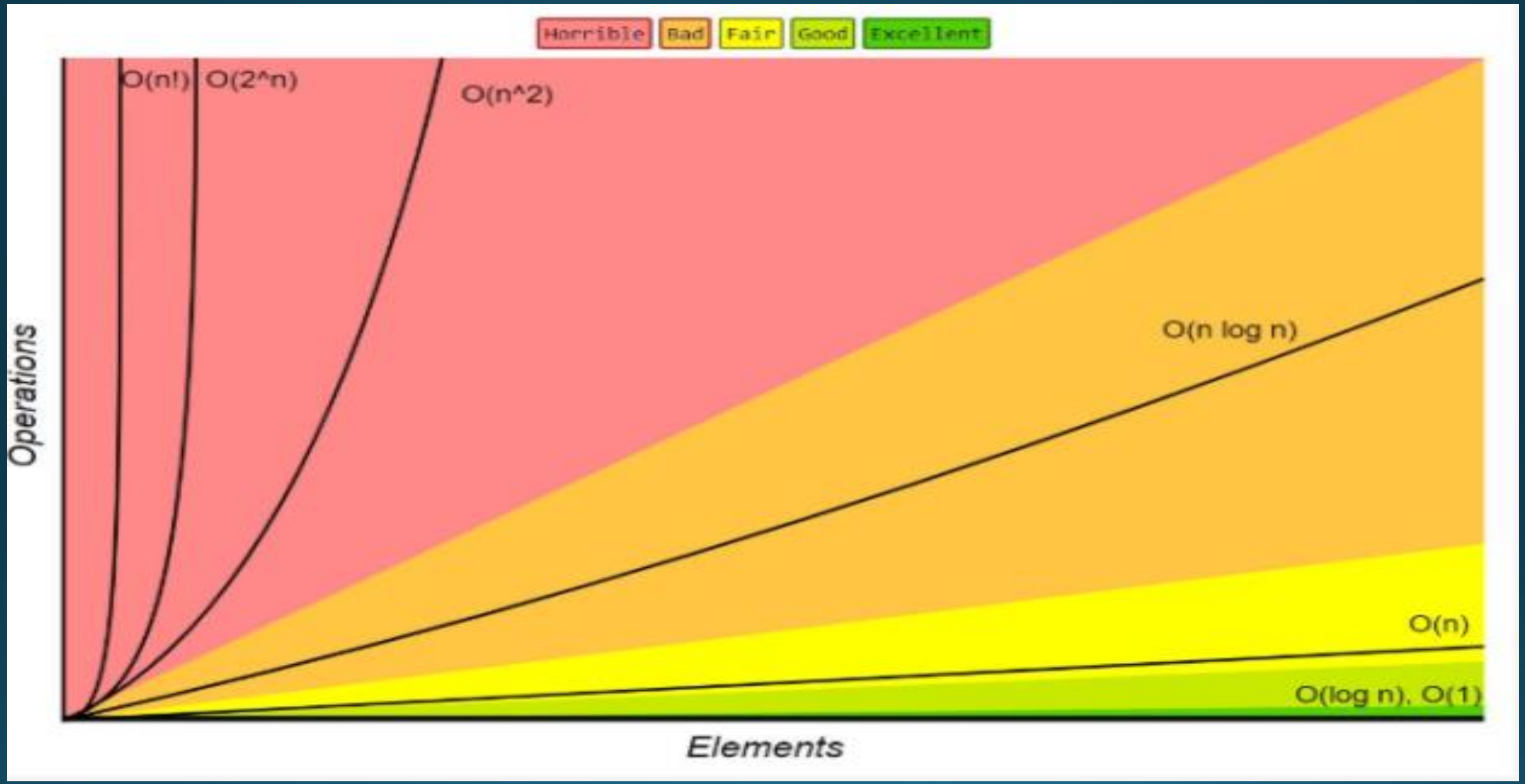
The time complexity of an algorithm is given by the number of steps taken by the algorithm to compute the function. The number of steps is computed as a function of some subset of number of inputs and outputs and magnitudes of inputs and outputs.

Count the number of steps in the sum of list elements

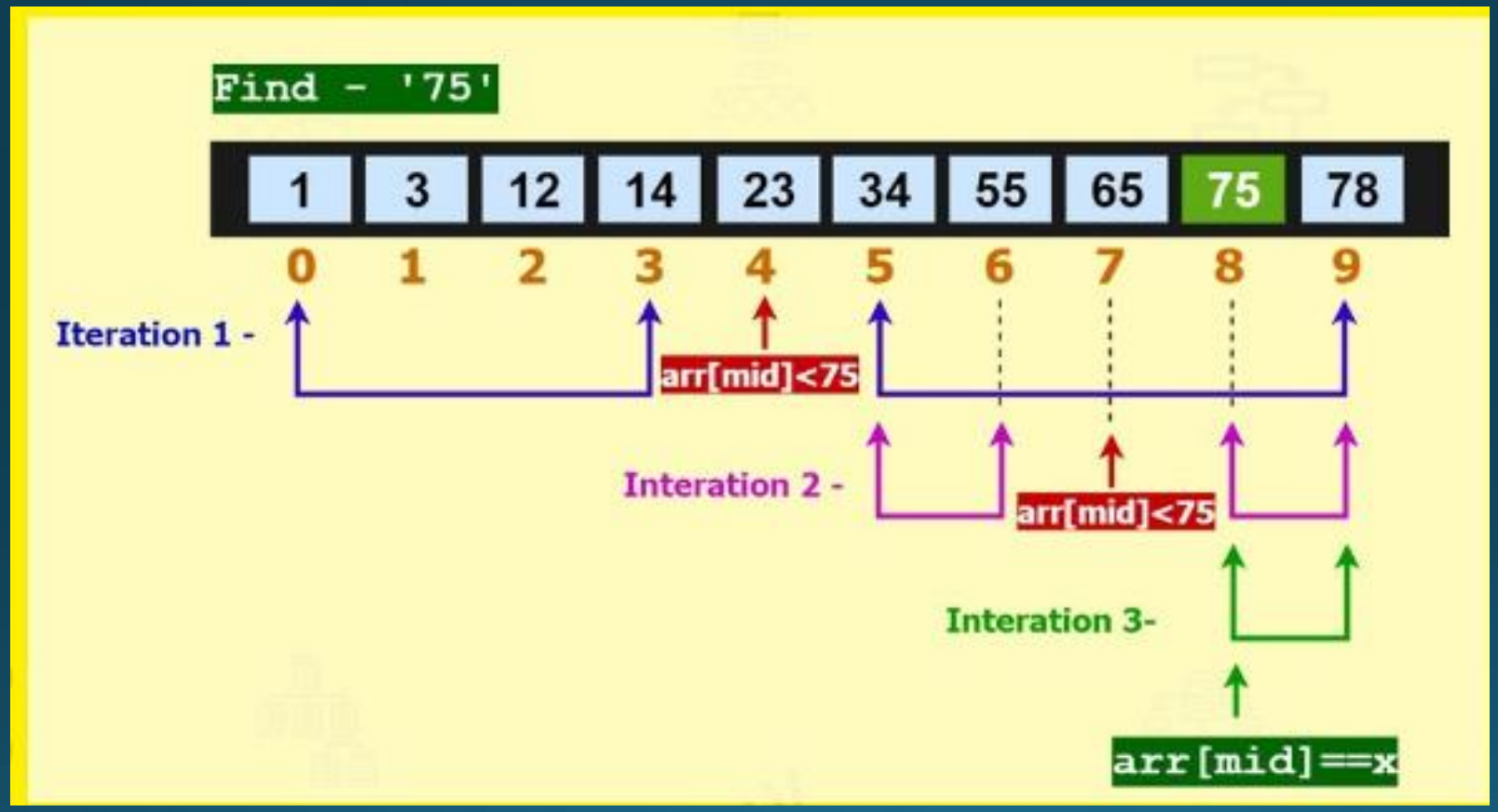
Big-Oh Notation

- defines an upper bound of an algorithm; it bounds a function only from above.
- used widely to characterize running time and space bounds in terms of some parameter n, which varies from problem to problem.
- Constant factors and lower order terms are not included in the big-Oh notation.

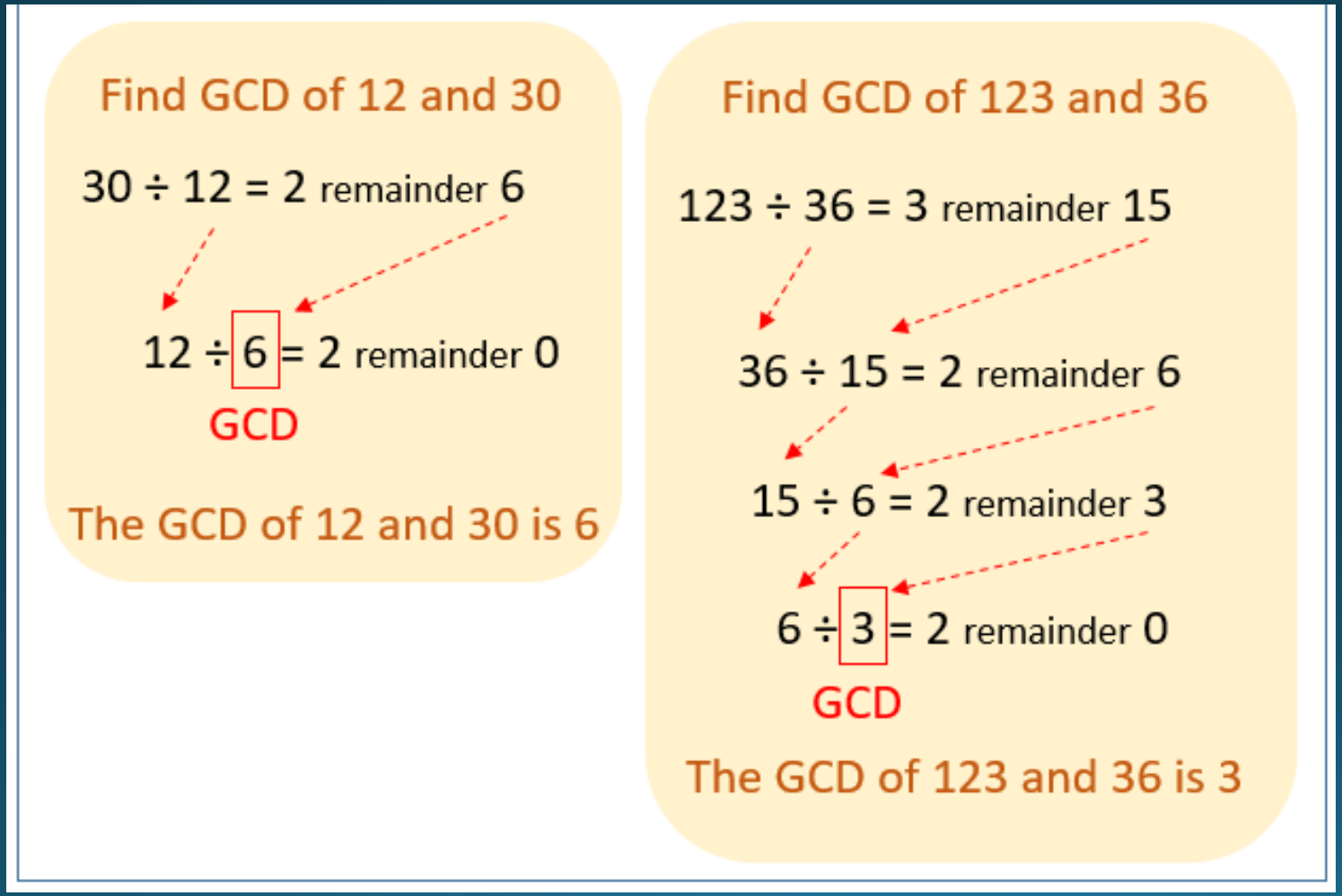



Horrible
Bad
Fair
Good
Excellent
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(log n), O(1)
Operations
Elements

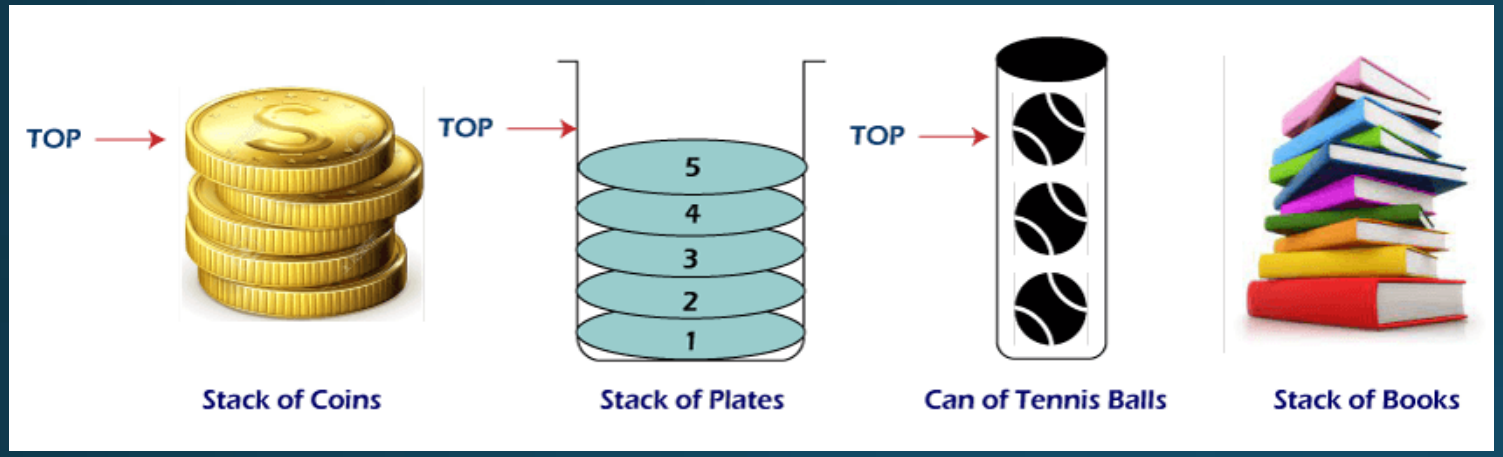
Binary Search

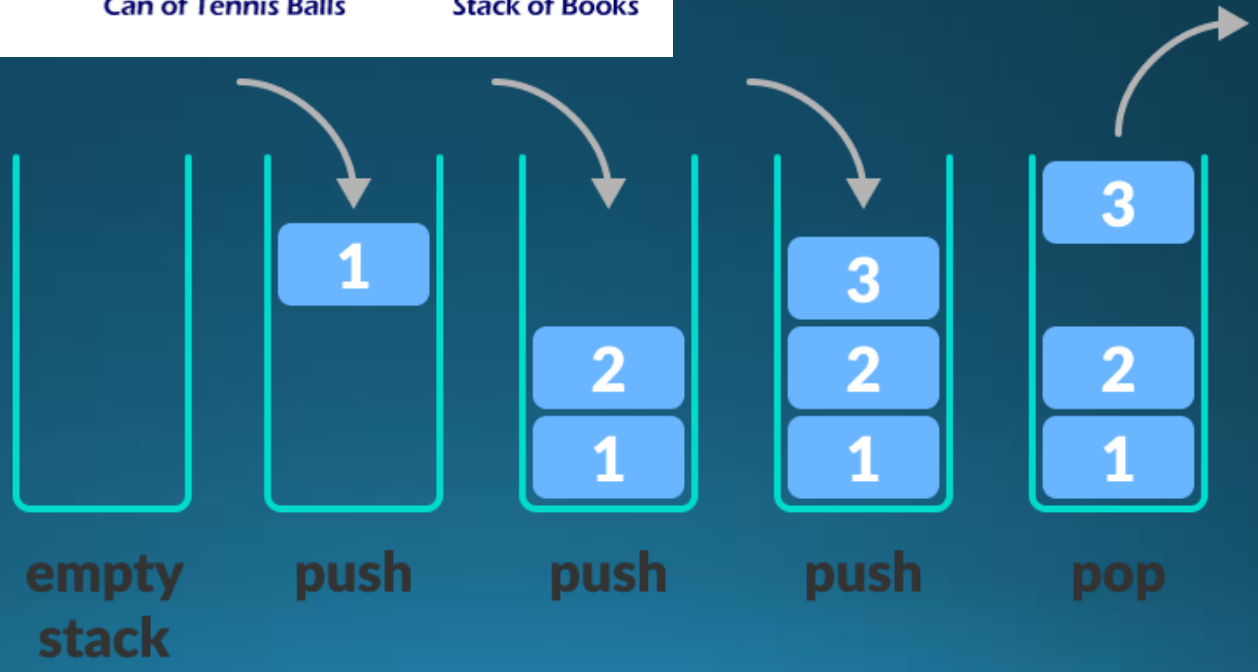


Euclids Algorithm



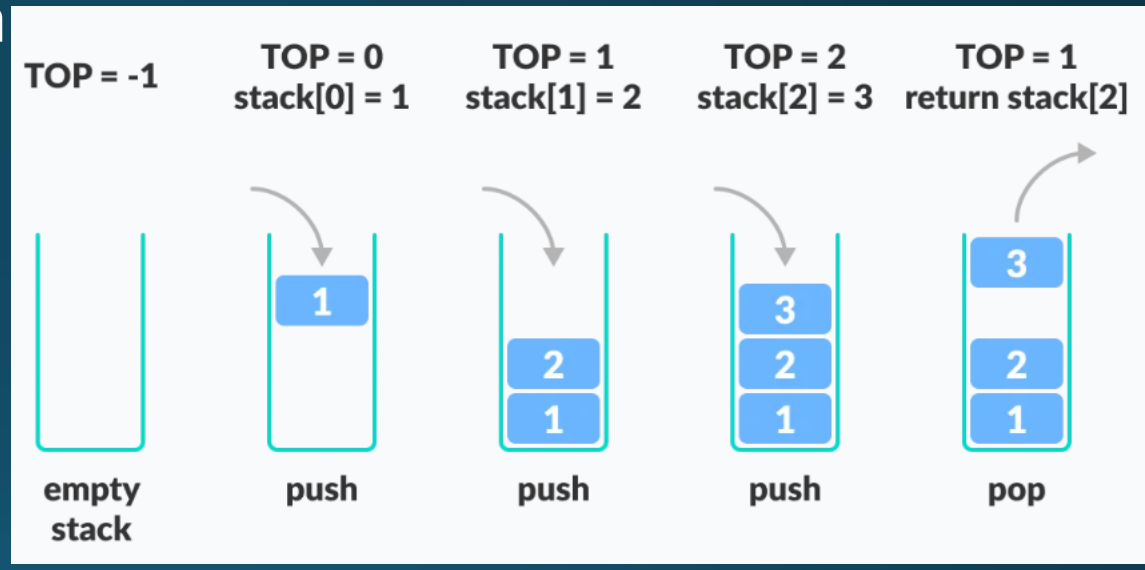
Stack data structure



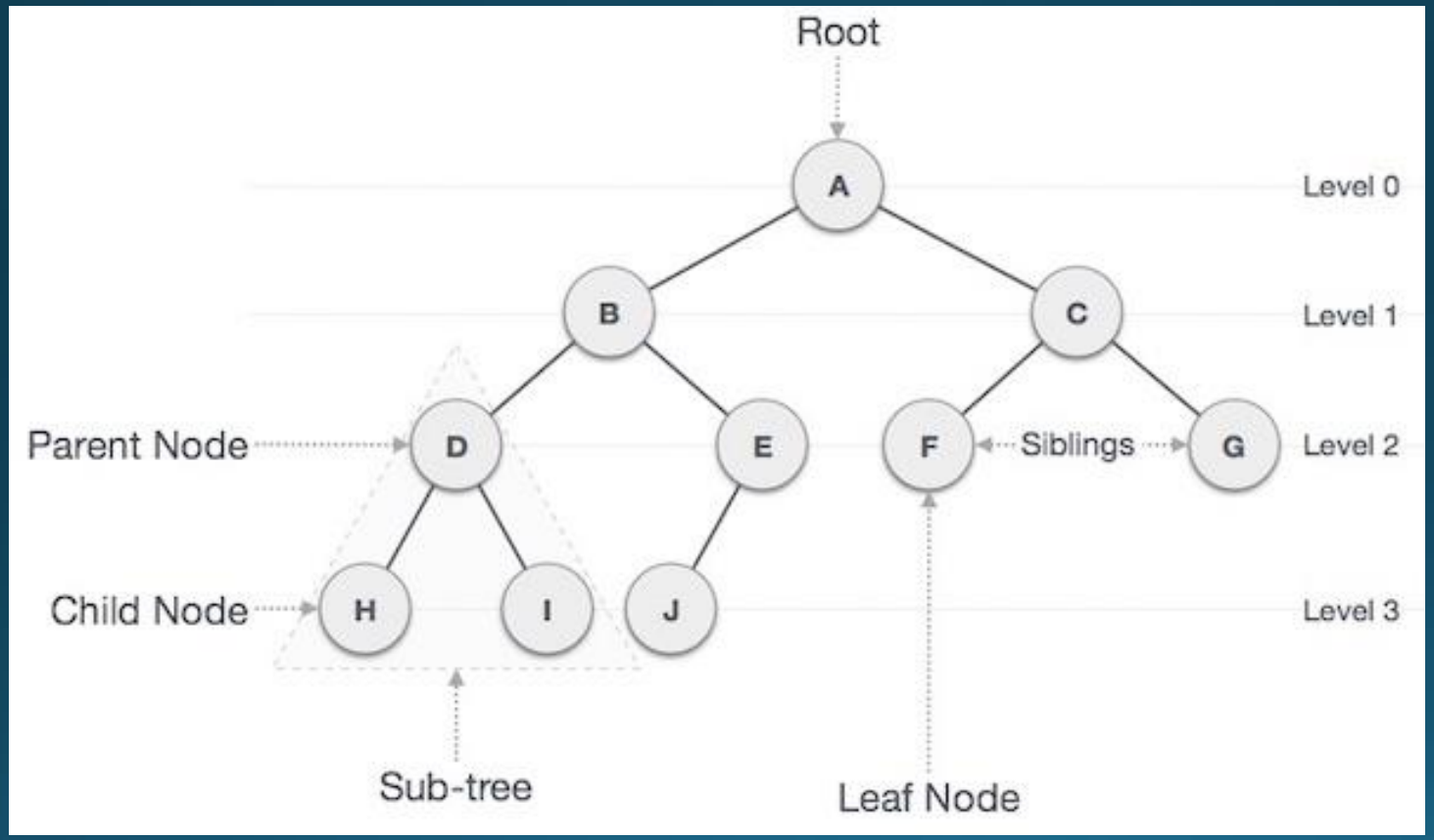


Operations on stack

- Push: Add an element to the top of a stack
- Pop: Remove an element from the top of a stack
- IsEmpty: Check if the stack is empty
- IsFull: Check if the stack is full
- Peek: Get the value of the top element without removing it

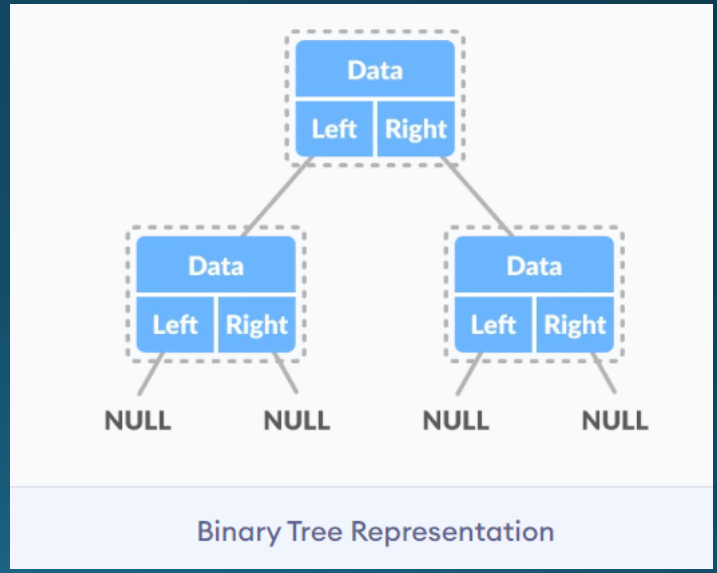


Binary tree


Root
A
B
C
D
E
F
G
H
I
J
Level 0
Level 1
Level 2
Level 3
Parent Node
Siblings
Child Node
Sub-tree
Leaf Node

Binary Tree

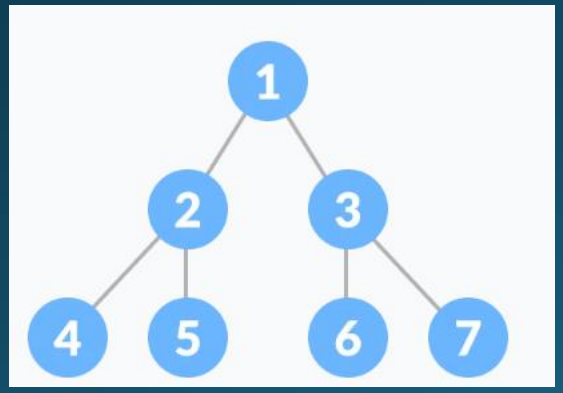
- A binary tree is a tree data structure in which each parent node can have at most two children. Each node of a binary tree consists of three items:
  - data item
  - address of left child
  - address of right child
- Binary Tree Applications
  - For easy and quick access to data
  - In router algorithms
  - To implement heap data structure

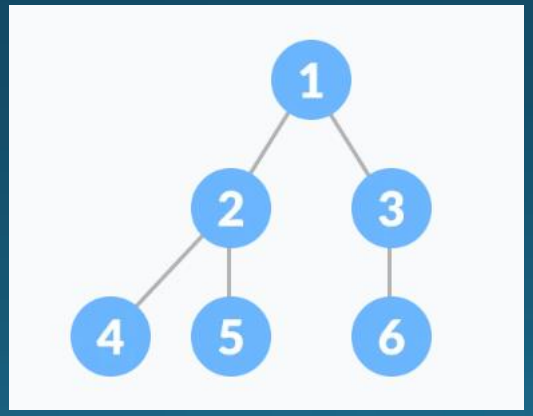


Binary Tree Representation

Complete Binary Tree

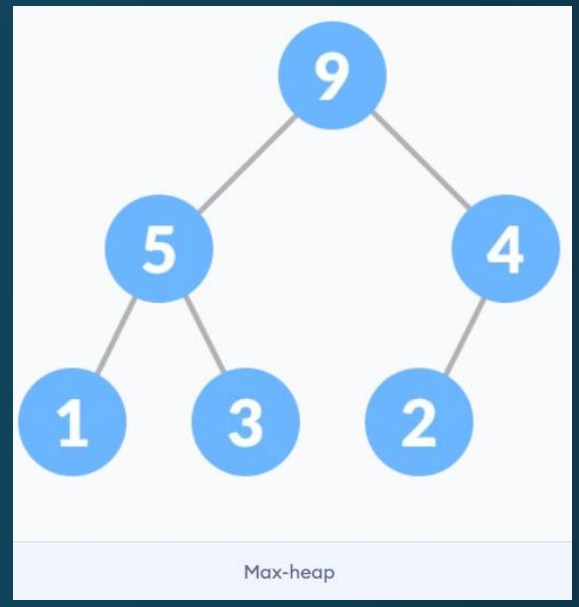
- A complete binary tree is just like a full binary tree, but with two major differences
  - Every level must be completely filled
  - All the leaf elements must lean towards the left.

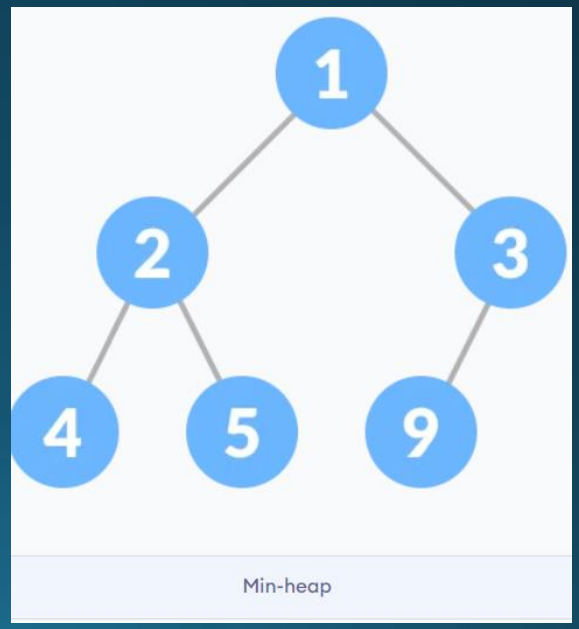




Heap Data Structure

- Heap data structure is a complete binary tree that satisfies the heap property
- max heap property :
  - any given node is always greater than its child node/s and the key of the root node is the largest among all other nodes.
- min heap property :
  - any given node is always smaller than the child node/s and the key of the root node is the smallest among all other nodes. This property is also called min heap property.
- Heap operations
  - Heapify
  - Heapify is the process of creating a heap data structure from a binary tree. It is used to create a Min-Heap or a Max-Heap.
  - Insert element into Heap
  - Deletion of element from the Heap
  - Finding minimum or maximum





Memory Management

- Memory management is
  - important to work efficiently.
  - related to writing memory-efficient code.
- If the memory is not handled well, it will take much time while preprocessing the data.
- The procedure of providing memory to objects is called allocation
- python follows Dynamic Memory Allocation.
- Python removes those objects that are no longer in use or can say that it frees up the memory space. This process of vanish the unnecessary object's memory space is called the Garbage Collector. The Python garbage collector initiates its execution with the program and is activated if the reference count falls to zero.
- To manually remove objects
  - import gc
  - gc.collect()

- Best Practices – Keeping it simple, dry code, naming Conventions, Comments and docs.

Best practices

- Write well structured code
  - having a well-structured code with proper names of the modules, correct indentations, and documentation improves the code's usability in the future.
  - Especially while making projects, you must include
    - a README file for describing your project,
    - the setup.py file for properly setting up your project in a new environment,
    - the requirements.txt for describing the dependencies that are needed. and
    - documentation describing how all the components of your projects work.
- Having proper comments and documentation
- Proper naming of variables, classes, functions and modules
- Write modular code
- Using virtual environments
  - The main purpose of a virtual environment in Python is to create a separate environment for Python projects.

Best Practices – Writing Clean code

- clean code is
  - focused
  - easy to read
  - easy to debug
  - easy to maintain
  - highly performant
- patterns for writing clean code
  - naming convention
    - use long descriptive names that are easy to read.
    - use descriptive intention revealing names
    - Avoid using ambiguous shorthand.
  - Functions should do one thing and do it well.

```
# Not recommended
# The au variable is the number of active users
au = 105

# Recommended
total_active_users = 105
```

```
# Not recommended
c = [“UK”, “USA”, “UAE”]

for x in c:
print(x)

# Recommended
cities = [“UK”, “USA”, “UAE”]
    for city in cities:
        print(city)
```

```
# Not recommended
fn = 'John'
Ln = ‘Doe’
cre_tmstp = 1621535852

# Recommended
first_name = ‘JOhn’
Las_name = ‘Doe’
creation_timestamp = 1621535852
```

Don’t write dry code

- The DRY or “Don't Repeat Yourself” principle is a software development practice aimed at reducing repetition of information.