**SOLUTION** N

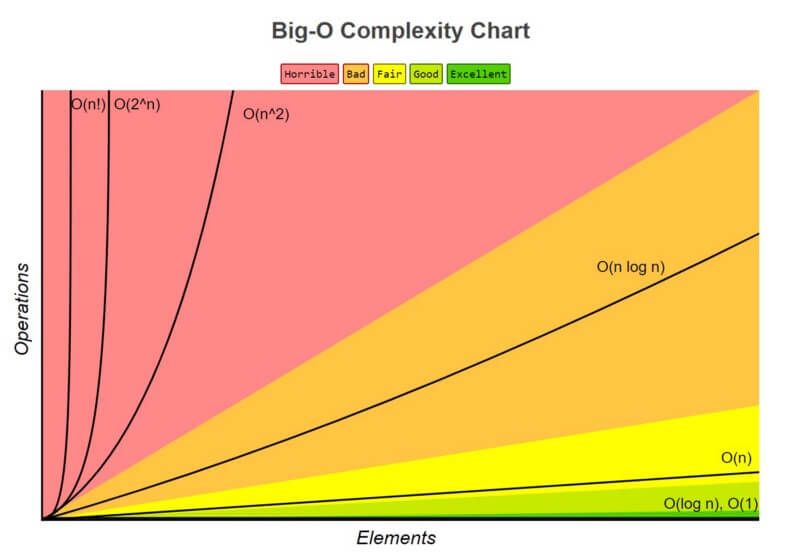
COMPLEXITY

* Complexity Analysis determines the amount of time and space resources required to execute it.
* It is used for comparing different algorithms on different input sizes.
* Complexity helps to determine the difficulty of a problem.
* Often measured by how much time and space (memory) it takes to solve a particular problem

*Time Complexity*

It is defined as the amount of time taken by an algorithm to run as a function of the length of the input.

#Note :- the time to run is a function of the length of the input and **not** the actual execution time of the machine on which the algorithm is running on.



*Space Complexity*

* It is the total space taken by the algorithm with respect to the input size.
* Space complexity includes both Auxiliary space and space used by input.
* It’s a parallel concept to time complexity

Time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of length of the input. While, the space complexity of an algorithm quantifies the amount of space or memory taken by an algorithm to run as a function of the length of the input.

*Optimization*

Optimization means modifying the brute-force approach to a problem. It is done to derive the best possible solution to solve the problem so that it will take less time and space complexity.

* We can reduce the time taken to run the program and increase the space occupied;
* we can reduce the memory usage of the program and increase its total run time, or
* we can reduce both time and space complexity by deploying relevant algorithms



DATA STRUCTURES

A data structure is a storage that is used to store and organize data. It is a way of arranging data on a computer so that it can be accessed and updated efficiently.

A data structure is not only used for organizing the data. It is also used for processing, retrieving, and storing data.

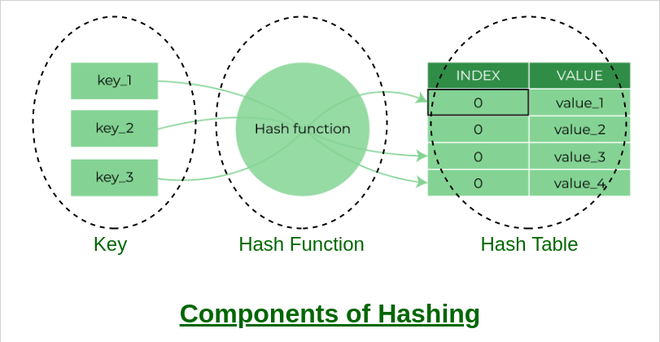


Classification of Data Structure

* Linear Data Structure: Data structure in which data elements are arranged sequentially or linearly, where each element is attached to its previous and next adjacent elements, is called a linear data structure.
  + Example: Array, Stack, Queue, Linked List, etc.
* Static Data Structure: Static data structure has a fixed memory size. It is easier to access the elements in a static data structure.
  + Example: array.
* Dynamic Data Structure: In dynamic data structure, the size is not fixed. It can be randomly updated during the runtime which may be considered efficient concerning the memory (space) complexity of the code.
  + Example: Queue, Stack, etc.
* Non-Linear Data Structure: Data structures where data elements are not placed sequentially or linearly are called non-linear data structures. In a non-linear data structure, we can’t traverse all the elements in a single run only.
  + Examples: Trees and Graphs.

HASHMAPS

* *Hashing* is a technique used in data structures that efficiently stores and retrieves data in a way that allows for quick access.
* It involves mapping data to a specific index in a hash table using a *hash function* that enables fast retrieval of information based on its key.
* This method is commonly used in databases, caching systems, and various programming applications to optimize search and retrieval operations.
* Hashing achieves all three operations (search, insert and delete) in O(1) time on average.



* Hash Function: You provide your data items into the hash function.
* Hash Code: The hash function crunches the data and give a unique hash code. This hash code is typically integer value that can be used an index.
* Hash Table: The hash code then points you to a specific location within the hash table.

A hash table is also referred as a hash map (key value pairs) or a hash set (only keys). It uses a hash function to map keys to a fixed-size array, called a hash table. This allows in faster search, insertion, and deletion operations.

The hash function is a function that takes a key and returns an index into the hash table. The goal of a hash function is to distribute keys evenly across the hash table, minimizing collisions.

Common hash functions include:

* Division Method: Key % Hash Table Size
* Multiplication Method: (Key \* Constant) % Hash Table Size
* Universal Hashing: A family of hash functions designed to minimize collisions

Causes of Hash Collisions:

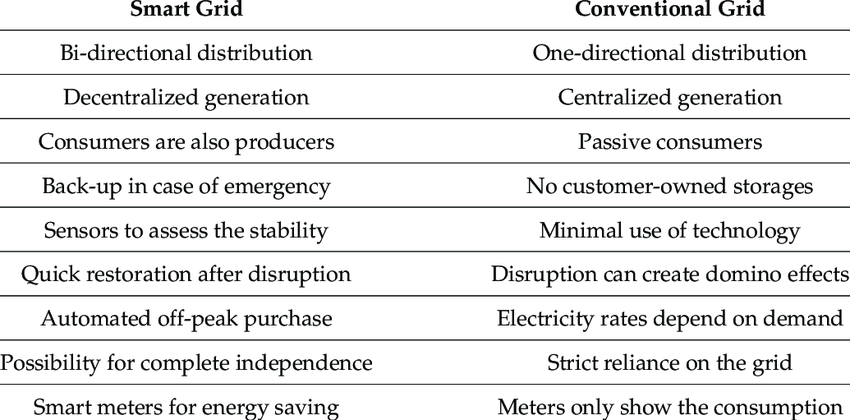
* Poor Hash Function: A hash function that does not distribute keys evenly across the hash table can lead to more collisions.
* High Load Factor: A high load factor (ratio of keys to hash table size) increases the probability of collisions.
* Similar Keys: Keys that are similar in value or structure are more likely to collide.

Collision Resolution Techniques

* Open Addressing:
  + Linear Probing: Search for an empty slot sequentially
  + Quadratic Probing: Search for an empty slot using a quadratic function
* Closed Addressing:
  + Chaining: Store colliding keys in a linked list or binary search tree at each index
  + Cuckoo Hashing: Use multiple hash functions to distribute keys

SMART GRIDS

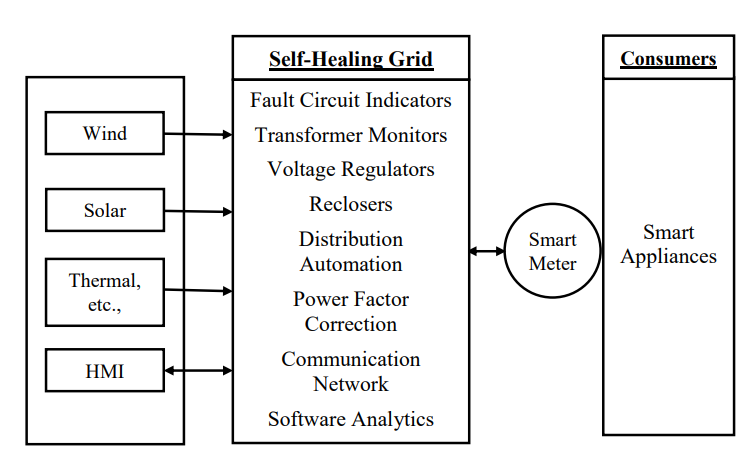
* A smart grid is an electric power grid that:
  + Establishes a communication network between the power supplier and consumer.
  + Utilizes smart sensors, smart meters, electric vehicles, and power-generating utilities.
* The smart grid incorporates an energy management system that:
  + Helps balance energy demand and supply.
  + Promotes efficient electricity production and consumption.
  + Reduces overall costs.



*Self-Healing*

A smart grid automatically detects and responds to routine problems and

quickly recover sif they occur, minimizing downtime and financial loss.



A smart grid automatically detects and responds to routine problems and

quickly recovers if they occur, minimising downtime and financial loss.

The Self-Healing Grid is a system comprised of sensors, automated controls,

and advanced software that utilizes real-time distribution data to detect and

isolate faults and to reconfigure the distribution network to minimize the

customers impacted.

One of the main goals of a Self-Healing Grid is to improve system reliability.

This can be accomplished by reconfiguring the switches and reclosers installed

on the distribution feeder to quickly isolate the faulted section of the feeder and

re-establish service to as many customers as possible from alternate

sources/feeders.

*Need for establishment of Smart Grid*

* Higher Penetration of renewable resources or distributed generation
* Extensive and effective communication overlay from generation to

consumers

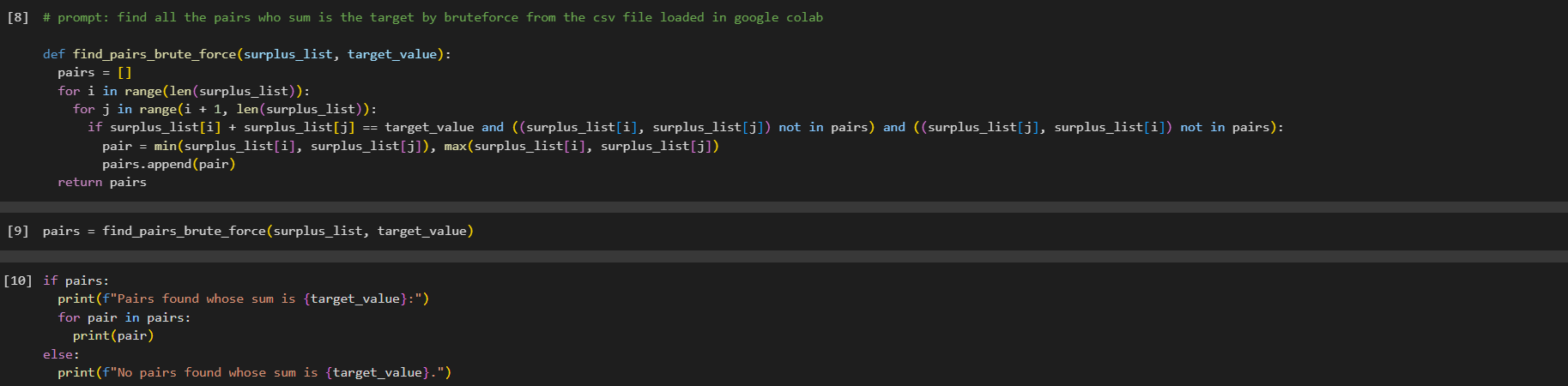
* Use of advanced sensors and high speed control
* Higher operating efficiency.
* Greater resiliency against attacks and natural disasters
* Automated metering and rapid power restoration
* Provided greater customer participation

Code Snippet

*Hash Map/Table*



*Brute Force*

**