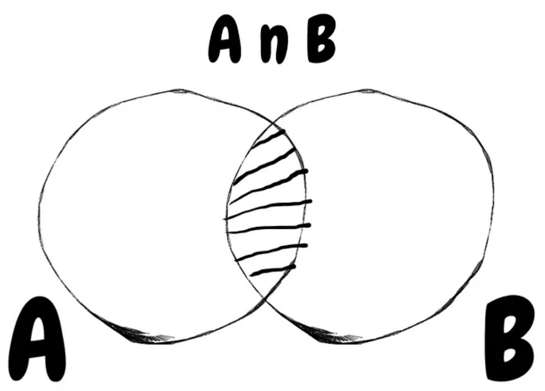
Set



💡 A set is a data structure that can store any number of `unique` values in any order you so wish. Set's are different from **arrays** in the sense that they only allow non-repeated, unique values within them.

> A set can be implemented in various ways but the most common ways are hash based and tree based.

→ **Hash-Based Set:** the set is represented as a hash table where each element in the set is stored in a bucket based on its hash code.

→ **Tree-based set:** In this implementation, the set is represented as a binary search tree where each node in the tree represents an element in the set.

Types of Set Data Structure:

The set data structure can be classified into the following two categories:

1. Unordered Set

An **unordered set** is an unordered associative container implemented using a hash table where keys are hashed into indices of a hash table so that the insertion is always randomized. All operations on the unordered set take constant time **O(1)** on an average which can go up to linear time **O(n)** in the worst case which depends on the internally used hash function, but practically they perform very well and generally provide a constant time lookup operation

2. Ordered Set

An Ordered set is the common set data structure we are familiar with. It is generally implemented using balanced BSTs and it supports O(log n) lookups, insertions and deletion operations.

Complexity Analysis of Operations on Set

→ **Insertion:** O(log n), where n is the number of elements in the set. This is because **std::set** is implemented as a balanced binary search tree, and inserting an element into a balanced binary search tree takes **O(log n)** time

→ **Searching:** O(log n), where n is the number of elements in the set.

→ **Deletion:** O(log n), where n is the number of elements in the set.

→ **Accessing the minimum/maximum element:** O(1), because std::set stores elements in sorted order, and the minimum and maximum elements can be accessed directly using the begin and end iterators.

→ **Size of the set: O(1),** because std::set stores the number of elements in a separate variable, and accessing this variable takes constant time.

Applications of Set Data Structure

Sets are abstract data types that can be used to store unique elements in a collection. Here are some common applications of sets:

→ **Removing duplicates:** If you have a large collection of data, you can use a set to easily remove duplicates and store only unique elements.

→ **Membership testing:** Sets provide efficient operations for checking if an element is a member of the set, which is useful in various algorithms and data structures.

→ **Set operations:** Sets can be used to perform operations such as union, intersection, and difference, which are useful in mathematical set theory and computer science.

→ **Graph algorithms:** Sets can be used to represent vertices and edges in graph algorithms, allowing for efficient operations such as finding connected components, detecting cycles, and computing minimum spanning trees.

→ **Cache replacement:** Sets can be used to implement cache replacement policies, where the oldest or least recently used elements are removed when the cache is full.

→ **Database indexing**: Sets can be used to implement indexes in databases, allowing for fast searching and retrieval of data based on specific keys or attributes.

Advantages of Set Data Structure

→ Set can be used to store unique values in order to avoid duplications of elements present in the set.

→ Elements in a set are stored in a sorted fashion which makes it efficient.→ Set is dynamic, so there is no error of overflowing of the set.

→ Searching operation takes **O(logN)** time complexity.

→ Sets provide fast and efficient operations for checking if an element is present in the set or not.

→ Sets can be implemented using different data structures, such as **HashSets** and **TreeSets**, each with its own advantages and use cases.

→ Sets can be used in a variety of applications, including algorithms, data analysis, and databases.

→ Sets can be used to improve performance in many algorithms by providing fast lookups.

Disadvantages of Set Data Structure:

→ Elements in a set can only be accessed with pointers, there is no indexing in a set like arrays.

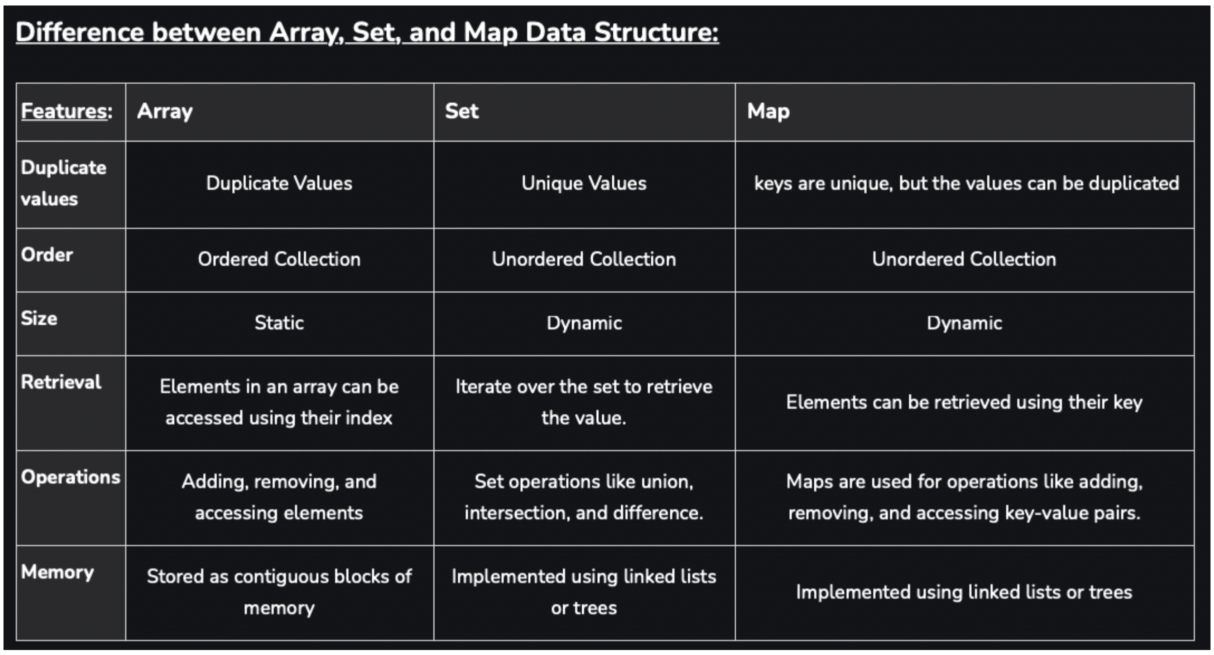
→ Set is very complex to implement because of its structure and properties.

→ A set takes **O(logN)** time complexity for basic operations like insertion and deletion.

→ Not suitable for large data sets.

→ Sets can only store elements of a specific data type.

→ Sets can use more memory than other data structures, such as arrays or lists because they store each element in a separate location.



Difference between Array, Set, and Map Data Structure:

Features:	**Array**	**Set**	**Map**
Duplicate values	Duplicate Values	Unique Values	keys are unique, but the values can be duplicated
Order	Ordered Collection	Unordered Collection	Unordered Collection
Size	Static	Dynamic	Dynamic
Retrieval	Elements in an array can be accessed using their index	Iterate over the set to retrieve the value.	Elements can be retrieved using their key
Operations	Adding, removing, and accessing elements	Set operations like union, intersection, and difference.	Maps are used for operations like adding, removing, and accessing key-value pairs.
Memory	Stored as contiguous blocks of memory	Implemented using linked lists or trees	Implemented using linked lists or trees

📚 References

GeeksforGeeks | A computer science portal for geeks

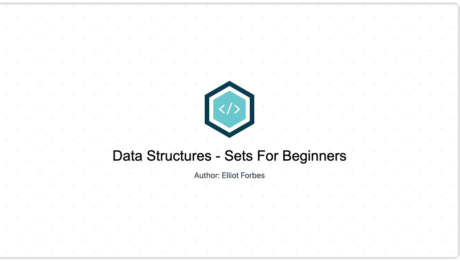
A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles, quizzes and practice/competitive

https://www.geeksforgeeks.org/

Data Structures - Sets For Beginners

In This Tutorial we look at sets, we look at how they work and what problems they can solve

https://tutorialedge.net/compsci/data-structures/sets-for-beginners/



✍🏼 Author → Serhat Kumas

https://www.linkedin.com/in/serhatkumas/

SerhatKumas - Overview

Computer engineering student who loves coding in different fields instead of focusing on a one spesific area. - SerhatKumas

https://github.com/SerhatKumas