[Data Structures And Algorithms](https://www.youtube.com/watch?v=RBSGKlAvoiM&ab_channel=freeCodeCamp.org)

A datastructure is a way of organising data in a organized fashion so that it can be used easily and efficently.

Why data structure ?

They are essential ingredients in creating fast and powerful algorithms.

**Abstract Data Types vs Data Structure**

An abstract data type is an abstraction of a data structure which provides only the interference to which a data structure must adhere to.

The integerface does not give any specific details about how something should be implemented or in what programming language.

**Eg:-**

| **Abstraction(ADT)** | **Implementation(DS)** |
| --- | --- |
| List | Dynamic Array, Linked List |
| Queue | Linked list based Queue,Array based queue,  Stack based queue |
| Map | Tree map, hash map/hash table |
| Vehicle | Golf cart, bicycle, smart car |

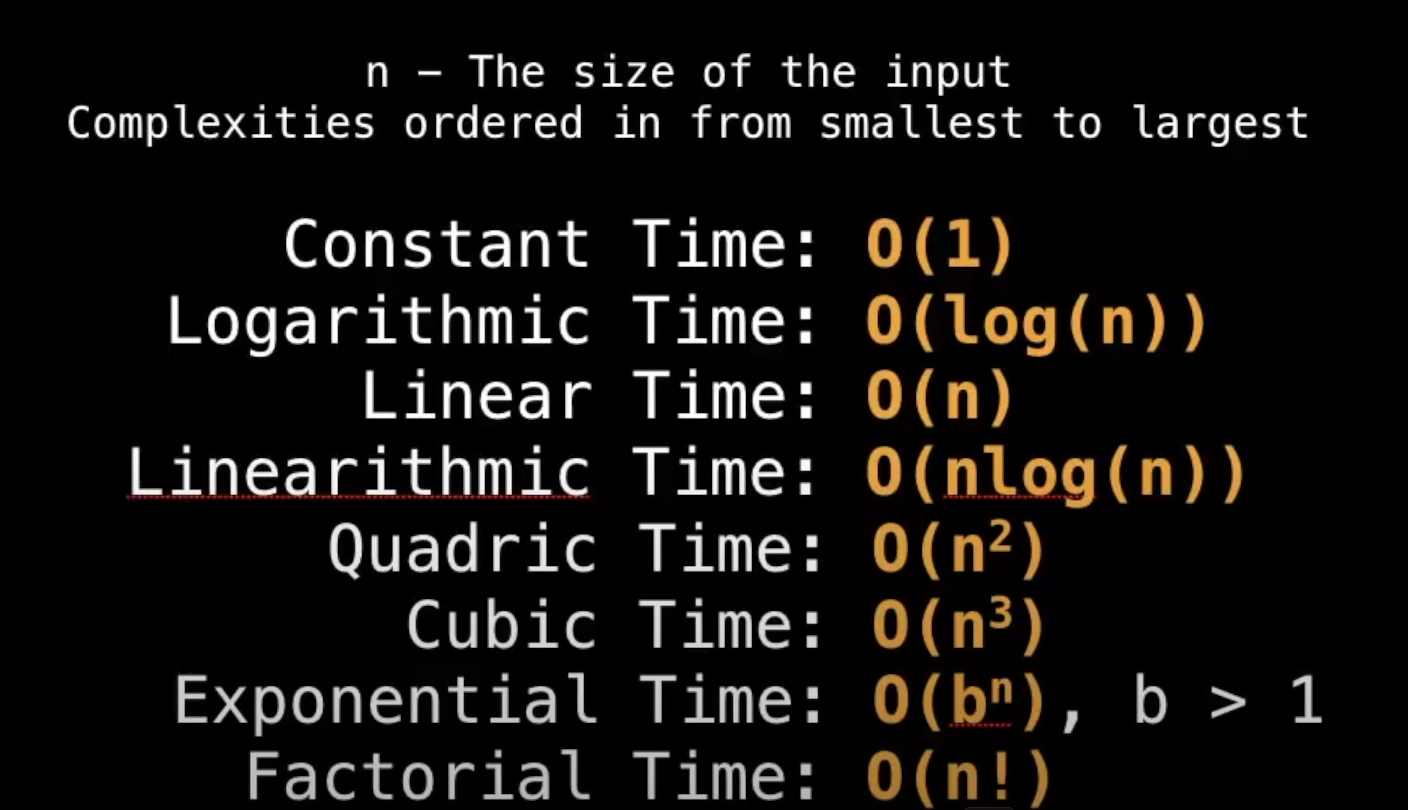
**Complexity Analysis**

Time and space

**Big-O Notation**

Big-O notation gives an upper bound of the complexity in the worst case, helping to quantify performance as the input size becomes arbitrarily large.

n - the size of the input



Let , f(n) = 7log(n^3) + 15n^2 + 2n^3 +8

Then ,

O(f(n)) = O(n^3)

Few examples:-

***🟢O(1) ;***

***A := 1***

***B := 2***

***C := a +5\*b***

***🟢O(n)***

***For i in range n;***

***i = i\*8+2***

***🟢O(n^2)***

***For i in range n:***

***i = i\*8+3***

***For j in range n:***

***J = j +1***

***🟢O(logn) binary search***

***Low := 0***

***High := n-1***

***While low <= high Do***

***Mid := (low/high)/2***

***If array[mid] == value ; return mid***

***Else if array[mid]< value: low = mid+1***

***Else if array [mid] >vlaue: high = mid-1***

***Return -1***

***Finding all subset of a set - O(2^n)***

***Finding all permutation of a string - O(n!)***

***Sorting all mergesort - O(nlog(n))***

***Iterating over all the cells in a matrix of size n by m - O(nm)***

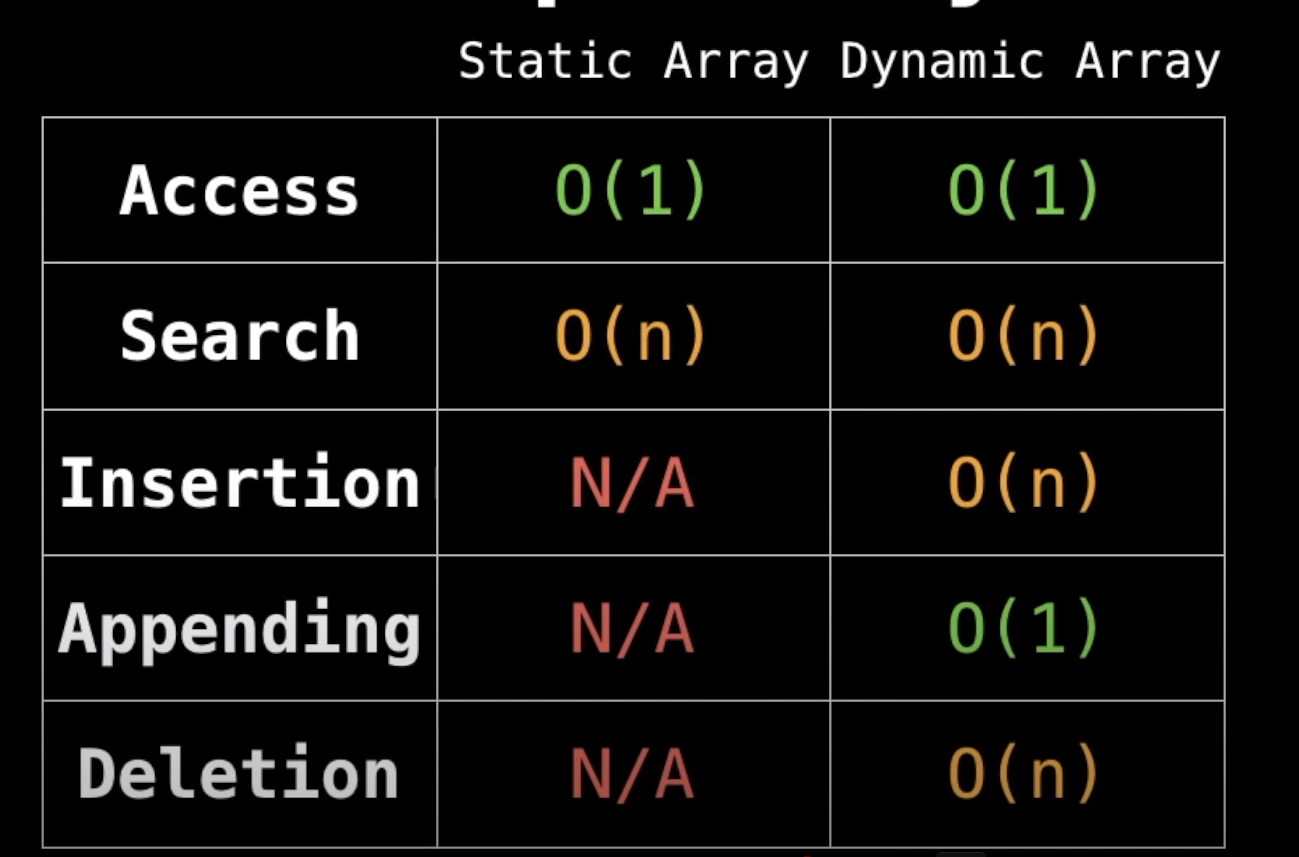
**Static and Dynamic Arrays**

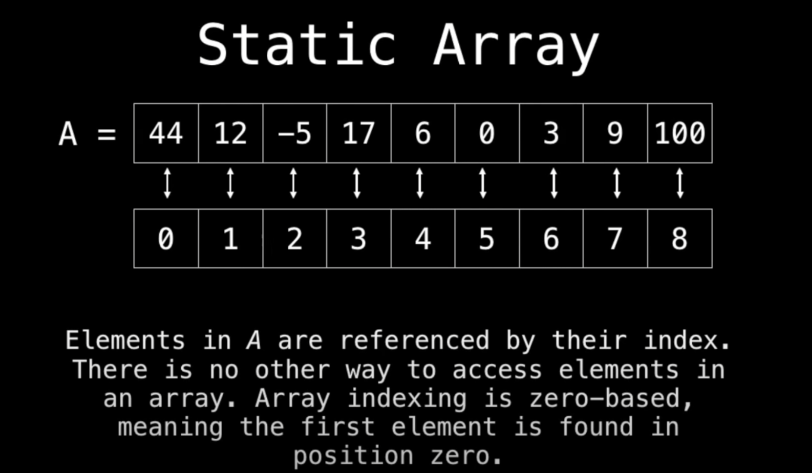
A static array is a fixed length container containing n elements indexable from range [0, n-1].

\*indexable mean that each slot’index can be referenced with a number.

Uses:-

* Storing and accessing sequential data.
* Temporarily storing object.
* Used by IO routines as buffers.
* Lookup tables and inverse lookup tables.
* Can be used to return multiple values from a function.
* Used in dynamic programming to cache answers to subproblems.

***Complexity in arrays:***



A[5] = 0

**Dynamic Array**

The dynamic array can grow and shrink in size.

eg:-

A = [3, 9]

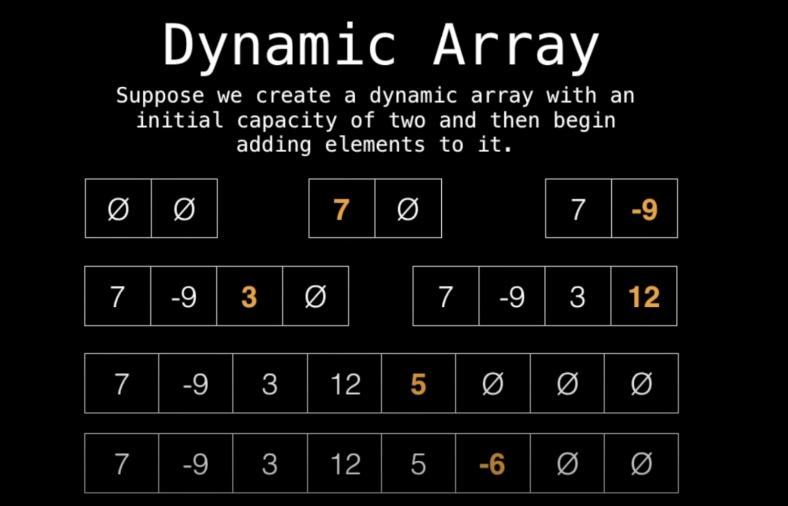
A.add(-7) A= [3, 9, -7]

A.remove(3)= [9,-7]

Implementation of an dynamic array:

One way is to use a static array.

* Create a static array with an initial capacity
* Add elements to the underlying static array, keeping track of the number of elements.
* If adding another element will exceed the capacity, then create a new static array with twice the capacity and copy the original elements into it.



[Implementation of dynamic arrays](https://github.com/atharvarya12/DsA-pRACtICE-AND-nOTES/blob/main/iMPImENTATION%20OF%20A%20dYNAMIC%20aRRAy.py)

**Linked Lists**

A linked list is a sequential list of nodes that holds data which points to another nodes also containing data.

****

**Uses**

List