RANKA
DATE / /
PAGE

ARRAY

(*) What is an Array?

Definition:

An array is a linear data structure that stores elements:

- of same data type
- in contiguous memory locations
- Accessed using an index

Example:

```
int arr[5] = {10, 20, 30, 40, 50};
```

(*) Array as an ADT (Important)

Before implementation, think ADT view

Array ADT defines:

Data:

- A finite ordered collection of elements of same type

Operations

- Create (size)
- access (index)
- update (index, value)
- traverse ()
- insert (index, value)
- delete (index)
- Search (value)

ADT does not say:

- Static or dynamic
- Stack or heap
- C or Python

RANKA
DATE / /
PAGE

(*) Memory Representation of Array

This is where C/C++ shine

Key property

Arrays use contiguous memory

if:

- Base address = B
- index = i
- Size of each element = w

Address formula

Address(arr[i]) = B + (i × w)

This is why arrays have O(1) access time

Types of Arrays

(1) One Dimensional Array

int arr[5];

- linear
- Single index

(2) Two dimensional Array

int arr[3][4]

- Matrix form
- Row-major (C/C++)

(3) Multi-Dimensional Array

int arr[2][3][4]

Operations On an Array

following Operations are Supported by an array

There can be many other Operations One can perform On arrays as well

eg:- Sorting asc, Sorting desc

Traversal

Insertion

Deletion

Search

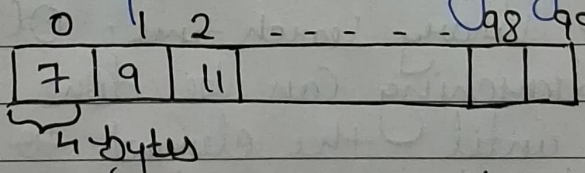
RANKA
DATE / /
PAGE

Traversal → Time: O(n)
visiting every element of an array
Once → Traversal

why traversal? → for use cases like:
→ Storing all elements → using Scanf
→ printing all elements → using printf

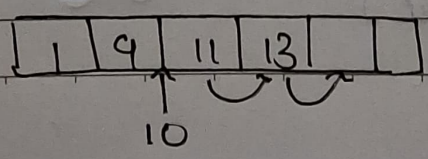
An important note about arrays
If we create an array of length 100 using a[100] in C language, we need not use all the elements. It is possible for a program to use just 60 elements out of these 100
→ But we cannot go beyond 100 elements

An array can easily be traversed using a for loop in C language



Insertion → Case I → At end → Time: O(1)
→ Case II → At Beginning → Time: O(n)
An element can be inserted in an array at a specified position

In Order for this Operation to be Successful, the array should have enough capacity

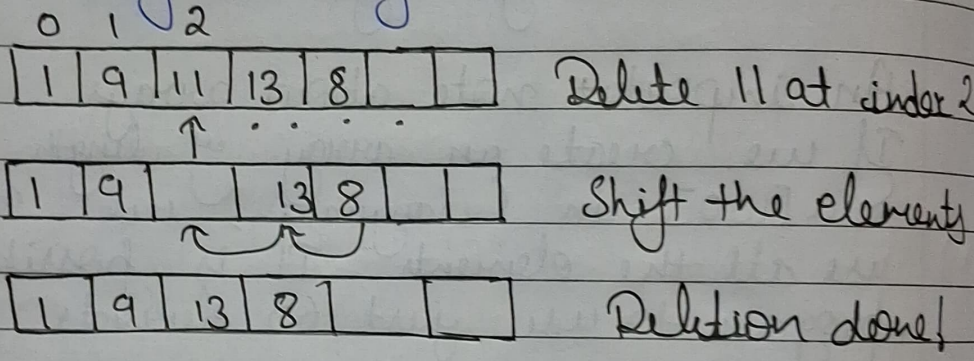


⇒ Elements need to be shifted to maintain relative order

when no position is specified its best to insert the element at the end

Deletion → Time: O(n)

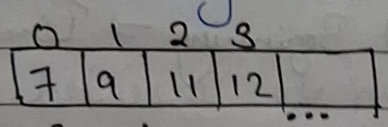
An element at specified position can be deleted creating a void which needs to be fixed by shifting all the elements to the left as follows:



We can also bring the last element of the array to fill the void if the relative ordering is not important

Searching → Linear Search time: O(n)
→ Binary search time: O(log n)

Searching can be done by traversing the array until the element to be searched is found



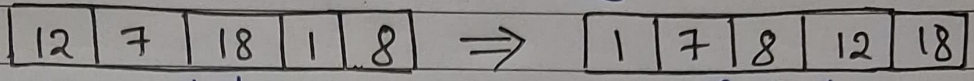
→ Search for sorted array time taken to search is much less than unsorted array!!

Sorting

Sorting means arranging an array in order (asc to desc.)

RANKA
DATE / /
PAGE

we will see various Sorting techique later



Unsorted array Sorted array

Static Vs Dynamic arrays

(*) Static Array

```
int arr[10];
```

- Size fixed
- Stack memory
- Faster
- Risk of overflow

(*) Dynamic array

```
int *arr = (int*) malloc (n* sizeof (int));
```

- Size decided at runtime
- Heap memory
- flexible
- Manual memory managment

Advantages of Arrays

- fast access
- Simple Structure
- Cache-friendly
- low memory Overhead

Disadvantages of Arrays

- fixed size (Static)
- Costly insertion/deletion
- memory wastage or Overfl-
- Homogeneous Elements Only

These limitation give birth to

- linked lists
- Vectors
- Deques

final Mental Model

Array ADT → Operation (insert, delete, access) → Rules (Ordered, homo) → Implementation (Static / dynamic)