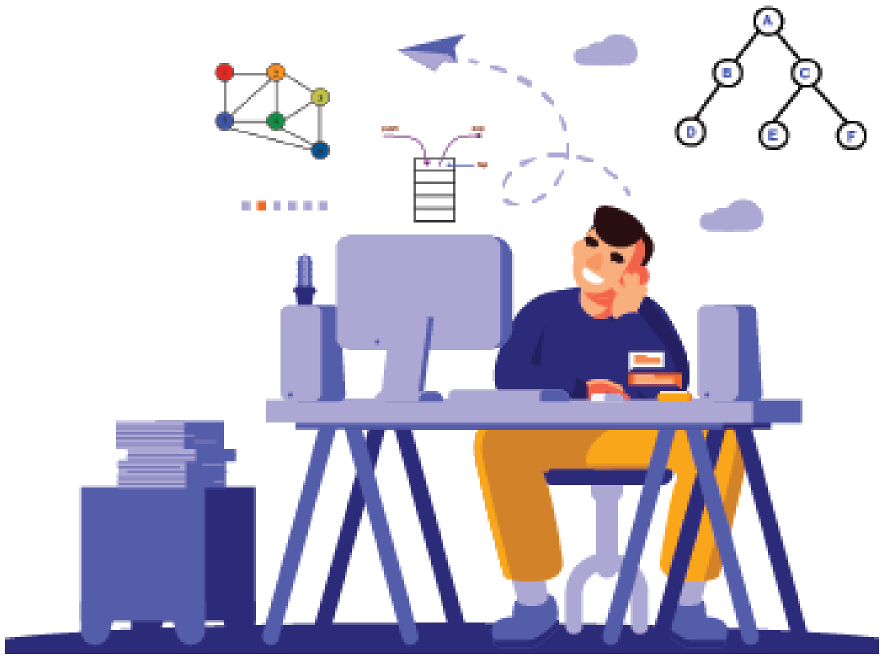
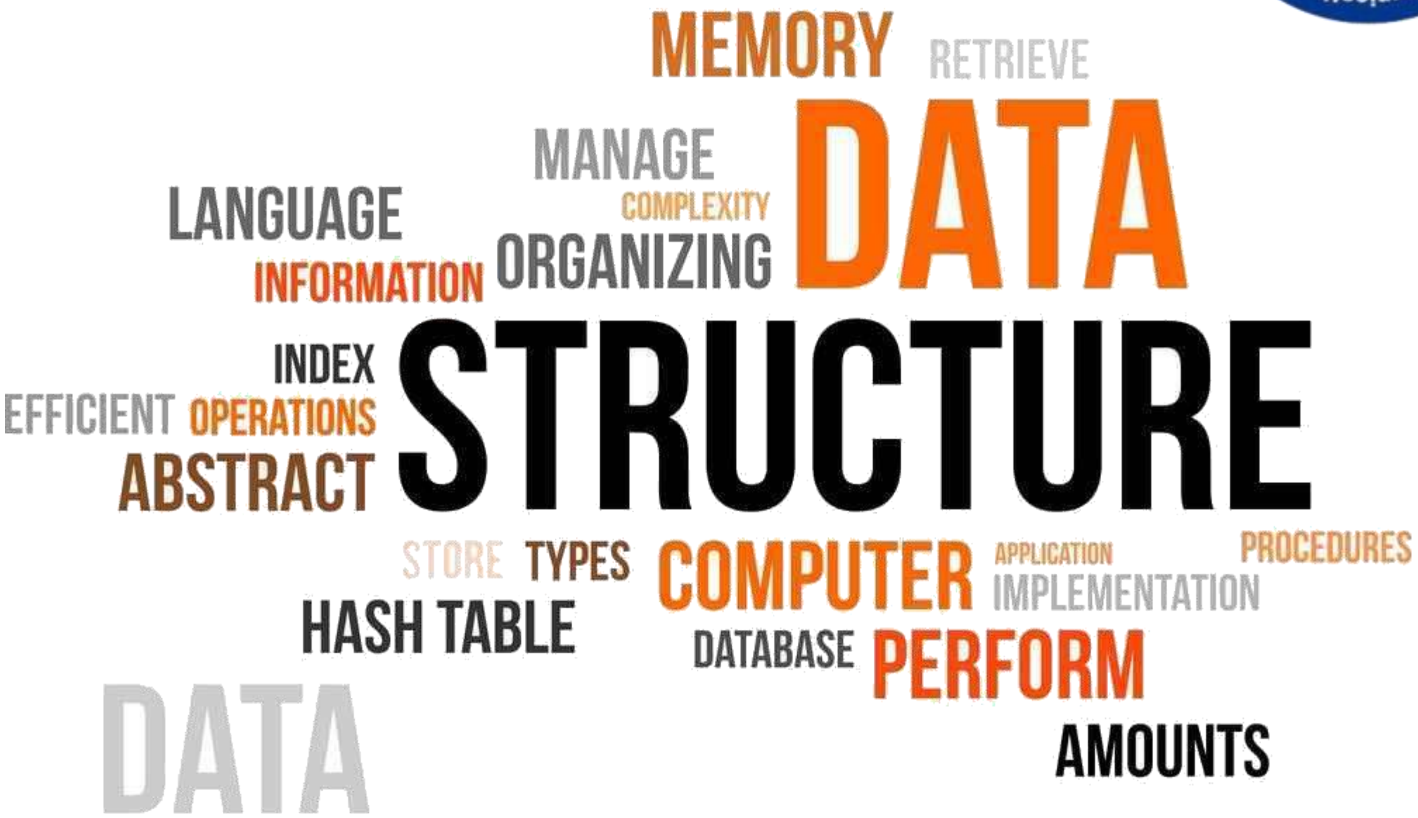


Data Structures

Course code: IT623



Dr. Rahul Mishra
Assistant Professor
DA-IICT, Gandhinagar

Array, Linear and Binary Search, and Matrices

Array

* We have studied linear and non-linear data structure.

* A data structure is said to be linear if its elements form a sequence, or, in other words, a linear list.

* Two basic ways to represent linear structures in memory

   a) One way is to have the linear relationship between the elements represented by means of sequential memory locations → arrays (This Chapter)

   b) Another is to have the linear relationship between the elements represented by means of pointers or links. ⟶ Linked list (Next chapter)

The operations one normally performs on any linear structure { array or linked list }

(a) Traversal : Processing each element

(b) Search : Finding the location

(c) Insertion : Adding new

(d) Deletion : Removing one

(e) Sorting : Arranging the elements in some type of order

(f) Merging : Combining two lists into a single list.

The particular linear structure that one chooses for a given situation depends on the relative frequency with which one performs these different operations on the structure.

Linear Array

A linear array is a list of a finite number (n) of homogeneous data element (i.e, data element of the same type) such that:

(a) The element of the array are referenced respectively by an index set consisting of (n) consecutive number.

(b) The element of the arrays are stored respectively in successive memory locations.

* Here, the number (n) of elements is called the length or size of the array.

* The length or the number of data elements of the array can be obtained as:

$$\boxed{\text{Length} = UB - LB + 1}$$

Representation we have already covered:

$A_1, A_2 \ldots A_n$

$A[1], A[2], \ldots, A[n]$

Representation

1> {DATA: 247, 56, 429, 135, 87, 156}

DATA[1] = 247
DATA[2] = 56
DATA[3] = 429
DATA[4] = 135
DATA[5] = 87
DATA[6] = 156

2>

	DATA
1	247
2	56
3	429
4	135
5	87
6	156

3>

DATA

247	56	429	135	87	156
1	2	3	4	5	6

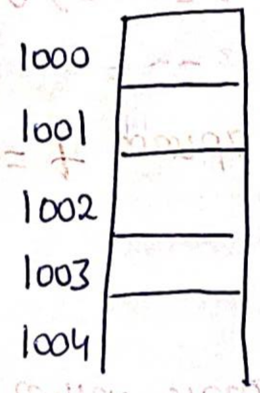
→ example

AUTO[K] = number of automatic sold in the year.

Representation of linear array in memory

> Let (LA) be a linear array in the memory of the computer.

$\boxed{LOC(LA[K])}$ = address of the element LA[K] of the array LA.



> The elements of LA are stored in successive memory cell. The computer does not need to keep track of the address of every element of LA, but need to keep track only of the address of the first element of LA. ⟶ Base(LA) base address

* Using Base (LA) ⟶ computer calculates the address of any element of LA using following relation:

$$LOC\ (LA[K]) = Base(LA) + w\ (K - \text{lower bound})$$ -(1)

where w is the number of words per memory cell for the array LA.

→ Scanning LA[K]th address is also easier.

Example:

Consider an array AUTO, which records the number of automobiles sold every year 1932 to 2023.

Base (Auto) = 200 and W = 4 words per memory cell for AUTO.

LOC (AUTO[1932]) = 200, LOC (AUTO[1933]) = 204, LOC (AUTO[1934]) = 208 ...

Determining the address of the array element for the year K = 1965 equation:

$$= Base(AUTO) + w(1965 - \text{lower bound})$$

$$= 200 + 4(1965 - 1932) = 332$$

LOC=BASE + w(Required-LB)

Do it for K = 200 ?

Consider the linear array — 5 minutes

A1(5:50), B1(-5:10) and C1(18)

(1) Find number of element in each array? { Length formula

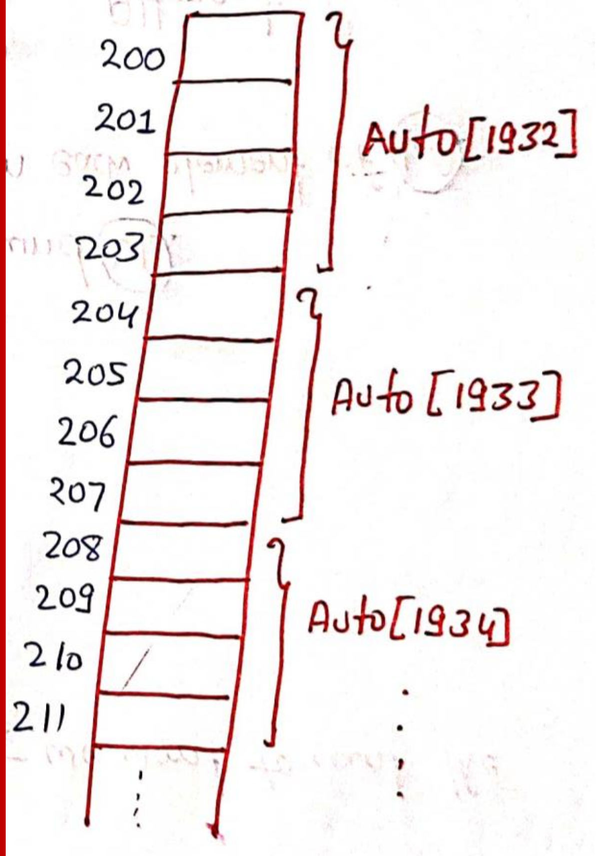
(2) Base(A1) = 300, w = 6

Find Address of?

A1[23] =

A1[37] =

A1[55] =



Traversing a Linear Array

→ Let A be a collection of data element stored in the memory.

→ We want to print the contents of each element of A; e.g., - we want to count the number of elements of A with a given property.

→ This is called Traversing.

>> Visiting each element of A exactly once.

@ Traversing a Linear Array: Procedure: 1

* LA is a linear array with lower bound (LB) and upper bound (UB).
* This procedure traverses LA applying an operation PROCESS on each element of LA.

1. [Initialize counter] Set K = LB;
2. Repeat steps 3 and 4 while K ≤ UB.
3. [Visit element.] Apply PROCESS to LA[K]
4. [Increase counter] Set K = K+1.
5. [End of step 2 Loop]
6. Exit.

Alternatively:
We can use repeat-for statement in place of while

Example

Consider previous example

(a) Find the number NUM of years during which more than 300 automobilies were sold.

1. [Initialization step.] Set NUM = 0
2. Repeat for K = 1932 to 2023

   IF AUTO[K] > 300, then: Set NUM := NUM + 1

   [End of loop]
3. Return

EXAMPLE OF "PROCESS"

(b) Print each year and the number of automobiles sold in that year.

1. Repeat for K = 1932 to 2023

   Write: K, AUTO[K]

   [End of loop]
2. Return

Inserting and Deleting

Inserting refers to the operation of adding another element to the collection (Array A).

Deleting refers to the operation of removing one element from A.

* Inserting at end of array is easier, whereas if we insert an element in the middle of the array

  → Average, half of the element must be moved downward to new locations to accomodate the new element and keep the order of the other elements.

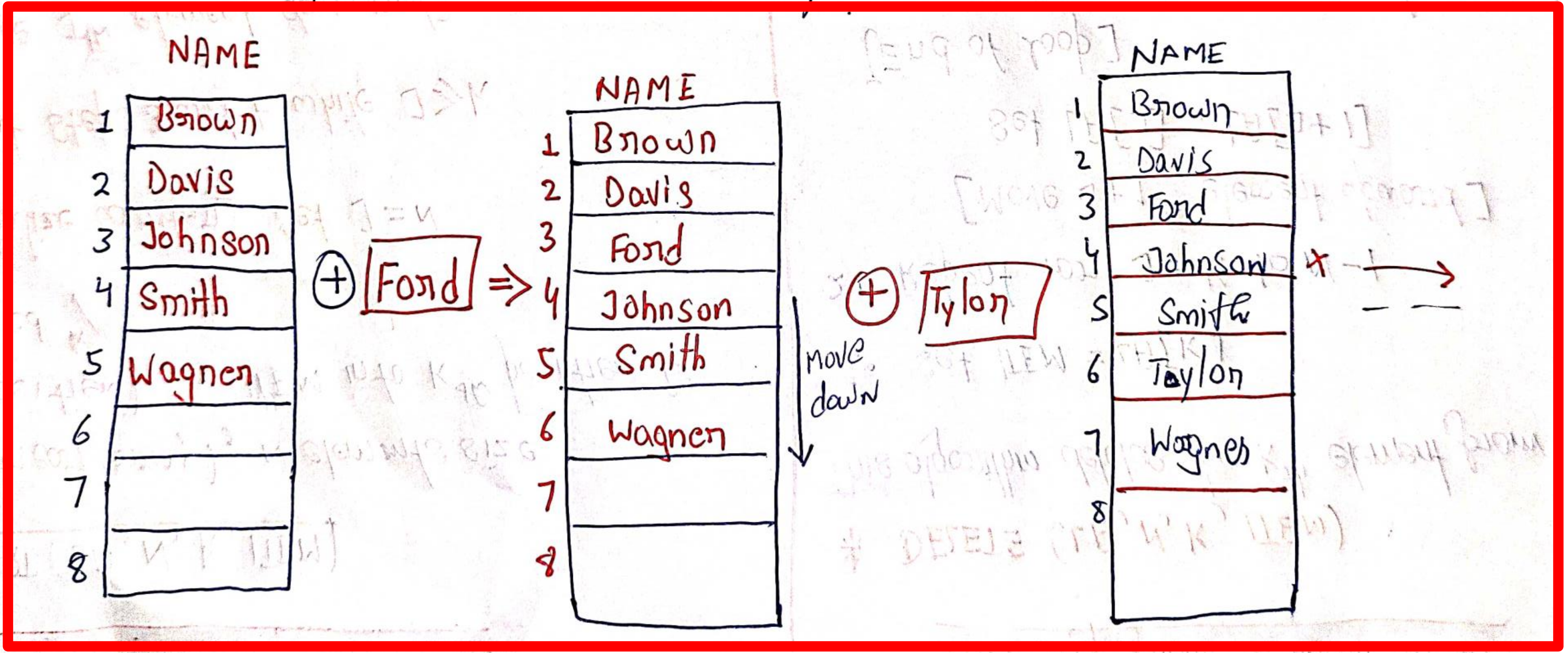
* In same manner, deleting an element from last is easier. However deleting an element somewhere in the middle array would require that each subsequent element be moved one location upward →

Example: 1

Suppose DSA has been declared to be an five-element array but data is recorded for first three element. Then we can add two more element's data. However, we cannot add any new (data) to the list[DSA].

Example: 2

Alphabetical Order is mandatory?



Inserting into linear array

* INSERT (LA, N, K, ITEM)

/* LA (Linear Array), N elements size
K (position) , ITEM into K^{th} position is
inserted */

1. [Initialize counter] Set J = N
2. Repeat Steps 3 and 4 while J ≥ K
3. [Move J^{th} element downword]
   Set LA[J+1] = LA[J]
4. [Decrease counter]
   set J = J-1
   [End of loop]
5. [Insert Element] set LA[K] = ITEM
6. [Reset N] set N = N+1
7. Exit.

Deleting from a linear array

* DELETE (LA, N, K, ITEM)

This algorithm deletes the K^{th} element from LA.

1. Set ITEM = LA[K]
2. Repeat for J = K to N-1
   [Move J+1^{st} element upward]
   Set LA[J] = LA[J+1]
   [End of loop]
3. [Reset the number N of elements in LA]
   Set N = N-1
4. Exit.