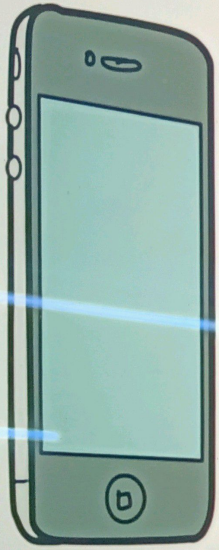


Mobile

Name = Mobile
Company = Apple
Price = 40000
Ram = 8 GB
Rom : = 64 GB
OS = iOS
Launching Date = 7th June 2020

Data :

Data are simply values or set of values.

Data Item : is a single unit of values.

1. Group Items : Data itmes those can divided into sub itmes.

Ex. Full Name : First Name , Middle Name, Last Name
Address : place, city, district, state, country
Date of Birth : Day, Month, Year

2. Elemenrty Items : Data items those can not be divided into sub items.

Ex. Age : 17
Mobile : 989xxxxxxx
Pin code : 442907

An entity :

An entity is something, that has certain attributes or properties which assigned values.

Name : Vinit Sharma
Age : 17
Gender : Male
Class: 12th
City : Warora
Date of Birth : 1st August 2003

Values can be
numeric,
non numeric, or
alpha numeric

Every attribute has certain value.
The collection of all entities form an entity set.

16

Age = 16

The term "information" is also used for data with given attributes.

Information means meanigful data.

Age = -45

Entity : Student

Name : Samiksha

Age : 18

Date of Birth : 14/09/2002

more attritbutes

Class :

City :

Gender :

Mobile No.

Feild : 'Field' is single elementary unit of information representing an attribute of entity.

Ex. Name, Gender, pincode, city etc.

Roll No. = 101
Name:
Gender :
Age:
DOB :
Mobile number:

Roll No. = 102
Name:
Gender :
Age:
DOB :
Mobile Number

Roll No. = 103
Name:
Gender :
Age:
DOB :
Mobile Number:

Record : A record is a collection of field values of a given entity

A record in a file may contain several fields.

File : A file is a collection of records of the entities.

There are certain fields having unique value. Such a field is called primary key.

In above file...
"Roll No". is a primary key

So data can be stored in feilds, records and files..

Apart from fields, records and files, the data can also be organised into more complex types of structures.
Data can be organised in different ways.

" The logical or mathematical model of a particular organisation is called a Data structure."

Linear Data structure

1. Array
2. Pointer array
3. Record
4. Linked list

Non-Linear Data structure

1. Tree
2. Binary Tree

Data Structure operations :

Array : Collection of homogenious(same type) elements is called as Array.

{ 8, 10, 5, 36, 7, 32, 4 } without decimal point (Integer Numbers)

{ 1.2, 6.3, 0.5, 9.5, 8.9 } with decimal point (Real Numbers)

Eg.

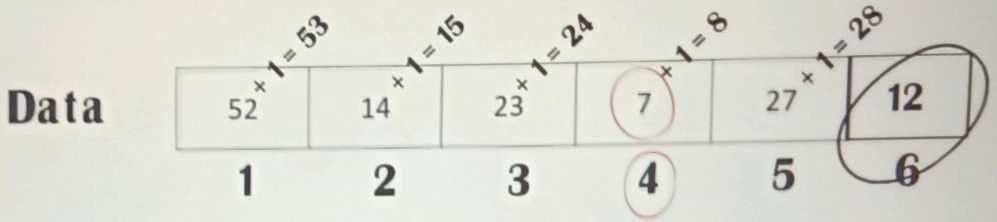
Data

Size / Length = 5

52	14	23	7	27
0	1	2	3	4
1	2	3	4	5

Index Number →

Data[1] = 52
Data[2] = 14
Data[3] = 23
Data[4] = 7
Data[5] = 27



Data Structure Operations : there are total six data structure operations.

1. **Traversing :** Accessing each element of Array(Structure) once so that it can be processed.
2. **Inserting :** Adding new element into the structure
3. **Deleting :** Removing element from the structure
4. **Searching :** Finding the location of element
5. **Sorting :** Arranging elements in some logical order { 7, 12, 14, 23, 27, 52 }
6. **Merging :** Combining two files into single file. { 4, 8, 3, 10 }

{ 7, 12, 14, 23, 27, 52, 4, 8, 3, 10 }

Algorithimic notation

An algoritm is a finite step by step list of well defined instructions for solving a particular problem,

Eg. **Addition of two int numbers**

Let A and B are two int numbers.

This algorithm will calculate SUM of these two numbers.

First Part
- Name of the Algorithm
- List of the variables
- Purpose of the Algorithm

Second Part

Step 1. Start
2. Read A and B — A = 10 B = 20
3. Calculate SUM = A + B — SUM = 10 + 20
4. Display SUM — 30
5. Stop

Stop statment complete the algorithm

List of steps

Steps are executed one after the other

Data /values may assigmed to variables by using read statement

Display statments displayed result or value of the variable

Arrays

1) Its the one of the linear data structure.
2) It is also called as Linear Array
3) It is a collection of finite n numbers of homogeneous data elements.
4) Data elements are of same type.
5) The elements of array are referenced by respectively by an index numbers.
6) The elements of array are stored respectively in successive memory locations.

Eg.

Marks

25	14	21	7
1	2	3	4

index numbers

LB
Lower Bound
Smallest Index Number

UB
Upper Bound
Highest Index Number

Elements of array can be denoted by subscript notaion

$Marks_1$, $Marks_2$, $Marks_3$, $Marks_4$

OR

Marks[1], Marks[2], Marks[3], Marks[4]

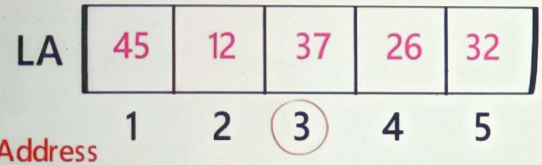
Marks[1] = 25
Marks[2] = 14
Marks[3] = 21
Marks[4] = 7

Representation of linear arrays in memory

1) Elements of linear array are stored in successive memory cells / blocks

2) The number of cells required depends on type of data elements.

Eg.



3) Computer keep track of address of the first element of array.

4) Here LA is linear array and address of first element is denoted by Base (LA) and called as base address of LA.

5) Using Base address computer calcuates the address of any elements of LA by using formlula -

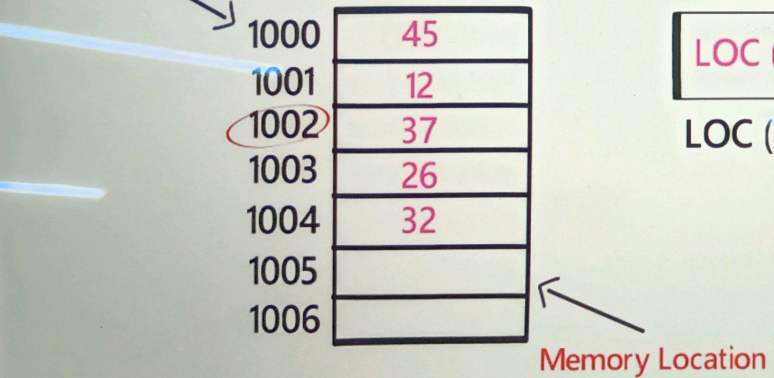


Fig. Memory representation of array

$$LOC(LA[K]) = Base(LA) + W(K - LB)$$

$$LOC(LA[3]) = 1000 + 1(3 - 1)$$
$$= 1000 + 2$$
$$= 1002$$

W = number of words (bytes) per memory cell

Travesrsing Linear Arrays :

Accessing each element of array only once so that it can be processed.

Algorithm : Traversing Linear Array.

Let LA is a linear array with lower bound LB and upper bound UB.
This algorithm will apply PROCESS to each element of LA.

Steps : 1) Start

2) Initialize counter
   set K = LB

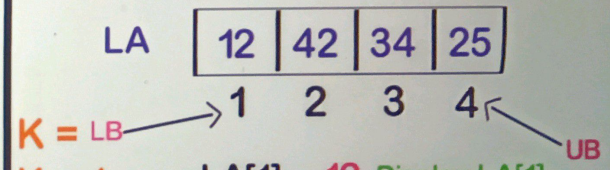
3) Repeat Steps 4th and 5th While (K <= UB)

4) Visit Element
   Apply PROCESS to LA[K]

5) Increment Counter
   K = K + 1

6) Stop

1 <= 4 ?	yes
2 <= 4 ?	yes
3 <= 4 ?	yes
4 <= 4 ?	yes
5 <= 4 ?	No



K = LB		
K = 1	LA[1]	12 Display LA[1]
K = 2	LA[2]	42
K = 3	LA[3]	34
K = 4	LA[4]	25
K = 5		

Sum of all the elements of the array

Let LA is a linear array of size N with lower bound LB and upper bound UB, this algorithm will find SUM of all the elements of the array.

Steps : 1) Start

2) Initialize SUM = 0

3) Initialize Counter
   set K = LB

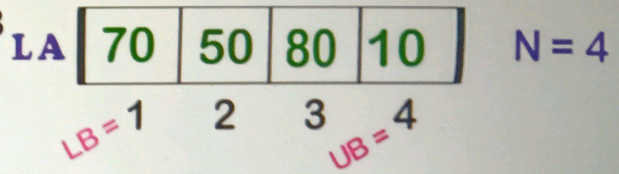
4) Repeat steps 5th & 6th while (K<= UB)

5) calculate SUM = SUM + LA[K]

6) Increment Counter
   K = K + 1

7) Display SUM

8) Stop



K = 1
(K<= UB)
(1 <= 4) ? YES
K = 2
(2 <= 4) ? YES
K = 3
(3 <= 4) ? YES
K = 4
(4 <= 4) ? YES
K = 5
(5 <= 4) ? NO

SUM = 0
SUM = SUM + LA[K]
SUM = 0 + LA[1]
SUM = 0 + 70
SUM = 70
SUM = SUM + LA[K]
SUM = 70 + LA[2]
SUM = 70 + 50
SUM = 120 + LA[3]
SUM = 120 + 80
SUM = 200 + LA[4]
SUM = 200 + 10
SUM = 210

Average of all the elements of the array

Let LA is a linear array of size N with lower bound LB and upper bound UB, this algorithm will find AVG all the elements of the array.

Steps : 1) Start

2) Initialize SUM = 0

3) Initialize Counter
   set K = LB

4) Repeat steps 5th & 6th while (K<= UB)

5) calculate SUM = SUM + LA[K]

6) Increment Counter
   K = K + 1

7) Display SUM

8) Calculate
   AVG = SUM / N

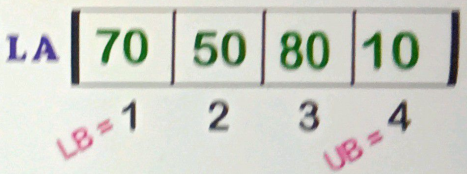
9) Display AVG

10) Stop

$$AVG = \frac{SUM}{N}$$

AVG = 210 / 4

AVG = 52.5



K = 1
(K<= UB)
(1 <= 4) ? YES
K = 2
(2 <= 4) ? YES
K = 3
(3 <= 4) ? YES
K = 4
(4 <= 4) ? YES
K = 5
(5 <= 4) ? NO

SUM = 0
SUM = SUM + LA[K]
SUM = 0 + LA[1]
SUM = 0 + 70
SUM = 70
SUM = SUM + LA[K]
SUM = 70 + LA[2]
SUM = 70 + 50
SUM = 120 + LA[3]
SUM = 120 + 80
SUM = 200 + LA[4]
SUM = 200 + 10
SUM = 210

Inserting a new element into Array

INSERT (LA, N, ITEM, K)

Let LA is a linear array of size N, this algorithm will insert ITEM at the Kth position of LA, Where K <= N + 1

Steps : 1) Start

2) Intialize Counter
   Set J = N

3) Repeat 4th & 5th While (J >= K)

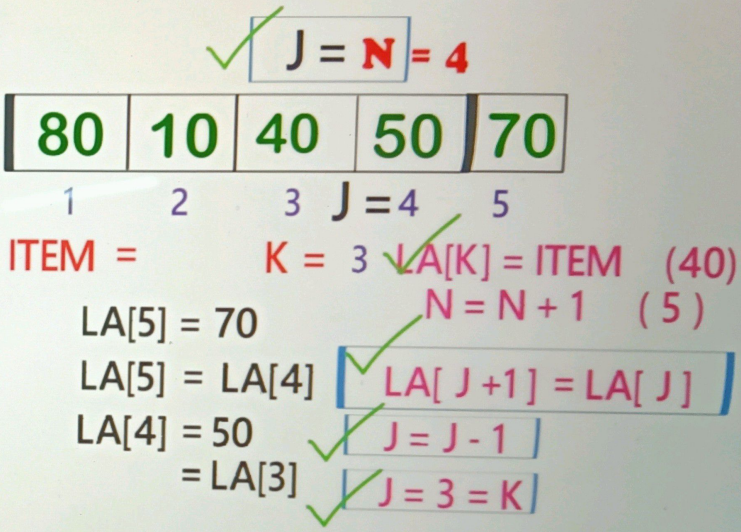
4) Move Jth element downword
   LA[J+1] = LA[J]

5) Decrement Counter
   J = J - 1

6) Insert ITEM
   LA[K] = ITEM

7) Reset N = N + 1

8) Stop



MacBook Air

DELETE(LA, N, ITEM, K)

Let LA is a linear array of size N, this algorithm will delete element ITEM from Kth position.
where K is any positive integer such that K <= N

Steps : 1) Start

2) set ITEM = LA[K]

3) Initialize counter
set J = K

4) Repeat 5th & 6th step while J <= N - 1

5) Move J+1 th element upword
LA[J] = LA[J+1]

6) Increment Counter
J = J + 1

7) Reset N = N - 1

8) Stop

N = 3

LA

80	50	70
1	2	3

K = 2 ITEM = 10

J = K

J = 2 J <= N - 1 2 <= 3 YES

LA[J] = LA[J+1]

LA[2] = LA[2+1]

LA[2] = LA[3]

LA[2] = 50

J = 3 J <= N - 1 3 <= 3 YES

LA[J] = LA[J+1]

LA[3] = LA[4]

LA[3] = 70

J = 4 J <= N - 1 4 <= 3 NO

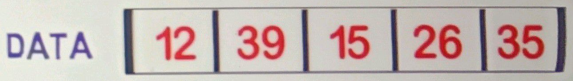
LINEAR(DATA, N, ITEM, LOC)

Let DATA is a linear array of size N.
this algorithm will find the LOC of ITEM in DATA and
if search is unsuccessfull it will set LOC = NULL

Steps : 1) Start

2) Initialize LOC
   Set LOC = 1
3) Repeat steps 4th & 5th while LOC <= N
4) Compare ITEM
   IF(ITEM == DATA [LOC]), then
      i) Display ITEM is found at LOC
      ii) go to step 7th
5) Increment LOC
   LOC = LOC + 1
6) IF (LOC > N), then
   Display ITEM is not found
7) Stop

N = 5



ITEM = 85 LOC = ?

LOC = 1 85 == 12 ? NO
INCREMENT LOC = LOC + 1
LOC = 2 85 == 39 NO
LOC = 3 85 == 15 NO
LOC = 4 85 == 26 NO
LOC = 5 85 == 35 NO
LOC = 6
6 <= 5 ? NO

BINARY(DATA, N, ITEM, LOC)

Let DATA is sorted linear array with size N, with lower bound LB and upper bound UB.
BEG denotes begining and END denotes ending and MID denotes middle segment of the array.
This algorithm will find LOC of ITEM in DATA and if search is unsuccesfull then it will set LOC = NULL

Steps : 1) start

2) Initialize BEG = LB and END = UB

3) Repeat step 4th & 5th while BEG <= END

4) Calculate MID = (INT) (BEG + END) / 2

5) Compare ITEM

IF(ITEM == DATA [MID])

i) Display ITEM is found at LOC = MID

ii) go to step 7th

ELSE IF(ITEM < DATA[MID])

Set END = MID - 1

ELSE Set BEG = MID + 1

6) IF (BEG > END)

i) Display ITEM is NOT found, Set LOC = NULL

7) stop

N = 3

DATA	10	20	30
	1	2	3

ITEM = 30 LOC = ?

BEG = 1 END = 3

is(BEG <= END) ?

(1 <= 3) ? YES

MID = (1 + 3) / 2 = 4/ 2 = 2

IF(ITEM == DATA[2])

IF(30 == 20) NO

ELSE IF(30 < 20) NO

ELSE Set BEG = 2 + 1 = 3

BEG = END = MID = 3

IF(30 == DATA[3])

Difference between Linear search and Binary search

Linear search	Binary search
1) In linear search array (data) may or may not be sorted. i.e. it is not neccessary that data should be sorted.	1) In binary search array (data) should be sorted.
2) Item to be search is compared with every element of array starting from first location.	2) Item to be search is compared with middle value of array.
3) Its less efficient than binary search	3) Its more efficient than linear search
4) Time complexity of linear search is O(n)	4) Time complexity of binary search is $O(\log_2 n)$
5) Example	5) Example

BUBBLE (DATA, N)

Let DATA is a linear array of size N and this algorithm will sort DATA in accending order.

Steps : 1) Start

2) Repeat steps 3rd and 4th for K = 1 to N - 1 by 1

3) Set PTR = 1

4) Repeat steps 5th and 6th while PTR <= N - k

5) Compare elements

IF(DATA[PTR] > DATA[PTR + 1]) then,

Interchange their values

i) Set TEMP = DATA[PTR]

ii) Set DATA[PTR] = DATA[PTR+1]

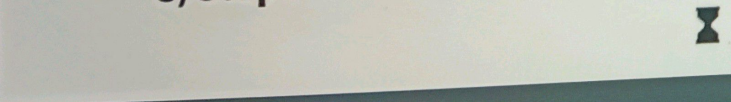
iii) Set DATA[PTR+1] = TEMP

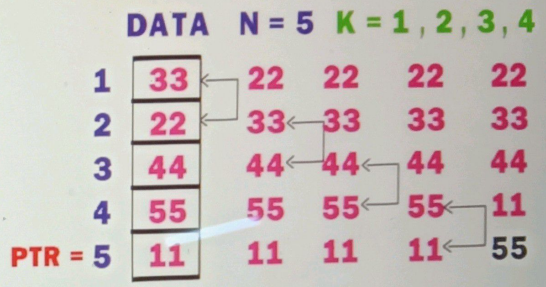
6) Increment counter

PTR = PTR + 1

7) Display Sorted DATA

8) Stop





is (PTR <= N - K) ?

TEMP = 33

DATA[1] = 22

DATA[2] = 33

is (5 <= 5 - 1) ?

is (5 <= 4) ? NO

IF(DATA[PTR] > DATA[PTR + 1])

IF(DATA[4] > DATA[5])

IF(55 > 11) YES

Pointer Array :

In array if each elements are pointer then its called as pointer array.

Pointer is a variable which contain the address of the elements.

Pointer points to an element in list.

Let we have four groups.
Each group consist of list of members.

Membership list is to be stored in memory.

The most efficient method is to form two arrays. One is members consisting of all members one after the other.

and another pointer array group contaning the starting locatons of different groups.

Observe that the elements of pointer array are starting addresses of each group

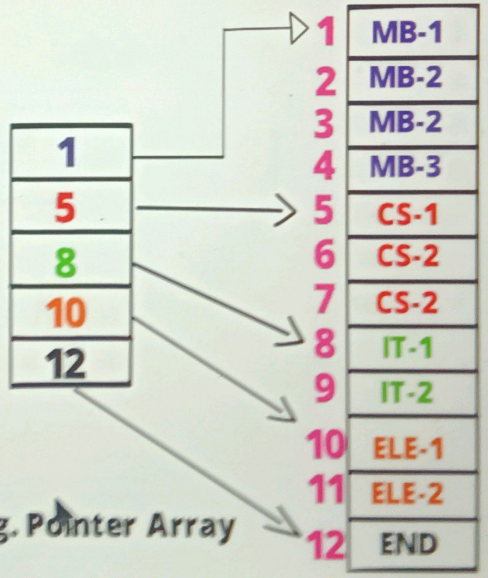


Fig. Pointer Array

XII Class : Students (Members)

MB = 4
CS = 3
IT = 2
ELE = 2

Linked list : Its a linear collection of data elements, called nodes.
linear order is given by means of pointer.
each node is divided into two parts, first part contains the information part of the elements and second part contains the address of the next node in the list

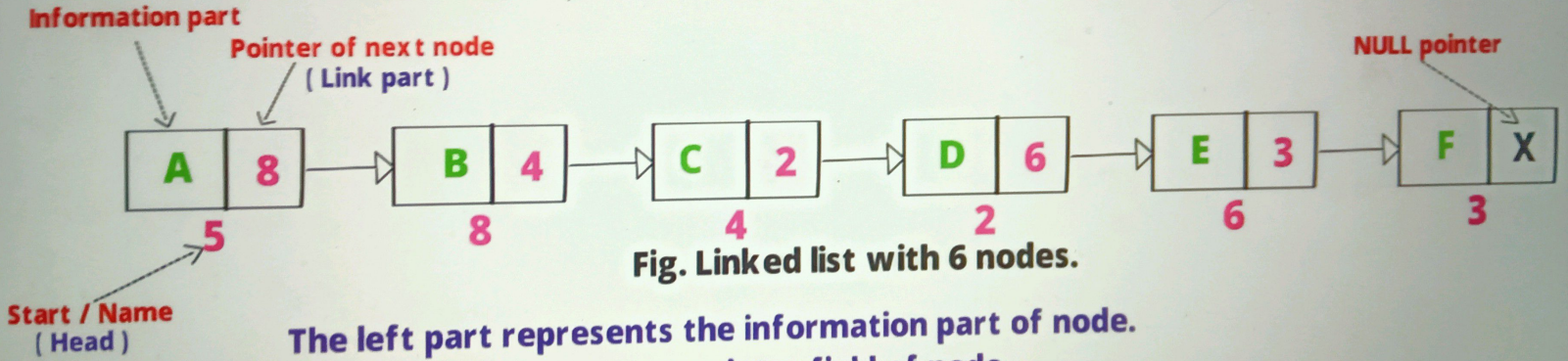


Fig. Linked list with 6 nodes.

The left part represents the information part of node.
The right part represent pointer field of node.
The arrow is drawn from this field to next node.
pointer field of last node conatin null pointer.
The address of first node in list called as start or name.
We need only this address to trace the linked list.

Linked list : Its a linear collection of data elements, called nodes.
linear order is given by means of pointer.
each node is divided into two parts, first part contains the information part of the elements and second part contains the address of the next node in the list

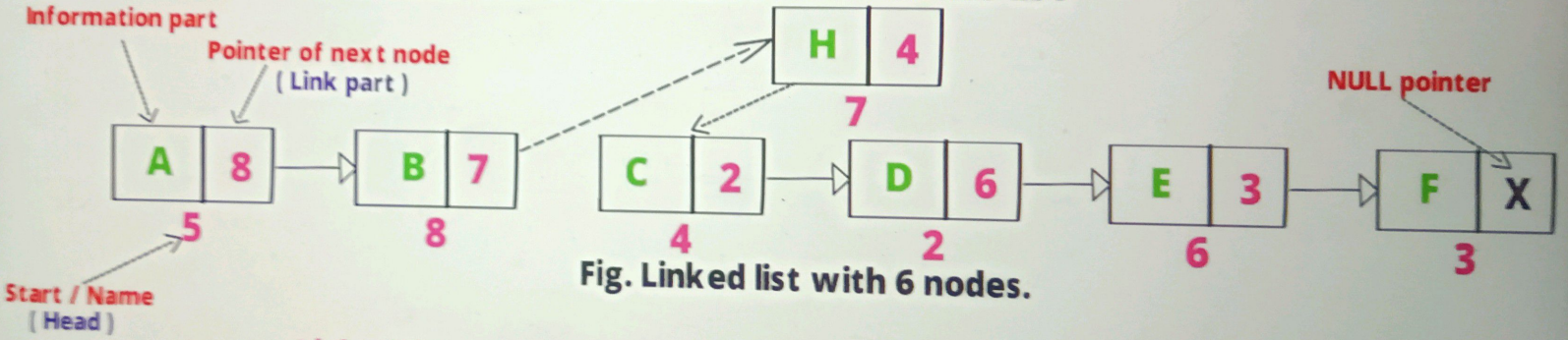


Fig. Linked list with 6 nodes.

Linked list

definition

linear order is given by means of pointer.

Inseration and deletion is easy

not required consecutive memory locations

Array

definition

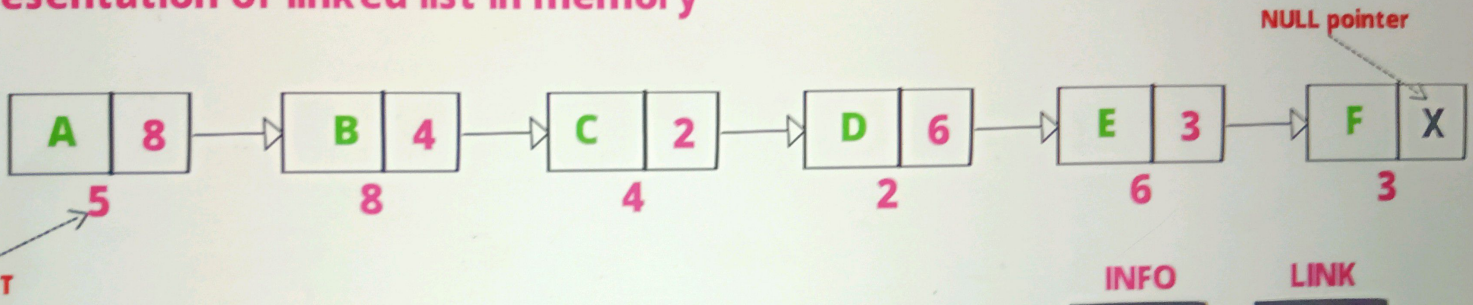
linear order is given by index number.

Inseration and deletion is expensive

required consecutive memory locations

Representation of linked list in memory

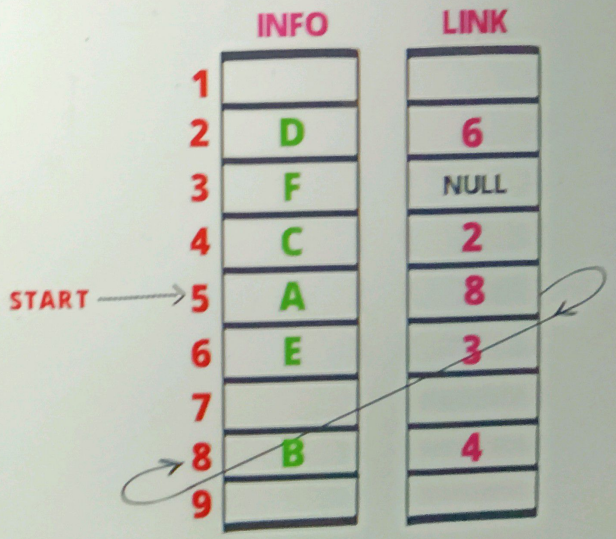
Eg.



Linked list can be represented by two parallel linear arrays.
One is INFO containing information part
and other is LINK containing next pointer field.
The name of linked list, start, contains beginnings of the list.

START points to 5th element

INFO[5] = A and LINK[5] = 8 so next node address is 8
INFO[8] = B and LINK[8] = 4 so next node address is 4
INFO[4] = C and LINK[4] = 2 so next node address is 2
INFO[2] = D and LINK[2] = 6 so next node address is 6
INFO[6] = E and LINK[6] = 3 so next node address is 3
INFO[3] = F and LINK[3] = NULL The list is ended here.



Stack :

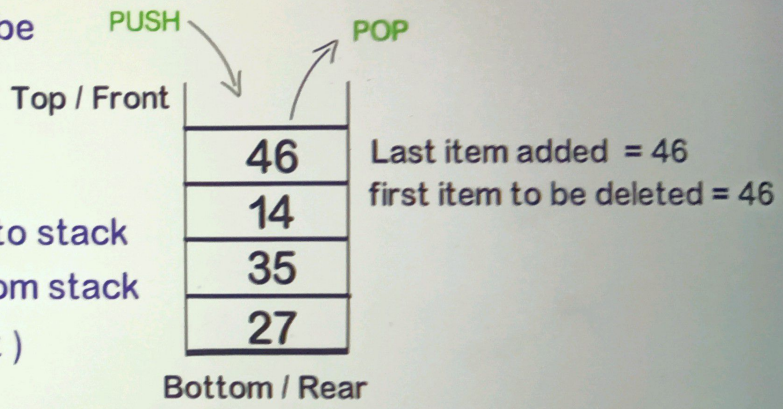
A stack is a data structure in which items may be added or removed only at one end.

the item can removed or added only from the top of the stack.

"PUSH" is the term used to insert an element into stack

"POP" is the term used to delete an element from stack

A stack is also called as LIFO. (Last In First Out)



Queue :

A queue is a linear list in which items may be added only at one end and items may be removed only at other end.

The queue is also called as FIFO. (First In First Out)