Lecture -2
Data structures

Array

Array

- Data structures are classified as either linear or nonlinear.
- A data structure is said to be linear if its elements form a sequence or a linear list.
- There are two basic ways of representing such linear structures in memory.
- One way is to have the linear relationship between the elements represented by means of sequential memory locations. These linear structures are called arrays.
- The other way is to have the linear relationship between the elements represented by means of pointers or links. These linear structures are called linked lists.
- Nonlinear structures are trees and graphs.

Linear Arrays

A linear array is a list of finite number n of homogeneous data elements such that :

a) The elements of the array are referenced respectively by an index set consisting of n consecutive numbers.

b) The elements of the array are stored respectively in successive memory locations.

The number n of elements is called the length or size of the array.

Three numbers define an array : lower bound, upper bound, size.

a. The lower bound is the smallest subscript you can use in the array (usually 0)

b. The upper bound is the largest subscript you can use in the array

c. The size / length of the array refers to the number of elements in the array , It can be computed as upper bound - lower bound + 1

Let, Array name is A then the elements of A is, by the bracket notation A[1], A[2], A[3],……., A[n]

The number k in A[k] is called a subscript and A[k] is called a subscripted variable.

Linear Arrays

Example :

A linear array DATA consisting of the name of six elements

DATA

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

DATA[1] = 247

DATA[2] = 56

DATA[3] = 429

DATA[4] = 135

DATA[5] = 87

DATA[6] = 156

Linear Arrays

Example :

An automobile company uses an array AUTO to record the number of 'automobile' sold each year from 1932 through 1984.

AUTO[k] = Number of auto mobiles sold in the year *K*

LB = 1932

UB = 1984

Length = UB – LB+1 = 1984 – 1932+1 =53

Representation of linear array in memory

Let *LA* be a linear array in the memory of the computer. The memory of the computer is a sequence of addressed locations.

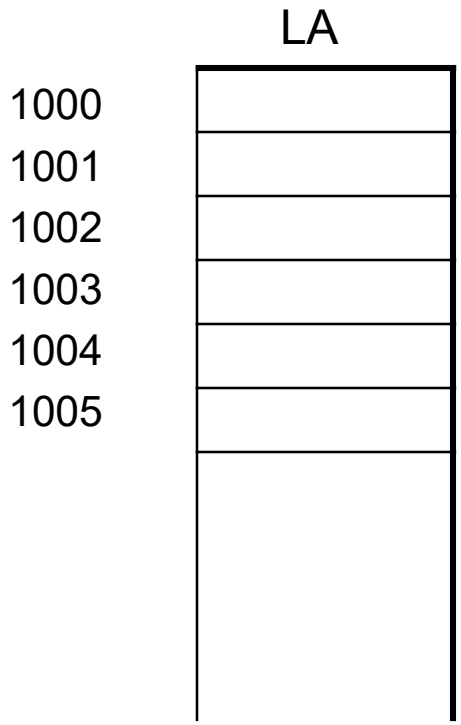


Fig : Computer memory

The computer does not need to keep track of the address of every element of LA, but needs to keep track only of the first element of LA, denoted by

Base(LA)

Called the base address of LA. Using this address Base(LA), the computer calculates the address of any element of LA by the following formula :

LOC(LA[k]) = Base(LA) + w(K – lower bound)

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

Representation of linear array in memory

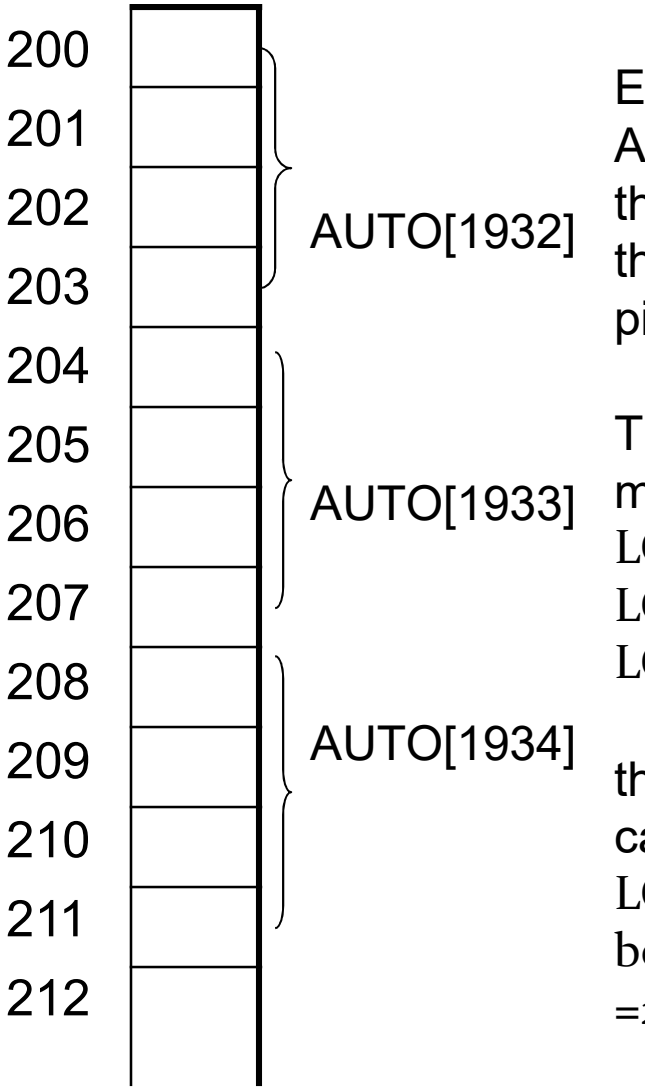


Fig : A

Example :
An automobile company uses an array AUTO to record the number of auto mobile sold each year from 1932 through 1984. Suppose AUTO appears in memory as pictured in fig A .

That is Base(AUTO) = 200, and w = 4 words per memory cell for AUTO. Then,
LOC(AUTO[1932]) = 200,
LOC(AUTO[1933]) =204
LOC(AUTO[1934]) = 208

the address of the array element for the year K = 1965 can be obtained by using :
LOC(AUTO[1965]) = Base(AUTO) + w(1965 – lower bound)
=200+4(1965-1932)=332

Inserting and Deleting

Inserting refers to the operation of adding another element to the Array .

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

Inserting an element somewhere in the middle of the array require that each subsequent element be moved downward to new locations to accommodate the new element and keep the order of the other elements.

Deleting an element somewhere in the middle of the array require that each subsequent element be moved one location upward in order to “fill up” the array. Fig shows Milon Inserted, Sumona deleted.

STUDENT

1	Dalia Rahaman
2	Sumona
3	Mubtasim Fuad
4	Anamul Haque
5	
6	

STUDENT

1	Dalia Rahaman
2	Sumona
3	Milon
4	Mubtasim Fuad
5	Anamul Haque
6	

STUDENT

1	Dalia Rahaman
2	Milon
3	Mubtasim Fuad
4	Anamul Haque
5	
6	

Insertion

INSERTING AN ELEMENT INTO AN ARRAY:
Insert (LA, N, K, ITEM)
Here LA is linear array with N elements and K is a positive integer such that K<=N. This algorithm inserts an element ITEM into the K^{th} position in LA.

ALGORITHM :

Step 1. [Initialize counter] Set J:=N

Step 2. Repeat Steps 3 and 4] while J >= K

Step 3. [Move J^{th} element downward] Set LA[J+1] := LA[J]

Step 4. [Decrease counter] Set J := J-1

[End of step 2 loop]

Step 5 [Insert element] Set LA [K] := ITEM

Step 6. [Reset N] Set N := N+1

Step 7. Exit

Deletion

DELETING AN ELEMENT FROM A LINEAR ARRAY
Delete (LA, N, K, ITEM)

ALGORITHM

Step 1. Set ITEM: = LA [K]

Step 2. Repeat for J=K to N-1

[Move J+1st element upward] Set LA [J]: =LA [J+1]

[End of loop]

Step 3 [Reset the number N of elements in LA] Set N:=N-1

Step 4. Exit.

Searching : Linear search

Searching refers to the operation of finding the location LOC of ITEM in DATA, or printing some message that ITEM does not appear there.

DATA is a linear array with n elements. The most intuitive way to search for a given ITEM in DATA is to compare ITEM with each element of DATA one by one. That is first we test whether DATA[1]=ITEM, and then we test whether DATA[2]=ITEM, and so on. This method, which traverses DATA sequentially to locate ITEM, is called linear search or sequential search.

Algorithm 4.5 : A linear array DATA with N elements and a specific ITEM of information are given. This algorithm finds the location LOC of ITEM in the array DATA or sets LOC = 0.

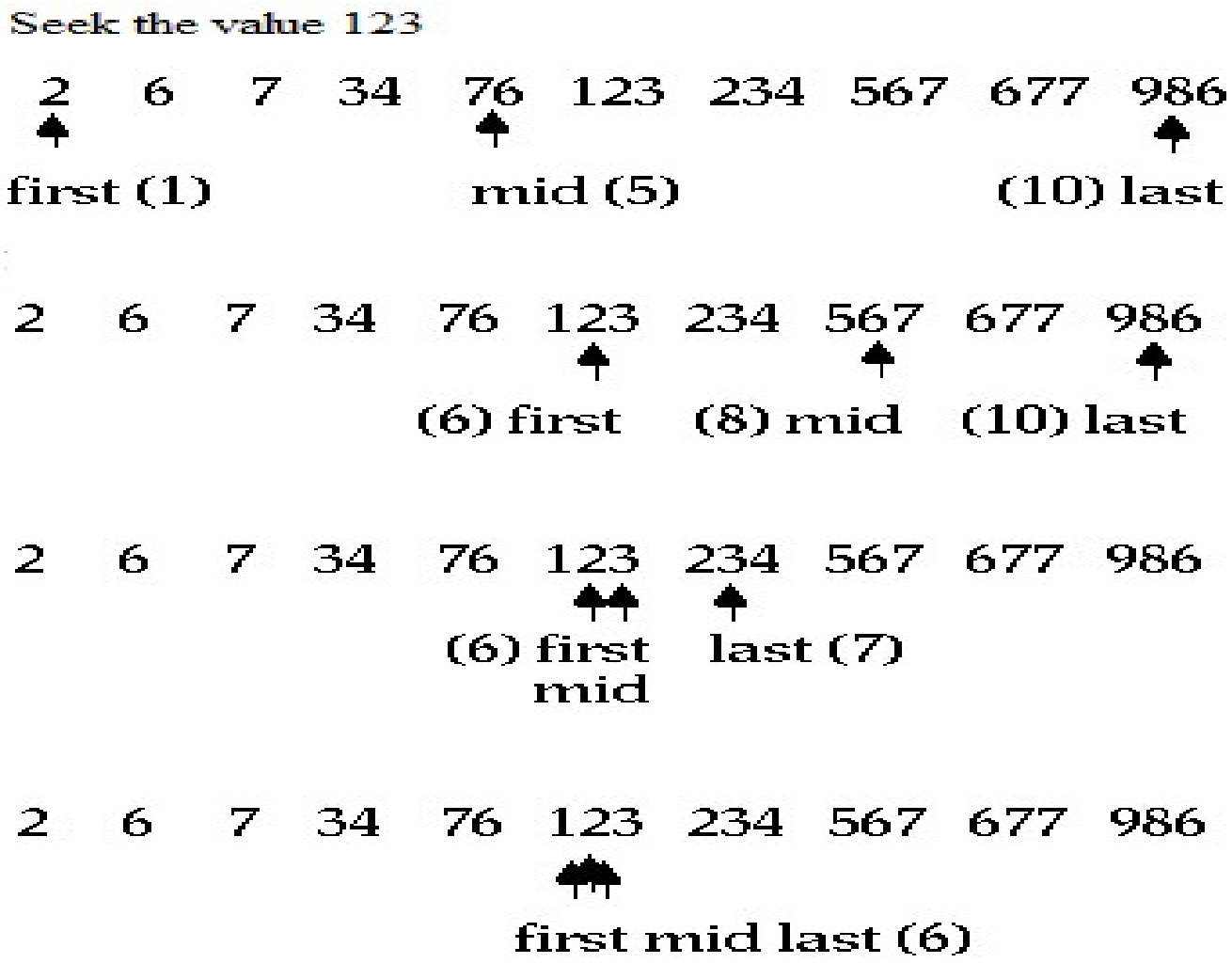
1. Read: ITEM.
2. Set LOC:=1.
3. Repeat while DATA[LOC]!= ITEM and LOC <=N:
   Set LOC := LOC +1.
   [End of loop]
4. If LOC = N+1, then :
   Write : ITEM is not in the array DATA.
   Else :
   Write : LOC is the location of ITEM.
5. Exit.

Binary Search Algorithm

BINARY(DATA, LB, UB, ITEM, LOC)

1. **Set BEG=LB; END=UB; and MID=INT((BEG+END)/2).**
2. **Repeat step 3 and 4 while BEG ≤ END and DATA[MID] ≠ ITEM**
3. **If ITEM < DATA[MID] then**
   Set END= MID - 1
   Else:
   Set BEG= MID+1
   [end of if structure]
4. **Set MID= INT((BEG+END)/2)**
   [End of step 2 loop]
5. **If ITEM = DATA[MID] then**
   Set LOC= MID
   Else:
   Set LOC= NULL
   [end of if structure]
6. **Exit.**

Binary Search example (Seek for 123)



Multidimensional arrays

Two dimensional, three dimensional arrays and ….. Where elements are referenced respectively by two, three and ……subscripts.

Two – dimensional Arrays

A Two – dimensional Arrays m x n array A is a collection of m . n data elements such that each element is specified by a pair of integers (such as J, K), called subscripts.

The element of A with first subscript J and second subscript K will be denoted by A[J, K]

Columns

Rows	1	2	3
1	A[1, 1]	A[1, 2]	A[1, 3]
2	A[2, 1]	A[2, 2]	A[2, 3]
3	A[3, 1]	A[3, 2]	A[3, 3]

Fig: Two dimensional 3 x 3 array A

Matrix Multiplication

Algorithm 4.7: MATMUL(A, B, C, M, P, N)

Let A be an MXP matrix array, and let B be a PXN matrix array. This algorithm stores the product of A and B in an MXN matrix array.

1. **Repeat steps 2 to 4 for I =1 to M:**
2. **Repeat steps 3 to 4 for J =1 to N:**
3. **Set C[I, J]:=0**
4. **Repeat for K =1 to P:**
   C[I, J]:= C[I, J]+A[I, K]*B[K, J]
1. **Exit**