Chapter 1 :- Concepts Of Data Structures

TYPES OF DATA STRUCTURES

There are various types of data structures which are classified accordingly to several ways, such as:

1. Linear data structure :

In this type of data structures data is stored sequentially using memory locations. Data are arranged in linear fashion. The example of Linear data structure are, Arrays, stacks, queues and linked list.

2. Non- Linear data structure :

In this type of data structures data is not stored sequentially, because memory is not allocated sequentially. Data are not arrange in order. The example of Non-Linear data structure are, Tree, Graph, Table and Sets.

3. Dynamic data structure :

The dynamic data structure are created using dynamic memory allocation, we use a pointer variable and its size is allocated during the execution (at run time) of the program. In dynamic memory allocation, we can increase or decrease the size of a data structure easily so it make flexible. Linked List is a example of dynamic data structure.

4. Static data structure :

In static data structure, the size of data structure is fixed and can not changed during execution of program. Using Array we can create static data structure.

5. Homogeneous data structure :

It is same as array, which stores all data of same data type.

6. Non- Homogeneous data structure :

It is same as structures and class, which contain different data types stored in a group.

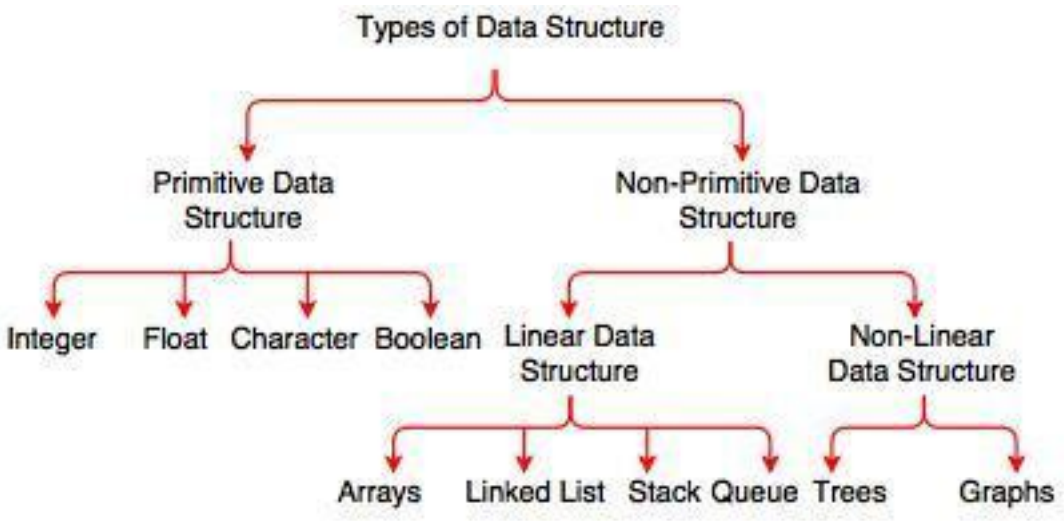


Fig. Types of Data Structure

ABOUT THE ALGORITHM

Introduction to Algorithm:

The word algorithm comes from the name of a Persian author, Abu Jafar Mohammed ibn Musa al Khowarizmi, who wrote a textbook on mathematics. This word has taken on a Significance in computer science, where "algorithm" has come to refer to a method that can be used by a computer for the solution of a problem. This is what makes algorithm different from words such as process, technique or method.

Definition: [Algorithm]: An algorithm is a finite set of instructions that, if followedaccomplishes a particular task. In addition, all algorithms must satisfy the following criteria :

1. **Input:** Zero or more quantities are externally supplied.
2. **Output:** At least one quantity is produced.
3. **Definiteness:** Each instruction is clear and unambiguous.
4. **Finiteness:** Each instruction is clear and unambiguous.
5. **Effectiveness:** Every instruction must be very basic so that it can be carried out, in principle, by a person using only pencil and paper. It is not enough that each operation be definite as in criterion 3; it also must be feasible.

An algorithm is composed of a finite set of steps, each of which may require one or more operations. The possibility of a computer carrying out these operations necessitates that certain constraints be placed on the type of operations an algorithm can include.

Applications:

Many algorithms are useful in a broad spectrum of computer applications, they include algorithm for sorting, searching, text processing. solving graph problem, geometric problem and perform common mathematical calculation.

KEY FEATURES OF AN ALGORITHM

(1) Name of Algorithm: Every algorithm is given an identifying name, written incapital letters.

(2) Steps: The actual algorithm is made up of a sequence of number of steps, eachbeginning with a phrase enclosed in a square brackets, which gives description of that steps.

(3) Assignment statements: The assignment statement is indicated by a placingan arrow( ) between the right hand side of the statement and the variable receiving the value.

Example: X 10
K 0

(4) If Statement: It has two form

(a) If (condition)
Then ---------

(b) If (condition)
Then ---------
Else ----------

(5) Case statement : The case statement is used for multiple choice type solution,the General case statement are,

Select case
case value 1:
case value 2:
:
case value N:
default:

(6) Repeat statement: While Loop, For looping.

(1) Repeat while (condition)

It is used to repeat a step until a given logical condition is false.

(2) Repeat thru step No for Variable name 1, 2,... N. (For Loop)

(7) Goto statement: The goto statement causes unconditional transfer of controlto the step referenced

Thus, "Go to step N" will cause transfer of control to step N.

(8) Exit statement: The Exit statement is used to terminate analgorithm. It is usually last step of algorithm.

Example:

Algorithm to find Big and Small number from N Numbers.

Algorithm: BIG AND SMALL NUMBER

Step 1: [Initialization]

Temp-0

Step 2:[Read the size of List]

Read N

Step 3: [Consider the first element of the list as Big and Small]

Big Small List [0]

Step 4:[Check whether the size of list is 1]

If N=1

then Big- Small- List [0]

Step 5: [Step for find Big and Small no]

Repeat thru step 6 while (Temp s N)

If List (Temp)> Big

then

Big List [Temp]

Else If (List (Temp] < Small)

Then

Small List [Temp

Step 6: Temp Temp+ 1

Step 7: Output Small and Big

Step 8: [Finished]

Exit.

Difference Between List And Array

CHARACTERISTICS

LIST	ARRAY
(1)List is an ordered set of element.	(1) Array is an unordered set of element.
(2) It consist of variable number of element.	(2) It consist of fixed no. of elements.
(3) Insertion and deletion can be performed on list.	(3) Insertion and deletion can not be performed on array.
(4) There are two types of list 1. Linear List 2. Non-Linear List	(4) There are two types of array 1. One dimensional Array 2. Multi-dimensional Array
(5) Location cannot be easily accessible.	(5) Location are easily accessible through subscripts.
(6) Address calculated is complicated.	(5) Address calculation is very easy because memory allocated sequentially.

ARRAY

1. Array have number of memory cells which is called elements.
2. Every element has assigned a unique number which is called address of particular element.
3. Element number is assign by name of array and index number. Always array start with 0 index number and end with one less than size of array.
4. All the elements of array share the common name, and they are distinguished from one another with the help of the element number.
5. The amount of memory occupied by the array is depending on the data type and number of elements of array.
6. The array elements are always stored sequentially or in linear fashion.
7. In array, lower bound cannot be changed but the upper bound may be expanded or changed.
8. In array, Insertion and deletion operation is slower and time consuming but searching and sorting are faster.

9. When an array is declared and not initialized, it contains garbage values. If array is declared as static, all elements are initialized to zero.

SEQUENTIAL / LINEAR SEARCHING :

- This is the simplest technique to find out searching an element value in an unsorted table or list.
- In this technique of searching, search the value of particular element in a table or list in sequential manner until the desired record is found.

Algorithm:

Let us consider, unsorted list L which contain N elements, we have to find the value x in the list L

L ⟶ represents the list of elements

N ⟶ number of elements in a list

X ⟶ value to be searched in the list

Step 1: [Initialize]

K 0 Flag 1

Step 2: Repeat step 3 for k=0,1,2,...,N-1

Step 3: if(L[k]=X)

Then (1) flag 0

(2) write("Search is successful")

(3) write("Value found at location : k+1)

Step 4: if(flag=1)

Then

Write("Search is successful ")

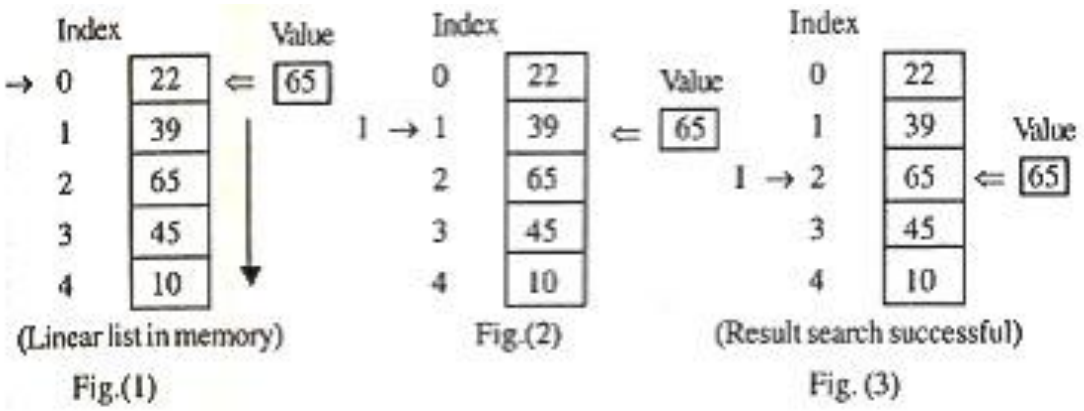
Step 5: [Finished]

Exit

EXAMPLE :

The above algorithm work on base of searching sequential in a list which is shown in below figure.

SEARCH VALUE = 65



PROGRAM

```
#include <stdio.h>
#include<conio.h>
void main()
{
    void linear_search(int[],int,int);
int a[20],n,x,i;
clrscr();
printf("\n How many num ber enter in list");
scanf("%d",&n);
printf("\n enter the numb er in list:");
```

```
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
         }
    printf("Enter the number to search in list");
    scanf("%d",&x);
    linear_search(a,n,x); //call function linear_search()
    getch();
}
void linear_search(int b[],intm,int y)
{
int flag=0;
    for(i=0;i<m;i++)
    {
          if(b[i]==y)
          {
          flag=1;
          break;
          }
    }
 if(flag=0)
        printf("search is unsuccessful");
else
        printf("Search is successful at location:%d",i+1);
getch();   }
```

OUTPUT

How many numbers enter in list:04

Enter the number in list: 69

55

45

22

Enter the number to search in the list:55

Search successful

Search value:55 at location:2

BINARY SEARCH

- Binary search techniq ue used for sorted list only and it is very efficient searching technique is used to fi nd location of a given element or record in a list.
- To find the location of a file in the computer directory are one of the use of binary searching technique.
- If we are on internet , Information are very wide. so a linear searchi ng is not possible to find out information about some record or file , it will take more time because searching sequentially one by one. For this purpose , a binary search technique is used more efficient.
- Binary search techniq ues works in following ways,

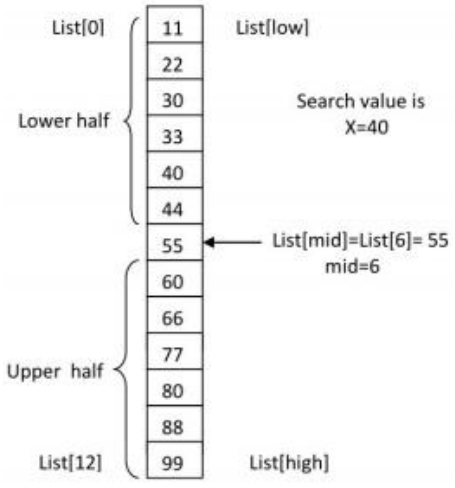


Fig. 1

- Let, low represent lower limit of list and high represent upper. mid is average of low and high.
- Mid = (low + high)/2
- we compare mid element with key to be searched , if the value of mid element is greater than search key , then key will exist in lower half of list , so take high=mid-1 and find new value of mid , i.e. mid=(low + high)/2.
- Otherwise key will exist in upper half of list , so take low=mid+1 and find new value of mid , i.e. mid=(low + high)/2 , this process is continues until the entire list is exhausted or key is searched.
- If the list is unsorted , so that list must be sorted first before starting the binary searching technique.

BINARY SEARCH EXAMPLE :-

Search Key = 40.

1. Initially from fig.1

   i. List[low] = List[0] = 11

   ii. List[high] = List[12] = 99

   Where low=0 and high=12

   iii. mid=(low + high)/2 = (0+12)/2 = 6
   Lower half list = 11,12,30,33,40,44
   Upper half list = 60,77,80,88,99

   iv. List[mid] = List[6] = 55
   since 40<55 means

   search key < List[mid] , so it will exist in lower half of the list. [
   Between 11,12,30,33,40,44].

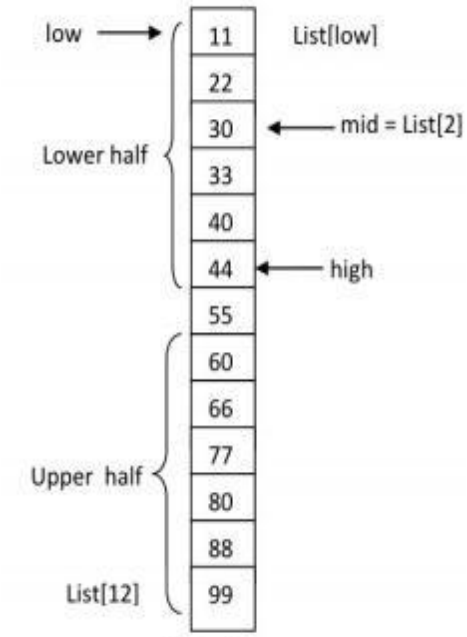


Fig. 2

2. 40<55

So , search key < List[mid]

i. high = mid-1

= 6 - 1

= 5

ii. mid = (low + high)/2

=(0+5)/2

= 2

iii. so , List[mid] = List(2) = 30

now search k ey 40 > 30 which is List[mid]

so , it will exist between List[mid] and List[high]

, means List[2] to list[5]

so , low =mid + 1.

3. sine search ke y > List[mid]

40>30

i. Low = mid+ 1 = 2+1 = 3

ii. New value of mid is,

mid=(low + high)/2 = (3+5)/2 = 4

iii. So now compare list[mid] And Search key,

List[mid]=Sea rch key

List[4]=Search key[40]

- 40=40 , it will match.

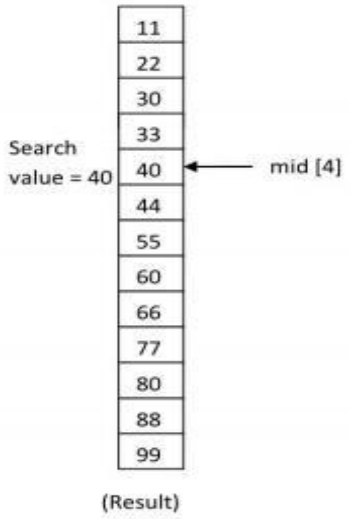


Fig. 3

So , search is successf ul.

Binary Search Algorith m :-

Algorithm :- Binary Search(List,n,X)

Step 1 : [initialization]

low 0

high n-1

flag 1

Step 2 : While (low < = high) repeat step 3 to 4.

Step 3 : mid = (low + high)/2.

Step 4 : if (X<List[mid])

Then

high = mid-1

else if (X > List[mid])

Then

low = mid+1

else if(X=List[mid])

Write("Search is successful at Location: mid+1").

Flag 0

Exit

Step 5: if (flag=1)

Then

Write("Search is Successful").

Step 6 : Exit