Batch Name : OC07 Crash Course Batch
Module Name : Data Structures

DS DAY-01:

- data structure is a programming concept

Q. What is a data structure?
It is a way to store data elements into the memory (i.e. into the main memory) in an oragnized manner, so that operations like addition, deletion, searching, sorting, traversal etc.... can be performed on it efficiently.

We want to store marks of 100 students:
int m1, m2, m3, m4, m5, ..., m100;//400 bytes – sizeof(int): 4 bytes – 32 bit compiler

int marks[100];//400 bytes

+ array: it is it is basic/linar data structure, which a collection/list of logically related similar type of data elements gets stored into the memory at contiguos locations.

- traversal on array => to visit each array element sequentially from first element max till last element.

```
for( index = 0 ; index < SIZE ; index++ ){
        printf("%4d", arr[ index ] );
}
```

- we want to store info about an employee:
empid : int
emp_name : char [ ]
salary : float

+ structure: it is basic/linear data structure, which is a collection/list of logically related similar and disimmilar type of data elements gets stored into the memory collectively as a single record/entity.

- there are 2 types of data structures:
1. linear/basic data structures
2. non-linear/advanced data structures

- to learn data structure is not to learn any programming language, it is nothing but to learn an algorithms in it: operations that can be performed on data elements:

What is an algorithm? Algorithms => Human Beings/End User
- An algorithm is a set of finite no. of instructions written in human understandable language like english, if followed, acomplishesh given task.

Algorithm to do sum of array elements:
step-1: intially take sum as 0
step-2: start traversal of an array and keep adding each array element into the sum sequentially from first element max till last element.
step-3: return final sum

What is a Pseudocode? => special form of an algorithm=> Programmer User
- An algorithm is a set of finite no. of instructions written in human understandable language like english **with some programming constraints**, if followed, acomplishesh given task, this form of an algorithm is referred as a pseudocode.

```
Algorithm ArraySum( A, n )//whereas A is an array of size n
{
        sum = 0;
        for( index = 1 ; index <= n ; index++ ){
                sum += A[ index ];
        }

        return sum;
}
```

Program is an implementation of an algorithm
OR
An algorithm is like a blue print of a program

What is a Program? Program => Machine
- Program is a set of finite no. of instructions written in any programming language given to the machine to do specific task.

```
#include<stdio.h>

int array_sum( int arr[ ], int n ){
        int sum = 0;
        int index;

        for( index = 0 ; index < n ; index++ ){
                sum += arr[ index ];
        }
        return sum;
}
```

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

	printf("sum = %d\n", array_sum(arr, 5);

	return 0;
}
```

- an algorithm is a solution of a given problem
- algorithm = solution
- one problem may has many solutions/algorithms, when one problem has many solutions one need to select an efficient solution/algo.
- to decide efficiency of an algorithm there is a need to do their analysis
- analysis of an algorithm is a work of calculating how much time i.e. computer time and space i.e. computer memory it needs to run to completion.
- there are two measures of analysis of an algorithm:

1. time complexity of an algorithm is the amount of time i.e. computer time it needs to run to completion.

2. space complexity of an algorithm is the amount of space i.e. computer memory it needs to run to completion.

Pune & Mumbai
if multiple paths exists between Pune & Mumbai then one need to select an optimized path
factors to decide optimized paths:
distance in km
time required to cover distance
mode of travelling
traffic conditions
etc....

Client => Manager (Not a Programmer)
Manager => Project Manager(Technical) => Developer(Programmer) => Program => Machine

searching: to search given key element in a collection/list of data elements.
- there are two searching algorithms:
1. linear search
2. binary search

1. Linear Search/Sequential Search:
step-1: accept key from the user which is to be search
step-2: start traversal of an array and compare value of key element with each array element sequentially from first element till either match is not found or max till the last element, if key is matches with any array element then return true otherwise return false.

```
Algorithm LinearSearch(A, n, key){
        for( index = 1; index <= n ; index++ ){
                if( key == A[ index ] )//compare value of key with each array ele
                        return true;//key is found
        }

        return false;
}
```

best case: if key is found in an array at very first position: O(1)
if size of an array = 10, no. of comparisons = 1
if size of an array = 20, no. of comparisons = 1
if size of an array = 50, no. of comparisons = 1
.
.
if size of an array = n, no. of comparisons = 1

worst case: if either key is found at last pos or key is not found in an array: O(n).
if size of an array = 10, no. of comparisons = 10
if size of an array = 20, no. of comparisons = 20
if size of an array = 50, no. of comparisons = 50
.
.
if size of an array = n, no. of comparisons = n

best case time complexity = if an algo takes min amount of time to run to completion. ==> lower limit

worst case time complexity = if an algo takes max amount of time to run to completion. ==> upper limit

average case time complexity = if an algo takes neither min nor max amount of time to run to completion.

There are 3 notations used to represent time complexities:
1. Big Omega (Ω) - this notation is used to denote best case time complexity asymptotic lower bound.

2. Big Oh (O) - this notation is used to denote worst case time complexity asymptotic upper bound.

3. Big Theta (θ)- this notation is used to denote an average case time complexity
asymptotic tight bound.

Asymptotic analysis: it is a mathematical way to calculate time complexity of an algorithm without implementing it in any programming language.

Descrete Maths:
Rule: if running time of an algo is having any additive /substractive /divisive / multiplicative contant then it can be neglected.
e.g.
O(n + 3) => O(n)
O(n – 5) => O(n)
O(n / 2) => O(n)
O(n * 3) => O(n)

- if any algorithm follows divide-and-conquer approach we get time complexity of that algorithm in terms of log.

2. Binary Search:

after caculating mid pos big size array divided logically into 2 subarrays => left subarray & right subarray
for left subarray, value of left remains as it is, and right = mid-1
for right subarray, value of right remains as it is, and left = mid+1

- binary search algo is also called as logarithmic search/half-interval search

first node/element => root node => root position
node which is not having further child/s => leaf node => leaf position
node which is having further child/s => non-leaf node => non-leaf position

sorting: to arrange data elements in a collection/list of elements either in an ascending order or in a desceding order.

- sort collection/list => bydefault in an ascending order

sorting algorithms:
1. selection sort:

total no. of comparisons = (n-1)+(n-2)+(n-3)+.....
total no. of comparisons = n(n – 1) / 2
=> n(n – 1) / 2

=> O((n^2 – n)) / 2) ... by using rule: if an algo is having divisive constant it can be neglected

=> O(n^2 – n)
=> O(n^2 – n) => <u>O(n^2)</u>

rule: if running time of any algo is having a polynomial then in ite time complexity we need to consider only leading term.
e.g.
O(n^3 + n + 2) => O(n^3)
O(n^2 + 5) => O(n^2)