Batch Name : PreCAT OM19
Subject : Data Structures

DS DAY-01:

- to store marks of 100 students
int m1, m2, m3, m4, ..., m100;//400 bytes

i want to sort marks in a descending

int marks[100];//400 bytes
operations can be performed on data ele's efficiently.

Array: it is a collection/list of logically related similar type of elements in which data ele's gets stored into the memory at contiguos location.

We want to store rollno, name & marks of student/record

Structure: it is a collection/list of logically related similar and disimmilar type of data elements in which data ele's gets stored into the memory collectively as a sinlge entity(record).

Class: it is a basic/linear data structure which is a collection of logically related similar and disimmiar type of data elements as well as functions.

typedef struct employee emp_t;

struct empolyee e1;//abstraction – abtract data type
emp_t e2;

```
class student
{
	//data  members
	private:
		int rollno;
		String name;
		float marks;
	public:
	//member functions
		student();
		~student();
		//getter functions
		//setter functions
		//facilitators
};
```

student s1;//abstract data type - abstraction

student s2;- reusability

struct employee e1;

to scan an array/traversal on an array: to visit each array element sequentially from first element till max last element.

- to learn data structure is not to learn any programming language, it is nothing but to learn an algorithms.

Q. What is a Program?
Program is a finite set of instructions written in any programming language (like C, C++, Java, Python, Assembly etc....), given to the machine to do specific task.

Program --> Machine

Q. What is an algorithm?
An Algorithm is a finite set of instructions written in human understandable language like english, if followed, acomplishesh given task.

Algorithm --> Programmer User

- A Program is an implementation of an algorithm.
- An algorithm is a like blueprint of a program.

Q. What is a pseudocode?
An Algorithm is a finite set of instructions written in human understandable language like english with some programming constraints, if followed, acomplishesh given task, such algorithm is referred as a pseudocode.

Algorithm to do sum of array elements: --> End User/Programmer User
Step-1: initially take value of sum variable as 0.
Step-2: start traversal of an array and keep adding each element into the sum variable one by one.
Step-3: return the final value of sum.

Pseudocode: its a special form of an algorithm --> Programmer User

```
Algorithm ArraySum( A, size ){
        sum = 0;
        for( index = 1; index <= size ; index++ ){
                sum += A[ index ];
        }

        return sum;
}
```

Program: --> Machine

```
int array_sum( int arr[], int size){
        int sum = 0;
        int index = 0;
        for( index = 0 ; index < size ; index++ ){
                sum += arr[ index ];
        }
        return sum;
}
```

Code <==> Program
Source Code – program written in any programming language.

- An algorithm is a solution of a given problem.
- algorithm = solution
- we can have many solutions for a the same problem, in this case one need to select an efficient solution/algo.

e.g.
searching: to search a given key element in a collection/list of data elements.
1. linear search
2. binary search

sorting: to arrange data elements in a collection/list of data elements either in an ascending order or in a descending order.
1. selection sort
2. bubble sort
3. insertion sort
4. quick sort
5. merge sort

Pune --> Mumbai
- multiple paths/routes may exists between two cities
- when we know multiple paths between 2 cities --> an optimized/efficient one
parameters/measures: distance, cost, status, traffic situation, time .....

searching:
+ Analysis of an algorithm:
- to decide efficiency of an algo's, we need to do their analsis.
- analysis of an algo, is nothing but to calculate how much time i.e. computer time and space i.e. memory it needs to run to completion.
- there are 2 measures of analysis of algo:
1. time complexity of an algo is the amount of time i.e. computer time it needs to run to completion.

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

- space ==> memory required to store variables, constants & instructions in a program/algo.

- Linear search:
step-1: accept value of key element (which is to search) from user
step-2: start traversal an array from first element and compare value of key with each array ele sequentially till match is found or max till the last element. If match found the n return true, otherwise return false.

Best case: If key is found at first position --> O(1) – constant time complexity
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 position or key is not exists --> 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

- We need to follow certains rules and we have to use some notations:

Asymptotic Notations:
1. Big Omega (Ω) - this notation is used
2. Big Oh (O)
3. Big Theta (θ)

- descrete mathemetics
Rule:
1. if running time of an algo is having any additive/substractive/multiplicative/ divisive constant, then it can be neglected.
e.g.
O(n+3) => O(n)
O(n-4) => O(n)
O(n/3) => O(n)
O(n/2) => O(n)
O(n*2) => O(n)

- if it is not mentioned specifically then (bydefault) we have to consider an average case time complexity for all algorithms.
- usually magnitudes of time complexities of an algo's are same in average case and worst case.

Q. What is the time complexity of a linear search?
Average Case = θ(n)

Q. What is the best case time complexity of a linear search?
Best Case = Ω(1)

Q. What is the average case time complexity of a linear search?
Average Case = θ(n)

space =

Prerequisite C Programming Language Topics: do revise following topics
1. storage classes
2. pointers
3. functions
4. structure

DS DAY-02:
- introduction to DS:
Why there is a need of data structure?
What is data structure? Types of data structure
- introduction to an algorithm, analysis of an algo:
- linear search:
best case : Ω(1)
worst case : O(n)
average case : θ(n)

binary search:
- by menas of calculating mid pos, big size array has been divided into two subarrays – left subarray & right subarray.

For left subarray value of left remains as it is, right = mid-1
For right subarray value of right remains as it is, left = mid+1

subarray is valid till (left <= right)
subarray becomes invalid as soon as left > right

if size of an array 1000
[0.....1000] ==> [0...499] 500 [501 .... 1000]

LSA => [0 ... 499] ==> [0.....249] 250 [251 ..... 500]
LSA => [0... 249] ==> [0...124] 125 [126.....249]

after iteration-1: n/2
after iteration-2: n/4
after iteration-3: n/8

after every iteration search space is getting reduced by half

for size of an array = n
for ietration search space = n
after iteration-1: $n/2 + 1 \Rightarrow n/2^1 + 1$
after ietartion-2: $n/4 + 2 \Rightarrow n/2^2 + 2$
after ietartion-3: $n/8 + 3 \Rightarrow n/2^3 + 3$
.
.
after k iterations: $n/2^k + k$

lets assume, $n = 2^k$
$\Rightarrow \log n = \log 2^k$ (by taking log on both sides)
$\Rightarrow \log n = k \log 2$
$\Rightarrow \log n = k$ (log 2 = 1)
$\Rightarrow k = \log n$

$T(n) = n/2^k + k$
put $n = 2^k$ in above equation, $k = \log n$
$T(n) = 2^k/2^k + \log n$
$T(n) = 1 + \log n$
$T(n) = O(1 + \log n)$
$T(n) = O(\log n + 1) \Rightarrow O(\log n)$

- algorithm which follows divide-and-conquer approach, we get time complexity in terms of log

- when we compare two algo's for deciding efficiency we need to consider an average case time complexities.

Rule:
if running time of an algo is having a polynomial, then only leading term gets considered in its time complexity.
e.g.
$O(n^3 + n^2 + 4) \Rightarrow O(n^3)$
$O(n^4 + n^2 + 2) \Rightarrow O(n^4)$

1. selction sort:

Total no. of comparisons = $n(n-1)/2 \Rightarrow (n^2 - n)/2$

$T(n) \Rightarrow O(n^2 - n/2) \Rightarrow O(n^2 - n) \Rightarrow \mathbf{O(n^2)}$

- C programming language: procedure oriented programming language
procedure/function: in C we need to divide logic of a program into functions

- C++ programming language: object oriented programming language

a = 20
b = 10

temp=10

temp = a;
a = b;
b = temp;

- bubble sort is also called as sinking sort

home work: implement bubble sort

for it = 0 => pos = 0,1,2,3,4
for it = 1 => pos = 0,1,2,3
for it = 2 => pos = 0,1,2
for it = 3 => pos = 0,1

for(pos = 0 ; pos < SIZE-1-it ; pos++)

if array is already sorted:

flag = 0

10 20 30 40 50 60 -> all pairs are in order -> array ele's are already sorted

best case: if array ele's are already sorted
total no. of comparisons = n-1
T(n) = O(n-1) => **O(n)**

 # DS DAY-03:
- binary search: algorithm, analysis & implementation
- sorting algo's:
selection sort:
bubble sort

3. Insertion Sort:

```
for( i = 1 ; i < SIZE ; i++ ){
        key = arr[ i ];
        j = i-1;
        while( j >= 0 && key < arr[ j ] ){
                arr[ j+1 ] = arr[ j ];//shift ele towards its right
                j--;
        }
        arr[ j+1 ] = key;
}
```

best case in insertion sort:
10 20 30 40 50 60 70

- in every iteration only 1 comparison takes place, and in insertion sort max n-1 no. of iterations are there
total no. of comparisons = n-1
T(n) = O(n-1) => O(n) => Ω(n).

- in any sorting algo, if relative order of two ele's having same key value remains same even after sorting then such sorting algo is referred as stable.

10 10' 20 30 40

Key = 10'

arr[1] = 10
arr[4] = 10

- to convert program into a menu driven program

10 20 30 40 50 60 70 80
[LP] 10 [20 30 40 50 60 70 80]
[LP] 20 [30 40 50 60 70 80]
[LP] 30 [40 50 60 70 80]

as in above case in every pass array is not getting divided equally into partitions and hence time complexity in this case do not get in terms of log n
worst case – $O(n^2)$.