PAGE No.
DATE / /

Chp-1 :- INTRODUCTION.

- Algorithm :- Algorithm is a set of well-defined rules to find the solution of a problem in limited no. of steps. It is a procedure to solve problem and evolved with time closely linked with programming.

- Data Structure :- Data structures are objects generate to store data and algorithms are set of instruction to perform task using data structure.

- Data type :- Data type is most basic classification of data through which the compiler gets to know the type of information that will be used through-out the code. Data type is transmission of information through which the programmer informs the compiler about what type of data is to be stored along which know much size of memory will be require to store it.

Q. Difference between Data type and data structure.

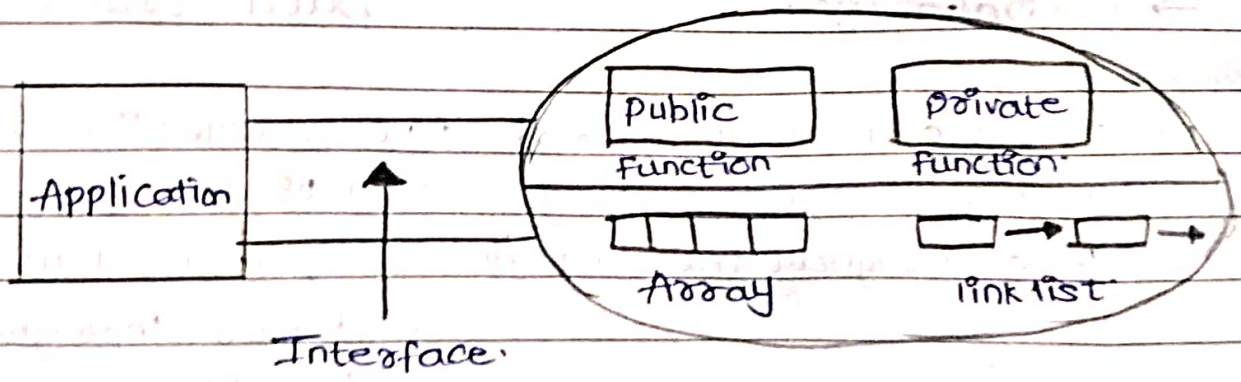
→

	Data type	Data structure.
1-	It tells up about the kind of variable which is being used throughout the program.	It is collection of different kind of data. The entire data is represented using object and used throughout the program.

2.	It can hold values but not data also it is data less	2- It can hold different kind of data with in one single object.
3.	No problem of time complexity.	3- Working with data structure introduces the time complexity.
4.	Values are directly assigned to the data type variable.	4- The data is assigned to the data structure object with some set of algorithm.
5	eg:- int, Char, float, etc.	5-eg :- Stack, Queue and tree

* ADT (Abstract Data types) :-

- It is type for object whose behaviour is defined by set of value and set of operation. It tell us about only what operations. It tell us about only what operations are to be performed but not how these operations will be implemented. It provides the information which is only essential and hides the other details. This process is known as Abstraction.



PAGE No.
DATE / /

* Algorithm analysis :-
The algorithm analysis focus on CPU usage, memory disk and network usage. It is mostly concerned about the CPU Time.

* Algorithm analysis is a part of computational complexicity theorey which provides ~~theriti~~ theoritical estimation for the required resources of an algorithm to solve a specific computational problem.

* CPU time have mainly two times/types

(i) Performances :-
How much time memory disk is used to run a particular program.

(ii) Complexity :-
It deals with what happening if the size of the program solved increases.

(iii) Time complexity :-
~~X~~ It deals with what happening if the size of the program solved increases.
It deals with qualification of amount of time taken by set of code to run as a function of amount of input.

(iv) Space complexity :-
It is total amount of memory space required by a program for its execution.

PAGE No.
DATE / /

* Frequency count :-

The frequency count method counts number of times one abstraction / instruction is executing. To determine step count / Frequency count we determine the number of steps per execution s/e. For each statement and total number of times each statement is executed.

eg:-

```
Algo sum (A,n)                    Frequency count.
{
  S=0                                   1
For (i=0; i<n; i++);                  n+1
{
  S=S+A[i];                             n
}
  return 0;                             1
}
```

∴ Frequency count of Algo sum (A,n)

$= 1 + (n+1) + n + 1$

$= 2n + 3$

n = Time complexity.

eg① add (int a [ ][ ], b[ ][ ] Frequency count

```
{
int c[i][j]=0;                    1
For (i=0; i<n; i++)               n+1
{
 For (j=0; j<n; j++)              n=(n+1)=n²+n
{
c[i][j]=a[i][j]+b[i][j];          n*n=2n²
}
}
}
return 0;
```

Frequency count of add (int a [ ][ ], b[ ][ ]) is =
$2n^2+2n+3$

* Asymptotic Notation :-

It refers to defining the mathematical boundry of its run time performance.

Using asymtotic analysis we can conclude best case, average case, and worst case scenario of an algorithm.

The time required by an algorithm.

Falls under three types :-

1) Best case :-

Minimum time required by program for execution.

eg② add (int a [] [], b[] [] Frequency count

```
{                                          1
int c [i] [j] = 0 ;                        n+1
For (i=0 ; i<n ; i++)                      n = (n+1) = n^2 + n
{                                          n * n = 2n^2
 For (j=0; j<n ; j++)
{
c[i] [j] = a[i] [j] + b[i] [j] ;
}
}
}
return 0;
```

Frequency count of add (int a [] [], b[] []) is = $2n^2 + 2n + 3$

* Asymptotic Notation :-

It refers to defining the mathematical boundry of its run time performance.

Using asymtotic analysis we can conclude best case, average case, and worst case scenario of an algorithm.

The time required by an algorithm.

Falls under three types :-

1) Best case :-

Minimum time required by program for execution.

2) Average case :-
Average ~~case~~ time required by program for execution

3) Worst case :-
Maximum time required by program for execution.

* Big oh (O) Notation :-

The notation is used to denote upper bound that is worst case behaviour.

Defination :- The function $f(n) = O(g(n))$ if and only if their exists positive constant (c) and no such that $f(n) <= (* g(n)$. For all n such that $n \geq n_0$.

* Omega (Ω) Notation :-

This is used to denote lower bound that is best case behaviour.

Defination :- The function $f(n) = \Omega(g(n))$ if and only if their exist positive constant c and no such that $f(n) >= c * g(n)$ where $n \leq n_0$.

* Theta (θ) Notation :-

This notation denotes both upper and lower bound of f(n).

PAGE No.
DATE / /

Defination:- The function $f(n) = \theta(g(n))$, if and only if their exists positive constant c_1 and c_2 and n_0 such that $c_1 * g(n) \leq f(n) \leq c_2 * g(n) \leq f(n)$ for all $n \geq n_0$.

* Graphs of Notations.

(i)

(ii)

(iii)

PAGE No.
DATE / /

* Searching Algorithm :-
The process of locating / searching a specific element from a set of elements is known as searching.
There are two methods of searching.
i) Linear search
ii) Binary search.

* Linear search :-

Linear search also known as sequential search is a simplest method of searching.

eg :- Series :- 15, 5, 20, 35, 2, 42, 67, 19.

element to be found :- 2.

→

0	1	2	3	4	5	6	7.
15	5	20	35	2	42	67	19.

Let the value to be searched :- $x = 2$.

1)

0	1	2	3	4	5	6	7.
15	5	20	35	2	42	67	19.

↓

$x \neq 15$

2)

0	1	2	3	4	5	6	7
15	5	20	35	2	42	67	19

$x \neq 5$.

3)

0	1	2	3	4	5	6	7
15	5	20	35	2	42	67	19

$x \neq 20$.

4)

0	1	2	3	4	5	6	7
15	5	20	35	2	42	67	19

$x \neq 35$.

5)

0	1	2	3	4	5	6	7
15	5	20	35	2	42	67	19

$x = 2$.

∴ The given element is found at 4th position.

* Binary Search :-

eg :- Series :- 1 4 7 9 16 56 70.
element to found :- 7.

(i)

0	1	2	3	4	5	6
1	4	7	9	16	56	70.
↓			↑			↓
low			mid			High.

mid = low + high / 2 = 3

mid > / < / = K.

9 > 7.

New high = mid - 1

= 3 - 1

= 2.

(ii)

0	1	2	3	4	5	6.
1	4	7	9	16	56	70.
↓	↓	↓				
Low	mid	High.				

mid = 0 + 2 / 2 = 1

= 4 < 7.

New low = mid + 1

= 1 + 1

= 2.

(iii)

0	1	2	3	4	5	6.
1	4	7	9	16	56	70.
		↙ ↓ ↘				
		low	high			
		mid.				

PAGE No.
DATE / /

mid = l + h/2
= 2 + 2/2
= 2.

mid > K = K
mid = 7 = key.

∴ The element 7 is found at position 2.

Eg (2) Series :- 2 5 8 12 16 23 38 56 72 91
element to be found = 16

(i)

0	1	2	3	4	5	6	7	8	9.
2	5	8	12	16	23	38	56	72	91

mid = low + high / 2
= 0 + 9 / 2.
= 4.5
= 4.

(ii) element to be found = 23.

mid.

(i)

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	23	28	56	72	91

↓ mid ↓ Low ↓ High

mid = low + high / 2
= 5 + 9 / 2
= 7.

(ii)

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	23	8	56	72	91
					↓		↓		↓
					low		med		High.

mid > 23 mid = 5 - 1 = 6.

New high = mid - 1
= 7 - 1
= 6.

(iii)

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	23	38	56	72	91
					↙	↓	↓		
					med	low	High.		

mid = low + high / 2
= 5 + 6/2
mid = 5
mid = 23 = key.

∴ The element 23 is found at position 5.

* Sorting techniques :=
A sorting algorithm is used to rearrange a given array according to a comparison operator (</>) the elements.

There are various types of sorting algorithm such as:-

1- Bubble sort.
2- Bucket sort.
3- Counting sort.
4- Heap sort.
5. Insertion Sort
6- Merge Sort.
7. Quick Sort.
8. Selection Sort
9. Radex Sort.
10. Shell Sort.

1- Bubble Sort :-

It is the simplest sort method which perform sorting by repeatedly moving the larget element to the highest index of the array.
It comprases of compairing each element to its adjest element and replace them accordingly.

eg:- Series :- 64, 34, 25, 12, 22, 11, 90.

pass 1 :- 64, 34, 25, 12, 22, 11, 90
34, 64, 25, 12, 22, 11, 90
34, 25, 64, 12, 22, 11, 90
34, 25, 12, 64, 22, 11, 90
34, 25, 12, 22, 64, 11, 90
34, 25, 12, 22, 11, 64, 90
34, 25, 12, 22, 11, 64, 90

PAGE No.
DATE / /

pass 2 :- 34, 25, 12, 22, 11, 64, 90
25, 34, 12, 22, 11, 64, 90
25, 12, 34, 22, 11, 64, 90
25, 12, 22, 34, 11, 64, 90
25, 12, 22, 11, 34, 64, 90
25, 12, 22, 11, 34, 64, 90
25, 12, 22, 11, 34, 64, 90.

pass 3 :- 25, 12, 22, 11, 34, 64, 90.
12, 25, 22, 11, 34, 64, 90
12, 22, 25, 11, 34, 64, 90
12, 22, 11, 25, 34, 64, 90
12, 22, 11, 25, 34, 64, 90
12, 22, 11, 25, 34, 64, 90
12, 22, 11, 25, 34, 64, 90.

pass 4 :- 12, 22, 11, 25, 34, 64, 90.
12, 22, 11, 25, 34, 64, 90
12, 11, 22, 25, 34, 64, 90
12, 11, 22, 25, 34, 64, 90
12, 11, 22, 25, 34, 64, 90.
12, 11, 22, 25, 34, 64, 90
12, 11, 22, 25, 34, 64, 90.

pass 5 :- 12, 11, 22, 25, 34, 64, 90
11, 12, 22, 25, 34, 64, 90
11, 12, 22, 25, 34, 64, 90
11, 12, 22, 25, 34, 64, 90
11, 12, 22, 25, 34, 64, 90
11, 12, 22, 25, 34, 64, 90.

PAGE No.
DATE / /

Pass 6 :- 11 , 12 , 22 , 25 , 34 ; 64 , 90

11 , 12 , 22 , 25 , 34 , 64 , 90

11 , 12 , 22 , 25 , 34 , 64 , 90

11 , 12 , 22 , 25 , 34 , 64 , 90

11 , 12 , 22 , 25 , 34 , 64 , 90

11 , 12 , 22 , 25 , 34 , 64 , 90

11 , 12 , 22 , 25 , 34 , 64 , 90.

Eg 2 :- Series :- 5 , 4 , 10 , 2 , 30 , 45 , 34 , 14 , 18 , 9 .

pass 1 :- 5 , 4 , 10 , 2 , 30 , 45 , 34 , 14 , 18 , 9

4 , 5 , 10 , 2 , 30 , 45 , 34 , 14 , 18 , 9

4 , 5 , 10 , 2 , 30 , 45 , 34 , 14 , 18 , 9

4 , 5 , 2 , 10 , 30 , 45 , 34 , 14 , 18 , 9

4 , 5 , 2 , 10 , 30 , 45 , 34 , 14 , 18 , 9

4 , 5 , 2 , 10 , 30 , 45 , 34 , 14 , 18 , 9

4 , 5 , 2 , 10 , 30 , 34 , 45 , 14 , 18 , 9

4 , 5 , 2 , 10 , 30 , 34 , 14 , 45 , 18 , 9

4 , 5 , 2 , 10 , 30 , 34 , 14 , 18 , 45 , 9

4 , 5 , 2 , 10 , 30 , 34 , 14 , 18 , 9 , 45

pass 2 :- 4 , 5 , 2 , 10 , 30 , 34 , 14 , 18 , 9 , 45

4 , 5 , 2 , 10 , 30 , 34 , 14 , 18 , 9 , 45

4 , 2 , 5 , 10 , 30 , 34 , 14 , 18 , 9 , 45

4 , 2 , 5 , 10 , 30 , 34 , 14 , 18 , 9 , 45

4 , 2 , 5 , 10 , 30 , 34 , 14 , 18 , 9 , 45

4 , 2 , 5 , 10 , 30 , 34 , 14 , 18 , 9 , 45

4 , 2 , 5 , 10 , 30 , 14 , 34 , 18 , 9 , 45

4, 2, 5, 10, 30, 14, 18, 35, 9, 45
4, 2, 5, 10, 30, 14, 18, 9, 35, 45
4, 2, 5, 10, 30, 14, 18, 9, 35, 45

pass 3:- 4, 2, 5, 10, 30, 14, 18, 9, 35, 45
2, 4, 5, 10, 30, 14, 18, 9, 35, 45
2, 4, 5, 10, 30, 14, 18, 9, 35, 45
2, 4, 5, 10, 30, 14, 18, 9, 35, 45
2, 4, 5, 10, 30, 14, 18, 9, 35, 45
2, 4, 5, 10, 14, 30, 18, 9, 35, 45
2, 4, 5, 10, 14, 18, 30, 9, 35, 45
2, 4, 5, 10, 14, 18, 9, 30, 35, 45
2, 4, 5, 10, 14, 18, 9, 30, 35, 45
2, 4, 5, 10, 14, 18, 9, 30, 35, 45

pass 4:- 2, 4, 5, 10, 14, 18, 9, 30, 35, 45
2, 4, 5, 10, 14, 18, 9, 30, 35, 45
2, 4, 5, 10, 14, 18, 9, 30, 35, 45
2, 4, 5, 10, 14, 18, 9, 30, 35, 45
2, 4, 5, 10, 14, 18, 9, 30, 35, 45
2, 4, 5, 10, 14, 18, 9, 30, 35, 45
2, 4, 5, 10, 14, 9, 18, 30, 35, 45
2, 4, 5, 10, 14, 9, 18, 30, 35, 45
2, 4, 5, 10, 14, 9, 18, 30, 35, 45
2, 4, 5, 10, 14, 9, 18, 30, 35, 45

pass 5:- 2, 4, 5, 10, 14, 9, 18, 30, 35, 45
2, 4, 5, 10, 14, 9, 18, 30, 35, 45
2, 4, 5, 10, 14, 9, 18, 30, 35, 45

2, 4, 5, 10, 14, 9, 18, 30, 35, 45
2, 4, 5, 10, 14, 9, 18, 30, 35, 45.
2, 4, 5, 10, 9, 14, 18, 30, 35, 45
2, 4, 5, 10, 9, 14, 18, 30, 35, 45
2, 4, 5, 10, 9, 14, 18, 30, 35, 45
2, 4, 5, 10, 9, 14, 18, 30, 35, 45
2, 4, 5, 10, 9, 14, 18, 30, 35, 45
2, 4, 5, 10, 9, 14, 18, 30, 35, 45

pass 6 :- 2, 4, 5, 10, 9, 14, 18, 30, 35, 45
2, 4, 5, 10, 9, 14, 18, 30, 35, 45
2, 4, 5, 10, 9, 14, 18, 30, 35, 45
2, 4, 5, 10, 9, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45

pass 7 :- 2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45
2, 4, 5, 9, 10, 14, 18, 30, 35, 45

PAGE No.
DATE / /

* Insertion sort :-
It is an sorting algorithm in which it inserts each elements of an array to its proper place.
It is very simple sort method which is used to arrange the deck of cards while playing bridge.

eg :- 1 5 4 10 1 6 4.

In this given array we consider 0th index element that is 5 to be sorted and rest all element to be unsorted.
In this algorithm a element in unsorted part will be compared with all elements from sorted part and will be directly place at proper position.

eg:-1 5 , | 4 , 10 , 1 , 6 , 4 .
4 , 5 , | 10 , 1 , 6 , 4 .
4 5 10 | 1 6 4
1 4 5 10 | 6 4
1 4 5 6 10 | 4
1 4 5 5 6 10

eg:2 12 | 11 13 5 6 .
11 12 | 13 5 6 .
11 12 13 | 5 6
5 11 12 13 | 6
5 6 11 12 13

eg:- 3.

4	3	2	10	12	1	5	6
3	4	2	10	12	1	5	6
2	3	4	10	12	1	5	6
2	3	4	10	12	1	5	6
2	3	4	10	12	1	5	6
1	2	3	4	10	12	5	6
1	2	3	4	5	10	12	6
1	2	3	4	5	6	10	12

- Comparison between Insertion and bubble sort.

	Bubble sort	Insertion.
1)	A simple sorting algorithm which compares adjacent pairs and swap them if they are in wrong order.	1- A simple sorting algorithm that builts the final sorted list by transfering one element at a time.
2)	Checks the neighbouring elements and swap them accordingly.	2) Transferse the element at a time to partially sorted array.
3)	More number of swaps.	3) Less number of swaps.
4-	Slow as compared to insertion sort.	4) Twice faster as compared to bubble sort
5)	It is simple	5) Complex than bubble sort.

PAGE No.
DATE 1 1

eg:- 3.

4	3	2	10	12	1	5	6
3	4	2	10	12	1	5	6
2	3	4	10	12	1	5	6
2	3	4	10	12	1	5	6
2	3	4	10	12	1	5	6
1	2	3	4	10	12	5	6
1	2	3	4	5	10	12	6
1	2	3	4	5	6	10	12

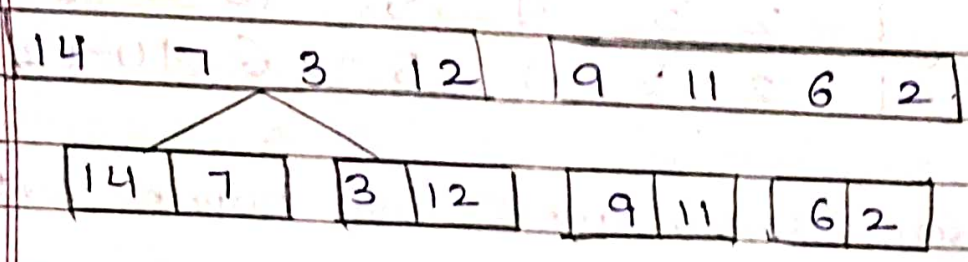
- Comparison between Insertion and bubble sort.

	Bubble sort	Insertion.
1)	A simple sorting algorithm which compares adjacent pairs and swap them if they are in wrong order.	1- A simple sorting algorithm that builts the final sorted list by transfering one element at a time.
2)	Checks the neighbouring elements and swap them accordingly.	2) Transferse the element, at a time to partially sorted array.
3)	More number of swaps.	3) Less number of swaps.
4-	Slow as compared to insertion sort.	4) Twice faster as compared to bubble sort.
5)	It is simple	5) Complex than bubble sort.

PAGE No.
DATE / /

* Time complexity:-
The best case -time complexity of bubble sort and insertion sort is $O(n)$ and worst case time complexity of both algorithm is both $O(n^2)$.

- Merge Sort → Recursive → DAC.



14 7 3 12 9 11 6 2

7 14 3 12 9 11 6 2.

3 7 12 14 2 6 9 11

2 3 6 7 9 11 12 14.

Merge sort:-
Merge sort is one of the most efficients sorting algorithm. It works on the principle DAC (Divide and conquer) It divides a input array into two parts ~~of the~~ calls it self for two parts and merge the two sorted parts. Merge sort repeatedly breaks down a list into several sub lists until each sub list consists of a single element and merging those sublists in a manner that results into sorted list.

PAGE No.
DATE / /

(2) 35 50 15 25 80 20 90 45.

35 50 15 25 80 20 90 45

35 50 15 25 80 20 90 45

35 50 15 25 20 80 45 90.

15 25 35 50 20 45 80 90.

15 20 25 35 45 50 80 90.

- ~~Data~~ Quick sort :-