Data Structures And Algorithms

Syllabus

Page:
Date: / /

1. Introduction to DSA
2. Analysis of Algorithm.
3. Searching & Sorting Algorithms.
4. Arrays.
5. Linked Lists
6. Stacks and Queues.
7. Trees
8. Graphs
9. Divide & Conquer Techniques.
10. Greedy Techniques.
11. Dynamic Programming
12. BackTracking & Branch & Bound
13. Hashing.

* Data Structure → storing, organising, retrieving
So that it can be used efficiently.

Variable :- It is a container which can hold/store a particular type of a value. Used for storing the values.

```
int x;          // integer datatype and integer type variable.
x = 25;
float y;
y = 5.5;
```

Datatype :- Type of data which we want to store in the memory.

```
int x;          // integer type value is stored.
```

Data Structure

① Linear DS
Elements are accessed in seq. order but it is not compulsory to store all elements sequentially.
Ex:- Linear Lists, Stacks, Queues.

② Non Linear DS.
Elements of this type are stored/accessed in a Non-Linear order
Ex:- Trees & Graphs.

Algorithm → An Algorithm is a step by step unambiguous instructions to Solve a given problem.

Page:
Date: / /

~~finite steps + finite Time~~
All steps are compulsory

Example:-

```
ATN(){
① Take 2 Nos. (a,b)
② Add a & b and store result in c
③ Print c
}
```

~~finite steps~~
~~finite Time~~
~~step is compulsory~~
sequence matter
Ambiguous

- ~~Algorithm always halts (terminate)~~ and say yes or no.
- ~~Algo. provides mapping to input to~~ output.

Properties of Algorithms:-

① Each step must be a basic operation like we can't write directly in algo to sort the given nos.

② It should terminate in finite number of time.

③ It should produce atleast 1 output.

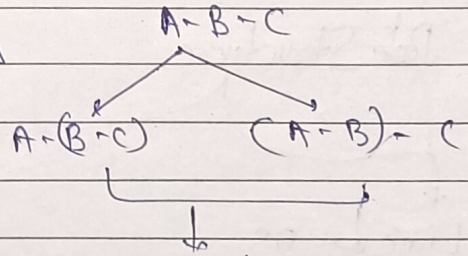
④ It should take 0 or more I/p.

```
Add 23(){          //here we take no inputs
Take variable x;
x = 2+3;
print(x);
}
```

⑤ Corresponding to given set of inputs we must have fixed o/ps. (propose mapping)

⑥ Platform Independent (no dependency) → Hardware, → Software

⑦ Every statement must be Unambiguous



Aisa nhi hona chahiye.

Steps Required to derive an Algorithm :-

Page:
Date: / /

Step1 → Problem Defination → we can predict the o/p by giving the set of inputs.
Identify and define Problem properly

Step2 → Development Model →
make DFD or any other model to describe problem Graphically

Step3 → Design →
Identify steps in algorithm & write them in proper sequence

Step4 → Testing →
Try to find errors in Algorithm.

Step5 → Analysis → (parameteres to find out which is best algo.° out of all algo.°)
measure its time and space complexity.

Step6 → Optimise →
Try to minimise time & Space Complexity of algorithm.

Step7 → Implementation →
Implement algo. in any Programming language.

★ Analysis Of Algorithm ★

performance is a subjected Term..
If an Algo°. Performance is good → where we are using that Algo°. in which it's performing good.

How to Analyse Performance of Algo°.)

① Aposteriori Algorithm.
(After Implementation)
Exact Time ⇒ CPU
Exact Memory (in main memory)

② Apriori Algorithm
(Before Implementation)
Approximate Time → in CPU time.
Approximate Space (in main memory)

A Posteriori Algorithm →