NOTES

1) **What is Problem solving? Explain the steps in Problem solving?**
   Problem solving is the process of overcoming issues, mistakes, errors, failures and risks to move forward.

   The steps followed to solve problems are:-
   a) Understanding the Problem
   b) Designing the algorithm
   c) Analysis of Algorithm
   d) Coding / Implementation
   e) Testing and Debugging

a) **Understanding the Problem:** It is important to clearly understand a problem before we begin to find the solution for it. We need to read and analyze the problem statement carefully in order to come up with a solution for it.
b) **Designing the algorithm:** Once the problem statement is understood, we need to design the steps to be followed to come up with the solution to the problem.
c) Analysis of Algorithm: the important task in algorithm development is to find out the time complexity and space complexity of an algorithm to prove that an algorithm is efficient.
d) Implementation or Coding: We need to convert the algorithm into a format which can be understood by the computer to generate the desired solution.
e) Testing and Debugging: Program created should be tested on various parameters. Debugging is the process of identifying and correcting or removing the bugs.

2) **What is an algorithm? Mention its Advantages and Disadvantages?**
   An algorithm can be defined as a step-by-step procedure for accomplishing a task.

Or

An algorithm is a sequence of unambiguous instructions for solving a problem.

Advantages:-

- It is easy to understand.
- An algorithm uses a definite procedure.
- It is easy to debug.
- It is easier for programmer to convert it into an actual program.

Disadvantages:-

- It is Time consuming.
- Difficult to show Branching and Looping in Algorithms.
- Big tasks are difficult to put in Algorithms.

3) Explain the Characteristics of Algorithm?

An algorithm must possess the following characteristics (features).

1. Finiteness: An algorithm must terminate in a finite number of steps
2. Definiteness: Each step of the algorithm must be precisely defined.
3. Effectiveness: Each step must be effective, in the sense that it should be easily convert able into program statement and should be performed exactly in a finite amount of time.
4. Input: The algorithm must be have zero or more inputs. These inputs are supplied to the algorithm.
5. Output: Each algorithm must produce one or more output values.

4) What is Flowchart? Mention its Advantages and Disadvantages?

A flowchart is visual or graphical representation of an algorithm.

Advantages:-

- Easy to make.
- Mistakes can be easily identified.
- Communication becomes effective and easy to understand.
- Logics can be easily interpreted.

Disadvantages:-

- It is a time consuming process.
- No scope for alteration or modification.
- No man to computer communication.

5) Examples of Algorithm?

Example

1) To find the average of three numbers, the algorithm is as follows

Step 1: Read the numbers a, b, c
Step 2 : Compute the sum of a, b and c
Step 3: Divide the sum by 3
Step 4 : Store the result in variable d
Step 5 : Print the value of d
Step 6 : End of the program

2) **Write an algorithm to calculate the simple interest**
using the formula Simple interest = P*N* R/100.

Where P is principle Amount, N is the number of years and R is the rate of interest.

Step 1: Read the three input quantities' P, N and R.
Step 2 : Calculate simple interest as
Simple interest = P* N* R/100
Step 3: Print simple interest.
Step 4: Stop.

3) Area of Triangle: Write an algorithm to find the area of the triangle.

Let b, h be the base and height of the triangle ABC
Step 1: Input the given elements of the triangle namely base b and height h.
Step 2: Area = (1/2) *b*h
Step 3: Output the Area
Step 4: Stop.

4). Write an algorithm to find the greatest among three numbers.

Step 1: Start.

Step 2: Read a, b , c.
Step 3: If a > b then go to step 6.
Step 4: if c > b then go to step 8
Step 5: Otherwise Print “b is largest” go to step 9.
Step 6: If c > a then go to step 8.
Step 7: Otherwise Print “ a is largest” go to step 9
Step 8: Print “c is largest”
Step 9: Stop.

5).**Write an algorithm to calculate the perimeter and area of rectangle.**

Step 1: Read length of the rectangle.
Step 2: Read width of the rectangle.
Step 3: Calculate perimeter of the rectangle using the formula perimeter = 2* (length + width)
Step 4: Calculate area of the rectangle using the formula area = length * width.
Step 5: Print perimeter.
Step 6: Print area.
Step 7: Stop.

6) What are the Differences between Algorithm and Flowchart?

Algorithm

- An algorithm can be defined as a step-by-step procedure for accomplishing a task.
- It is easy to debug.
- In the algorithm, plain text is used.
- For complex programs, algorithms prove to be inadequate.
  It is complex to understand.

Flowchart

- A flowchart is a visual or graphical representation of an algorithm.
- It is hard to debug.
- In the Flowchart, symbols/shapes are used.

- For complex programs, Flowcharts prove to be adequate.
  It is easy to understand.

7) Explain Pseudo code with example?

Pseudo code is a simple way of describing a set of instructions that does not have to use specific syntax.

Common pseudo code notation:-

- Input
- Output
- While
- For
- Repeat until
- If then else

Example:-

```
REPEAT
OUTPUT "What is the best subject you take?"
INPUT user inputs the best subject they take
STORE the user's input in the answer variable
IF answer = 'Computer Science' THEN
OUTPUT 'of course it is!'
ELSE
OUTPUT 'try again!'
UNTIL answer = 'Computer Science'
```

8) Explain the Practical Applications of Algorithm?

- Internet: With the aid of these algorithms, various sites on the Internet are able to manage and manipulate this large volume of data. Finding good routes on which the data will travel and using search engine to find pages on which particular information is present.
- E-Commerce: The day-to-day electronic commerce activities is hugely dependent on our personal information such as credit/debit card numbers, passwords, bank statements, OTPs and so on. The core technologies used include public-key cryptography and digital

signatures which are based on numerical algorithms and number theory.

- Google's search engine uses Page Ranking algorithm to find the best matches for search terms. It decides which pages are listed first when you search for something.
- Weather Forecasting: These algorithms are used to model weather patterns and make predictions.
- Shortest path algorithm: This also has an extensive use as In a transportation firm such as a trucking or railroad company, may have financial interest in finding shortest path through a road or rail network because taking shortest path result in lower labour or fuel costs.
- Other important applications of algorithms: Speech recognition, image processing, facebook's friend suggestion algorithm, product recommendation in e-commerce sites like amazon etc.

9) Explain algorithm as a technology?

- System performance depends on choosing efficient algorithms which can compute the correct solution in least given time as much as on choosing fast hardware.
- Algorithms are widely used throughout all areas of IT. In mathematics and computer science, an algorithm usually refers to a small procedure that solves a recurrent problem.
- Algorithms as a technology is widely prevalent in aerospace applications, healthcare applications, automotive applications, ecommerce, social media etc.

10)What do you mean by "instance of a problem." and "correctness of an algorithm"?

An "instance of a problem" consists of the input needed to compute a solution to the problem.

An algorithm is said to be correct if, for every instance, it gives the correct output. An incorrect algorithm might not yield correct answers for all the input instances.

11)Explain different design approaches to solve a problem or Designing Algorithm?

There are many ways to design algorithms for a given problem. The following are some of the popular design approaches.

- **Brute Force Algorithm**: This is a straightforward approach to a problem. Here we will be iterating every possibility available to solve that problem.
- **Recursive Algorithm**: This type of algorithm is based on recursion. Here the problem is solved by breaking it into sub-problems of the same type until the problem is solved.
  Examples:- Problems solved using recursion are Fibonacci series, Tower of Hanoi, Factorial of a number etc.
- **Divide and Conquer**: This involves in solving particular problem by dividing it into one or more sub problems of smaller size, recursively solving each sub problem and then merging the solutions to the original problem.
  Examples:- This algorithm is applicable to Quick Sort and Merge Sort, Multiplying a large number etc.
- **Greedy approach**: A greedy algorithm is used to solve optimization problems. It does not always guarantee the optimal solution however it generally produces solutions that are very close to the optimal solution.
  Examples:- This algorithm is used for the problems on Prim's Algorithms, Kruskal Algorithm etc.
- **Dynamic Programming**: This algorithm works by remembering the results of a previous run and using them to arrive at new results.
  Examples:- This algorithm is used to solve the Knapsack Problem, Dijkstra Shortest path Algorithm etc.
- **Backtracking algorithms:** This algorithm is used to find a solution in an incremental manner. There is often recursion / repetition involved and attempts are made to solve the problem one part at a time.
  Examples:- The problems that can be solved through the this algorithm are M-Colouring Problem, Rat in maze Problem, Hamilton cycle etc.

12) Explain Analysis or Performance Analysis of Algorithm?

Performance analysis of an algorithm is the process of calculating space required by that algorithm and time required by that algorithm.

Algorithm can be analyzed in two ways:

- By checking the correctness of an algorithm.
- By measuring time and space complexity of an algorithm.

Time Factor: Time is measured by counting the number of key operations such as comparisons in the sorting algorithm.

Space Factor: Space is measured by counting the maximum memory space required by the algorithm.

To analyze the algorithm, 2 phases are required.

- **Priori Analysis** – "Priori" means before. This analysis is done before its implementation.
- **Posteriori Analysis** – "Posterior" means after. This analysis is done after implementing the algorithm in any programming language.

13) Explain Priori Analysis and Posteriori Analysis?

Priori Analysis – "Priori" means before. This analysis is done before its implementation.

- Total time taken by the algorithm = The number of times the statement will be executed (frequency count) x time taken for one execution
- The notations used in Priori analysis are Big-oh (O), Omega (Ω), Theta(θ), small-oh(o).

Posteriori Analysis – "Posterior" means after. This analysis is done after implementing the algorithm in any programming language.

- Testing a program consists of 2 major phases.

a) **Debugging:** After execution of program if there are faulty results, then they are corrected using this approach.

b) **Profiling:** It is the actual time taken by the algorithm to process the data.

Procedure: **Profiling**
{
1. Read data
2. Time (t1)
3. Process (data)
4. Time (t2)
5. Write (time=t2 – t1)

}

14) What are the difference between Priori Analysis and Posteriori Analysis?

Priori Analysis	**Posteriori Analysis**
Analysis is the process of determining how much computing time and storage an algorithm will require.	Profiling will measure the exact time and space required to compute the results.
Independent of programming language and machine.	Depends on machine, compiler and programming language used
It will give approximate answer.	It gives exact answer.
Uses asymptotic notations to represent the time taken.	It does not depend on the asymptotic notations to represent the time taken.
Analysis is done on the algorithm and not the code.	Analysis is done on the actual code 1. Read data 2. Time (t1) 3. Process (data) 4. Time (t2) Write (time=t2 – t1)

15) What is Complexity of an algorithm? What are its types?

- Complexity of an algorithm is a measure of the amount of time and /or space required by an algorithm for an input of a given size
- **Types of algorithm complexity are:**
  - Time complexity – Time required to complete the task of that algorithm
  - Space complexity – Space required to complete the task of that algorithm.

16) Explain Space Complexity of an Algorithm?

- The total amount of computer memory required by an algorithm to complete its execution is called as space complexity of that algorithm.
- During program execution the computer memory is used for
  - **Instruction space:-** It is the amount of memory used to store compiled version of instruction.
  - **Environmental stack:-** It is the amount of memory used to store information of partially executed functions at the time of function call.
  - **Data space:-** It is the amount of memory used to store all the variables and constants.

The space needed by an algorithm consists of the following components:-

a) **The fixed static part:-** A fixed part that is a space required to store certain data and variables, that are independent of the size of the problem. For example, simple variables and constants used, program size, etc. Let Cp be the space required for the static part.

b) **The variable dynamic part:-** A variable part is a space required by variables, whose size depends on the size of the problem. Let Sp be the space required for the static part.

The overall space requirements for an algorithm is the sum of both the fixed static part storage and variable dynamic part storage. If P be a program, then space required for program P will be denoted by S(P).

$$S(p)=Cp + Sp$$

Example:- Algorithm for sum of two numbers.

```
algorithm sum(p,q,r)
  {
     p=1;
     q=1;
    r=p + q;
}
```

S(p)= Cp + Sp
=1 + 1 + 1 + 0
=3 =>0(1)

17) Explain Time Complexity of an Algorithm?

The total amount of time required by an algorithm to complete its execution is called Time complexity.

There are 3 cases:

1. **Best case**: When minimum time is required to complete its execution.
2. **Average case**: The amount of time is neither more nor less for its execution.
3. **Worst case**: Maximum amount of time is required to complete its execution.

Example:-

Let us consider three different algorithms such as A1, A2 and A3 with execution time 5 seconds, 2 seconds and 3 seconds respectively.

Algorithm	Running time of an algorithm	
A1	5s	→ **Worst case**
A2	2s	→ **Best case**
A3	3s	→ **Average case**

18)What is Time-Space Tradeoff?

Space time tradeoff is a way of solving a problem or calculation in less time by using more storage space.

OR

By solving a problem in a very little space by spending more time.

19) Explain the Asymptotic Notations?

Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

There are mainly three asymptotic notations:-

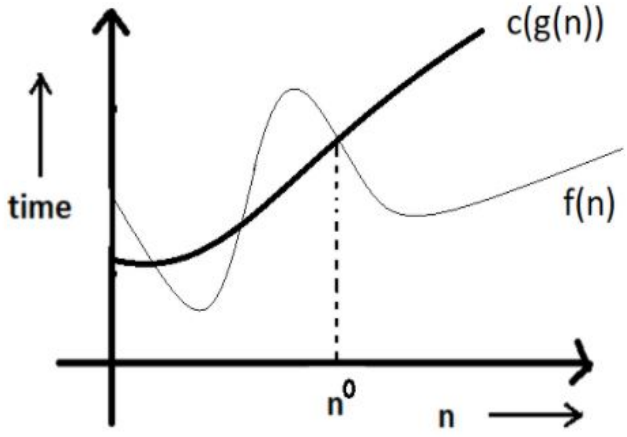
- Big Oh Notation (O-notation)
- Big Omega Notation (Ω-notation)
- Big Theta Notation (Θ-notation)

Big oh notation (O):

- Big oh notation is used to describe an asymptotic upper bound.
- Mathematically, if f(n) describes the running time of an algorithm; f(n) is O(g(n)) if and only if there exist positive constants c and n_0 such that:

  $0 \leq f(n) \leq c\ g(n)$ for all $n \geq n_0$
- Here, n is the input size, and g(n) is any complexity function, for, e.g. n, n2, etc. (It is used to give upper bound on a function)
- If a function is O(n), it is automatically $O(n^2)$ as well. Because it satisfies the equation given above.

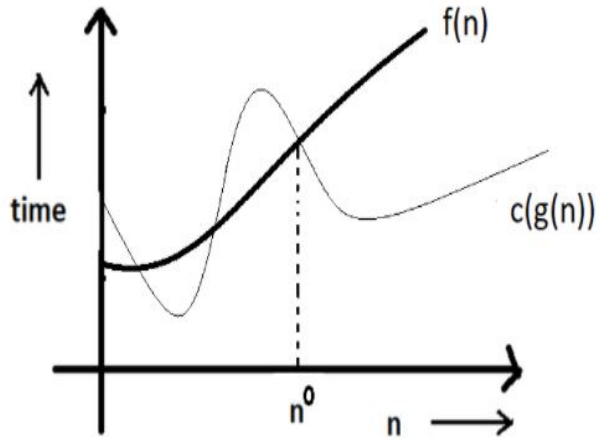
Graphic example for Big-Oh notation:-



Big Omega Notation (Ω):

- Big Omega notation is used to describe an asymptotic upper bound.
- Let f(n) define the running time of an algorithm; f(n) is said to be **Ω** (g(n)) if and only if there exist positive constants c and $n_0$such that:
  $0 \leq c\ g(n) \leq f(n)$ for all $n \geq n_0$
- It is used to give the lower bound on a function.
- If a function is $O(n^2)$, it is automatically O(n) as well. Because it satisfies the equation given above.

Graphic example for Big-Omega notation:-



Big Theta notation (θ):

- Let f(n) define the running time of an algorithm.
- f(n) is said to be θ (g(n)) if f(n) is O (g(n)) and f(x) is **Ω** (g(n)) both. Mathematically,

$$0 \leq f(n) \leq c_1 g(n) \quad \forall \quad n \geq n_o$$
$$0 \leq c_2 g(n) \leq f(n) \quad \forall \quad n \geq n_o$$

for sufficiently large value of n

Merging both the equations, we get:

$0 \leq c2\ g(n) \leq f(n) \leq c1\ g(n) \quad \forall \quad n \geq no.$

The equation simply means that there exist positive constants c1 and c2 such that f(n) is sandwiched between c2 g(n) and c1 g(n).

Graphic example of Big Theta notation(θ):-