Tutorial No - 1

Q 1) Write note on 'recursive algorithms'. Explain recursive algorithms with examples.

→ • Recursive Algorithms :-

1. Recursive function is a function that is a partially defined by itself.
2. Recursion is usually slower than iteration due to overhead of maintaining stack
3. Recursion uses more memory than iteration
4. Infinite recursion can crash the system.
5. Recursion makes code smaller.
6. Recursive algorithm is a method of simplification that divides the problem into subproblems of the same nature.
7. Generation of factorial, fibonacci number series are the examples of recursive algorithms.
8. It is specially good for working on things that have many possible branches and are too complex for an iterative approach.

a) Give recursive and iterative algorithm for fibonacci series

→ • Recursive →

```
int fibonacci (int n)
{
    if (n<=1)
        return 1;
    else
        return (fibonacci (n-1)+fibonacci (n-2));
}
```

- iterative →

```
for (i=2 ; i<n ; i++)
{
    n3 = n1 + n2 ;
    n1 = n2 ;
    n2 = n3 ;
}
```

b) Give recursive & iterative algorithm to find factorial of given number.

→

- Recursive →

```
int factorial (int n)
{
    if (n=0)
    return 1;
    else
    return (n* factorial (n-1));
}
```

- Iterative →

```
if (n>1) {
for (i=n ; i<= n, i--)
{
    fact = fact * n;
}
}
else
    return 1;
```

c) Give recursive and iterative algorithm to find GCD of given numbers.

→ • Recursive →

```
int gcd (int m, int n)
{
  if ((m % n == 0)
   return n;
  else return gcd (n, m % n);
}
```

• Iterative →

```
for (i=1 ; i<=n1 && i<=n2 ; i++)
{
   if (n1 % i==0 && n2 % i==0)
      gcd = i ;
}
```

d) Give recursive and iterative algorithm to solve the problem of "Towers of Hanoi".

→ • Recursive →

```
Void towers (int num , char frompeg , char tapeg,
                    char allxpeg)
{
  if (num==1)
  {
    printf ("\n Move disk 1 from peg %c
       to peg %c", frompeg , tapeg);
    return;
  }
```

towers (num-1, frompeg, auxpeg, topeg);
printf ("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);
towers (num-1, auxpeg, topeg, frompeg);

- Iterative →

Q 2) What is algorithm ? What are the different characteristics of algorithm ?

→ An algorithm is defined as the sequence of instructions written in simple english that are required to get the desired results. It helps to develop the fundamental logic of a problem that leads to a solution.

Algorithm is a step by step procedure, which defines a set of instructions to be executed in a certain order to get a desired output.

Algorithms are generally created independent of underlying languages, that is an algorithm can be implemented in more than one programming language.

There are some characteristics which every algorithm should follow. There are five different characteristics which deal with various aspects of algorithm. they are as follows :-

1) Input specified :-

An algorithm should have 0 or more well defined inputs. Input precision requires that you know what kind of data, how much and what form the data should be

2) Output specified :-

The output is the data resulting from the computation (intended Result). An algorithm should have 1 or more well defined outputs and should match the desired output

3) Definiteness -

definiteness means specifying the sequencing of operations for turning input into output. Algorithm should be clear and unambiguous. Details of each step must be also spelled out (including how to handle errors).

4) Effectiveness :-

for an algorithm to be effective, it means

that all those steps that are required to get to output must be feasible with the

5) Finiteness :-

The algorithm must stop, eventually Algorithms must terminate after a finite number of steps. An algorithm should not be infinite and always terminate after definite number of steps.

6) Independent -

An algorithm should have step-by-step directions, which should be independent of any programming code. It should be such that it could be run on any of the programming languages.

(Q3) Write a note on "Algorithm specification".

→ We can depict an algorithm in many ways.-

- Natural language :
implement a natural language like English.

- Flow Charts :
Graphic representations denoted flowcharts. only if the algorithm is small and simple.

- Psedo code :
This psedo code skips most issues of

ambiguity : no particularly regarding syntax programming language

Psedo code conventions -

# while (condition) do
{
(statement 1)

(statement 2)
}

# for variable = value 1 to value2 step · step do
{
(statement +1)
:
(statement +n)
}

# A repeat-until statement is constructed as follows
repeat
(statement 1)
:
(statement n)
until (condition)

# if (condition) then (statement)
if (condition) then (statement 1) else (statement 2)

Q4) Write note on "Asymptotic notations"
or
Explain different asymptotic notations.
or
Define -
(a) Big oh (o) (b) omega (Ω)
(c) theta (Θ) (d) Little oh (o)
(e) little omega (ω).

→ Asymptotic notations is used to describe the running time of an algorithm - how much time an algorithm takes with a given input, n.

There are three different notations: big O, big theta (Θ), big Omeg (Ω)

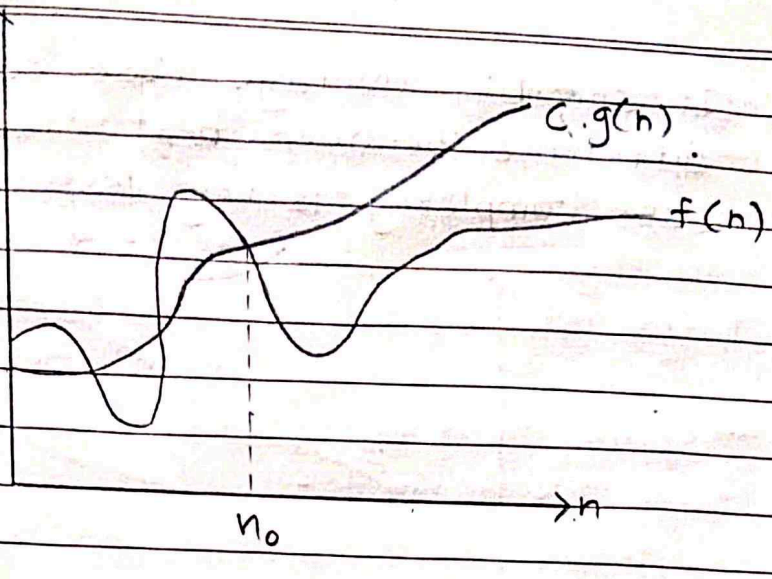
Asymptotic notations are ~~one big Omeg (~~ used to represent the complexities of algorithms

a) Big oh (o) →

The function $f(n) = o(g(n))$
iff there exist positive constants c and no such that,

$f(n) <= c * g(n)$ for all n, $n > no$.

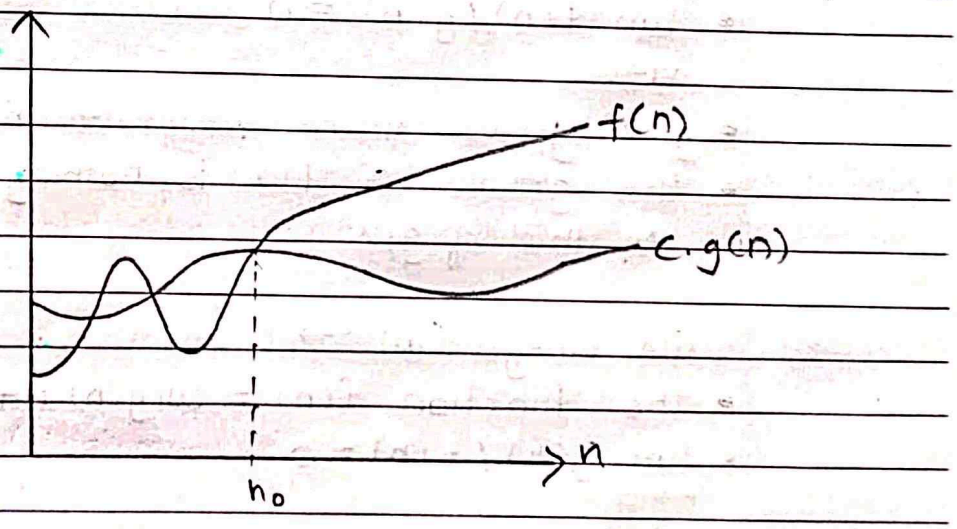
- It specifies upper Bound worst case complexity.



b) Omega (Ω) →

The function $f(n) = \Omega(g(n))$
iff there exist positive constants c and no such that $f(n) >= c * g(n)$ for all n, $n > no$

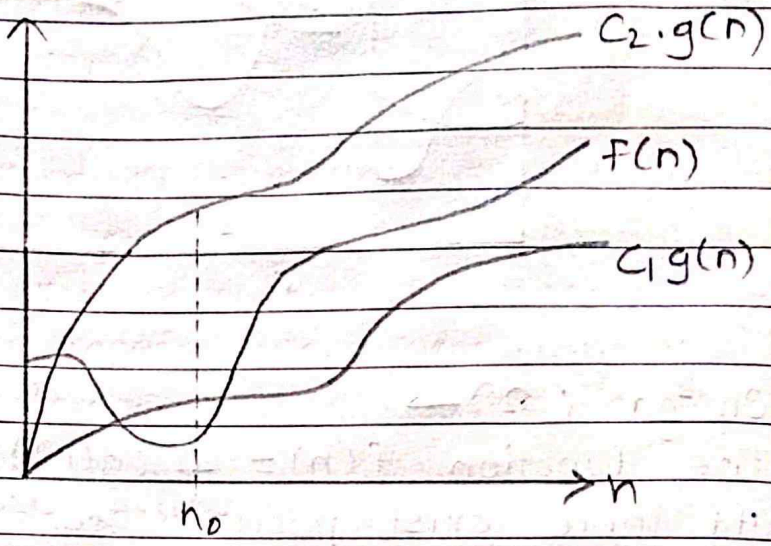
- It specifies lower Bound Best case compelxity



c) Theta (Θ) →

The function $f(n) = \Theta(g(n))$
iff there exist positive constants c_1, c_2 and no such that $c_1 * g(n) <= f(n) <= c_2 * g(n)$
for all n, $n > n_0$

It specifies average case complexity difference between best case and worst case complexity is very less.



d) little oh (o) :→

- The function $f(n) = o(g(n))$ iff
- $\lim_{n \to \infty} f(n) / g(n) = 0$
- $g(x)$ grows much faster than $f(x)$.
- the growth of $f(x)$ is nothing compared to that of $g(x)$

e) little omega (ω) →

- The function $f(n) = \omega(g(n))$ iff
- $\lim_{n \to \infty} g(n) / f(n) = 0$

(Q 5) Explain time and space complexity.

→ Generally, there is always more than one way to solve a problem in computer science with different algorithms. Therfore, it is highly required to use a method to compare the solutions in order to judge which one is more optimal.

There are two such methods use,
1) Time complexity
2) space complexity

1) Time complexity :-
- The time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of length of the input.
- The time T(P) taken by an program P is the sum of the compile time and the run time (execution time)
- compile time does not depends on instance characteristics.
- Program compiled once can be run several times.
- so only run time is concerned with time complexity tp.
- key operations like comparison are considered

2) Space

2) Space complexity :→

- The space complexity of an algorithm quantifies the amount of space taken by an algorith to run as a function of length of input.

- Space need by algorithm is sum of

1) A fixed part -
   Which is independent of characteristics of the inputs and outputs.
   e.g. Instruction space, space for variables

2) A variable part -
   Which consist of space needed by component variables whose size is depedent on the particular problem instance being solved
   e.g. Space needed by reference variable
   recursion stack space.
   $S(P) = C + Sp$

Q6) Explain performance analysis and performance measurement.

→

Q 7) Write note on "Randomized Algorithms".
Differentiate between las vegas and Monte Carlo Algorithms.
Give solution to the problem of "finding repeated element" and "Primality Testing".

→ Randomized Algorithms →

- A randomized algorithms is one that makes use of a randomizer.
- such as a random number generator.
- Some of the decisions made in the algorithm depend on the output of the randomizer.
- The output of a randomized algorithm could also differ from run to run for the same

input.

- The execution time of a randomized algorithm could also vary from run to run for the same input.
- The inputs for a randomized algorithm are similar to those of deterministic algorithms along with sequence of random bits that can be used by the algorithm for making random choices.
- Randomized algorithms are known for their simplicity.
- An algorithm that uses random numbers to decide what to do next anywhere in its logic is called randomized Algorithm.