Part - A.

1. Write Consecutive integer checking algorithm for computing gcd (m,n)

```
if (m>n)
        t = n
else
        t = m
if (m/t == 0) {
    if (n/t == 0)
            return t.
    else
            t = t-1
}
```

2. Find GCD (31415, 14142) by applying Euclids algorithm.

gcd(31415, 14142)
= gcd (14142, 3131)
= gcd (3131, 1618)
= gcd (1513 , 105)
= gcd (1513 , 105)
= gcd(105, 43)
= gcd (43, 19)
= gcd(19, 5)
= gcd (5 ,4)
= gcd (4,1)
= gcd(1,0)
=1.

3. How do you measure the efficiency of an algorithm?

We can measure the efficiency of an algorithm using

(•) Time effiency :

indicates how fast an algorithm runs

(•) Space efficiency:

refers to the amount of memory units required by the algorithm in addition to the space needed for its input & output.

4. List the reasons for choosing an approximate algorithm.

(•) An approximation algorithm gurantees to run in polynomial time though it does not gaurantee the most effective solution.

(•) An approximation algorithm guarantees to seek out high accuracy and top quality solution

(•) Approximation algorithm are used to get an answer near the (optimal) solution of an optimization problem in polynomial time.

5. What i

To id

algorithm

contributi

and c

execute

6. List

from

(n-2)

7. Comp

8. Compa

using

lim

n→

5. What is Basic operation in algorithm?

To identify the most important operation of the algorithm called the basic operation, the operation contributing the most to the total running time, and compute the number of times the basic operation executed.

6. List the following according to their order of growth from lowest to the highest:

$(n-2)!$, $0.001n^4 + 3n^3 + 1$, 3^n, $n \log n$, $5n^2+6$, $\log n$

7. Compare the orders of growth of $n!$ and 2^n using limit

8. Compare the orders of growth of $\frac{1}{2}n(n-1)$ and n^2 using limits.

$$\lim_{n \to \infty} \frac{\frac{1}{2}n(n-1)}{n^2} = \frac{1}{2} \lim_{n \to \infty} \frac{n^2-n}{n^2} = \frac{1}{2} \lim_{n \to \infty} \left(1 - \frac{1}{n}\right) = \frac{1}{2}$$

$$\frac{1}{2} n(n-1) \in \Theta(n^2)$$

9. List the general plan for the empirical analysis of algorithm time efficiency.

(i) Select a specific (typical) sample of inputs

(ii) use physical unit of time eg., (milli seconds)
(or)
count actual number of basic operation's execution

(iii) analyse the empirical data.

10. Define algorithm visualization.

The algorithm visulization refers to the diagramatical representation of the efficiency of an algorithm. The efficiency of an algorith is verified by varying or increasing the input size. Example: Graph, pie chart.

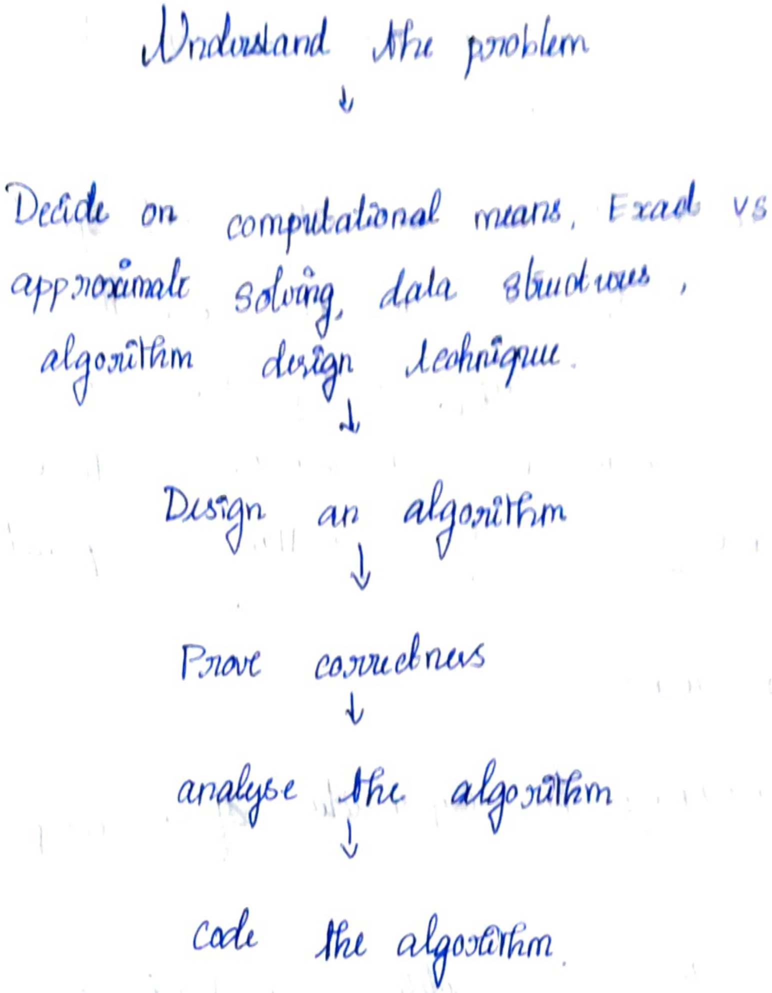
11a. D

anal

Wn

Part - B

11a. Discuss the sequence of steps in designing and analysing an algorithm with neat flowchart



Understand the problem

Decide on computational means, Exact vs approximate solving, data structures, algorithm design technique.

Design an algorithm

Prove correctness

analyse the algorithm

code the algorithm.

Understanding the problem:

- understand the problem before designing an algorithm
- Read the problem description carefully & ask questions if you have any doubts about the problem, solve few small examples by hand and think about special cases.

- * Select the known algorithm
- * How the algorithm work with the particular problem. Find the strength and weakness of the particular algorithm.
- * If the algorithm is not available, then design your own algorithm.
- * An input to an algorithm ~~spt~~ specifies an instance of the problem the algorithm solves. Import to specify exactly the range of instances the algorithm needs to handle so that the boundary value of algorithm get fixed.

2. Decision making:

a) Asscertaining the capabilities of a computational device:

- * Once you completely understand the problem, you need to ascertain the capabilities of the computational device the algorithm is intended for.
- * Based on computational device, classify the algo.

→ Sequential: instruction are executed one after another, one operation at a time.

→ parallel : concurrently or parallel execution.

b) Choosing between exact and approximate problem solving:

- * Solving the problem exactly is called exact problem solving eg: sorting.
- * solving the problem approximately is called approximation algorithm eg: Integral, TSP.

Deciding on appropriate data structures.

- * To design a better program, use data structure concept. Data structure and algorithm work together for these are interdependent.

Algorithm + data structure = Programs.

Algorithm design Techniques:

- * General approach to solve the problem algorithmically and that must be suitable for all types of input.

It should produce desired output in a finite time

Technique : Brute force, Divide & conquer, dynamic programming

Prove Correctness of an algorithm

- * A correct algorithm is not only works most of the time but one that work correctly for all types of input.

* The algorithm yields required result for all possible input in a finite amount of time. For all input it should produce desired output.
* Proof the correctness using mathematical induction

Analysing the algorithm:

* Time efficiency - How fast the algorithm works
* Space - How much extra memory the algorithm needs
* Simplicity - which are easy to understand.
* Generality - one algorithm must solve all kinds of algo problems.

Coding the algorithm.

* Convert the algorithm to program using any one high-level language.

-> Range of i/p - whether giving correct input

-> optimizing code - reduce unnecessary code

-> Redesign - modification can be done in any steps

-> analyse the o/p - Getting correct op or not for all positive combination of input.