Shahul Hameed. S
1KN18CS097
CSE 'A' Sec

01. What is an algorithm? Explain the properties of an algorithm. Explain the Notion of an Algorithm with an example.

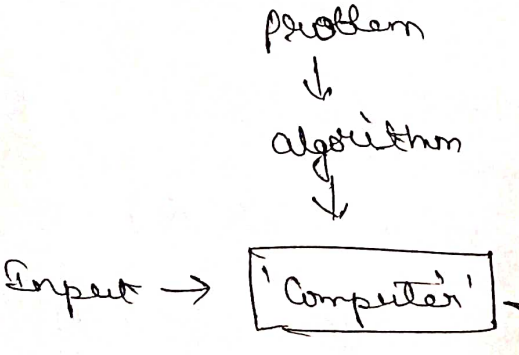
Algorithm: An algorithm is a finite sequence of unambiguous instructions to solve a particular problem.

Properties of algorithm:

1. Input: Zero or more quantities are externally supplied.
2. Output: Atleast one quantity is produced.
3. Definiteness: Each instruction is clear and unambiguous. it must be perfectly clear what should be done.
4. finiteness: if we trace out the instructions of an algorithm, then for all cases, the algorithm terminates after a finite number of steps.
5. Effectiveness: Every instructions must be very basic so that it can be carried out, in principle it is not Enough that each operation be definite as in Criterion C', it also must be feasible.

Notion of an algorithm:

> The Non-ambiguity requirement for each step of an algorithm Can't be compromised.



The range of inputs for which an algorithm works has to be specified carefully.

The same algorithm can be represented in several different ways.

Several algorithms for solving the same problem may exist.

Algorithms for the same problem can be based on very different ideas can resolve the problem with dramatically different speeds.

Ex:- gcd (greatest common divisor) of two non-negative m, n integers

$gcd(m,n) = gcd(n, m \bmod n)$

By Euclids algorithm.

$gcd(60,24) = gcd(24,12) = gcd(12,0) = 12$

(2) P.T (i) $\frac{1}{2}(n-1)n \in \Theta(n^2)$

for comparison we will consider value of n

if $n=2$

$f(n) = \frac{1}{2} n(n-1)$ $g(n) = n^2$

$= \frac{1}{2}(2)(2-1)$ $= 2^2$

$f(n) = 1$ $g(n) = 4$

if $n=4$

$f(n) = \frac{1}{2} 4(4-1)$ $g(n) = 4^2 = 16$

$f(n) = 6$

All such comparisons indicate that

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

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

Θ - notation says $c_2 g(n) \leq f(n) \leq c_1 g(n)$

(ii) $n! \in \Omega(2^n)$

If $n=2$ $\quad f(n) := n! \quad g(n) = 2^n$

$= 2! \qquad = 2^2$

$= 2 \qquad = 4$

If $n=3$ $\quad f(n) = 3! \quad g(n) = 2^3$

$= 6 \qquad = 8$

$f(n) \not\geq c * g(n)$

If $n=5$ $\quad f(n) = 5! \quad g(n) = 2^5 = 32$

$= 100$

$\therefore f(n) \geq c * g(n)$

(3) With suitable example, Explain the significance of order of growth in analysing algorithms?

7 Measuring the performance of an algorithm in relation with the input size called order of growth.

Significance: All Exponential functions belong to the same order of growth regardless of the base of the Exponent. Exponential functions grows very quickly so Exponential algorithm are only useful for small problems.

Similar for the log terms, the base of log doesn't matter. Changing bases is the equivalent of multiplying by a Constant, which doesn't change order of growth.

Ex:-

n	$\log n$	$n \log n$	n^2	2^n
1	0	0	1	2
2	1	2	4	4
4	2	8	16	16
8	3	24	24	256
16	4	64	256	65,536

from the above table log function is the slowest growing function. And the Exponential function 2^n is fastest and grows rapidly with varying I/p size n.

(4) Explain general plan of mathematical Analysis of non-recursive algorithms with Example.

> * Decided on a parameter indicating an input size.
> * Identify the algorithms basic operation.
> * check whether the number of times the basic operation is executed depends only on the size of an input. If it also depends on some additional property, the worst case average case and if necessary, best case efficiences have to be investigated seperately.
> * set up a sum expressing the number of times the algorithms basic operation is executed.
> * using standard formulas and rules of sum manipulation, Either find a closed form, formula for count, or, at the very least, Establish its order of growth.

Ex: To find maximum Element in the given array.

Algorithm: max Element A[0 ... n-1]

max val ← A[0]

for i ← 1 to n-1 do

if A[i] > maxval

max val ← A[i]

return max value

Here comparision is basic operation

$$C(n) = \sum_{i=1}^{n-1} 1 = n-1 \in \theta(n)$$

(5) Explain about towers of Hanoi problem and S.T the Efficiency of this algorithm is Exponential.

There are 3 pegs A, B & C. The 5 disks of different diameters are placed on peg A. The arrangement of the disks is such that every small disk is placed on the larger disk.

Problem of "Towers of Hanoi" states that move the given five disks from peg A to peg C using peg B as an auxillary.

The conditions are:-

(i) Only the top disks on any peg may be moved to any other peg.

(ii) A larger disk should never rest on the smaller one

The solution can be stated as,

- Move top n-1 disks from A to B using C as auxillary
- Move the remaining disk from A to C.
- Move the n-1 disks from B to C using A as auxillary

```
algorithm :- TOH (n, A, C, B)
  {
   if (n = 1) then
   write ("The peg moved from A to C")
    return ;
  }
  Else
     {
      TOH (n-1, A, B, C);
      TOH ( n-1, B, C, A);
     }
```

Proof to prove algorithm is exponential is by mathematical Induction.

Basic step:

$$n - i = 1 \Rightarrow i = n - 1$$

$$M(n) = 2^i M(n-i) + 2^i - 1$$

$$= 2^{n-1} M(n-(n-1)) + 2^{n-1} - 1$$

$$= 2^{n-1} M(1) + 2^{n-1} - 1$$

$M(n) = 2^n - 1$ Now if $n = 1$ then

$M(1) = 2^1 - 1 = 1$ is proved.

Induction step: $M(n) = 2M(n-1) + 1$

$$M(n) = 2(2^{n-1} - 1) + 1$$

$$= 2^n - 2 + 1$$

$M(n) = 2^n - 1$ is proved.

we get recurrance as,

$$M(n) = 2^n - 1$$

$$\theta(2^n - 1) = \theta(2^n)$$

∴ The efficiency of Tower of Hanoi algorithm is exponential.

(6) Briefly explain important fundamental data structures used in its design.

A data structure can be defined as a particular scheme of organizing related data items.

1. Linear data structure: The two most important Elementary data structures are the array and the linked list.

(a) array :- An array is a sequence of n items of the same data type that are stored contiguously in computer memory and made accessible by specifying a value of the array's index.

Array of n-Elements.

(b) Linked list :- A linked list is a sequence of zero or more elements called nodes.

(c) Stack :- It is a list insertions and deletions can be done only at one end. This is called top.

(d) Queue :- A queue is a list from which elements are deleted from one end of the structure called the front, and new elements are added to the other end called rear.

(e) graph :- A graph is informally thought of as a collection of points in plane called vertices.

(f) trees :- A tree is connected Acyclic graph. A graph may no cycles but is not necessarily connected.

(3) Explain the general concept of divide and conquer method. general algorithm D and C (P) [where p is the problem to solve] to illustrate this technique. Discuss general divide and conquer technique.

* In divide and conquer method, a given problem is
(i) divided into smaller sub problems.

(ii) These sub problems are solved independently.

(iii) Combining all the solutions of sub problems into a solution of the whole.

If the sub problems are large enough then divide and conquer is reapplied.

* The generated sub problems are usually of same type as the original problem.

* The control abstraction for divide and conquer is as given below using control abstraction as flow of control of procedure is given.

* The computing time of above procedure of divide and conquer is given by the recurrence relation.

$$T(n) = \begin{cases} g(n) & \text{where } n \text{ is small} \\ T(n_1) + T(n_2) + \dots + T(n_i) + f(n) & \text{where } n \text{ is sufficiently large.} \end{cases}$$

Here,

$T(n)$ is the time for divide and conquer of size n.

$g(n)$ is the computing time required to solve small inputs.

$f(n)$ is the time required in dividing problem (P) and combining the solutions to sub problems.

=> algorithm : DC (P):

```
{
  if P is too small then
  return solution of P
  Else
  {
    divide(P) and obtain P1, P2 ... Pn
```

where n ≥ 1

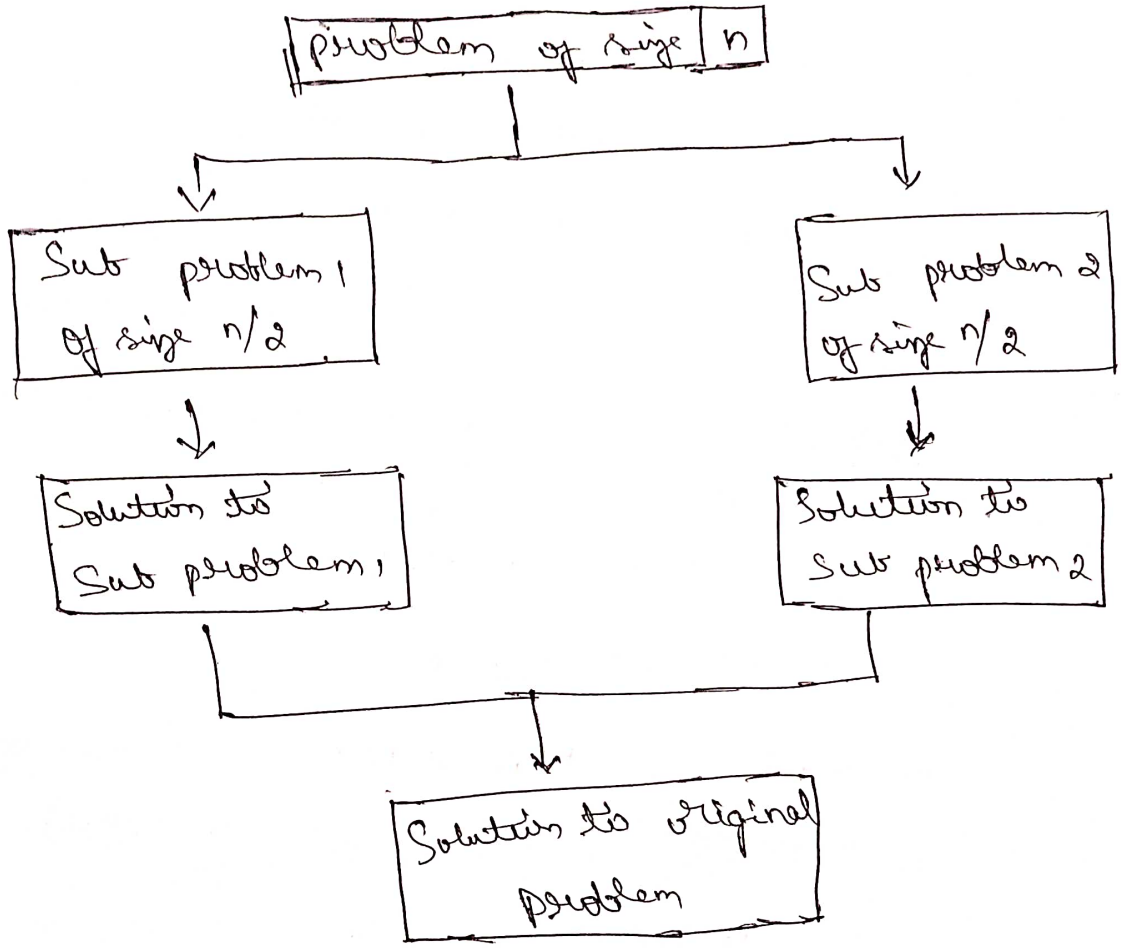
Apply Dc to each sub problem

Return combine (Dc(P1), Dc(P2) .... Dc(Pn))

}

}

Divide and conquer technique



(8) Write an algorithm for merge sort with an example. Derive the time efficiency (best, average, worst) case of the algorithm. Sort the list E, X, A, M, P, L, E in alphabetical order using merge sort algorithm. Draw the tree of recursive call.

Merge sort is a sorting algorithm that uses the divide and conquer strategy in this method division is dynamically carried out.

Algorithm:-

```
Mergesort (int, A[0....n-1], low, high)
    Sorted Array A[0.....n-1]
      if (low < high) then
    {
    mid ← (low+high)/2
    Merge Sort (A, low, mid)
    Merge Sort (A, mid+1, high)
    Combine (A, low, mid, high)
    }
```

=> Time efficiency class of algorithm by master theorem let the recurrance relation for merge sort is

$$T(n) = T(n/2) + T(n/2) + C_n$$

$$T(n) = 2T(n/2) + C_n$$

Here, $T(n) = aT(n/b) + f(n)$ is a recurrance relation by comparing $a=2$, $b=2$, $C_n = f(n)$

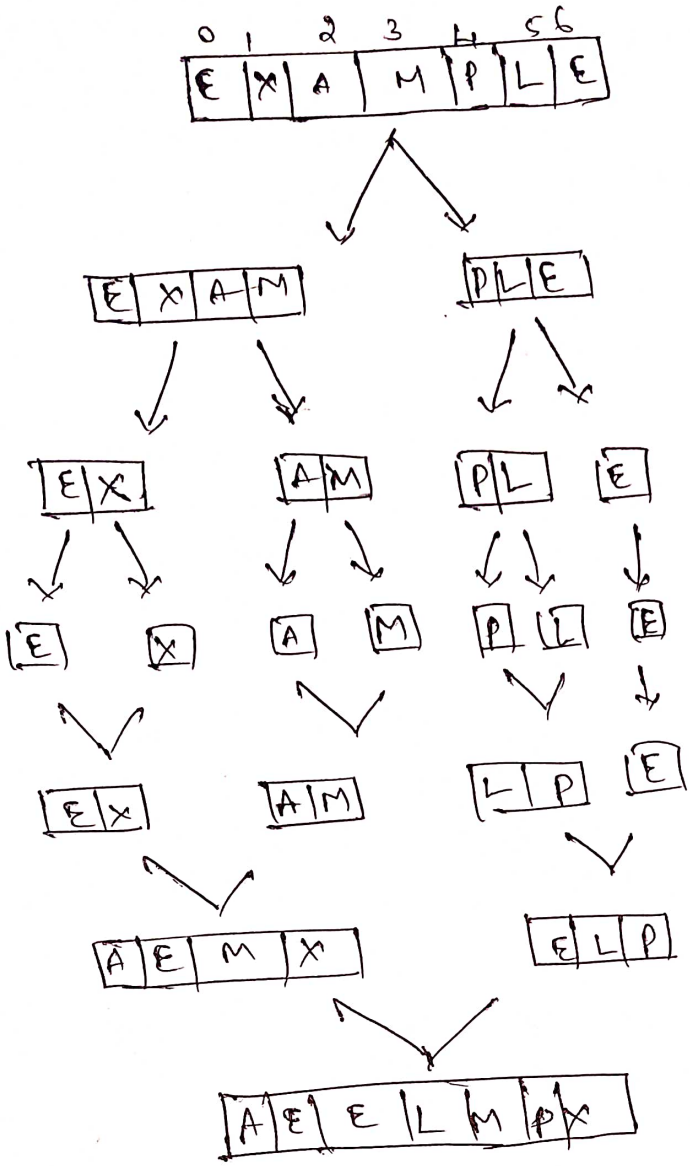
$$T(1) = 0$$

As per master theorem,

$T(n) = \theta(n^d \log n)$ if $a = b^d$,

As Eqn n^d where $d=1$

=> Merge sort : E, X, A, M, P, L, E



is sorted list.