Internals - 01

Design and analysis of algorithm

Shahul Hameed S
1KN18CS097
CSE A'sec

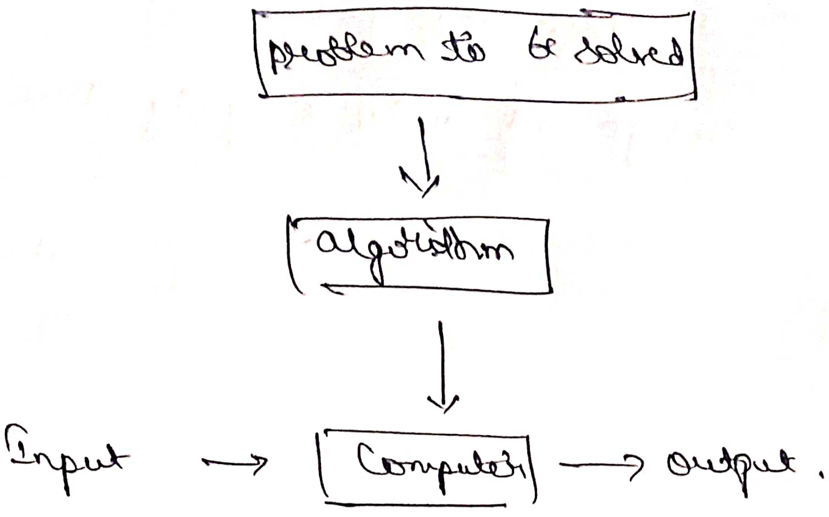
(1) a. An algorithm is a finite sequence of unambiguous instructions to solve a particular problem.

properties:-

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 instruction of an algorithm, then for all cases, the algorithm terminates after a finite number of steps.
5. Effectiveness:- Every instruction must be very basic, so that it can be carried out.

Notion of an algorithm:-

· ~~The non-ambiguity requirement for each step of an~~



Notion of an algorithm

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

If $n = 2$

$\frac{1}{2} n(n-1) = \frac{1}{2} * 2(2-1)$

$= 1$

$n^2 = 2^2 = 4$

If $n = 4$,

$\frac{1}{2} n(n-1) = \frac{1}{2} * 4(4-1) = 6$

$n^2 = 4^2 = 16$

If $n = 8$

$\frac{1}{2} n(n-1) = \frac{1}{2} * 8(8-1) = 28$

$n^2 = 8^2$

$n^2 = 64$

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

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

Θ notation says $c_2 g(n) \le t(n) \le c_1 g(n)$

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

if $n=2$

$f(n) = 2! = 2$

$g(n) = 2^n = 2^2 = 4$

if $n=3$

$f(n) = 3! = 6$

$g(n) \Rightarrow 2^n = 2^3 = 8$

if $n=1$

$f(n) \Rightarrow 1! = 1$

$g(n) = 2^1 = 2$

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

$n=5$

$f(n) \Rightarrow 5! = 120$

$g(n) = 2^5 = 32$

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

(4a) MergeSort (int A[0....n-1], low, high)

if (low < high) then

{

mid ← (low+high)/2

Merge Sort (A, low, mid)

MergeSort (A, mid + 1, high)

Combine (A, low, mid, high)

}

Algorithm combine (A[0....n-1], low, mid, high)

{

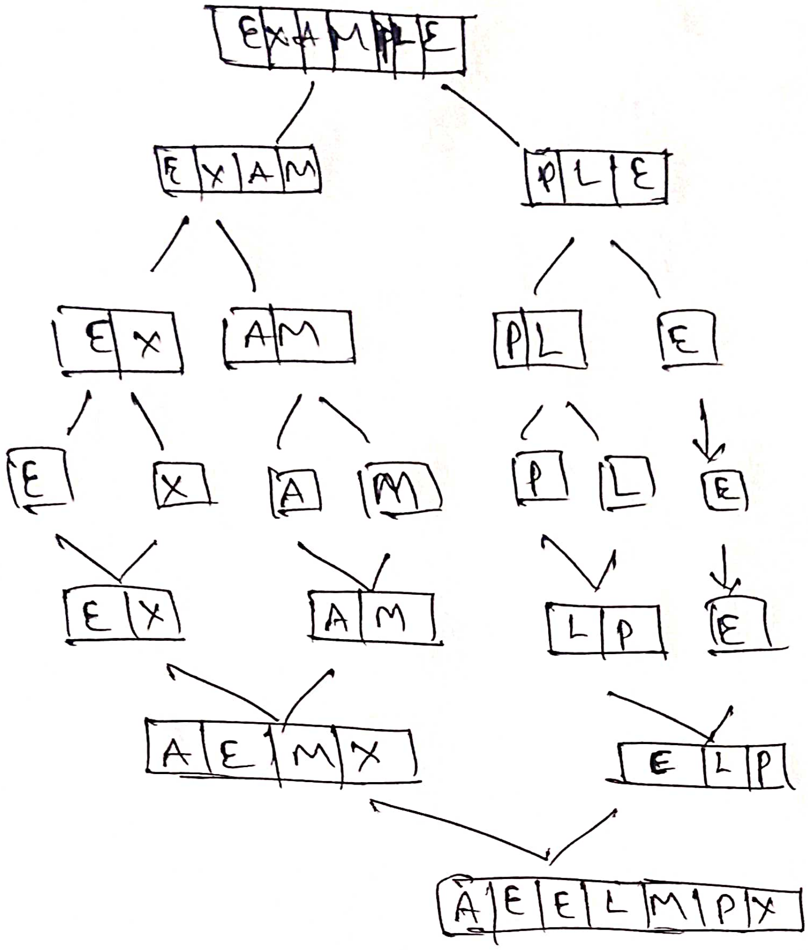
k ← low; i ← low; j ← mid+1;

while (i <= mid and j <= high)

{

```
if (A[i] <= A[j]) then
{
    // smaller element present in left sublist
    ~~if A[i] <= A[j] then~~
    {
        temp [k] <- A[i]
            i <- i+1
            k <- k+1
    }
  else      // smaller element present in right sublist
    {
        temp[k] <- A[j]
          j <- j+1
          k <- k+1
        }
  }
while ( i <= mid)
  {
   temp [k] <- A[i]
   i <- i+1
   k <- k+1
  }
```

2. b.



Sorted list //..