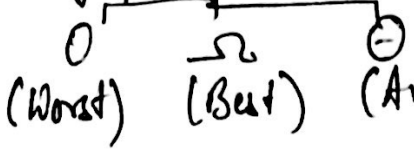
IAT-1 Design & Analysis of Algorithms.

Q.1 (a) Asymptotic Analysis.



Labelled Graph. Inequation — (10) (5)

(b) $t_1(n) \in O(g_1(n))$ — (1)

$\therefore t_1(n) \leq c_1 g_1(n) \quad \forall n \geq n_1$ — (5)

$t_2(n) \in O(g_2(n))$ — (2)

$\therefore t_2(n) \leq c_2 g_2(n) \quad \forall n \geq n_2.$

Let $c_3 = \max(c_1, c_2)$ and let $n \geq \max(n_1, n_2)$.

$$\therefore t_1(n) + t_2(n) \leq c_1 g_1(n) + c_2 g_2(n)$$
$$\leq c_3 g_1(n) + c_3 g_2(n)$$
$$\leq 2c_3 [\max(g_1(n), g_2(n)]$$

$$\therefore t_1(n) + t_2(n) \in O(\max[g_1(n), g_2(n)]).$$

Q.2 (a).

n	$\log_2 n$	$\sqrt[2]{n}$
10^1	3.3	3.16
10^2	6.6	10
10^3	10	31.62
10^4	13	100
10^5	17	316.22.

↓ faster than $\log \frac{n}{2}$. — (5)

b) Fibonacci .. // Iterative.

```
f[0] ← 0 ; f[1] ← 1
for i ← 2 to n do
    f[i] ← f[i-1] + f[i-2]
return f(n)
```

// Recursive. fib(n).

```
if ((n==0) or (n==1)) // base cond^n.
    return n.
else
    return fib(n-1) + fib(n-2)
```

(5)

Recurrence. $T(n) = T(n-1) + 1$.

$T(1) = 1$.

Solve. to get. $T(n) = n$. $O(n)$. Linear.

Q.3 (a) 5 specify algo — (5)
+ defin + Time + space complexity

(b) Recursive algo for Tower of Hanoi — (5)
+ Recurrence

$$T(n) = \begin{cases} 2 \cdot T(n-1) + 1 & \text{for } n > 1 \\ 1 & \text{for } n = 1 \end{cases}$$

Solve = $\theta(2^n)$. Exponential

Q.4 (a) Quick Sort - Algo — 5

$O(n^2)$

$\Omega(n \log n)$, $\Theta(n \log n)$

(b) 65, 70, 75, 80, 85, 60, 55, 50, 45 — 5

pivot

OR

Q.4 (a) Recursive algo for Max, Min. — 10

& partitions: MM(l, mid, max, min)

MM(mid+1, r, max1, min1)

Q.5 (a) — 5

D & C - control abstl.

```
MergeSort (low, high)
if (low < high) then
  { mid = ⌊(low+high)/2⌋;
  MergeSort (low, mid);
  MergeSort (mid+1, high);
  Merge (low, mid, high);
}
```

Call Algo to

Merge (low, mid, high)

Q.5(b). E, X, A, M, P, L, E.

E	X	A

M	A	L	E

M	P

L	E

M	P	L	E

(5)

Sorted list:

E	A

A E E L M P X

Q.6(a). Recursive Max Min -

Recurrence. $T(n) = 2T(n/2) + b$. (5)

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

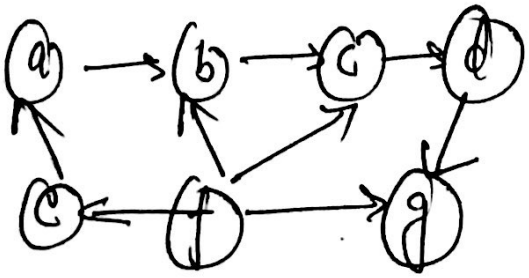
(b) (i) $T(n) = 2T(n/2) + n$.

Solve. $O(n \log n)$.

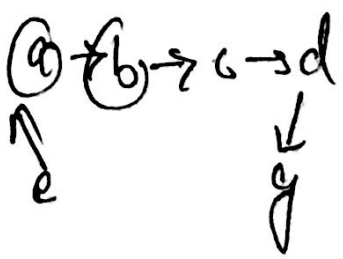
(ii) $T(n) = T(n/2) + 1$.

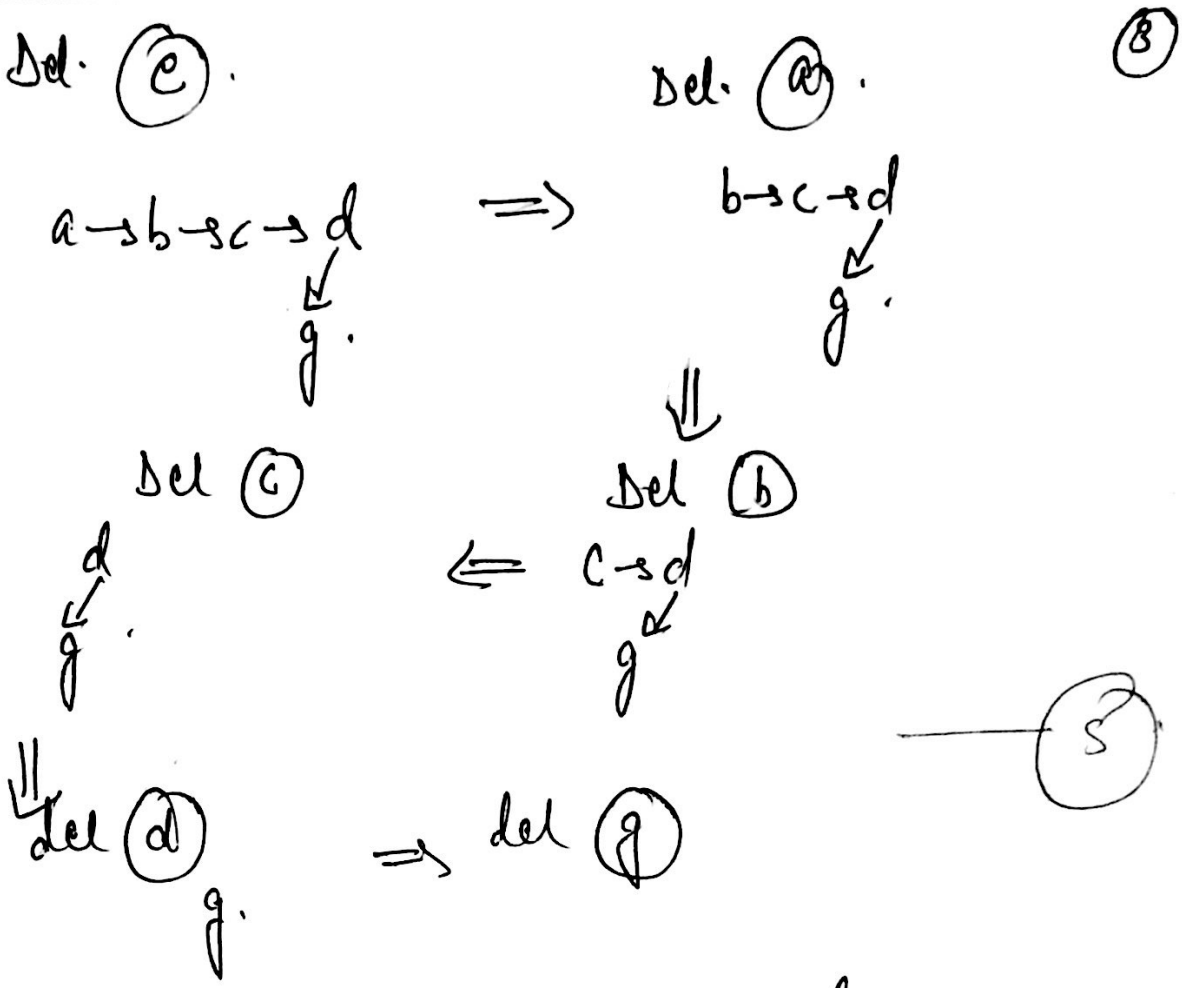
Solve. $O(\log n)$. (5)

Q.7 (a).



Del. (f).





Answer. (f), e, a, b, c, d, g.

Q.7(a). Matrix Multip. DXC

$T(n) = 8T(n/2) + n^2$.

$T(1) = 0$

.

.

Solve $O(n^3)$

Strassen's. — 5

$$T(n) = \begin{cases} 7 \cdot T(n/2) + 18n^2 ; & \text{if } n > 2 \\ 1 ; & \text{if } n \leq 2. \end{cases}$$

.

.

Solve to

$$7^{\log_2 n} + 18n^2 \left[\sum_{i=0}^{k-1} 7^k / 2^{2k} \right]$$