Divanshika Gupta
DS1 - 45

23/2/24

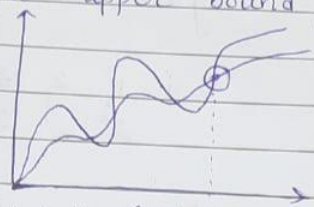
Design and Analysis of Algorithm
Assignment

Ans 1 Asymptotic notations are used to tell the complexity of an algorithm when input is very large.

Types of asymtotic notation :

→ Big O (O) : $f(n) = O(g(n))$

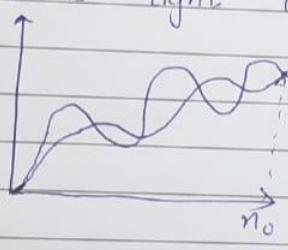
g(n) is 'tight' upper bound of func.



$f(n) = O(g(n))$ iff $f(n) < cg(n)$ $\forall$ $n > n_0$ & some constant $c > 0$

→ Big Omega Notatatio (Ω) : $f(n) = \Omega(g(n))$

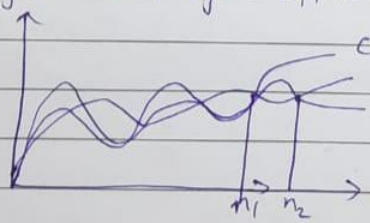
g(n) is 'tight' lower bound of f(n)



$f(n) \geq cg(n)$ $\forall$ $n \geq n_0$ & some constants $c > 0$.

→ Theta Notation (Θ) : $f(n) = \Theta(g(n))$

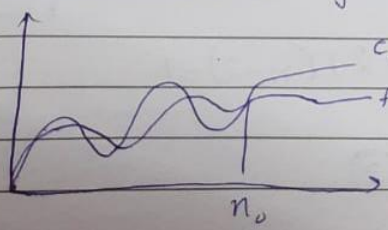
It gives both 'tight' upper & tight lower bounds.



$\forall$ $n > \max(n_1, n_2)$ & some constant $c_1, c_2 > 0$

n_1 & n_2 can be same also.

→ Small Theta (θ) : g(n) upper bound of f(n), $f(n) = \theta(g(n))$



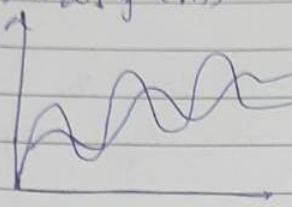
iff $f(n) < cg(n)$

$n > n_0$ & const. $c > 0$

ARCHITECTURE

→ Small Omega (ω) : $g(n)$ is lower bound of $f(n)$

$f(n) = \omega(g(n))$



iff $f(n) > c\,g(n)$

$\forall\ n > n_0$ & $\forall$ constant $c > 0$

Ans (2) Time complexity is $O(\log n)$

TC :

Ans (3) pos ($i = 1$ to n)

$i = i * 2$

i =	1	2	4	8	...	n
	2^0	2^1	2^2	2^3	...	2^k

GP $\Rightarrow ar^{k-1}$

$n = 1.2^{k-1}$ $\qquad n = \frac{2^k}{2}$

$\log 2n = k \log 2$

$\log 2 + \log n = k \log 2$

$\boxed{T(n) = O(\log(n))}$

Ans (5)

Ans (3) $T(n) = 3T(n-1)$, $n > 0$ otherwise 1

$T(0) = 1$

$T(1) = 3T(0) = 3$

$T(2) = 3T(1) = 9 = 3^2$

$T(3) = 3T(2) = 27 = 3^3$

$\boxed{T(n) = 3^n = O(3^n)}$

Ans (4) $T(n) = 2T(n-1) - 1$ — (1) , $n > 0$ otherwise 1

let $n = n-1$

$T(n-1) = 2T(n-1-1) - 1 = 2T(n-2) - 1$

put $T(n-1)$ in (1)

$T(n) = 4T(n-2) - 3$ — (2)

tant c>0

put $n = n-2$

$T(n-2) = 2T(n-2-1)-1 = 2T(n-3)-1$

put in (2)

$T(n) = 4(2T(n-3)-1)-1 = 8T(n-3)-4-1$

$= 8T(n-3)-5 = 2^k T(n-k)$

$\therefore (n-k)=1$

$\therefore k = n-1$

$T(n) = 2^{n-1} T(n-n+1) - 5 = 2^{n-1} T(1) - 5$

$= \frac{2^n}{2} = 2^n = O(2^n)$

$T(n) = \Theta(2^n)$

Ans (5)

```
while (s<=n)
{ i++;
  s=s+i;
  printf("#");
}
```

i = 1 ⇒ i++ , i = 2

s = 3 , i = 3

s = 6 , i = 4

s = 10 , i = 5

s = 15 , i

i = 2 3 4 5

s = s+1+2 s+1+2+3 s+1+2+3+4 s+1+2+3+4+5

$S = 1 + 1 + 2 + 3 + 4 \ldots k$

$S(R) = R(R+1)/2 \leq n$

$= k^2 + \frac{k}{2} \leq n$

$k^2 \leq n \Rightarrow k \leq \sqrt{n}$

$T(n) = O(\sqrt{n})$

Ans (6) void function (int n)
{ int i, count = 0; — (1)
for (i=1; i*i <= n; i++)
{ count++
}
i = 1, 2, 3, 4 ... $\sqrt{n}$
i = 1, 4, 9, 16 ... n^2
Complexity : $O(\sqrt{n})$

Ans (7) void function (int n)
{ int i, j, n, count = 0;
for (i=n/2; i<=n; i++) — n/2
for (j=1; j<=n; j=j+2) — $\log_2(n)$
for (k=1; k<=n; k=k*2) — $\log_2(n)$
count++;
}
~~complexity~~ = $O(\frac{n}{2} \times \log_2(n) \times \log_2(n))$

complexity = $O(n\log^2(n))$

Ans (8) function (int n) — T(n)
{ if (n == 1) return;
for (i=1 to n) { — n
for (j=1 to n) { — n^2
print ("*"); — n^2
}
}
function (n-3) — T(n-3)
}

complexity : $T(n) = O(n^2) + T(n-3)$

Date.
Page No.

Ans (9) void function (int n)
{ for (i = 1 to n) {
    for (j = 1; j <= n; j = j+1)
        print ("*");
}
}

i = 1, 2, 3, 4 ....
j = 1, 3, 6, 10 ....

i = 1 , n times
i = 2 , n/2 times
i = 3 , n/3 times

$1 + \frac{n}{2} + \frac{n}{3} + \cdots + 1$

complexity = $O(n \log n)$

Ans 10. n^n & c^n

$n = 1$

$n^k = 1^k$, $c^n = c$

$n^k = 2^k$, $c^n = c^2$

$n = k$, $n^k = k^k$, $c^n = c^k$

∴ we can say that
for any value of $n > 0$

$n^k \geq c^n$

but $n^k = f(n)$, $c^n = C_0 g(n)$

$\because f(n) \geq C_0 g(n)$ $C_0 > 0$, $n_0 \geq n_0$

$\therefore f(n) = O(C_0 g(n))$

$n^k = O(c^n)$

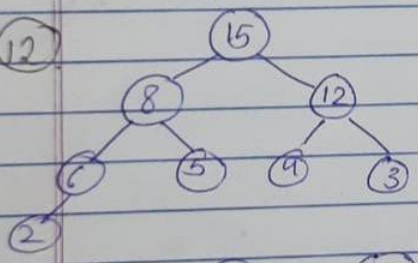
Date
Page No.

Ans (11)
```
int extractMin( Vector<int> & heap)
{ if (heap.empty())
  { return -1;        -> O(1)
  }
  swap (heap[0], heap.back()); -> O(1)
  int minElement = heap.back();
  heap.popback();              - O(1)
  heapify (heap, 0);           - O(log n)
  return min element;
}
```

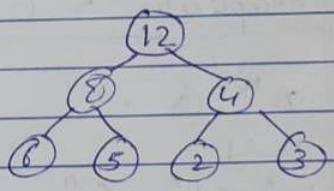
No. of comp $= \left[1 * \frac{n}{4}\right] + 2 * \frac{n}{8} + 3 * \frac{n}{16} + (n-1) * 1$

$= \boxed{O(\log(n))}$

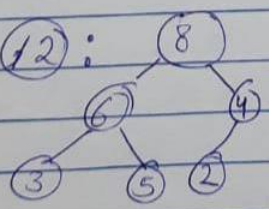
Ans (12)



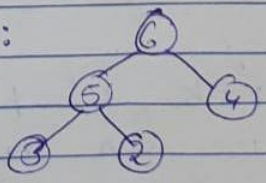
delete (15) :



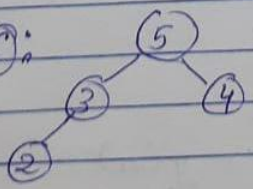
delete (12) :



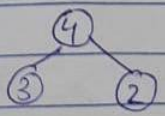
delete (8) :



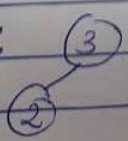
delete (6) :



delete (5) :



delete (4) :



delete (3) : (2)

delete (2) : heap is empty.