Page No.
Date

Que 4. $T(n) = 2T(n-1) - 1$ — ①

Let $n = (n-1)$

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

~~$T(n) = 4T(n-2) - 2$ — ②~~

~~$T(n-2) = 4T(n-4) - 2$~~
~~put value in eq ②~~

$T(n) = 8$

~~$T(n) = 8T($~~
$T(n) = 4T(n-2) - 2$ — ②

$n = n-2$

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

$T(n) = 8T(n-3) - 3$

$T(n) = 2^k \cdot T(n-k) - k$

Let $n-k=1$
$k = n-1$

$T(n) = 2^{n-1} T(\cancel{n} - \cancel{n}-1) - n-1$
$= 2^{n-1} - n - 1.$
$= 2^{n-1}$
$= \frac{2^n}{2} - n - 1 = O(2^n)$

Page No.

Date

4

Qu 5. $i = 1 \Rightarrow i{+}{+},\ i=2$

$s=3,\ i=3$

$s=6,\ i=4$

$s=10,\ i=5$

$s=15$

$i = 2 \quad 3 \quad 4 \quad 5$

$s = s+1+2,\ s+1+2+3,\ s+1+2+3+4,\ s+1+2+3+4+5$

$s = s+1+2+3+4+5 \ldots\ldots k.$

$S(k) = k(k+1)/2 \le n$

$= (k^2+k)/2 \le n$

$k^2 \le n$

$k \le \sqrt{n}$

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

Qu 6 $i = 1 \quad 2 \quad 3 \quad 4 \ldots\ldots k$

$i^2 = 1 \quad 4 \quad 9 \quad 16 \ldots\ldots k^2$

$k^2 \le n$

$k \le \sqrt{n}$

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

Qu 8. $T(n) = n * n * [T(n-3)]$

$T(n) = n^2 + T(n-3)$

$\boxed{T(n) = O(n^2)}$

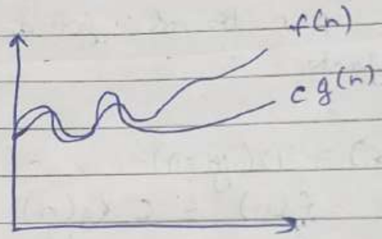
(iii) Big ~~Theta~~ Omega

Gives lower bound for a f(n) within constant factor

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

if $f(n) \geq c\,g(n)$

for $c > 0$, an $n \geq n_0$



Que 2

$i = 1 \quad 2 \quad 4 \quad 8 \ldots n$

$= 2^0 \quad 2^1 \quad 2^2 \quad 2^3 \ldots 2^k$

$G.P = 1 \cdot 2^{k-1}$

$n \Rightarrow \frac{2^k}{2}$

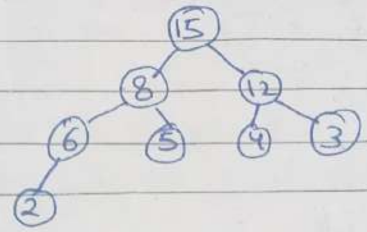
$2n = 2^k$

$\log 2n = k \log 2$

$\log n = k$

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

and show heap after every step.



Assignment

Que 3. $T(n) = 3T(n-1)$ if $n>0$

When $n=1$,

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

$= 3$

When $n=2$

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

$= 3\times 3 = 9$

When $n=3$

$T(3) = 3T(2)$

$= 9\times 3 = 27.$

$T(n) = 3, 9, 27, 81 \ldots\ldots 3^n.$

$T(n) = O(3^n)$

Assignment.

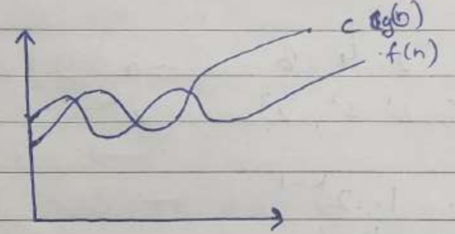
Section - DS1
Name - Divyansh Dewgan
Roll No. - 47.

Que 1. Asymptotic notations are used to represent the complexities of algorithms for asymptotic analysis.

(i) Big Oh notation:-

Gives an upper bound for a $f(n)$ within a constant factor.

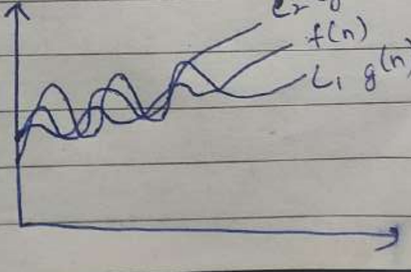
$f(n) = O(g(n))$
if $f(n) \leq c\,g(n)$
for $c > 0$ & $n \leq n_0$



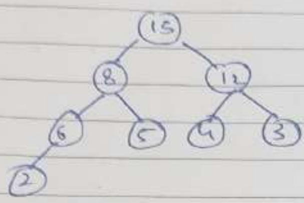
(ii) Big ~~Omega~~ Theta:-

Give ~~lower~~ bound for a $f(n)$ within constant factor

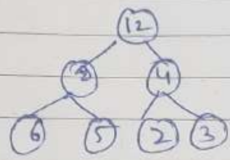
$f(n) = \theta(g(n))$
if $c_1 g(n) \leq f(n) \leq c_2 g(n)$
for $c_1, c_2 > 0$ & $n \geq n_0$



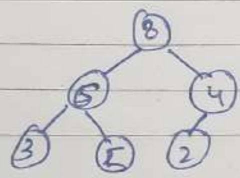
Qu 11.



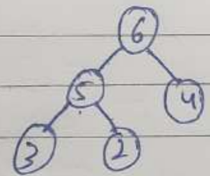
root 15 erased



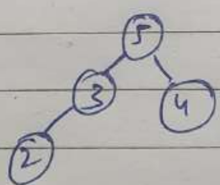
root 12 erased



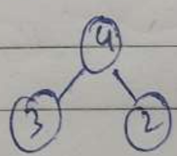
root 8 erased



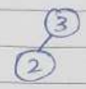
root 6 erased



root 5 erased



root 4 erased



root 3 erased

root 2 erased

heap is empty.

Page No.
Date

Qu 9. $i=1$, $j = 1\ 2\ 3\ 4 \ldots\ldots n/1$
$i=2$, $j = 1, 3, 5, 7 \ldots\ldots n/2$
$i=3$, $j = 1, 4, 7, 11 \ldots\ldots n/3$
$i=n-1$, $j = 1, n$ $\qquad n/n = 1$

$$n + \frac{n}{2} + \frac{n}{3} \cdots \frac{n}{n} = \log(n)$$

Harmonic series.

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

Qu 11.

```
int extractMin (vector<int> & arr)
{
    if (arr.empty())
        return -1;                      → O(1)
    swap (arr[0], arr.back());          — O(1)
    int min = arr.back();
    arr.popback();                      → O(1)
    heapify (arr, 0);                   → O(log n)
    return min;
}
```

$T(n) = O(\log(n))$