DESIGN and ANALYSIS of ALGORITHMS

Page No.
Date : / /

ASSIGNMENT 1

Submitted by :- ARJUN AHALAWAT
Section :- DS1 Class Roll no. :- 23

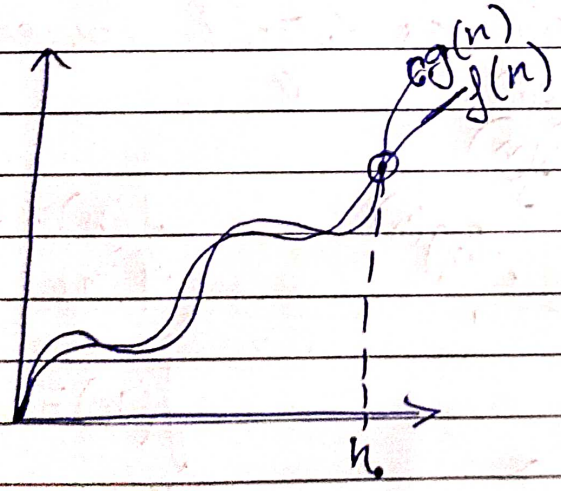
Q1) What do you understand by Asymptotic Notations? Define different notations with examples.

Ans 1) Asymptotic notations is a mathematical framework that analyses the performance of algorithms and functions as their input sizes approaches infinity.

- Big Oh Notations (O)

$$f(n) = O(g(n))$$

$g(n)$ is the "tight" upper bound of $f(n)$

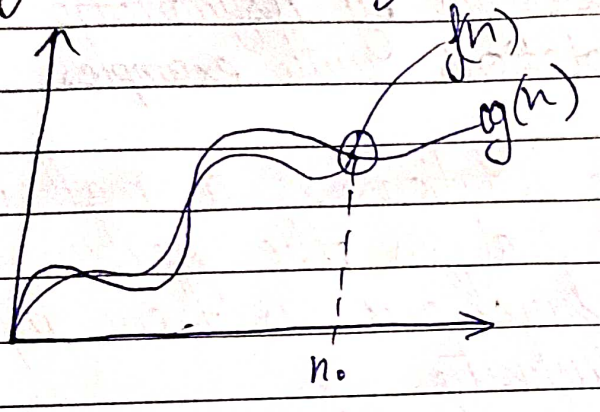


if and only if;
$f(n) \leq cg(n)$
$\forall\ n \geq n_0$, and
c is some constant.

- Big Omega Notations (Ω)

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

$g(n)$ is the "tight" lower bound of $f(n)$



iff, $f(n) \geq cg(n)$

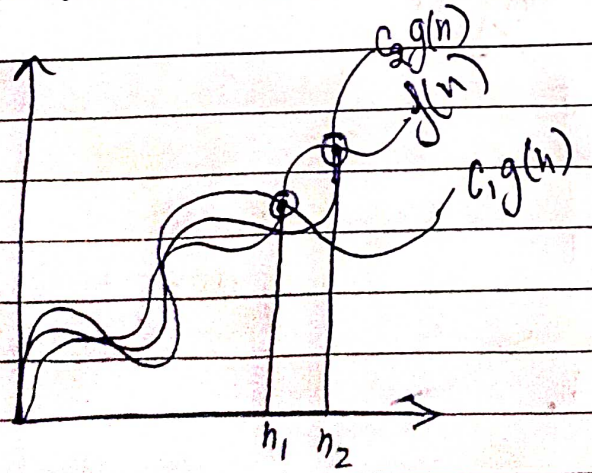
$\forall\ n \geq n_0$ and

c is some constant

- Theeta Notations (Θ)

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

Theeta here gives both tight ~~as well as~~ upper bound as well as tight lower bound.



$f(n) = Og(n)$ and

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

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

iff, $c_1 g(n) \leq f(n) \leq c_2 g(n)$

$\forall\ n > \max(n_1, n_2)$ and

some constants $c_1, c_2 > 0$

Page No.
Date : / /

- Small Oh notations (o)

$f(n) = o(g(n))$

$g(n)$ is the upper bound of $f(n)$

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

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

- Small omega notations (ω)

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

$g(n)$ is the lower bound of $f(n)$

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

$\forall \; n > n_0$ & $\forall \; c > 0$.

Q2) What should be the time complexity of —

```
for (i=1 to n) { i = i*2; }
```

Ans 2)

```
for (i=1 to n)  ——————→  1 to n → G.P
{
    i = i*2;  ——————  O(1)
}
```

G.P :- $n = ar^{n-1}$

$n = \frac{2^n}{2}$ → $\log_2 n$

$\Rightarrow O(\log n)$

Page No.
Date : / /

Ans 3) $T(n) = \{3T(n-1)$ if $n>0$, otherwise $1\}$

$T(1) = 3^1$

$T(2) = 3T(1)$ $\quad$ $T(2) = 3 \times 3$

$T(3) = 3T(2)$ $\quad$ $T(3) = 3 \times 3 \times 3$

$T(4) = 3(T(3))$ $\quad$ $T(4) = 3 \times 3 \times 3 \times 3$

$T(n) = 3^n$

Ans 4) $T(n) = \{2T(n-1) - 1\}$

$T(1) = 1$

$T(2) = \{2T(1) - 2\}$

$= 2 \times 2 - 1$

$= 3$

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

$= 1$

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

$= 1$

$T(n) = 1$

Page No.
Date : / /

Ans 5)
```
int i=1, s=1;
while (s<=n)        ---> n/2
{
    i++;            ---> 1
    s=s+i;          ---> 1
    print ("#");    ---> 1
}
```

=>> $n/2 + 1 + 1 + 1 = n/2 + 3$ (ignoring the constants).

=>> $O(n)$

Ans 6)
```
void function (int n)
{
    int i, count = 0;
    for (i=1; i*1<=n; i++)
        count++;
}
```

Time Complexity = $O(\sqrt{n})$

Ans 7)
```
void function (int n)
{
    int i, j, k, count=0;
    for (i=n/2; i<=n; i++)          ---> n/2
        for (j=1; j<=n; j=j*2)      ---> log2 n
            for (k=1; k<=n; k=k*2)  ---> log2 n
                count++;            ---> 1
}
```

=>> ~~$n/2 * (\log_2 n) * (\log_2 n)$~~

= $n/2 (\log_2 n (\log_2 n))$

Page No.
Date : / /

Time Complexity = $O(n/2 (\log n)^2)$

Ans 8)

```
function (int n)                          ---- T(n)
{
    if (n == 1)
        return;
    for (i=1 to n)
    {
        for (j=1 to n)
        {
            printf ("*");
        }
    }
    function (n-3);                       ---- T(n-3)
}
```

$O(n^2)$

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

$T(1) = 1$

putting $n = n-3$

$T(n-3) = (n-3)^2 + T(n-6)$ ---- (2)

$\Rightarrow T(n) = (n-3)^2 + T(n-6) + n^2$ ---- (3)

putting $n = n-6$

$T(n-6) = T(n-9) + (n-6)^2$ ---- (4)

$\Rightarrow T(n) = T(n-9) + T(n-6) + (n-6)^2 + (n-3)^2$

Page No.
Date : / /

Ans 9)
```
void function (int n)
{
    for (i=1 to n)          ----> n times
    {
        for (j=1 to ; j<=n; j=j+i) ---> log n times
            printf ("*")
    }
}
```

$T(n) = n/1 + n/2 + n/3 + \ldots\ldots + n/n$

$= O(n \log n)$

Ans 10) n^k and c^n ; where $k \geq 1$ and $c > 1$

let $k=1$ and $c=2$

n^1 and 2^n

here, c^n grows faster than n^k i.e. growth rate of c^n is higher.

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

c^n is the upper bound of n^k.

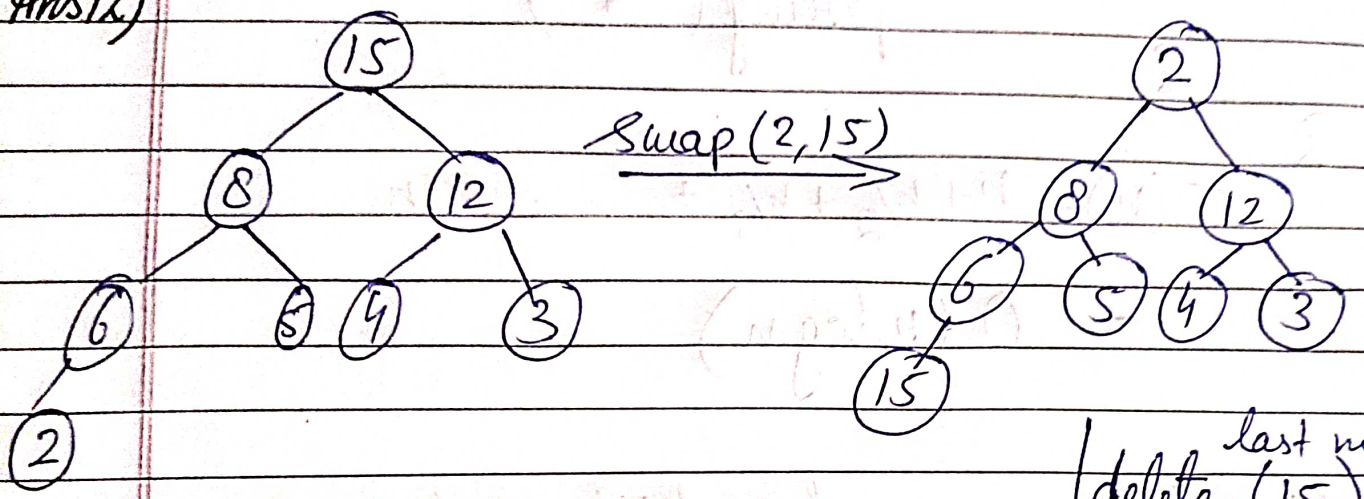
$(n^k \leq c^n)$

Page No.
Date : / /

Ans 11) Extract min time complexity (when it returns only 1 minimum element from heap)

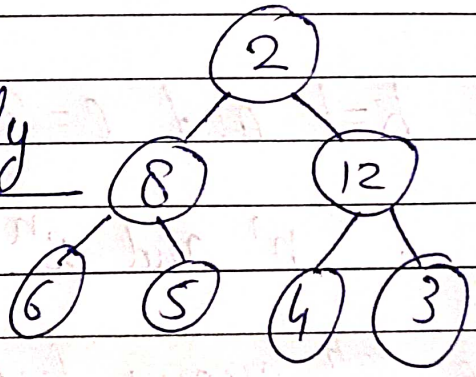
Time Complexity = O(1) (as only root node will be deleted).

Ans 12)

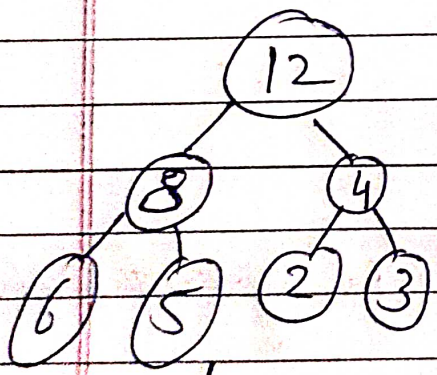


Swap (2, 15)

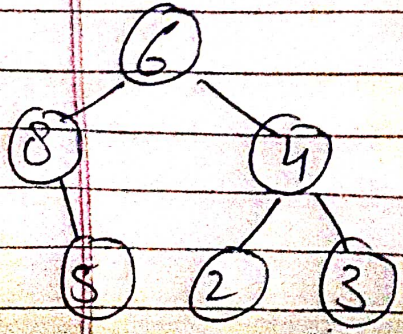
delete (15) last node.



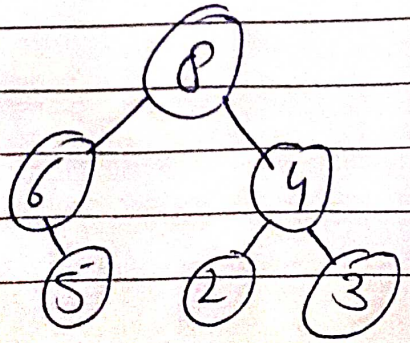
heapify



swap (12, 6) and deleting last node

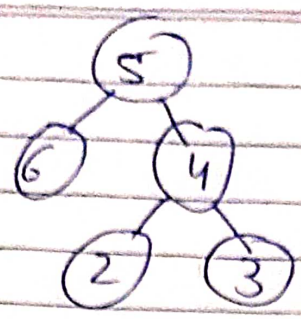


heapify

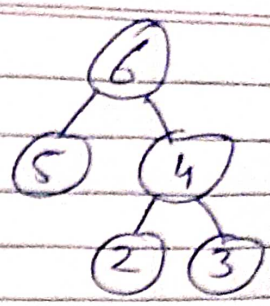


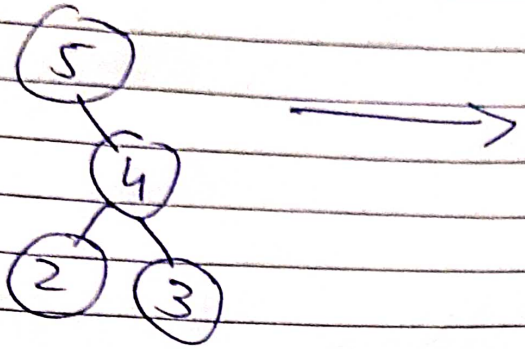
Page No.
Date : / /

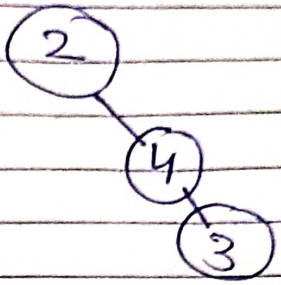
Swap(6,5) and delete(6)



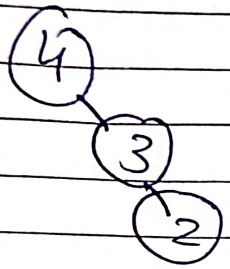
heapify →

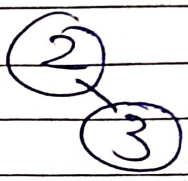




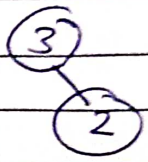


heapify →





heapify →



→ (2) Delete last node(2)

→ All nodes gets deleted.

X X

Submitted by :- ARJUN AHALAWAT

Section :- DS1

Class Roll no:- 23