Name → Karan Singh Bisht
Sec → A
Roll.No → 14

TUTORIAL - 2

1) When while loop executes -

At first pass $i = 1$
2^{nd} pass $i = 1 + 2$
3^{rd} pass $i = 1 + 2 + 3$
Similarly 4^{th} pass $i = 1 + 2 + 3 + 4$

for 1 ------ n^{th} $i = 1 + 2 + 3 ---- + n$

for i^{th} time $i = (1 + 2 + 3 + 4 + --- i) < n$

$$= \frac{i(i+1)}{2} < n$$

$$= \frac{i^2 + i}{2} = \left(\frac{i^2}{2} + \frac{i}{2}\right) < n$$

ignoring $\frac{i}{2}$ & $\frac{1}{2}$.

After neglecting we left with

$$= i^2 < n$$

$$= i < \sqrt{n}$$

Hence, the time Complexity is $O(\sqrt{n})$

```
int rectib (int n)
{
    if (n <= 1)
    return n;
    else
    return rectib (n-1) + rectib (n-2);
```

Time Complexity :-

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

when $n = 0$ & $n = 1$

i.e, $T(0) = T(1) = 0$

for ($T(n) = ?$)

Here $T(n-2) \approx T(n-1)$

On Substituting the Value of $T(n-1) - T(n-2)$ into $T(n)$,

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

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

On Substituting

$T(n) = 2 \times [2 \times T[n-2] + 1] + 1$

$T(n) = 4T(n-2) + 3$

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

$T(n) = 2 * [2 * [2T(n-3) + 1] + 1] + 1$

$T(n) = 8 \times T(n-3) + 7$

!

!

$T(n) = 16 * T(n-4) + 15$

Similarly for k^{th} term ___

$$T(n) = 2^k \cdot T(n-k) + (2^k - 1)$$

$$n - k = 0$$

$$n = k$$

Hence, $T(n) = 2^n + T(0) + (2^n - 1)$

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

So, time Complexity is $O(2n)$

<u>Space Complexity</u> :-

Here n are the no. of entries in a stack & for each function Call one.

So space Complexity for each Case (Call) is 1, i.e $O(1)$

& for 'n' no. of Cases. $< = n$

i.e <u>$O(n)$</u>

3). for (int i = 0 ; i < n ; i++)

```
{
    for (int j=0 ; j<n; j = j*2
    {
        O(1) // statements
    }
}
```

$O(n(\log n))$

```
for(int i=0; i<n; i++)
{
    for(int j=0; j<n; j++)
    {
        for(int k=0; k<n; k++)
        {
    O(1)                    //statements
        }
    }
}
```

$O(n^3)$

```
for(int i=0; i<n; i=i/2)
{
    for(int j=0; i<n; j=j*2)
    {
            O(1)
        }                   // statements
        O(1)
    }
}
```

$O(\log(\log n))$

4). $T(n) = T(n/4) + T(n/2) + n^2$

On removing $T(n/4)$ as smaller term

$T(n) = T(\frac{n}{2}) + cn^2$

On applying Master Theorem on R.H.S.

$a = 0$, $b = 2$ $K = 2$, $P = 0$.

$\log_b a = \log_2 0 = 0$

$0 < 2$ i.e $\log_b a < K$

& $P \geq 0$

$O(n^K \log^P n)$

so, $O(n^2 \log^0 n)$

$\underline{O(n^2)}$

5). time Complexity of the function fun() is $O(n \log n)$

for $i=1$, inner loop Executed n times.
for $i=2$, inner loop Executed $n/2$ times.
for $i=3$, inner loop Executed $n/3$ times.
for $i=n$, inner loop Executed n/n times.

So, Complexity is as

$$\left(\frac{n}{1} + \frac{n}{2} + \frac{n}{3} + \cdots \frac{n}{n} \right)$$

$$n \left(1 + \frac{1}{2} + \frac{1}{3} + \cdots \frac{1}{n} \right)$$

from h.p $\rightarrow \underbrace{\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots \frac{1}{n} \right)}$

particular term time Complexity is $O(\log n)$

so for total n loops.
time Complexity is $\underline{O(n \log n)}$

6). for (int i = 2 ; i < n ; i = pow (i, k))
{
// O(1)

}

K is Constant.

"i" takes the Value like $2, 2^k, 2^{k^2}, 2^{k^3} \cdots 2^{k^{\log_k(\log n)}}$

last term must be less than or equal to n.

$\underline{O(\log_k(\log(n)))}$

8 a). $100 < \log n < \log(n+1) < \log(\log n) < n < n! < n\log n <$
$\log^2 n < 2^n < 4^n < 2^{(2n)} < n^2$

b) $1 < \sqrt{\log(n)} < \log(n) < \log(n!) < \log(\log n) < \log(2n) <$
$2\log(n) < \log(n!) < n\log(n) < n < 2n < 4n < n! < 2^{(2^n)}$

c) $96 < \log_8(n) < \log_2(n) < \log(n!) < n! < n\log_6(n) < n\log_2(n)$
$< 5n < 8n^2 < 8^{(2n)} < 7n^3$