Q.1 what is the time complexity of below code & why

Sol^n

```
void fnc (int n)
{
    int j=1, i=0;
    while (i<n)
    {
        i+ = j;
        j++;
    }
}
```

$j=1 \qquad i=1$

$j=2 \qquad i=1+2$

$j=3 \qquad i=1+2+3$

$i = 1+2+3+ \cdots m < n$

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

$$\frac{m^2+m}{2} < n \Rightarrow m^2 < \sqrt{n}$$
$$\Rightarrow m = \sqrt{n}$$

By summation method

$$\Rightarrow \sum_{i=1}^{m} 1 \Rightarrow 1+1+1+ \cdots m = 1+1+1 \cdots \sqrt{n}$$
$$= \sqrt{n}$$

$$\underline{T(n) = \sqrt{n}} \text{ Ans}$$

order of ...

Q.2 Write recurrence relation for the recursive func that prints Fibonacci Series. Solve the recurrence relation to get time complexity of the program. What will be the space complexity of this program & why?

Sol^n. Fibonacci Series - it is the sum of previous two terms.

0, 1, 1, 2, 3, 5, 8

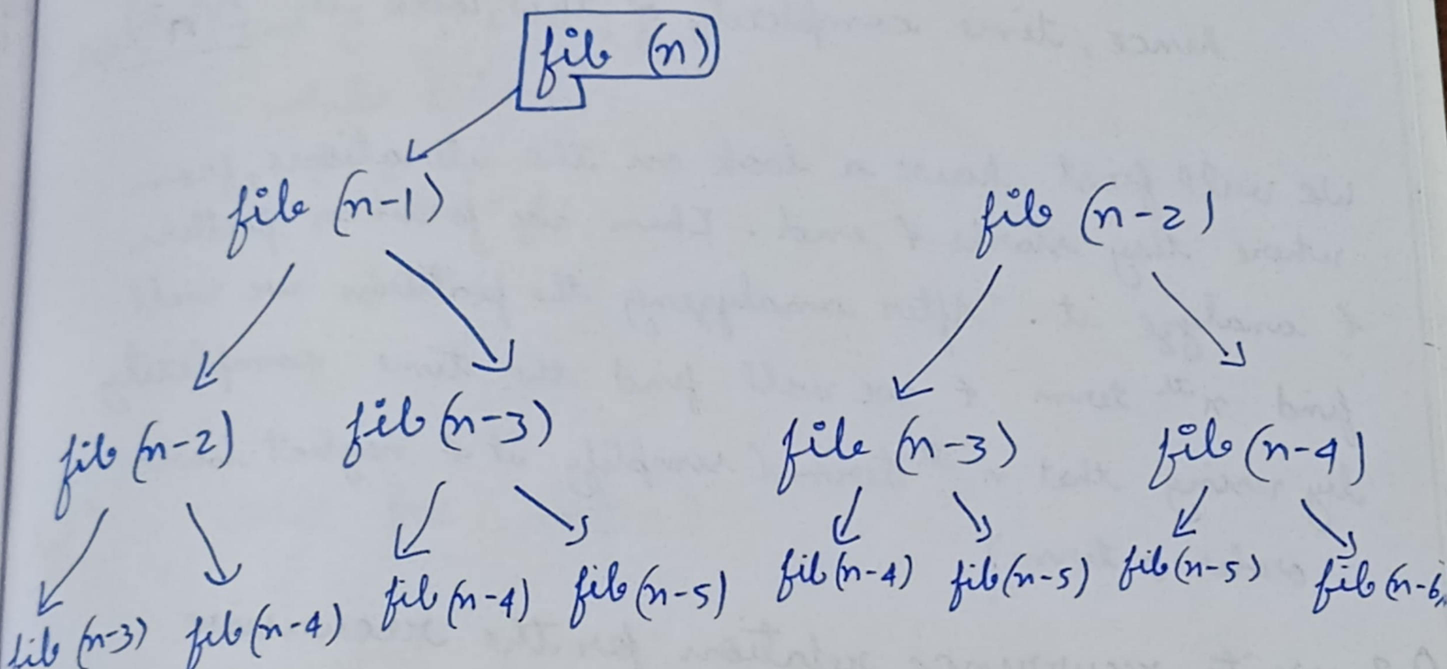
```
int fib (int n)
{
    if n <= 1
        return n;
```

else

return fib(n-1) + fib(n-2);

}

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



$1 + 2 + 4 + 8 + \cdots n$

$a = 1$

$r = 2$

<u>Space complexity</u> - depends upon max depth of the tree so, $\boxed{O(n)}$.

$$S_n = a\frac{(r^n - 1)}{(r-1)}$$

$$S_n = 1\frac{(2^{n+1} - 1)}{2-1} \Rightarrow \frac{2^{n+1}-1}{1} = 2 \cdot 2^n \Rightarrow O(2^n)$$

Q.3 Write programs which have complexity

(i) $n(\log n)$

(ii) n^3

(iii) $\log(\log n)$

sol^n

(i) $n(\log(n))$

```
for(i=1; i<=n; i*=2)
  {
     for(j=1; j<=n; j++)
       {
          sum = sum + j;
       }
  }
```

(i) $i = 1, 2, 4, 8 \; ---- \; n \rightarrow$ G.P ($a=1$, $r=2$)

(ii) $j = 1, 2, 3, 4, 5 \; ---- \; n \rightarrow$

(i) GP $\rightarrow$ $n = 1 * 2^{K-1}$

$$n = 2^{K-1}$$

$$\log n = (K-1).\log_2(2)$$

$$\log n = K-1.(1)$$

$$K = \log n + 1$$

$K = \lg(n + \cancel{1})$ neglect lower order term

$$K = \log_2 n$$

$O(\log_2 n)$ —— (a) eq

~~Ans~~

(ii) $O(n + \cancel{n} + \cancel{n} + \cancel{1})$

$O(n)$ ——— (b) eq

multiplying (a) with (b)

total complexity = $O(n \cdot \log(n))$ Ans

(ii) n^3

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

i	j	k
1	$\left(\frac{n+1}{2}\right)$	$\frac{(n+1)}{2}$
2		
3	$\left(\frac{n+1}{2}\right)$	$\frac{(n+1)}{2}$
4		
5	$\left(\frac{n+1}{2}\right)$	$\left(\frac{n+1}{2}\right)$
6	$\left(\frac{n+1}{2}\right)$	$\frac{(n+1)}{2}$
⋮	⋮	⋮

$$n * \frac{(n+1)}{2} * \frac{(n+1)}{2}$$

$$\frac{n^2+n}{2} \times \frac{(n+1)}{2}$$

$$\frac{n^3+n^2+n^2+n}{4}$$

now we will neglect lower order term

$$\frac{n^3 + \cancel{n^2} + \cancel{n^2} + \cancel{n}}{\cancel{4}}$$

$O(n^3)$

(iii) $\log(\log n)$

```
for (i=2; i<n; i=i*i)
  {
    count ++;
  }
```

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

Solve the recurrence relation

sol: $T(n) = T(n/4) + T(n/2) + cn^2$

we will neglect lower order term $T(n/4)$

$T(n) = T(n/2) + cn^2$

by master's method

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ $\quad a = 1$

$b = 2$

$\Rightarrow c = \log_b a$

$\Rightarrow c = \log_2 1 \quad \Rightarrow 0$

$\Rightarrow n^c = 1$

by comparing $n^c < f(n)$ we get

$n^c < f(n)$

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

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

Q.5 what is the time complexity of following fnc();

Sol^n

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

i	j
1	1
2	1 + 3 + 5
3	1 + 4 + 7
⋮	⋮
n	

$$\sum_{i=1}^{n} \frac{n-1}{i}$$

$$T(n) = \frac{n-1}{1} + \frac{n-1}{2} + \cdots\cdots + \frac{n-1}{n}$$

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

$\Rightarrow n \log n - \log n$

$\Rightarrow O(n \log n)$ Ans

Q.6 what should be the time complexity of

sol^n

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

where k is constant

$$T.C = 2,\ 2^{K},\ 2^{K^2},\ 2^{K^4} \ldots\ldots\ 2^{K \log_K (\log n)}$$

$$2^{K \log_K (\log n)} = 2^{\log n} = n$$

so, there are total

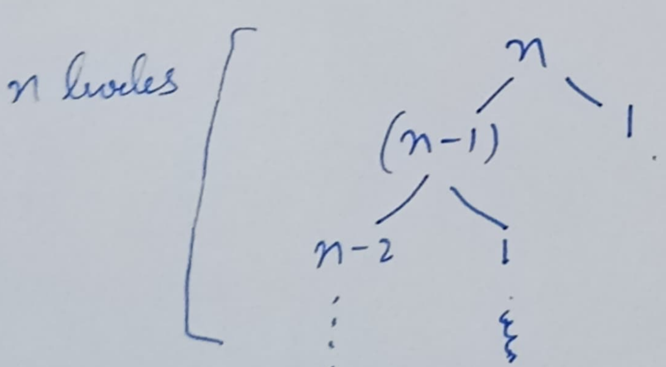
$\log_K (\log n)$ iterations

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

Q.7 Given algo divided array in 99% and 1% part

sol^n

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



$$T(n) = T(n-1) + T(n-2) + \ldots T(1) + O(1)$$
$$= n$$

$$T(n) = O(n)$$

lowest height = 2
highest height = n

The given algo provides linear result.

Q.8 Arrange the following in increasing order of rate of growth

(a) n, $n!$, $\log n$, $\log(\log n)$, root(n), $\log(n!)$, $n \cdot \log(n)$, $\log^2 n$, 2^n, 2^{2^n}, 4^n, n^2, 100

(b) $2(2^n)$, $4n$, $2n$, 1, $\log(n)$, $\log(\log n)$, $\sqrt{\log(n)}$, $\log 2n$, $2\log(n)$, n, $\log(n!)$, $n!$, $n2$, $n\log(n)$

(c) 8^{2n}, $\log_2(n)$, $n\log_6(n)$, $n\log_2(n)$, $\log n!$, $n!$, $\log_8(n)$, 96, $8n2$, $7n3$, $5n$

a- $\log(\log(n))$ < $\log^2(n)$ < $\log(n)$ < $\log(n!)$ < root(n) < $n(\log(n))$ < n < n^2 < 2^n < 100 < $n!$ < 4^n < 2^{2^n}

b- $\log(\log n)$ < $\log n$ < $\sqrt{\log(n)}$ < $\log 2n$ / 1 < $2\log(n)$ < $\log n!$ < $n \cdot \log(n)$ < n < $n2$ / $2n$ < $4n$ < $2(2^n)$ < $n!$

c-

$\log_2(n)$ < $\log_8(n)$ < $\log(n!)$ < $n\log_2(n)$ < $n\log_6(n)$ < $5n$ < $8n2$ < 96 < $7n3$ < $n!$ < $(8)^{2n}$