

Name - Mohammad Anas Khan

Section - CST SPL-1

University Roll No. - 2017514

Class Roll No. - 20

Year - IV Sem / 2 Year

Course - BTech CSE with ML & AI

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

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

j=1 i=1
j=2 i=1+2
j=3 i=1+2+3

i = 1 + 2 + 3 + ... 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

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

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

order @ 2000;
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?

Solⁿ - Fibonacci Series - it is the sum of previous two terms.

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

```
int fibo (int n)
{
```

```
    if  $n \leq 1$ 
```

```
        return n;
```

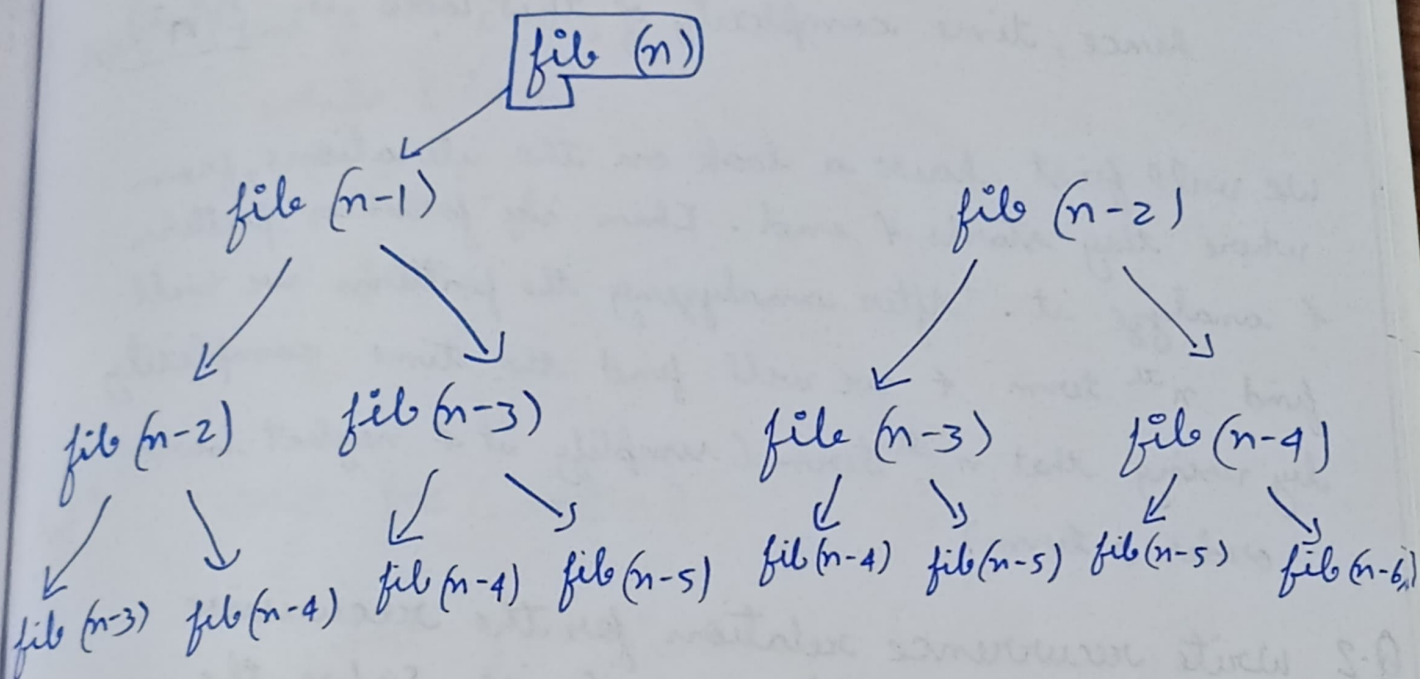


```

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

```

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



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

$$a = 1$$

$$r = 2$$

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

space complexity - depends upon max depth of the tree
so, $O(n)$.

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

Space Complexity: It is $O(n)$ as we haven't use any extra space ~~array~~ to store anything.

Q.3 write programs which have complexity

(i) $n(\log n)$

(ii) n^3

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

solⁿ
(i) $n(\log n)$

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

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

(ii) $j = 1, 2, 3, 4, 5 \dots n \rightarrow A.P.$

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

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

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

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

$$K = \log n + 1$$

$$K = \log(n+1) \quad \text{neglect lower order term}$$

$$K = \log_2 n$$

$$O(\log_2 n)$$

— (a) eq

~~Ans = n~~

(ii) $O(n + n + n + 1)$

$O(n)$ ————— (b) eq

multiplying (a) with (b)

total complexity = $O(n \cdot \log(n))$ ~~→~~

(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

1

2

3

4

5

6

⋮

⋮

j

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

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

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

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

⋮

⋮

k

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

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

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

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

⋮

⋮

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

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

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

now we will neglect
lower order term

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

$$O(n^3)$$

(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

solⁿ $T(n) = T(n/4) + T(n/2) + Cn^2$

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

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

by master's method

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \quad \begin{matrix} a=1 \\ b=2 \end{matrix}$$

$$\Rightarrow C = \log_b a$$

$$\Rightarrow C = \log_2 1 \Rightarrow 0$$

$$\Rightarrow n^C = 1$$

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

$$n^C < f(n)$$

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

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

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

Solⁿ

```
int fun (int n)
```

```
{
  for(i=1; i<=n; i++)
```

```
{
  for(j=1; j<=n; j+=i)
```

```
    // O(1);
```

```
  }
```

```
}
```

```
1
```

```
2
```

```
3
```

```
...
```

```
n
```

```
...
```

```
n
```

```
...
```

```
n
```

```
...
```

```
n
```

```
...
```

```
n
```

```
1
```

```
1 + 3 + 5
```

```
1 + 4 + 7
```

```
...
```

```
n
```

```
...
```

```
n
```

```
...
```

```
n
```

```
...
```

```
n
```

```
...
```

```
n
```

```
...
```

$\frac{n-1}{i}$

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

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

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

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

Q.6 what should be the time complexity of
solⁿ

```
for (int i=2; i<=n; i=pow(i, k))
{
```

```
    // O(1)
}
```

where k is constant

$$T.C = 2, 2^k, 2^{k^2}, 2^{k^3}, \dots, 2^{k \log_k (\log n)}$$

$$2^{k \log_k (\log n)} = 2^{\log n} = n$$

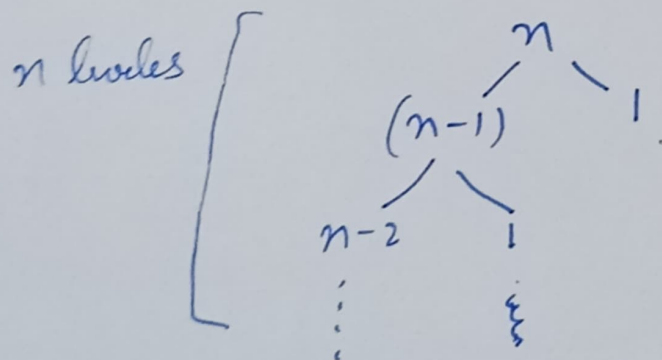
so, there are total

$\log_k (\log n)$ iterations

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

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

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



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

$$= n$$

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

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) $n, n!, \log n, \log(\log n), \text{root}(n), \log(n!), n \cdot \log(n), \log^2 n, 2^n, 2^{2^n}, 4^n, n^2, 100$
- (b) $2(2^n), 4n, 2n, 1, \log(n), \log(\log n), \sqrt{\log(n)}, \log 2n, 2 \log(n), n, \log(n!), n!, n^2, n \log(n)$
- (c) $8^{2^n}, \log_2(n), n \log_6(n), n \log_2(n), \log n!, n!, \log_8(n), 96, 8n^2, 7n^3, 5n$

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

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

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