Tutorial-2

Name - Anshika aarg
Section - F
Roll No. - 46
Univ Rno - - 2016649

Ques What is the time complexity of below code & how?

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

j=1	i=1	] m-level
j=2	i = 1+2	
j=3	i = 1+2+3	

for (i)

$\therefore 1+2+3+\ldots+ < n$

$\therefore 1+2+3+m < n$

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

$m \simeq \sqrt{n}$

By summative method

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

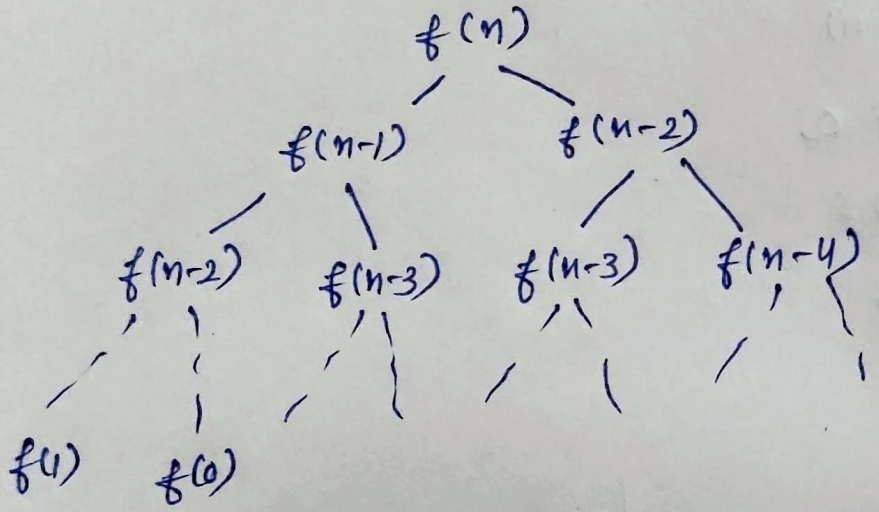
$\boxed{T(n) = \sqrt{n}}$ - Ans

Ques-2 Write recurrence relation for function that prints Fibonacci series. Solve it to get the time complexity what will be the space complexity & why?

→ For Fibonacci series

$f(n) = f(n-1) + f(n-2)$ $\quad f(0) = 0$, $f(1) = 1$

By forming a tree



∵ At every function call we get 2 fn calls.

∵ for n levels

we have $= 2 \times 2 \ldots$ n times

∴ $\boxed{T(n) = 2^n}$

MAXIMUM SPACE - considering Recursive stack:

no. of maximum calls $= n$

For each call we have space complexity $O(1)$

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

without considering recursive stack:

each call we have time complexity $O(1)$

$\boxed{\therefore T(n) = O(1)}$

Q3 Write programs which have complexity, $n(\log n)$, n^3, $\log(\log n)$

1) $n \log n$ → Quick sort

```
void quicksort (int arr[], int low, int high)
{
    if (low < high)
    {
        int pi = partition (arr, low, high);
        quicksort (arr, low, pi-1);
        quicksort (arr, pi+1, high);
    }
}
int partition (int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = (low - 1);
    for (int j = low; j <= high-1; j++)
    {
        if (arr[j] < pivot)
        {
            i++;
            swap (&arr[i], &arr[j]);
        }
    }
    swap (&arr[i+1], &arr[high]);
    return i+1;
}
```

2) n^3 → Multiplication of 2 square Matrix -

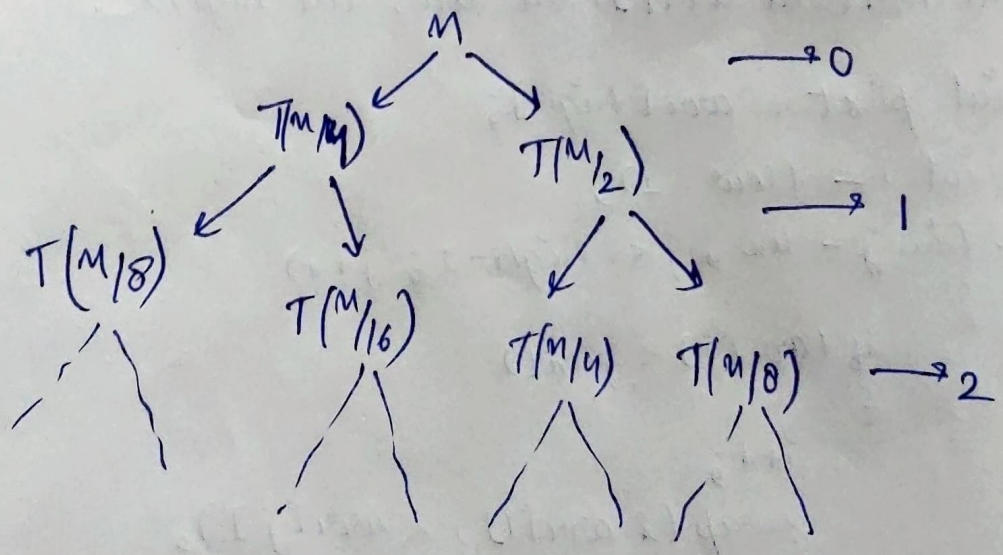
```
for (i=0; i<r1; i++)
    for (j=0; j<c2; j++)
```

```
for ( k=0 ; k<c1 ; k++)
  {
    res [i][j] += a[i][k] * b[k][j];
  }
```

3.
```
log(log(n))
for (i=2; i<n ; i=pow(i,i))
  {
    count++;
  }
```

Ques.4 Solve the following resonance relation.

$T(n) = T(\frac{n}{4}) + T(\frac{n}{2}) + cn^2$



At level

$0 \rightarrow cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{c\,5n^2}{16}$

$2 \rightarrow \frac{n^2}{8^2} + \frac{n^2}{16^2} + \frac{n^2}{4^2} + \frac{n^2}{8^2} = \left(\frac{5}{16}\right)^2 n^2 c$

max level $= \frac{n}{2^k} = 1$

$= K = \log_2 n$

$$T(n) = c\left[n^2 + \left(\frac{5}{16}\right)n^2 + \left(\frac{5}{16}\right)^2 n^2 + \cdots + \left(\frac{5}{16}\right)^{\log n} n^2\right)$$

$$T(n) = cn^2\left[1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \cdots \left(\frac{5}{16}\right)^{\log n}\right]$$

$$T(n) = Cn^2 \times 1 \times \left(\frac{1 - (5/16)^{\log n}}{1 - (5/16)}\right)$$

$$T(n) = cn^2 \times \frac{11}{3} \times \left(1 - \left(\frac{5}{16}\right)^{\log n}\right)$$

$$T(n) = O(n^2 c)$$

$O((n^2)$ Ans

Ques-5 What is the time Complexity of following fun()?

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

for	i	j
	1	1
	2	1+3+5
	3	1+4+7
	!	1+5+9
	n	

$j = (n-1)/i$ times

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

$$\therefore T(n) = \frac{(n-1)}{1} + \frac{(n-1)}{2} + \frac{(n-1)}{3} + \cdots + \frac{(n-1)}{n}$$

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

$$= n\log n - \log n$$

$T(n) = O(n \log n)$ — Ans

Ques 6 What should be time complexity of

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

where k is a constant.

for

i
2^1
2^k
2^{k^2}
2^{k^3}
.
.
2^{k^m}

where

$2k^m <= n$

$k^m <= \log_2 n$

$m = \log_k \log_2 n$

$\therefore \sum_{i=1}^{m} 1$

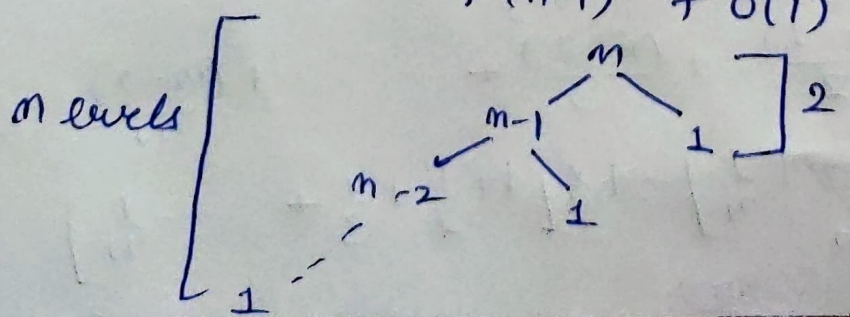
$1 + 1 + 1$ ——— m times

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

Ques-7 Write a recurrence relation when quick sort repeatedly divides array into 2 parts 99% and 1%. Derive time complexity in this case. Show the recurrence tree while deriving time complexity & find difference in heights of both extreme parts. What do you understand by this analysis?

Given algo divides array in 99% and 1% part.

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



work is done at each level

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

$= n \times n$

$\therefore T(n) = O(n^2)$

lowest height $= 2$

highest height $= n$

$\therefore$ difference $= n-2$ $\quad n > 1$

The given algorithm produces linear result.

Ques- Arrange following in increasing order of rate of growth:

a) $n, n!, \log n, \log\log n, \text{rest}(n), \log(n!), n\log n, \log^{2(n)}, 2^n, 2^{2^n}, 4^n, n^2, 100$

→ $100 < \log\log n < \log n < (\log n)^2 < \sqrt{n} < n < n\log n < \log(n!) < n^2 < 2^n < 4^n < 2^{2^n}$.

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

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

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

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