TUTORIAL-2

Name:- Srijan
Section :- F
Roll No :- 30
University Roll No :- 2017067.

1Ans :-
Solution :-

$$\left.\begin{array}{ll} j=1 & i=1 \\ j=2 & i=1+2 \\ j=3 & i=1+2+3 \end{array}\right] n\text{- level}$$

for (i)

$\because 1+2+3+ \dots + < n$

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

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

$$m \simeq \sqrt{n}$$

By summation method

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

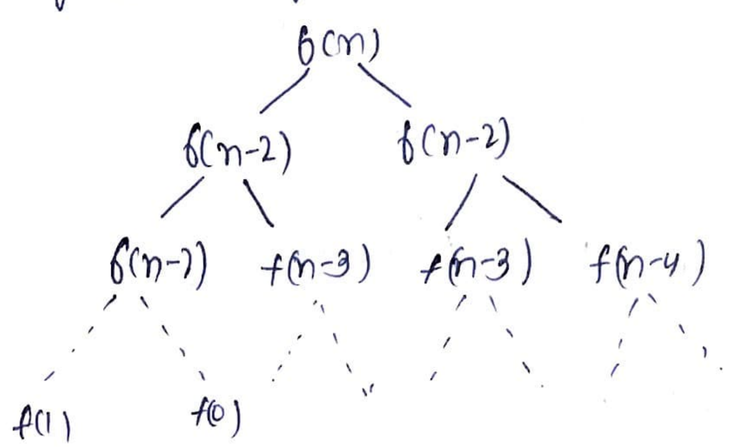
$$\boxed{T(n) = \sqrt{n}}$$

2Ans :-
Solution :- for Fibonacci series

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

$f(0)=0$
$f(1)=1$

By forming a tree



$\because$ At every function call we get 2 function calls

for n levels.

$\therefore$

we have = 2x2 - - - - n time

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

MAXIMUM SPACE

considering recursion

Stack:

No of calls maximum = n

for each call we have space complexity O(1)

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

without considering recursion stack:

each call we have time complexity O(1)

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

3 Ans:→

Solution:→ (1) nlogn → Quick sort

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

(2) $n^3 \rightarrow$ Multiplication of 2 square matrix :-

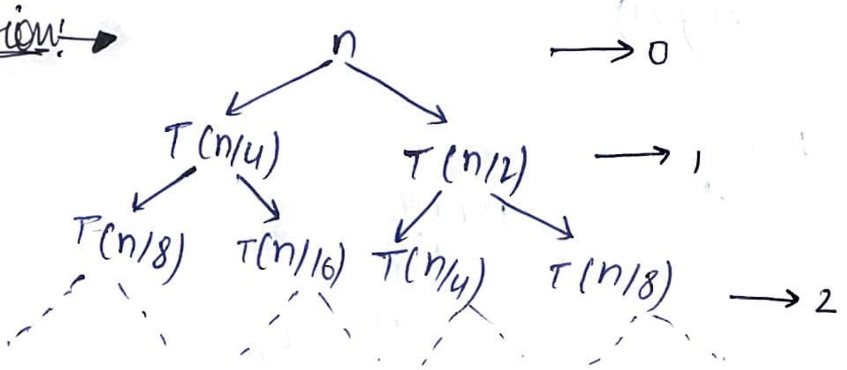
```
for (i=0 ; i<r1 ; i++){
    for (j=0 ; j<c2 ; j++){
        for (k=0 ; k<c1 ; k++){
            res [i][j] += a[i][k] * b[k][j];
        }
    }
}
```

(3) log (log n)

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

Ques:→

Solution:→



At level

$0 \rightarrow cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{c5n^2}{16}$

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

$\vdots$

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

$k = \log_2 n$

$T(n) = c\,(n^2 + (5/16)\,n^2 + (5/16)^2\,n^2 + \cdots + (5/16)^{\log n}\,n^2)$

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

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

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

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

$O(cn^2)$

5 Ans :→
Solution :→ for

i	j
1	1
2	1+3+5
3	1+4+7
⋮	
n	1+5+9

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

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

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

$$T(n) = n[1 + 1/2 + 1/3 + \cdots + 1/n] - 1 \times [1 + 1/2 + 1/3 + \cdots + 1/n]$$

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

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

6 Ans :→
Solution :→ Given algorithm for i

2^1
2^k
2^{k^2}
2^{k^3}
⋮
2^{k^n}

where $2^{k^m} <= n$

$k^m = \log_2 n$

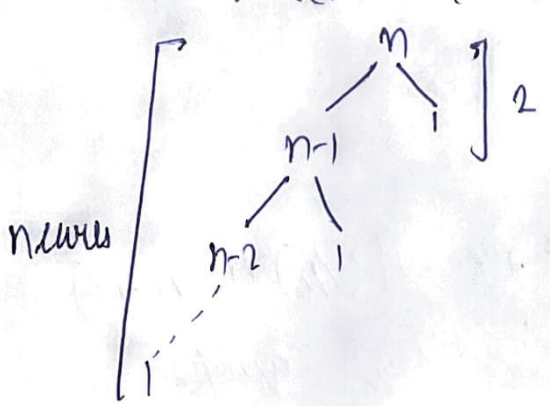
$m = \log_k \log_2 n$

$1 + 1 + 1 \cdots m$ times

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

7 Ans :→
Solution :→ Given Algorithm divides array in 99% and 1% part

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



'n' work is done at each level

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

$$= n \times n$$

$\boxed{\therefore \; T(n) = O(n)^2}$

lowest night $=2$

highest night $= n$

$\therefore$ $\boxed{\text{difference} = n-2}$ $n > 1$

The given algorithm produces linear results

8 Ques:→

Solution:→

(a)

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

(b)

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

(c)

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