TUTORIAL → 2

Ques1

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

for j=1   i=1
     j=2   i=1+2;
     j=3   i=1+2+3;

(m levels)

for $1+2+3+ - - - - < n$

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

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

$m \approx \sqrt{n}$

$\therefore$ By Summation method

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

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

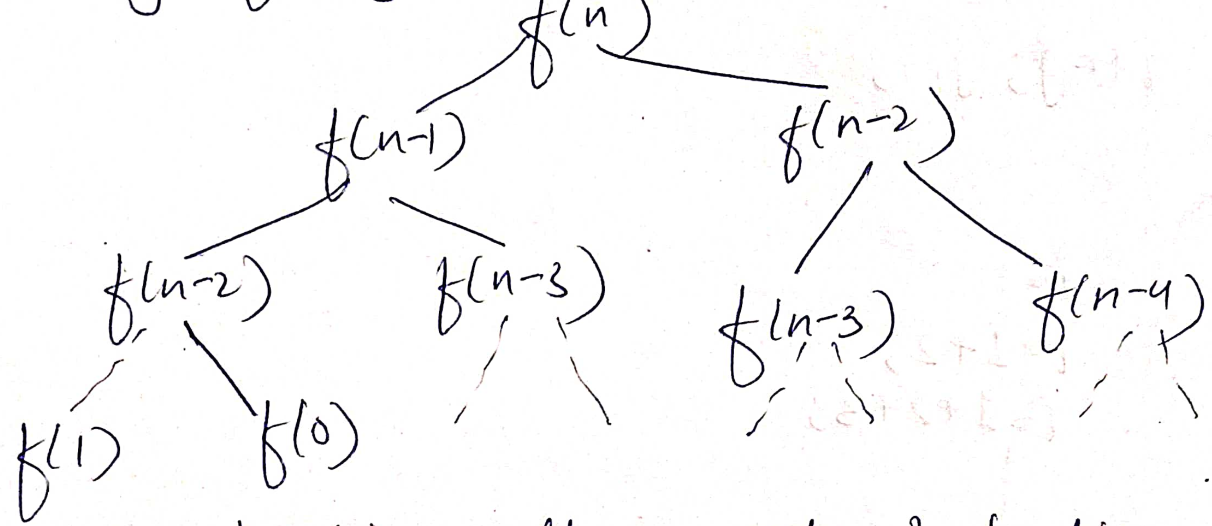
Ques2

for fibonacci series-

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

$$f(0) = 0, \quad f(1) = 1$$

By forming tree-



At every function call we get 2 function calls-

$\therefore$ for n levels-

we have, $2 \times 2 \times$ - —— n times

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

Maximum space-

considering recursive stack,

no. of calls maximum $= n$

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

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

Without considering recursive stack, for each we have time complexity $O(1)$

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

Ques 3

(1) n logn

```
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

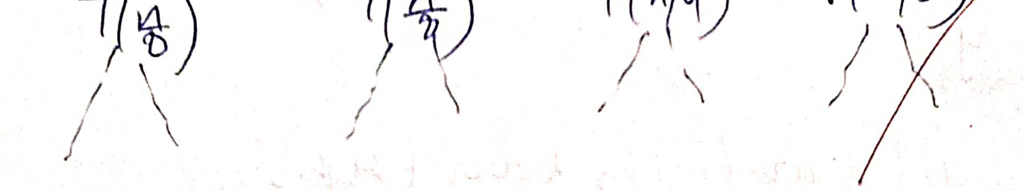
multiplication of two square matrix

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

(3) $\log(\log n)$

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

Ques 4 $T(n) = T(n/4) + T(n/2) + c \times n^2$



At level -

$0 \rightarrow cn^2$

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

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

|

max levels $= \frac{n}{2^k} = 1 \Rightarrow k = \log_2 n$

$\therefore T(n) = c(n^2 + (5/16)n^2 + (5/16)^2 n^2 + \text{———} (5/n)^{\log n} n^2)$

$= cn^2[1 + (5/16) + (5/16)^2 + \text{———} (5/16)^{\log n})$

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

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

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

Ques 5

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

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

for	i	j
	1	1
	2	1+3+5
	3	1+4+7
	|	|
	n	

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

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

$T(n) = n[1 + \frac{1}{2} + \frac{1}{3} + \text{——} \frac{1}{n}] - 1[1 + \frac{1}{2} + \frac{1}{3} + \text{——} \frac{1}{n}]$

$= n \log n - \log n$

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

Ques6

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

for i
2^1
2^k
2^{k^2}
2^{k^3}
|
2^{k^n}

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

$k^m = \log_2 n$

$m = \log_k \log_2 n$

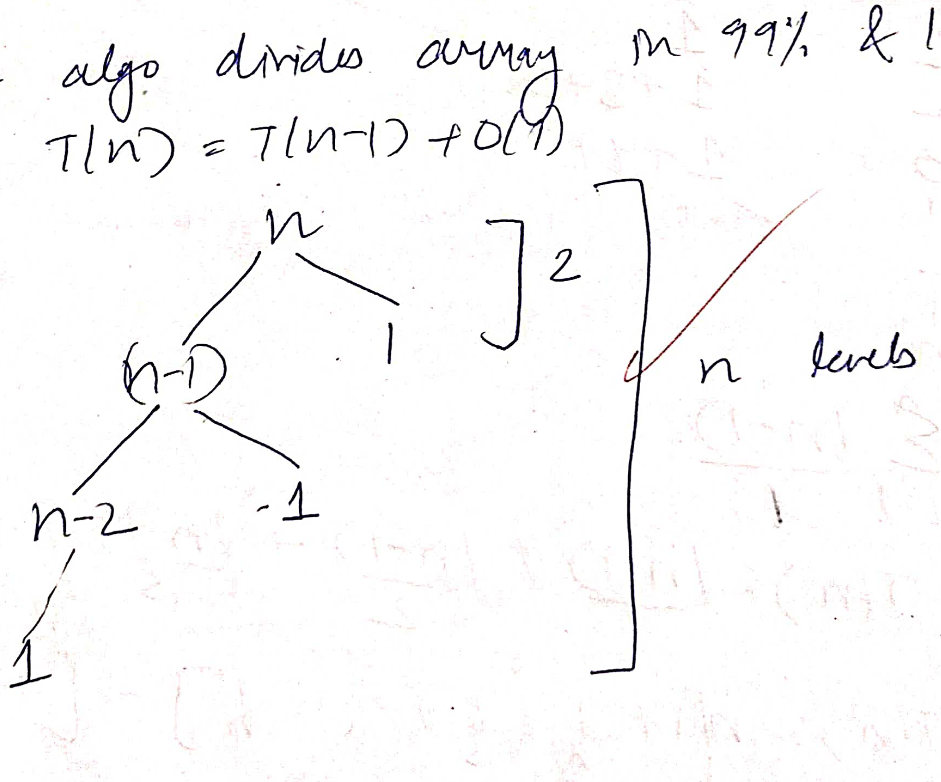
$\therefore \sum_{i=1}^{m} 1$

$\Rightarrow 1+1+1+ \text{— —} \ m$ times

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

Ques7 Given algo divides array in 99% & 1% part

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



n' work is done at each level for merging.

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

$= n \times n$

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

Lowest height = 2

Highest height = n

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

The given algo produces linear result

Ques8- Considering for large values of 'n'.

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

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

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