(1)

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

⇒ for $j = 1 \quad i = 1;$
$j = 2 \quad i = 1+2;$
$j = 3 \quad i = 1+2+3;$] m levels

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

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

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

$m \approx \sqrt{n}$

by summation method:

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

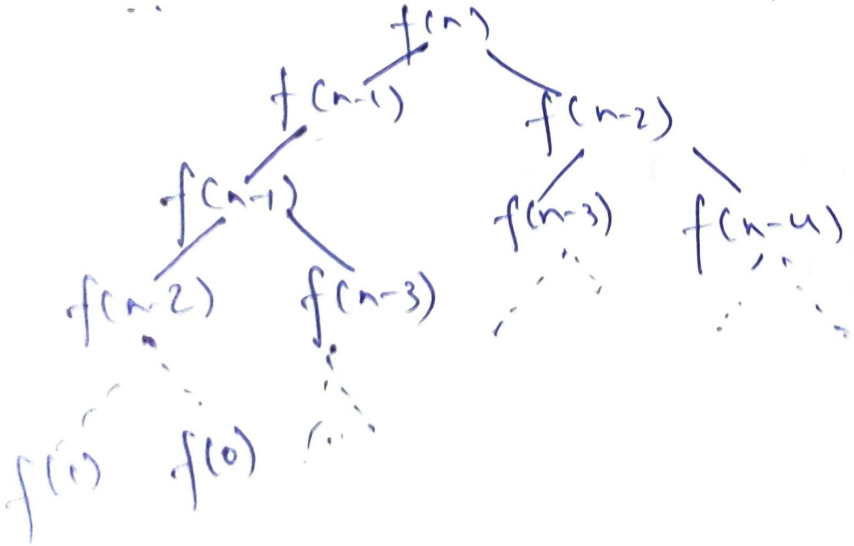
$\therefore T(n) = \sqrt{n}$

(2) For fibonacci series:-

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

$f(0) = 0 \qquad f(1) = 1$

$\therefore$



∵ at every function call we get two function calls for n levels:-

we have :> 2×2 .... n times

∴ $T(n) = 2^n$

Maximum Space :- considering recursive stack:-

no of calls max. = n

for each call we have space complexity O(1)

∴ $T(n) = O(n)$

(3)

a) n log n :-

Quick sort

```
void func ( int arr[], int l, int h)
    {  if ( l < h)
       { int pi = partion (arr, lo, h);
         func ( arr, l, pi-1);
         func ( arr, pi+1, h);
    } }

int partion ( int arr[], int l, int h)
    { int pi = arr[h];
      int i = ( l-1);
```

```
for(int j=l ; j<=h; j++)
{ if(arr[i]<pi)
    { i++;
      swap(arr[i], arr[j]);
    }
}
swap(arr[i+1], arr[h]);
return (i+1);
}
```

(b) n^3 :

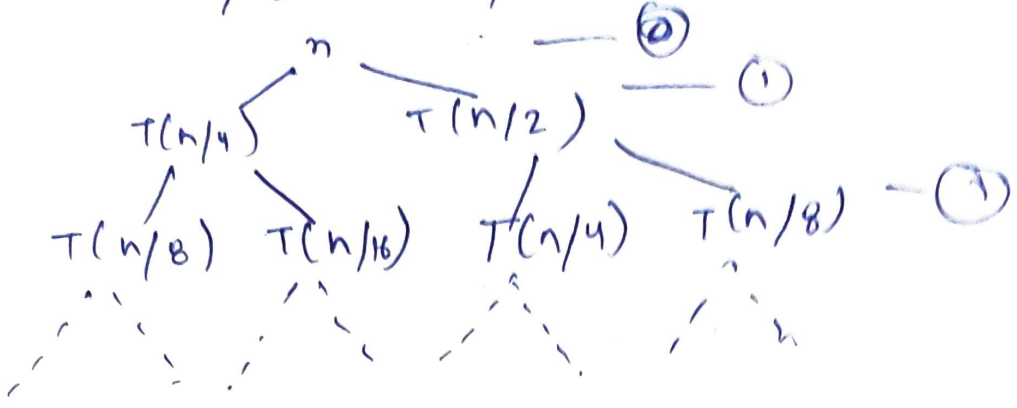
Multiplication of two square matrix

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

(c) $\log(\log n)$

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

④ $T(n) = T(n/4) + T(n/2) + cn^2$



At level → 0 → cn^2

1 → $\frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{5cn^2}{16}$

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

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

$\Rightarrow k = \log_2 n$

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

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

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

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

$\therefore T(n) = O(n^2 c) \Rightarrow O(cn^2)$

(5)
```
int fun(int n)
{
    for(i=1; i<=n; i++)
    {
        for(j=1; j<n; j+=i)
        {
        }
    }
}
```

for → i : 1, 2, 3, 4, ..., n

j : 1, 1+3+5, 1+4+7, 1+5+9, ...

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

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

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

$= n \log n - \log n$

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

(6)
```
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^m}

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

$k^m = \log_2 n$

$m = \log_k \log_2 n$

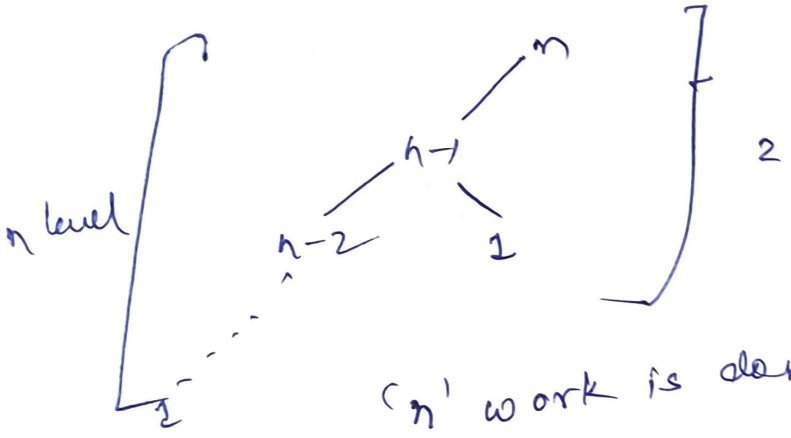
$\because \sum_{i=1}^{m} 1$

$\Rightarrow 1 + + + \ldots m$ times

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

(7) 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

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

$= n \times n$

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

lowest height = 2

height height = n

$\therefore$ diff $= n - 2$ $\because (n > 1)$

(8) 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) $1 < \log\log n \leq \sqrt{\log n} < \log n < \log 2n < 2\log n < n < n\log n < 2n < 4n < \log(n!) < n^2 < n < 2^{\log 2n} < 5n$

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