(Krishna Rathi)
F-62

Tutorial-2

Ques1) What is the time complexity of below code and how?

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

Ans)

$j = 1 \quad i = 1$
$j = 2 \quad i = 1+2$
$j = 3 \quad i = 1+2+3$] m-level

for (i)

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

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

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

$m \simeq \sqrt{n}$

By summation method

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

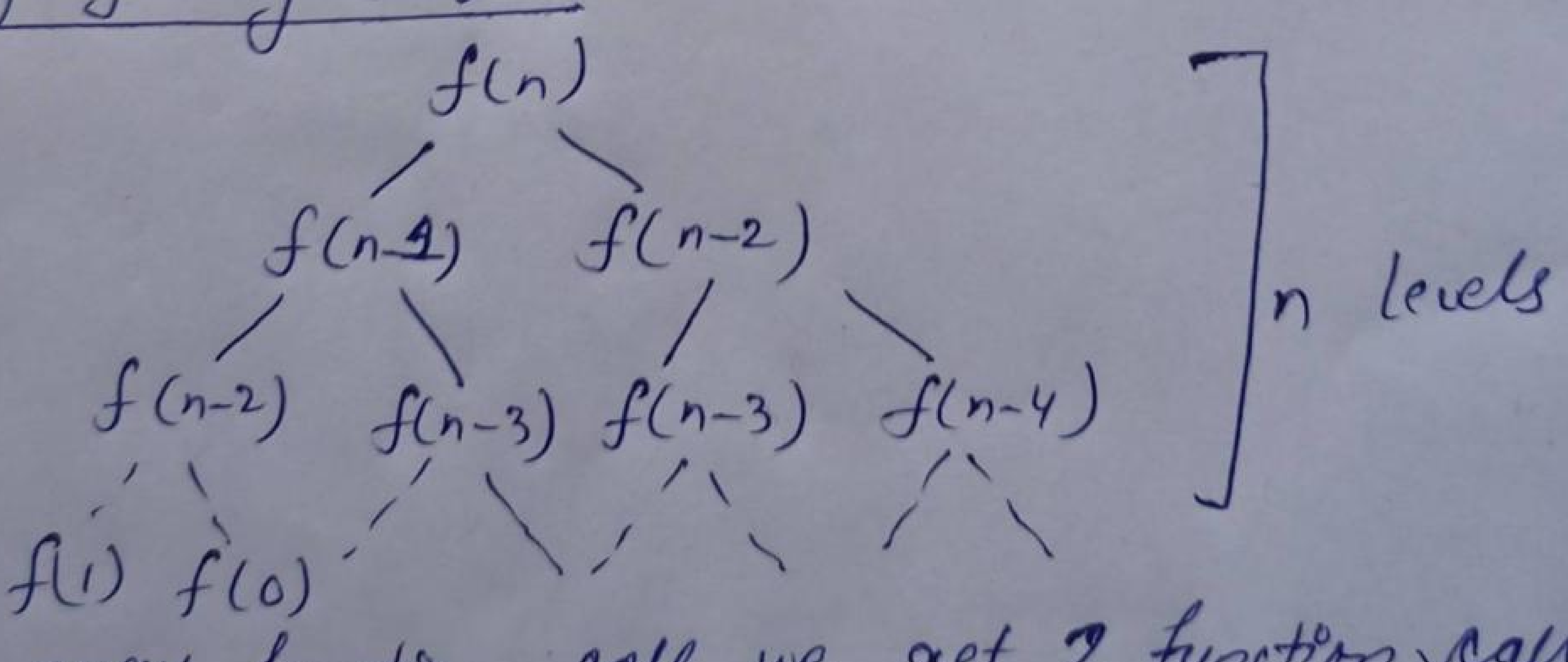
$\boxed{T(n) = \sqrt{n}}$ Ans

Ques2) Write recurrence relation for function that prints Fibonacci series. Solve it to get the time complexity. What will be the space complexity and why?

Ans) For fibonacci Series

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

By forming a tree



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

$\therefore$ for n levels

We have $= 2 \times 2 \times \cdots$ n times

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

Scanned by TapScanner

MAXIMUM SPACE

Considering Recursive Stack:

no. of calls maximum = n

For each call we have space complexity O(1)

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

Without considering Recursive stack:

each call we have time complexity O(1)

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

Ques3) Write programs which have space complexity: n(log n) ; n^3, log(log n)

Ans) 1) n logn → 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 [i] < 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];
    }
```

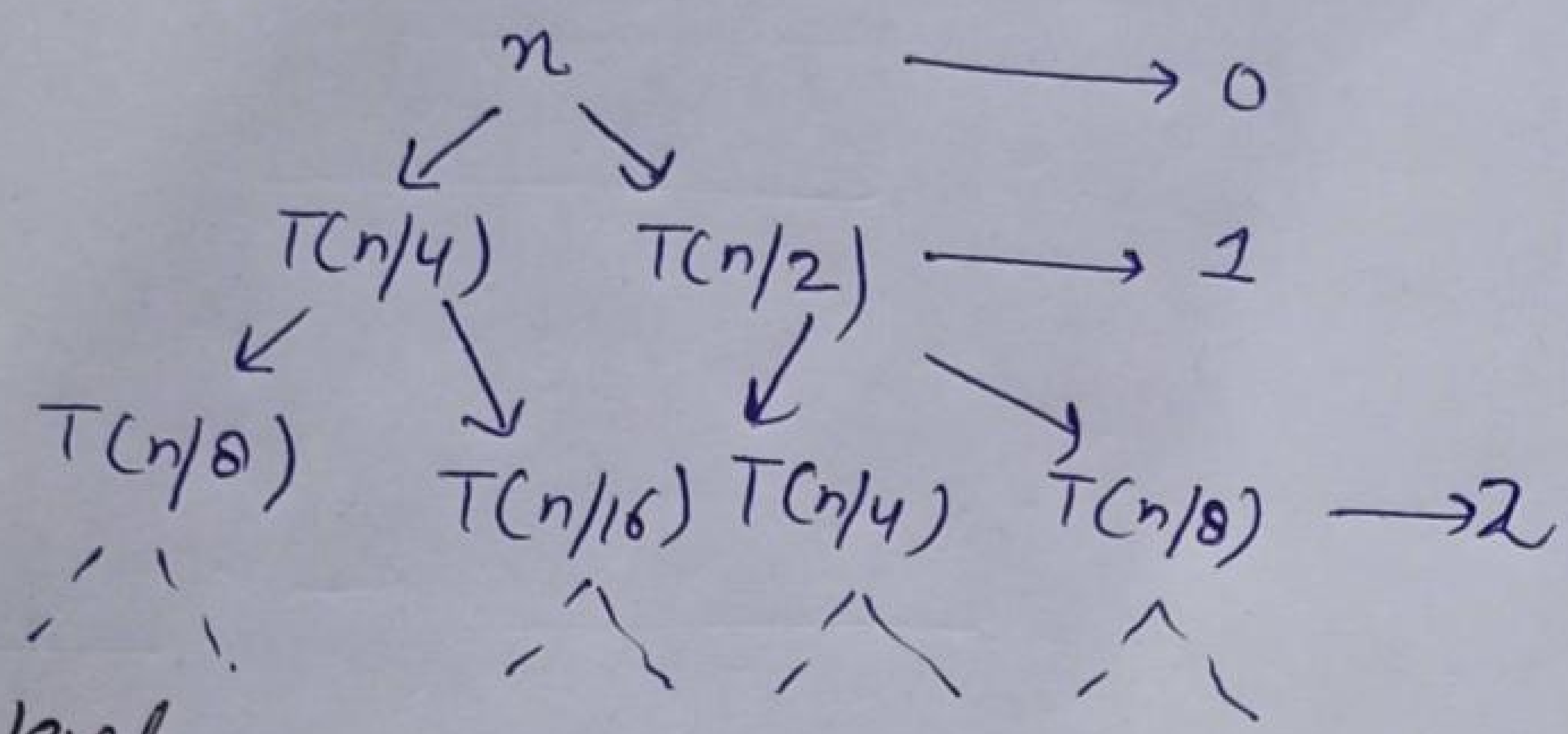
log (log n)

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

Ques4) Solve the following recurrence relation

$T(n) = T(n/4) + T(n/2) + Cn^2$

Ans)



At level

$0 \rightarrow Cn^2$

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

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

$\vdots$

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

$= k = \log_2 n$

$T(n) = C\left(n^2 + (5/16)n^2 + (5/16)^2 n^2 + \dots + (5/16)^{\log n} n^2\right)$

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

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

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

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

$\boxed{O(Cn^2)}$ Ans

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
 } } }
```

Ans) for

i	j
1	1
2	1+3+5
3	1+4+7
⋮	1+5+9
n	

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

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

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

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

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

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

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

Ans) for

i

2^1

2^k

2^{k^2}

2^{k^3}

⋮

2^{k^m}

where

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

$k^m = \log_2 n$

$m = \log k \log_2 n$

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

$1 + 1 + 1 + \dots$ m times

$$\boxed{T(n) = O(\log_k \log n)}$$

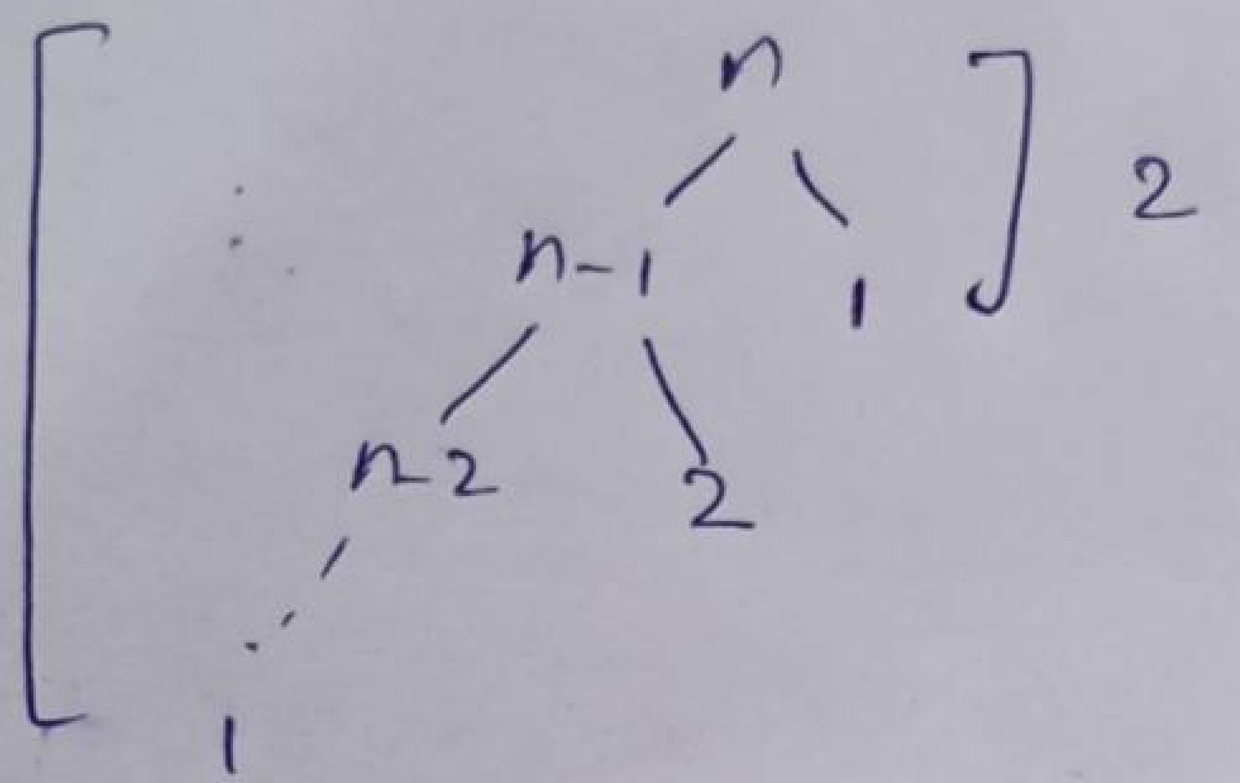
Scanned by TapScanner

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

Ans) Given :- Algorithm divides array in 99% and 1% part

$\therefore T(n) = f\cancel{k}T(n-1) + O(1)$

n times



'n' work is done at each level.

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

$$= n \times n$$

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

Lowest Height = 2

Highest Height = n

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

The given algorithm produces linear result.