Aditi Banerji
38 - F
20[redacted]6575

Tutorial - 2

Q1. What is the time complexity of below code and how?

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

So, $j = 1$ $\quad$ $i = 1$

$j = 2$ $\quad$ $i = 1+2$

$j = 3$ $\quad$ $i = 1+2+3$

⋮ $\quad$ ⋮

n times $\quad$ n times

ie for(i)

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

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

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

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

Aditi.

By Summation Method,

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

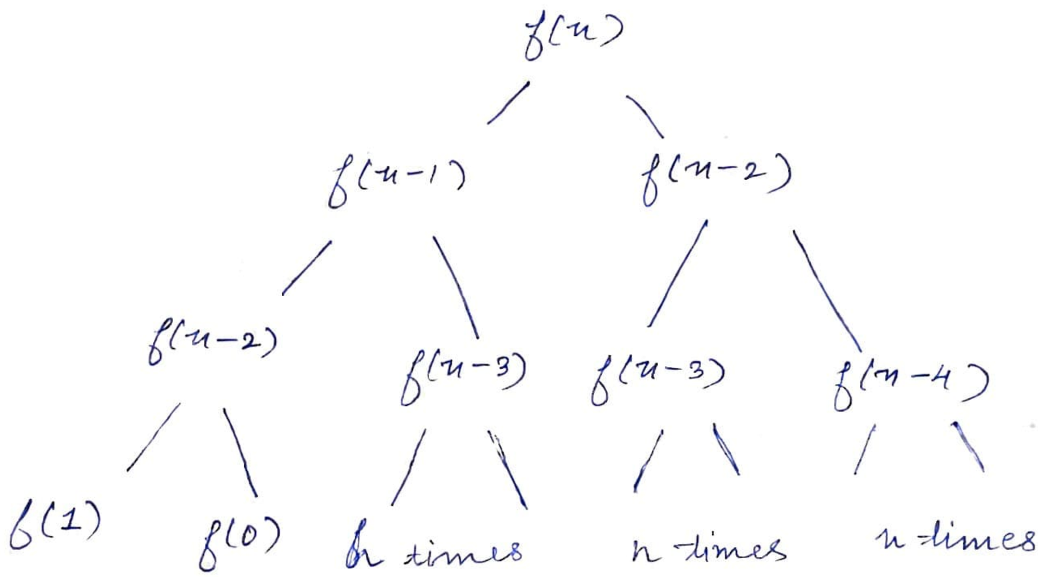
So, $T(n) = \sqrt{n}$ Ans

Q2. Write Reccurance Relation for function that prints Fibonnaci Series. Solve it to get time complexity. What will be the space complexity and why? (2)

→ for Fibonnaci Series :

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

∴ using Tree formation,



Since at every level we get 2 functions, so for n levels : we have → 2 × 2 × ... n times

∴ Time Complexity → $T(n) \Rightarrow 2^n$

* for space complexity :

No. of maximum calls $= n$

For each call Space Complexity $= O(1)$

∴ $T(n) = O(n)$

(In case of Recursive stack)

-Aditi

Q3. write programs which have Complexity :
$n(\log n)$, n^3, $\log(\log n)$

1) Program with $n(\log n)$ Complexity :-
(Quick Sort)

```
void quick_sort ( int a[], int l, int r)
{
    if (l<r)
    {
        int p = partition ( a, l, r );
        quick_sort ( a, l, p-1);
        quick - sort ( a, p+1, r);
    }
}
int partition ( int a[], int l, int r)
{
    int pivot = a[r];
    int i = (l-1);
    for ( int j=l ; j<=r-1; j++)
    {
        if ( a[j] < pivot )
        {
            i++;
            swap( &a[i], &a[j]);
        }
    }
    swap ( &a[i+1], & a[r]);
    return (i+1);
}
```

2) Program with n^3 Complexity :- multiplication of Square matrix

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

Aditi

```
        for( int k=0; k<4; k++)
    {
            res[i][j]+ = a[i][k] * b[k][j];
    }
  }
}
```


3) Program with log(logn) Time complexity :-

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

Q4. Solve the following Reocurrence Relation
$T(n) = T(n/4) + T(n/2) + Cn^2$

$n \rightarrow 0$

$T(n/4) \quad T(n/2) \rightarrow 1$

$T(n/8) \quad T(n/16) \quad T(n/4) \quad T(n/8) \rightarrow 2$

At level :

$0 \rightarrow Cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{C5n^2}{16}$

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

$\vdots$

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

ie $k = \log_2 n$

Aditi.

$$T(n) = c(n^2 + (5/16)n^2 + (5/16)^2 n^2 + \dots + (5/10)^{\log n} \cdot n^2)$$

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

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

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

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

Q5. what is the time Complexity of following func()?

```
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
⋮	⋮
n times	n times

using Summation Method,

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

Aditi

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

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

$= n \log n - \log n$ (neglecting $\log n$)

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

Q6. what should be time Complexity of

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

where k is a Constant

for = i

2^1
$2k$
$2k^2$
$2k^3$
$\vdots$
$2k^m$

where $2k^m <= n$

$k^m = \log_2 n$

$m = \log k \log_2 n$

using Summation Method,

$\sum_{i=1}^{m} 1$ ie $1+1+1+ \dots$ m times

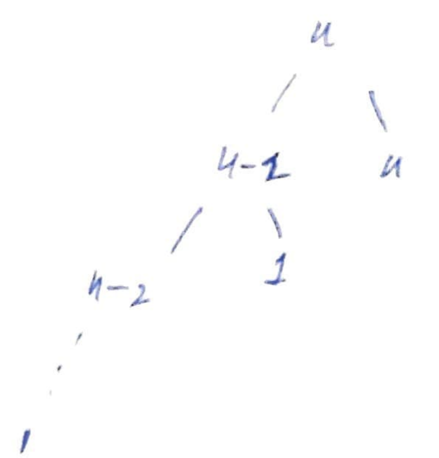
$T(n) = O(\log_k \log n)$ Ans.

Aditi

Q7. Write a Reocurrence Relation when quick sort repeadely divides array into 2 parts of 99% and 1%. Derive time Complexity in this case. Show the Reocurrence tree while deriving T.C. and find difference in height of both extreme parts. What do you understand by this analysis? (7)

→ ATQ, Algorithm divides array into 99% and 1%

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



'n' times 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 $= n-2$ where $n > 1$

Aditi.

→ This Algorithm produces 'Linear' Result

Q8. Arrange following in Increasing order of Rate of growth:

a) n, $n!$, $\log n$, $\log\log n$, $\sqrt{n}$, $\log(n!)$, $n\log n$, $\log^2(n)$, 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, $\log(n!)$, $n!$, n^2, $n\log(n)$

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

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

96, $8n^2$, $7n^3$, $5n$

$\rightarrow 96 < \log_8 n < \log_2 n < 5n < n\log_6(n) < n\log_2 n < \log(n!) < 8n^2 <$

$7n^3 < n! < 8^{2n}$

Aditi.