Name :- Shikha Singh
Section :- CST
Roll no :- 18

Tutorial - 2

Que ① What is the time complexity of below code and how?

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

→ Value after execution of while loop

1st time = i = 1

2nd time = i = 1 + 2

3rd time i = 1+2+3

4th time i = 1+2+3+4

let for ith time $i = (1+2+3+ \dots i) < n$

$= i(i+1)/2 < n$

$= i^2 < n$

$i = \sqrt{n}$

therefore time complexity $= O\sqrt{n}$ Ans.

Shikha Singh

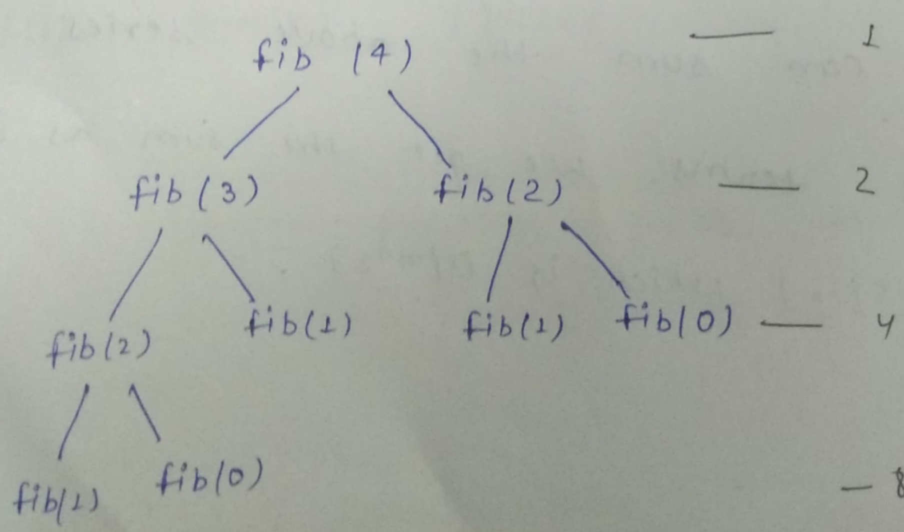
Que (2) Write recurrence relation for the recursive function that prints Fibonacci series. Solve the recurrence relation to get time complexity of the program.

→ What will be the space complexity of this program and why?

→ 0 1 1 2 3 5 . . . n

```
int fib(int n)
{
    if (n <= 1)
        return n;                                  — O(1)
    return fib(n-1) + fib(n-2) ← T(n-1)+T(n-2)
}
```

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



$T(n) = 1 + 2 + 4 + 8 + \ldots + n$

$$S(n) = \frac{a(r^{n+1} - 1)}{r - 1}$$

$$= \frac{1(2^{n+1} - 1)}{1}$$

$$= T(n) = [2^n \cdot 2 - 1]$$

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

space complexity $O(1)$

As recursive implementation doesn't store any values from and calculates every value from scratch. So as complexity of call is $O(1)$

$\therefore$ total space complexity $= O(1)$

Que (3) Write programs which have complexity - $n(\log n)$, n^3, $\log(\log n)$.

→ (i) $n(\log n)$

```
for (i=1; i<=n; i=i*2)       // log n
for (j=1; j<=n; j++)         // n
{   int s =1;
}
```

$\Rightarrow O(n \log n)$

(ii) n^3

```
for (i=0; i<=n; ++i)          n times
  for (j=0; j<=n; ++j)        n times
    for (k=0; k<=n; ++k)      n times
    {
      cout<<"Hey;"}
```

$\Rightarrow O(n^3)$

(iii) $\log(\log n)$

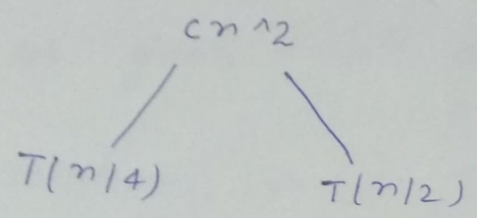
→
```
for (int i=2; i<n; i= pow (i, c))
{
   cout <<"Hi" ;}
```

where c is any constant

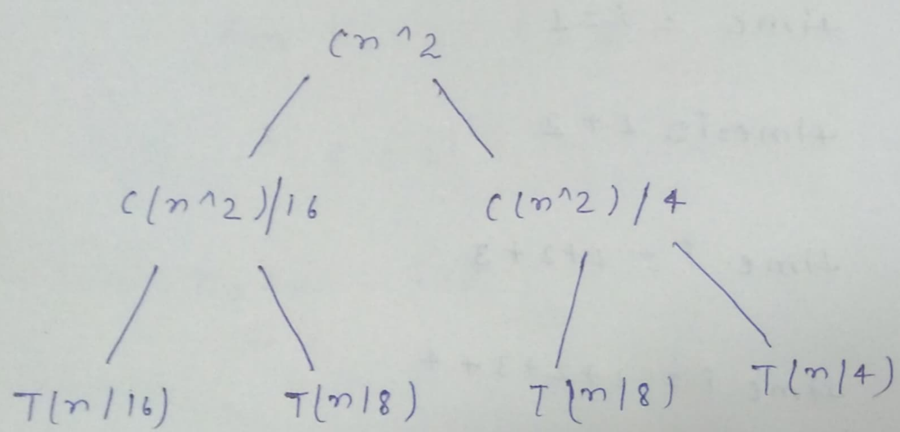
Shubham
Sup

Que (4) Solve the following recurrence relation $T(n) = T(n/4) + T(n/2) + cn^2$

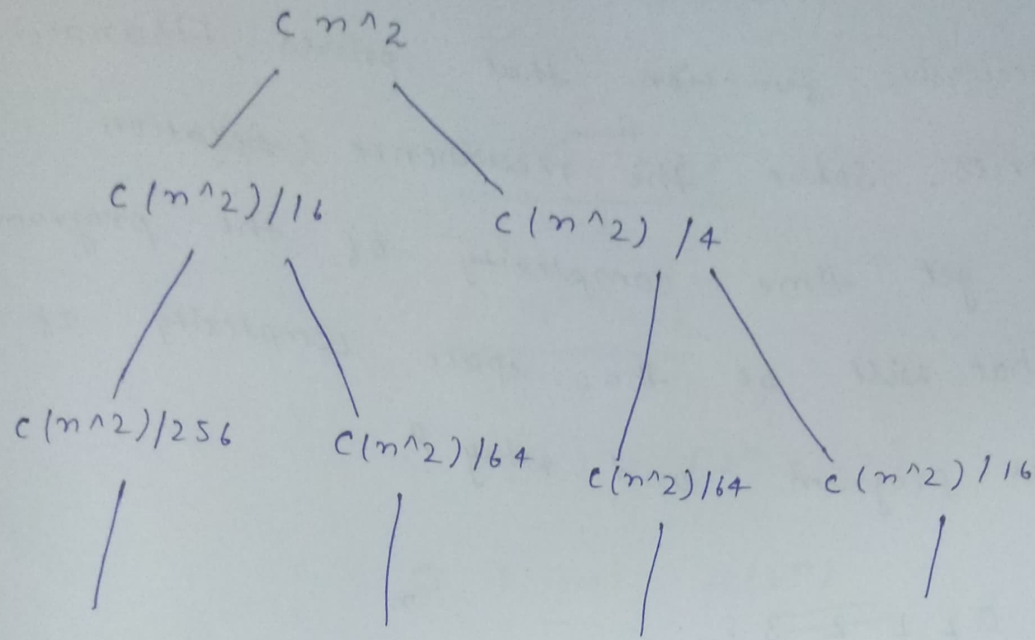
→ The intial recursion tree for the given recurrence relation.



If we further break down the expression $T(n/4)$ and $T(n/2)$, we get following recursion tree.



Breaking down further gives us



If we sum the above tree level by level. We get the following series

T(n) = c (n^2 + 5(n^2)/16 + 25(n^2)/256)+ ...

The above series is geometrical progression with ratio 5/16. To get an upper bound. We can sum the above series for finite terms. We get the sum as (n^2)/(1-5/16) which is O(n^2).

Que (5) What is the time complexity of following function fun( )?

→

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

→

for $i = 1 \rightarrow j = 1, 2, 3, 4 \ldots n$ (run for n times)

for $i = 2 \rightarrow j = 1, 3, 5 \ldots$ (run for n/2 times)

for $i = 3 \rightarrow j = 1, 4, 7 \ldots$ (run for n/3 times)

$T(n) = n + n/2 + n/3 + n/4 + \ldots$

$= n(1 + 1/2 + 1/3 + 1/4 + \ldots)$

$= n \int_1^n 1/x \Rightarrow n \int_1^x \frac{dx}{x} \Rightarrow \log]_1^n$

$= n \log n$

The time complexity of following function is $n \log n$

Shubham Singh

Que ⑥ What should be the time complexity of

```
for (int i = 2; i <= n; i = pow (i, K))
{
    // Some O(1) expressions or statements
}
```

Where, K is a constant.

→ for first iteration $i = 2$

" second " $i = 2^K$

" third " $i = (2^K)^K = 2^{K^2}$

|

|

|

n^{th} iteration $i = 2^{K^i}$ loops ends at $2^{K^i} = n$

apply log $\log n = \log 2^{K^i}$

$K^i = \log n$

again apply log $\log(K^i) = \log n$

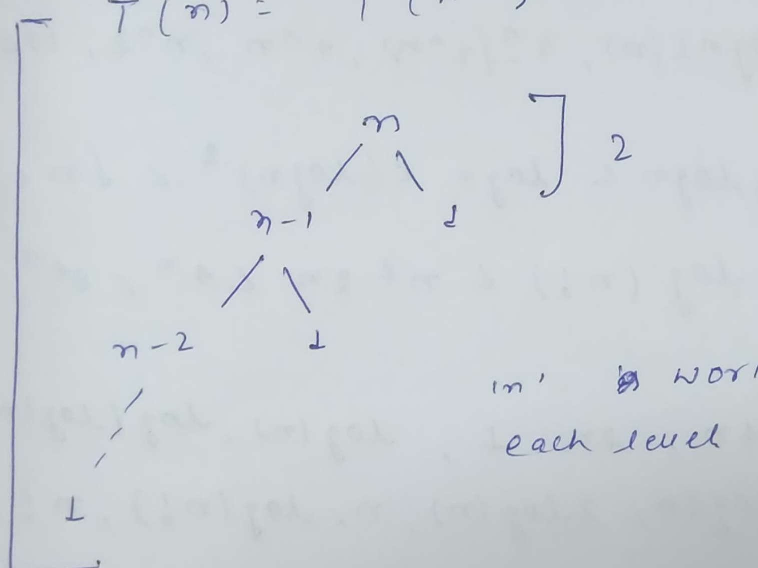
$i = \log_K(\log n)$

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

Que (7) Write a recurrence relation which quick sort repeatedly divides the array into two parts of 99% and 1%. Derive the time complexity in this case. Show the recursion tree while deriving time complexity and find the difference in heights of both the extreame parts. What do you understand by this analysis?

→ $T(n) = T(n-1) + O(1)$

n levels



'n' work is done on each level for merging

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

$= n \times n$

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

Shubh Gupta

Lowest height = 2

highest " = n

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

The given algorithm provides linear result.

Que ⑧ Arrange the following in increasing order of rate of growth.

→ considering for large value of 'n'

(a) n, n!, log n, log log n, root(n), log(n!),
n log n, log n 2(n), 2^(2^n), 4^n, n^2, 100

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

(b) 2(2^n), 4n, 2n, 1, log(n), log(log(n)),
$\sqrt{\log(n)}$, log 2n, 2 log(n), n, log(n!), n!,
n2, nlog(n)

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

Shubham Singh

1(c) $8^{(2n)}$, $\log_2(n)$, $n \log_6(n)$, $n \log_2(n)$,

$\log(n!)$, $n!$, $\log_8(n)$, 96, $8n^2$, $7n^3$, $5n$

→ $96 < \log_8 n < \log_2 n < 5n < n \log_6 n$

$< n \log_2 n < \log(n!) < 8n^2 < 7n^3 < n! < 8^{2n}$