DATE: / 20
PAGE No

Tutorial - 2

(1) What is the time complexity of below code show

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

Time complexity - O(sqrt(n))

1st time $i=1$

2nd time $i=3$ $(i=i+2)$

3rd time $i=6$ $(i=1+2+3)$

⋮

nth time $i = \frac{x(x+1)}{2} = x^2$

$x^2 < n$

$x = \text{sqrt}(n)$

(2) Write recurrence relation for the recursive function that prints Fibonacci series. Solve the recurrence relation to get complexity of program what will the space complexity of this program & why

Solⁿ.

* fib(n) = fib(n-1) + fib(n-2)

let T(0) = 1

```
fib(n):
      if n <= 1
            return 1
      return fib(n-1) + fib(n-2)
```

Time complexity :

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

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

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

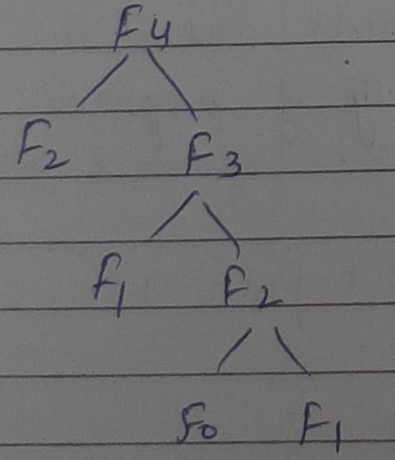
$= 2 * (2T(n-y)+c)+c$
$= 4T(n-y)+3c$

$T(n-4) = 2*(4T(n-y)+3c)+c$
$= 8T(n-3)+7c$
$= 2^k * T(n-k) + (2^k-1)c$

$n-k = 0 \Rightarrow n = k \quad k = n.$

$T(n) = 2^n * T(0) + (2^n-1)c$
$= 2^n * 1 + 2^n c - c$
$= 2^n(1+c) - c$
$\simeq 2^n$ // contents can be ignore
$O(2^n)$

Space Complexity : The space is proportional to the maximum depth of the recursion tree



Hence the space complexity of Fibonacci recursive is O(N)

(3) Write program which have complexity $n(\log n)$, n^3, $\log(\log n)$

→ Merge sort = $n\log n$

for time complexity = n^3
we can use three nested loops - $O(n^3)$

```
for(int i=0; i<n; i++)
{
    for(int j=0; j<n; j++)
    {
        for(int k=0; k<n; k++)
        {
            some O(1) expression
        }
    }
}
```

for time complexity - $\log(\log n)$
We can use the following func

```
for(int i=2; i<n; i= pow(i,c))
{
    //some O(1) expression
}
```

where k is constant

→ for time complexity $n\log n$
we can use the following function

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

Some O(1) expression
}
}

(4) Solve the following reccurence relation
$T(n) = T(n/4) + T(n/2)$

$T(\frac{n}{2}) \geq T(\frac{n}{4})$

Sol → $T(n) = 2T(n/2) + cn^2$
using masters method.
$T(n) = aT(n/b) + f(n)$

$a \geq 1$, $b > 1$, $c = \log_b a$ comparing n^c < f(n)
we get,
$c = \log_2 2 = 1$
$\Rightarrow f(n) > n^c$
$\Rightarrow T(n) = \Theta(f(n))$
$= \Theta(n^2)$

(5) What is the time complexity of the following function

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

DATE: / 20
PAGE No

Sol^n → for i = 1 → j = 1, 2, 3, 4 - - - n (run for n time)
for i = 2 → j = 1, 3, 5 - - - - (run for n/2 times)
for i = 3 → j = 1, 4, 7 - - - - - (run for n/3 times)

$T(n) = n + n/2 + n/3 + n/4 + - - - -$
$= n(1 + 1/2 + 1/3 + 1/4 + - - - -)$
$= n \int_1^n \frac{1}{x} \Rightarrow n \int_1^n \frac{dx}{x} = \log x$

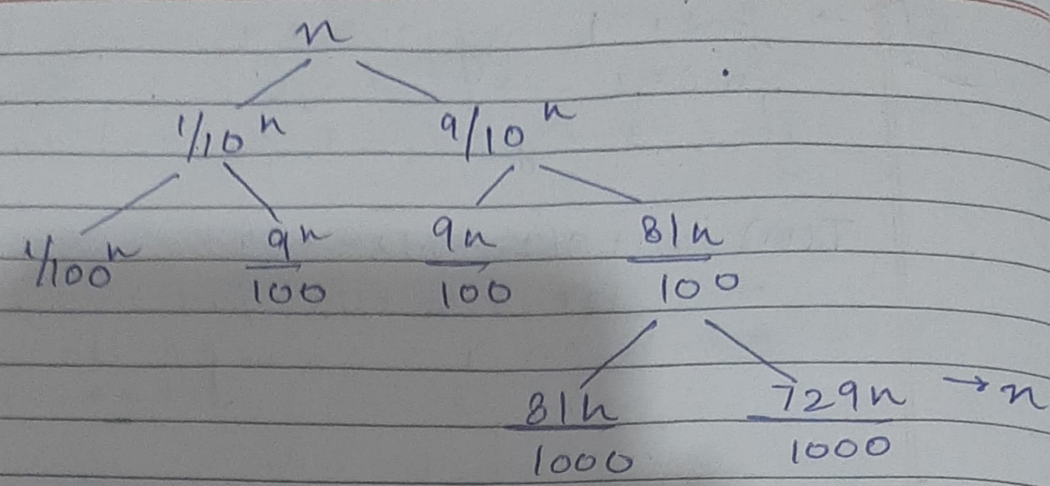
$n\log n$
∴ The time complexity of following func is $n\log n$

(6) What should be the time complexity of following function

```
for(int i = 2; i < n; i = pow(i, k))
{
    // Some O(1) expression or statements
}
```
where k is constants

Sol. for first iteration $i = 2$
2nd iteration $i = 2^{k}$
3rd iteration $i = (2^k)^k = 2^{k^2}$
|
|
|
nth iteration $i = 2^k$ loop ends at $2^k = n$
apply log $\log n = \log 2^{k^i} \Rightarrow k^i = \log n$
again apply log $\log(k^i) = \log n \Rightarrow i = \log_e(\log n)$

(7)



If we split in this manner

Recurrence Relation -

$$T(n) = T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right) + O(n)$$

when first branch is of size $9n/10$ & second one is $n/10$

Solving the above using recursion approach calculating values

at 1^{st} level, value $= n$

at 2^{nd} level, value $= \frac{9n}{10} + \frac{n}{10} = n$

Value remains same at all levels i.e. n

Time Complexity = Summation of values

$\Rightarrow O(n \times \log_{10/9} n)$ (upper bound)

$= \Omega(n \log_{10} n)$ (lower bound)

2) $\boxed{O(n \log n)}$

DATE : / 20
PAGE No

(8) Considering large value 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) $96 < \log_8 n < \log 2n < 5n < n(\log_6 n)$

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

$< 8^{2n}$