Name: Aditya Negi Section: D Roll No: 60

Date.
Page No.

Tutorial 2

Q1.

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

$i = \cancel{1+}\ i + 1$

$i = i + 2$

$i = i + 3$

$i = i + 4$

⋮

$i = i + x < n$

The value of i after x iteration

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

$x(x+1) < 2n$

$x^2 + x < 2n$

Time Complexity $= O(\sqrt{n})$

Q2. * $Fib(n) = fib(n-1) + fib(n-2)$ Let $T(0) = 1$

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

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

$T(n-2) = 2 * \{2T(n-4) + c\} + c$
$= 4T(n-4) + 3c$
$T(n-4) = \cancel{2*}\ 8T(n-6) + 7c$

$T(n) = 2^k \cdot T(n-2k) + (2^k - 1)c$

$n - 2k = 0$
$k = \frac{n}{2}$

$T(n) = 2^{n/2} \cdot (T(0)) + (2^{n/2} - 1)c$
$= (1+c) \times 2^{n/2} - c$

$T(n) \propto 2^{n/2}$ (Lower Bound)

$T(n-2) \simeq T(n-1)$
$T(n) = 2T(n-1) + c$ $c=4$
$T(n-1) = 4T(n-2) + 3c$

Date. ____________
Page No. ____________

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

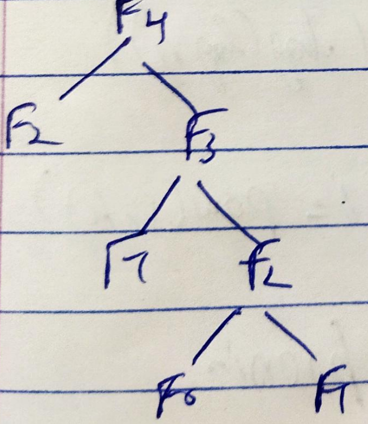
$$n - k = 0 \Rightarrow k = n$$

$$\Rightarrow T(n) = (1+c)\,2^n - c$$

$$T(n) \propto 2^n \quad \text{(upper Bound)}$$

Time Complexity $\Rightarrow O(2^n)$

* The space is proportional to the maximum depth of the recursion tree.



Hence the space complexity will be $O(n)$

Date.
Page No.

Q3 (i) Time Complexity = 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
```

(2) Time complexity = $O(n.\log n)$

```
for (int i=1; i<=n; i++)
  for (int j=1; j<n; j=j*2)
    Some O(1) expression
```

(3) Time Complexity = $O(\log(\log n))$

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

4. $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + cn^2$

Ignoring lower order term.

$T(n) = T\left(\frac{n}{2}\right) + cn^2$

Using master Theorem

$a = 1, \quad b = 2, \quad f(n) = n^2$

$c = \log_b a = \log_2 1 = 0$

$\boxed{0 < n^2}$ True

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

5.

i	j
1	n
2	$n/2$
3	$n/3$
4	$n/4$
:	
:	

Time complexity will be sum of series

$S = \frac{n}{1} + \frac{n}{2} + \frac{n}{3} + \cdots$

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

Complexity $= n \times \sum_{i=1}^{n} \left(\frac{n}{i}\right)$

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

6. $2, 2^k, (2^k)^k, \ldots$

Let no. of terms be λ

Given last term is n

$$2^{k^{\lambda-1}} = n$$

$$k^{\lambda-1} \log 2 = \log n$$

$$k^{\lambda-1} = \log n$$

$$(\lambda-1) \log k = \log(\log n)$$

$$\boxed{\lambda = \log(\log n)}$$

Time complexity $T(n) = O(\log(\log n))$