

Tutorial Sheet - 2

Q17

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

i	j
0	1
1	2
3	3
6	4
10	5
15	6

Series = 0, 1, 3, 6, 10, 15, ...

$$n = 0 + 1 + 2 + 3 + \dots + k$$

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

$$n = \frac{k^2 + k}{2}$$

$$n \approx k^2$$

$$k \approx \sqrt{n}$$

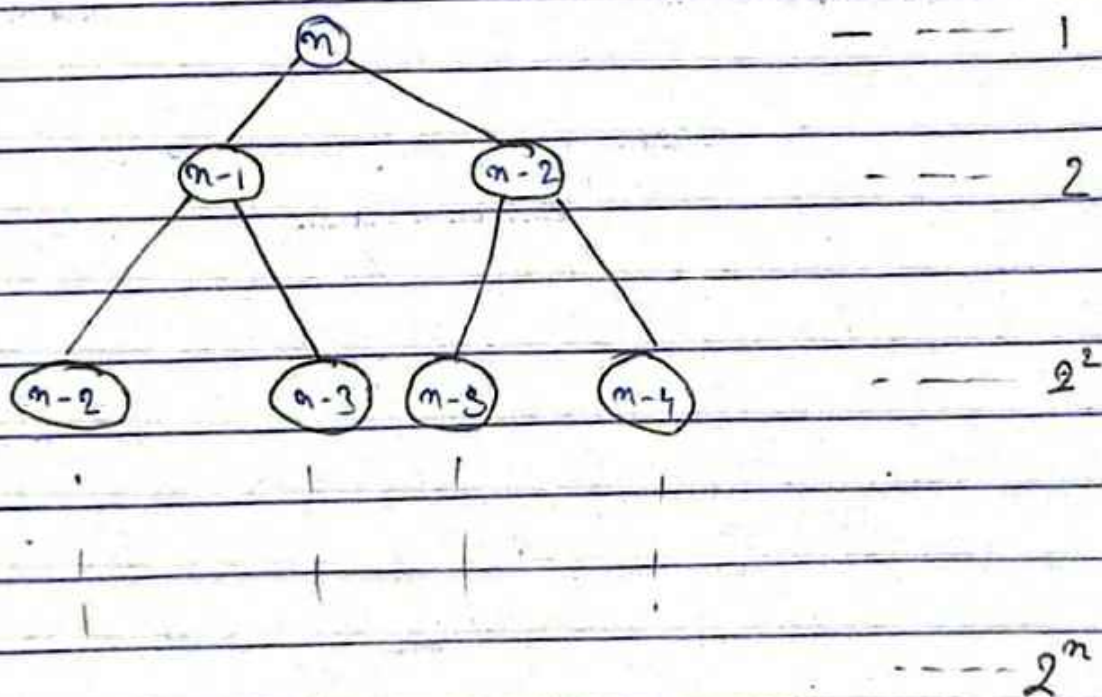
$$\text{Time Complexity} = O(\sqrt{n})$$

Q12 >

Recurrence relation for fibonacci series

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

Using Recurrence tree method



$$\text{Time Complexity} = 1 + 2 + 4 + \dots + 2^n$$

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

$$\text{or Time Complexity} = O(2^n)$$

Space Complexity: Space complexity of fibonacci series using recursion is proportional to height of recurrence tree.

So, Space Complexity = $O(n)$

Q3) Write code for complexity

(i) $n \log n$

```
for (i to n)  
{
```

```
    for (j=1; j<=n; j*=2)
```

```
         $O(1)$  statements
```

```
}
```

(ii) n^3

```
for (i to n)
```

```
    for (j to n)
```

```
        for (k to n)
```

```
             $O(1)$  statements
```

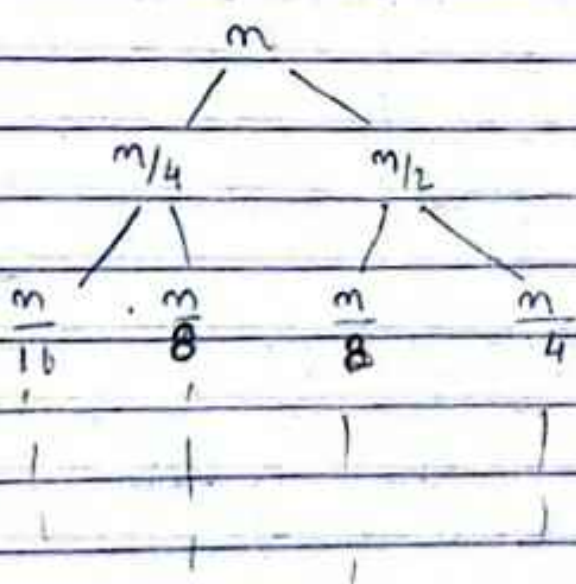
(iii) $\log(\log n)$

```
int i = n;
while (i > 0)
{
```

$i = \sqrt{i};$

}

Q4) $T(n) = T(n/4) + T(n/2) + cn^2$



$= cn^2$

$= \frac{cn^2}{16} + \frac{cn^2}{4} = \frac{5cn^2}{16}$

$= \frac{cn^2}{256} + \frac{cn^2}{64} + \frac{cn^2}{64} + \frac{cn^2}{16} =$

$\frac{25}{256} cn^2$

So, $T(n) = c \left(n^2 + \frac{5n^2}{16} + \frac{25n^2}{256} + \dots \right)$

$$\text{here } r = \frac{5}{16} \quad \text{So, } S_{\infty} = \frac{1}{1-r}$$

$$T(n) = Cn^2 \left(1 + \frac{5}{16} + \frac{25}{256} + \dots \right)$$

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

$$= Cn^2 \cdot \frac{16}{11}$$

$$\text{Time Complexity} = \Theta(n^2)$$

Q.5)

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

i	j	time
1	1 to n	n-1
2	1 to n	(n-1)/2
3	1 to n	(n-1)/3
⋮	⋮	⋮
n	1 → n	(n-1)/n
		$n \log n$

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

Q6 >

```
for (int i=2; i<=n; i=pow(i,k))
```

```
{
```

// some $O(1)$ expressions or statements

```
}
```

$$i = 2, 2^k, 2^{k^2}, 2^{k^3}, \dots, 2^{k^n}$$

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

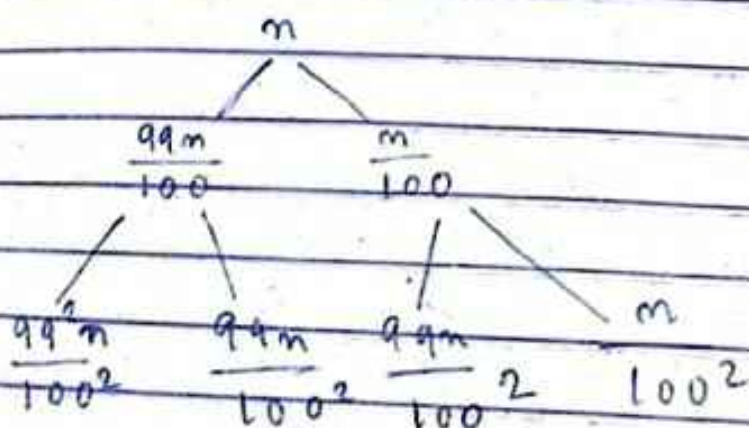
$$\log n = k^n \log 2$$

$$\frac{\log \log n}{\log 2} = \alpha \log K$$

$$\alpha = \frac{\log \log n}{\log 2 + \log K}$$

$$\text{Time Complexity} = O(\log \log n)$$

Q77



Taking longer branch that is $\frac{99n}{100}$

$$\text{Time Complexity} = \log_{\frac{99}{100}} n$$

$$\approx \log n$$

$$m = \left(\frac{99}{100} \right)^k$$

$$\text{or } k = \log \left(\frac{100 m}{99} \right)$$

$$T(m) = m \left(\log \frac{100}{99} \right)^m / 100$$

$$= O(n \log_{99} n)$$

Q8) Increasing order of rate of growth.

@ $n, n!, \log n, \log \log n, \text{root}(n), \log(n!),$

$n \log n, \log^2 n, \log 2^n, 2^{2^n}, 4^n, n^2, 100$

$100 < \log \log n < \log n < \sqrt{n} (\text{root}(n)) < n < n \log n < n^2 < 2^n < 2^{2^n} < 4^n < n!$

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

$$(c) \quad 96 < \log_{16} n < \log_2 n < 5n < n \log_8(n) < n \log_{16} n < 8n! < 7n^3 \\ < \log n! < 8^{2^n} < n!$$