Name- Gaurav Bhandari Sec- F Roll no.- 58
U. Roll no.- 2016747 ID- 20021182

Tutorial : 02

1.

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

j	i
1	1
2	3
3	6
4	10
⋮	⋮
k	$1+2+3+\dots k = \frac{k(k+1)}{2}$

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

OR $$k^2 < n$$

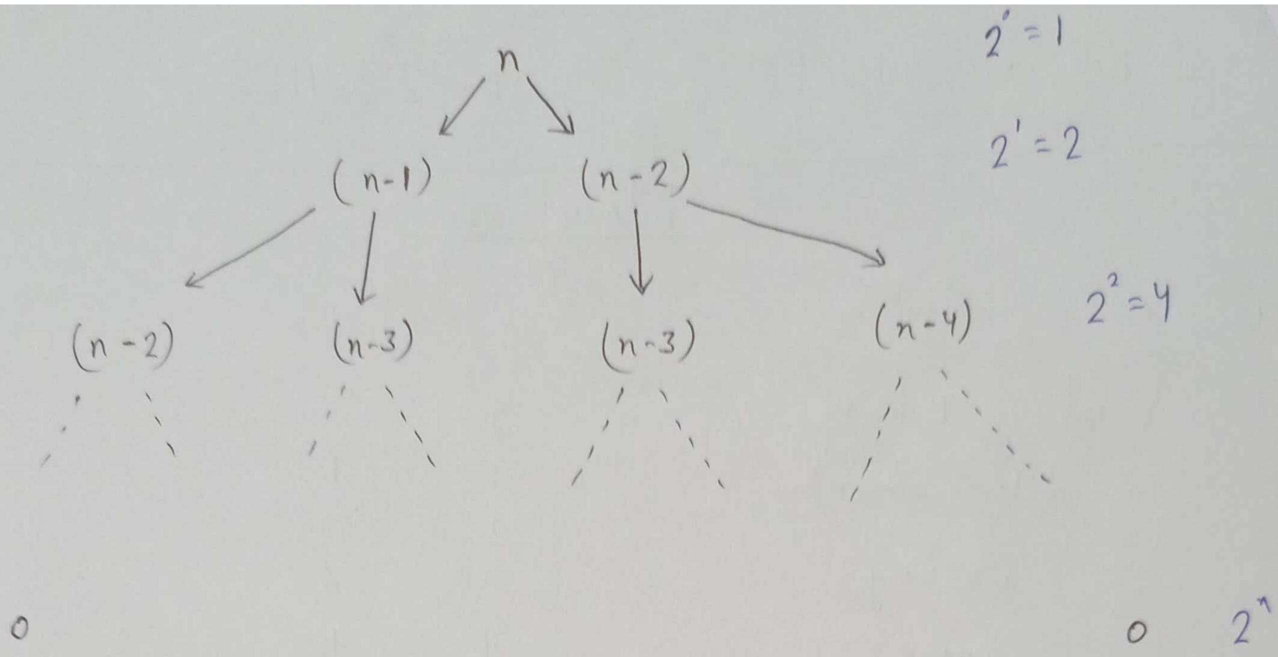
$$k < \sqrt{n}$$

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

2. Fibbonacci Series with recursion.

```
int fibbo (int n)
{ if (n==1)
    return 1;
  else if (n==0)
    return 0;
  else
    return fibbo(n-1) + fibbo(n-2);
}
```

3.



$\therefore$ Time Complexity of function is $O(2^n)$

Space Complexity $O(1)$

$\because$ no extra space

Using Recursion Function

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

for $n = 0$

$T(0) = 1$

for $n = 1$

$T(1) = 1$

$\because$ only one condition

no recursive calls.

3. (i) Program with Time Complexity $n(\log n)$

```
int main()
{ int n, count = 0;
  cin >> n;
  for (int i= 0 ; i<n ; i++)
  {  for (int j=0; j<n ; ~~j++)~~ j*=2)
     {  count ++;
     }
     cout << count << endl;
  }
```

(ii) Program with Time Complexity n^3

```
int main()
{  int n, count = 0;
   cin >> n;
   for( int i = 0 ; i<n ; i++)
   {
     for (int j = 0 ; j<n ; j+=2)
     {
       for (int k = 0 ; k<n ; k++)
       {
         count ++;
       }
     }
   }
   cout << count << endl;
}
```

iii) Program with Time Complexity $\log(\log n)$

```
int main()
{ int n, count=0; p=0;
  cin >> n;
  for (int i=0; i<n; i++) i*=2)
    p++;
  for (int j=1; j<p; j*=2)
    cout << j;
}
```

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

$T(n/4)$ will be ignored as it is of lower order.

$\Rightarrow T(n) = T(n/2) + cn^2$ —— ①

put $n = n/2$ in eq ①

$T(n/2) = T(n/4) + c\frac{n^2}{4}$

Put the value of $T(n/2)$ in eq ①

$T(n) = T(n/4) + \frac{cn^2}{4} + cn^2$ —— ②

Put $n = \frac{n}{4}$ in eq ①

$T(n/4) = T(n/8) + \frac{cn^2}{16}$

Put the value of $T(n/4)$ in eq ②

$T(n) = T(n/8) + \frac{cn^2}{16} + \frac{cn^2}{4} + cn^2$ —— ③

from eq 1, 2 & 3

$$T(n) = T(n/2^k) + cn^2\left(1 + \frac{1}{4} + \frac{1}{16} + \dots\dots \frac{1}{4^k}\right)$$

put

$$\frac{n}{2^k} = 1$$

$$2^k = n$$

substituting,

$$\Rightarrow T(1) + cn^2\frac{\left[1 \times \left(1 - \left(\frac{1}{4}\right)^k\right)\right]}{1 - \frac{1}{4}}$$

$$1 + cn^2\left[\frac{4}{3} - \frac{4}{3} \times \left(\frac{1}{4}\right)^k\right]$$

$$1 + cn^2 \times \frac{4}{3} - cn^2 \times \frac{4}{3} \times \left(\frac{1}{2^2}\right)^k$$

$$1 + cn^2 \times \frac{4}{3} - c\cancel{n^2} \times \frac{4}{3} \times \frac{1}{\cancel{n^2}}$$

$$1 + cn^2 \times \frac{4}{3} - c \times \frac{4}{3}$$

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

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	
1	1, 2, 3	n times
2	$\underbrace{1, 3, 5, \ldots\ldots n}$	n/2 times

$= 1 + (k-1)/2 = n$

$k = \dfrac{n+1}{2}$

i	j	
3	1, 4, 7, ...	n/3 times
⋮		
n	1, ...	n/n times

Total Time complexity $\Rightarrow n + \dfrac{n}{2} + \dfrac{n}{3} + \cdots\cdots + \dfrac{n}{n}$

$\Rightarrow n\left[1 + \dfrac{1}{2} + \dfrac{1}{3} + \cdots\cdots + \dfrac{1}{n}\right]$

$\Rightarrow n \sum_{k=1}^{n} \dfrac{1}{k}$

$\Rightarrow n \cdot \log(n)$

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

6.
```
for ( int i=2; i<n ; i= pow (i, k))
{
    // some O(1) expressions
}
```

$i \rightarrow 2^k, 2^{2^k}, \ldots\ldots \quad 2^{k^i}$

for the termination of loop

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

Taking log,

$$k^i \log_2 2 = \log_2 n$$

$$k^i = \log_2 n$$

again taking log.

$$i \log_k k = \log_k \log_2 n$$

$$\Rightarrow \quad i = \log_k \log_2 n$$

Time complexity $\Rightarrow O\{\log_k \log_2 n\}$

8\. a) $100 < \log\log n < \log n < \log^2 n < \sqrt{n} < n < \log n! < n\log n$
$< n^2 < 2^n < n! < 4^n < 2^{2^n}$

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

c) $96 < \log n < \log_8 n < \log_2 n < 5n < \log n! < n\log_6 n < n\log_2 n$
$< 8n^2 < 7n^3 < n! < 8^{2^n}$