Tutorial-2

Q1-

Initially, $i=0, j=1$.

	Value at i	Value at j	Run
1st time	$i=1$	$j=2$	1
2nd time	$i=3$	$j=3$	1
3rd time	$i=6$	$j=4$	1
4th time	$i=10$	$j=5$	1
5th time	$i=15$	$j=6$	1
6th time	$i=21$	$j=7$	1
kth time	$i=n$	$j=k$	1

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

$$K^2+K=2n$$

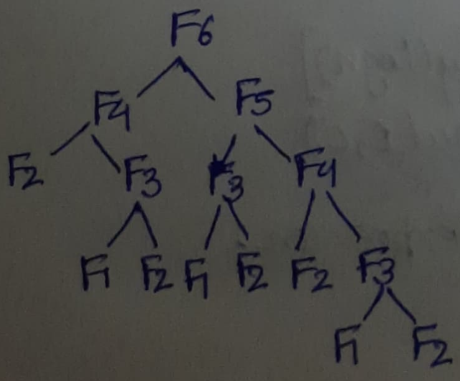
$$K^2<n$$

$$K<sqrt(n)$$

Time complexity → $O(sqrt(n))$

Q2-

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



SPACE COMPLEXITY:
The space is proportional to the max depth of recursion tree.

$T = O(2^n)$

Space complexity $= O(N)$

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

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

$T(n-2) = 2*(2T(n-2) + C$

$= 2^2 T(n-2) + C$

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

$= 8T(n-3) + 7C$

$= 2^K T(n-K) + (2^K - 1)C$

* Let, $n-K = 0$

$n = K$

$= 2^K T(0) + (2^n - 1)C$

$2^n 1 + 2^n C - C$

$2^n(1+C) - C$

$\simeq 2^n$

Q3- Mergesort- nlogn

1)- we can use three 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.
       }
    }
 }
```

2)- For time Complexity-[log(logn)]

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

3)- For time Complexity-$n\log n$
we can use the following function:

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

Q4- $T(n/2) \geq T(n/4)$

$T(n) = 2T(n/2) + Cn^2$

using Master's method

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$c = \log_2^2 = 1$

$f(n) > n^c$

$T(n) = \theta f(n)$

$\theta(n^2)$

Q5- for $i=1 \rightarrow j = 1, 2, 3, 4, ---n$ (Run for n)

for $i=2 \rightarrow j = 1, 3, 5 -----$ (Run for $n/2$)

for $i=3 \longrightarrow j = 1, 4, 7, ---n$ (Run for $n/3$ times)

$T(n) = n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + ----$

$= n\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + ---\right)$

$= n \int_1^n \frac{1}{x} = n \int_1^n \frac{dx}{x}$

$= n[\log n]_1^n$

$T(n) = \theta(n\log n)$

Q6- For first iteration, $i = 2$

2nd iteration, $i = 2^{k}$

3rd iteration, $i = (2^{k})^{k}$

|

|

|

nth iteration, $i = (2^{k})^{i}$

Loop ends at $2^{k^{i}} = n$

Apply log:

$\log n = \log_{2}^{k^{i}}$

$k^{i} = \log n$

Applying log again:

$(\log k^{i}) = \log n$

$i = \log(\log n)$

$T(n) = \Theta(\log(\log n))$

Q8-

a)- $100 < \log\log n < \log n < \log^{2}_{n} < \sqrt{n} < n < \log n < n\log n$
$< n^{2} < 2^{n} < 4^{n} < n! < 2^{2n}$

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

9- $96 < \log_{2} n = \log_{e} n < 5n < n\log_{2} n = n\log_{e} n < 8n^{2} < 7n^{3}$
$< 8^{2n} < n!$