Design and Analysis of Algorithm

Tutorial - 2

Ans-1)

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

$j=1 \quad i=0+1$

$j=2 \quad i=0+1+2$

$j=3 \quad i=0+1+2+3$

⋮

terminate at $i>=n$

$0+1+2+3+\cdots+k>=n$

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

$k^2 >= n$

$\boxed{k=\sqrt{n}}$

$= O(\sqrt{n})$

Ans-2) Recurrance Relation for fibonacci series.

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

and $T(0)=T(1)=1$

$\Rightarrow$ if $T(n-1) \approx T(n-2)$

(Lower Bound) $T(n) = 2T(n-2)$

$= 2\times 2(T(n-4))$

$= 4T(n-4)$

$= 8T(n-6)$

$= 16T(n-8)$

⋮

$T(n) = 2^k T(n-2k)$

$n-2k=0$

$\boxed{n=2k}$ $\boxed{k=\frac{n}{2}}$

$$T(n) = 2^{n/2}T(0)$$
$$= 2^{n/2}$$
$$T(n) = \Omega(2^{n/2})$$

$\Rightarrow$ if $T(n-2) \approx T(n-1)$

(Upper bound) $T(n) = 2T(n-1)$
$$= 2(2T(n-2)) = 4T(n-2)$$
$$= 4(2T(n-3)) = 8T(n-3)$$
$$= 2^k T(n-k)$$

$$n-k=0$$
$$\boxed{n=k}$$

$$T(n) = 2^k \times T(0) = 2^n$$
$$= T(n) = O(2^n)$$

Ans-3) $O(n\log n) \Rightarrow$
```
for(int i=0 ; i<n ; i++){
    for(int j=1 ; j<n ; j=j*2){
        // O(1) statement
    }
}
```

$O(n^3) \Rightarrow$
```
for(i=1 to n){
    for(j=1 to n){
        for(k=1 to n){
            // some O(1) statement
        }
    }
}
```

$O(\log(\log n)) \Rightarrow$
```
for(i=1; i<=n; i=i*2){
    for(j=1; j<=n; j=j*2){
        // some O(1)
    }
}
```

Ans - 4)

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

Let assume

$$T(n/2) >= T(n/4)$$

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

applying master's theorem $\left[T(n) = aT\left(\frac{n}{b}\right) + f(n)\right]$

$a = 2$, $b = 2$, $f(n) = n^2$

$$c = \log_b a = \log_2 2 = 1$$

$$n^c = n.$$

compare n^c and $f(n) = n^2$

$$f(n) > n^c$$

so, $T(n) = \theta(n^2)$.

Ans - 5)

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

i	j		
1	1, 2, 3, 4 -- n	$\longrightarrow$ n time.	
2	1, 3, 5, 7, -- k	$\longrightarrow k > \frac{n}{2}$	$\rightarrow$ n times.
3	1, 4, 7, 10, -- k	$\longrightarrow k > \frac{n}{3}$	$\rightarrow$ n times.
⋮			
n	-- --	$\longrightarrow$ ntimes.	

So, total complexity $= O(n^2 + n^2 + \dots)$

$$= O(n^2)$$

Ans-6)

```
for(int i=2; i<=n; i=Pow(i,k)) {
    // some O(1)
}
```

k is constant.

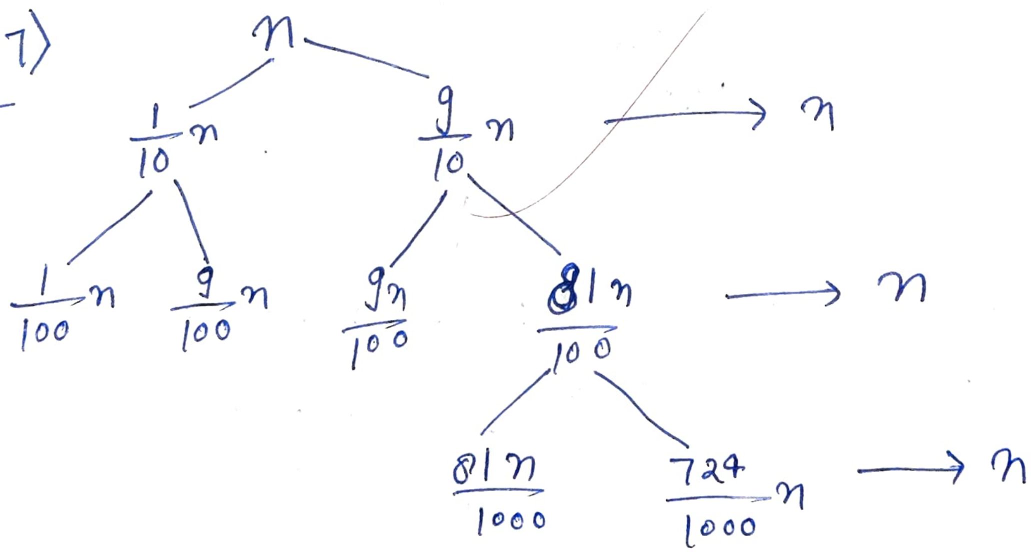
i
2
2^k
2^{k^2}
2^{k^3}
⋮
2^{k^m}

termination at

$$i > n$$
$$2^{k^m} > n$$
$$k^m \log_2 2 > \log n$$
$$k^m > \log n$$
$$m \log k > \log(\log n)$$
$$m > \frac{\log(\log n)}{\log k}$$

$$\boxed{T.C. = O(\log(\log n))}$$

Ans-7)



If we split it 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 $9n/10$ and second one is $n/10$ showing the above using recursion tree approach.

Calculating values.

At level-1 $= n$

At level-2 $= \frac{9n}{10} + \frac{n}{10} = n$

Value remains same at all levels i.e. n

T. C. = Summation of values.

$= O(n \log n)$ [Upper bound]

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

$\Rightarrow O(n \log n)$

Ans - 8)

(a) $100 < \log n < \sqrt{n} < n < \log(\log n) < n \log n < \log n!$

$< n! < n^2 < \log^2 n < 2^n < 2^{2n} < 4^n$

~~(b) $1 < \sqrt{\log n} < \log n < 2\log n < \log 2N < N < 2N <$~~

~~$4N < \log(\log N) <$~~

(b) $1 < \log(\log n) < \sqrt{\log n} < \log n < 2\log n < \log(2n) < n$

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

(c) $96 < \log_8 n < n \log_6 n < \log_2 n < n \log_2 n < \log(n!) < 5n <$

$8n^2 < 7n^3 < n! < 8^{2n}$