DAA

Tutorial - 2

Name : Karan Maurya

Class Roll no : 11

Sec : F

Ans. 1.

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

i	j	steps
0	1	1
1	2	2
3	3	3
6	4	4
10	5	5
15	6	6
:	:	:
$i+(j-1)$	j	K

let, $i+(j-1) = n$

$i+(K-1) = n$

$K = n - i + 1$

as $K \propto n$

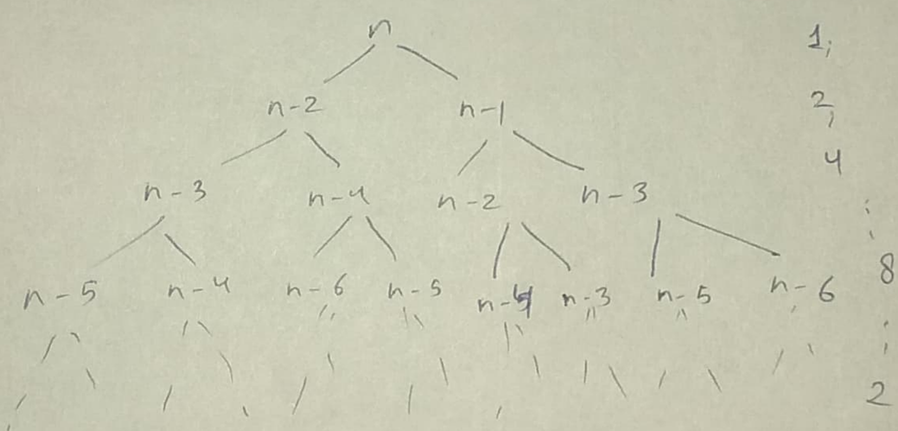
so, $\boxed{T.C = O(n)}$

Ans. 2.

```
int fib(int n)          - T(n)
{
    if (n <= 1)             T(1)
        return n;
    return  fib(n-1) + fib(n-2);
}                T(n-1)      T(n-2)
```

recurrence relation

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



$$T(n) = 1 + 2 + 4 + \ldots + 2^n$$

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

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

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

Space complexity of fibonaci series is O(1) as space required is proportional to maximum depth of the recursive tree, because, that is maximum no. of element that can be present in the implicit function call stack.

Ans. 3. for $n(\log n)$

```
int sum = 0;
```

(i).

```
for(i=0; i<n; i++)
{
    for( j=0; j<n; j*=2)
        sum = i+j;
}
```

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

(ii).

```
int sum = 0
for( i=0; i<n; i++)
{
    for( j=0; j<n; j++)
    {
        for( k=0; k<n; k++)
            sum = i+j+k;
    }
}
```

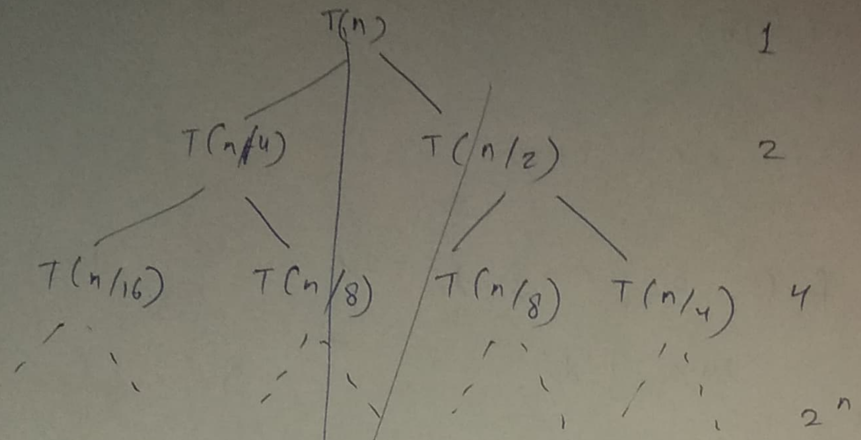
$$\boxed{T.C = O(n^3)}$$

(iii).

```
for( int i=0; i<n; i*=2)
{
    for( int j=0; j<n; j*=2)
    {
        sum = i+j;
    }
}
```

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

Ans. 4



$$T(n) = 1 + 2 + 4 + \dots + 2^n + \log n \; cn^2$$

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

$$T(n) = 2^{n+1} - 1 + cn^2 \log n$$

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

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

$$\therefore \; T(n) = T(\alpha n) + T(\beta n) + f(n)$$

$$\alpha = \frac{1}{4}, \quad \beta = \frac{1}{2}, \quad f(n) = cn^2$$

as, $\alpha + \beta = 0.75 < 1$

so, $T(n) = O(f(n))$

$$T(n) = O(cn^2)$$

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

Ans. 5

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

i	j	times
1	1, 2, 3, ..., n	n
2	1, 3, 5, 7, ..., 2n-1	$\frac{n}{2}$
3	1, 4, 7, ..., 3n-2	$\frac{n}{3}$
⋮		
n	1, 1+n	1

So, $T.C = n + n/2 + n/3 + \cdots + 1$

$T.C = n\left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots \frac{1}{n}\right) + 1$

$\boxed{T.C \approx O(n^2)}$

Ans 6

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

i

2

2^k

$(2^k)^k = 2^{k^2}$

2^{k^3}

2^{k^4}

$\vdots$

2^{k^m}

let, $2^{k^m} = n$

$\log 2^{k^m} = \log n$

$k^m = \log n$

$m \log k = \log(\log n)$

$$m = \frac{\log(\log n)}{\log k}$$

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

Ans. 7.

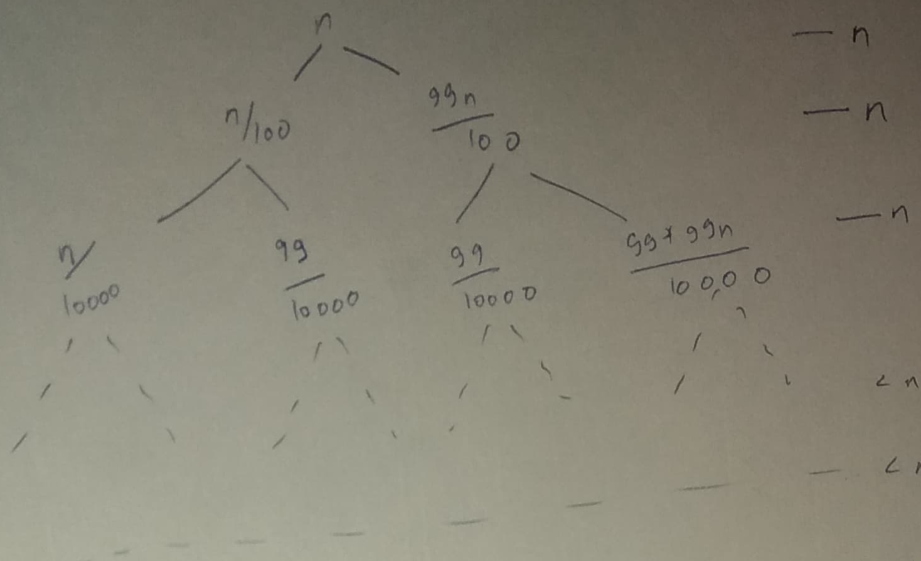
$$T(n) = T\left(\frac{99}{100} n\right) + T\left(\frac{1}{100} n\right) + kn$$

when quick sort divides array into 99 : 1

$$T(n) = T(\alpha n) + T(\beta n) + f(n)$$

$$\alpha + \beta = \frac{99}{100} + \frac{1}{100} = 1$$

so, $\boxed{T(n) = \theta(n \log n)}$



height of tree is log n & at each level time complexity is approximated to O(n)

So, Total time complexity : 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) < N log N < log N! < N! < N^2 < 2×2^N

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