Date ___/___/______

Saathi

TUTORIAL-2

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

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

$S = 0+1+3+6+\cdots + \sqrt{k}$ — (1)

also,

$S = 0+1+3+6+\cdots + \sqrt{k-1} + \sqrt{k}$ — (2)

from (1) & (2)

$0 = 1+2+3+4+\cdots + k - \sqrt{k}$

$\sqrt{k} = 1+2+3+4+\cdots\cdots + k$

$\sqrt{k} = \frac{1}{2}k(k+1)$ — (3)

for k iterations,

$1+2+3+6+\cdots + k < n$

Page No.

Date ___/___/___

From (3)

$k(k+1)/2 < n$

$\sqrt{\frac{k^2+k}{2}} < \sqrt{n}$

$k \cong O(\sqrt{n})$

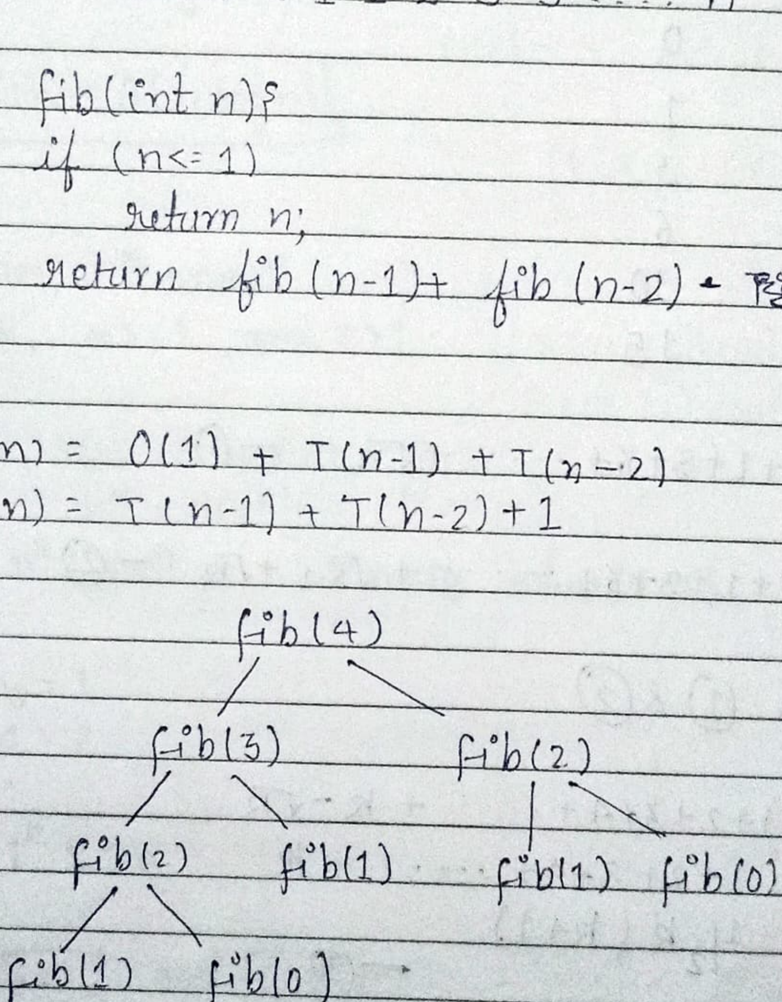
$\boxed{T(n) < O(\sqrt{n})}$

Q 2. fibonacci series: 0 1 1 2 3 5 .... n

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

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

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



$T(n) = 1+2+4+8+ \dots + n$

$S(n) = \frac{a(r^n - 1)}{r-1}$

$a = 1$, $n = \log n + 1$, $r = 2$

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

$S(n) = 2^n \cdot 2 - 1$

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

Space Complexity: As, the complexity of 1 call is O(1) and only variable that is stored is O(1).

∴ Total space complexity = O(1)

Sol. 3 (i) n(logn)

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

        int val = 0;
    }
}
```

Time complexity : O(n logn)

(ii) n^3

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

            cout << "Hello";
        }
    }
}
```

$\therefore$ Time Complexity $= O(n^3)$

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

```
for (int i=0; i<n; i= pow(i,2)){

        cout << "Hello";
}
```

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

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

Neglecting lower order term,

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

$a=1, \quad b=2 \quad . \quad c = \log_2(1)$

$n^c = n^0 = 1 < cn^2$

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

Saathi

Date ___/___/___

Sol. 5
```
int fn(int n){
    for (int i = 1; i<=n; i++){
        for(int j =1; j<n; j+=i){
            cout << "Hi";
        }
    }
}
```

for $i=1$, $j = 1+2+3+4+\cdots+n$
for $i=2$; $j = 1+3+5+7+\cdots+n$
for $i=3$; $j = 1+4+7+10+\cdots+n$

$T(n) = n + n/2 + n/3 + n/4 + \cdots + n/n$

$T(n) = n\left(1 + 1/2 + 1/3 + 1/4 + \cdots\ 1/n\right)$

$T(n) = n\int_1^n (1/x)$

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

Sol. 6 Time Complexity:

```
for(int i = 2; i<=n; i= pow(i, k)){
{ }
```
where, k is constant

for first iteration, $i = 2$
for second iteration, $i = 2^k$
for third iteration, $i = (2^k)^k = 2^{k^2}$
$\vdots$
for n iteration, $i = (2^k)^i = n$

Applying log.

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

Applying log again,

$$i = \log_k \log(n)$$

$$\boxed{T(n) = \log_k \log(n)}$$

Sol:7 Given, array in quick sort is divided into two parts of 99% and 1%.

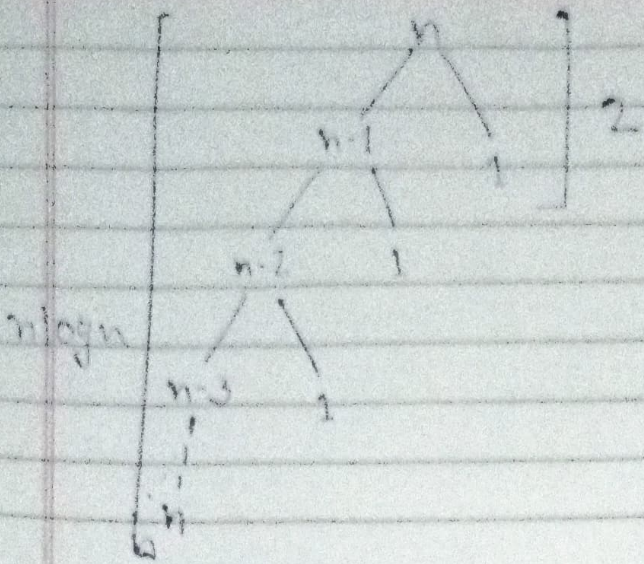
This means the pivot always split the list into 99:1. So, each node will be branch to node as, $1/100$ and $99/100$

So, the depth of such tree is by a factor of 10/9
So, the depth will be $\log_{(100/99)} N$

at each level of tree we have to go N values.

$$N * \log_{(100/99)} N = N * (\log_2 N) / \log(100/99)$$

$$\therefore \boxed{T(n) = N \log_2 N}$$



lowest weight = 2
heighest weight = n

$\boxed{\therefore \text{ difference} = n-2}, \forall n>1$

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

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

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