Tutorial-2

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

Soln.

when $j = 1$, $i = 1$

$j = 2$, $i = 3$

$j = 3$, $i = 6$

$j = 4$, $i = 10$

till n terms

$$\Rightarrow S = 1+3+6+10+15+ \cdots + t_n$$

$$S = \quad 1+3+6+10+15 \cdots + t_{n-1} + t_n$$

$$0 = 1+2+3+4+5+ \cdots + \cancel{(n-1)}(t_n - t_{n-1}) - t_n$$

$$t_n = 1+\underbrace{2+3+4+5+ \cdots \cdots (n-1)}_{\text{AP (Sum)}} \text{ terms} \quad ①$$

$\Rightarrow$ for AP,

$$\text{Sum} = \frac{n-1}{2}\left[2\times2 + (n-2)\right]$$

$$\text{Sum} = \frac{n-2}{2}\left[2+n\right]$$

from ①

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

$$\Rightarrow t_n = \frac{n^2+n}{2} = \frac{n^2}{2} + \frac{n}{2}$$

$\Rightarrow$ Sum of n terms $= \Sigma t_n$

$$= \frac{1}{2}\left[\Sigma n^2 + \Sigma n\right]$$

$$= \frac{1}{2}\left[\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right]$$

$$= \frac{n}{2} \cdot \frac{(n+1)}{2}\left[\frac{2n+1}{3} + 1\right]$$

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

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

(2) Recurrance relation for recursive fibonacci-

$$T(n) = T(n-1) + T(n-2) \quad (\text{i}) \quad ; n > 1$$

$$T(n) = 1 \quad ; n = 0, \; n = 1$$

for lower bound

$$T(n-1) \sim T(n-2) \quad (\text{ii})$$

$\Rightarrow$ from (i) & (ii)

$$T(n) = 2T(n-2) \quad (\text{iii})$$

putting $n = n-2$ in (iii)

$$T(n-2) = 2T(n-4) \quad (\text{iv})$$

from (iii) & (iv)

$$T(n) = 2[2T(n-4)] = 2^2 T(n-4) \quad (\text{v})$$

$n = n-4$ in (iii)

$$T(n-4) = 2T(n-6) \quad (\text{vi})$$

from (v) & (vi)

$$T(n) = 2^3 T(n-6)$$

Similarly after k operations,

$$T(n) = 2^k T(n-2k) \quad (\text{vii})$$

if $n-2k=0$

$k = n/2$

from (VII)

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

$\boxed{T(n) = 2^{n/2}}$ for lower bound

Now for upper bound,

$T(n-2) \sim T(n-1)$ —(VIII)

$\Rightarrow T(n) = 2T(n-1)$ —(IX) [from (I) & (VIII)]

$n = n-1$ in (IX)

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

$\Rightarrow T(n) = 2^2\,T(n-2)$ —(X)

$n = n-2$ in (X)

$T(n-2) = 2\,T(n-3)$ —(XI)

from (X) & (XI)

$T(n) = 2^3\,T(n-3)$

$\Rightarrow$ after k operations,

$T(n) = 2^k\,T(n-k)$ —(XII)

if $n-k=0$

$k=n$

from (XII)

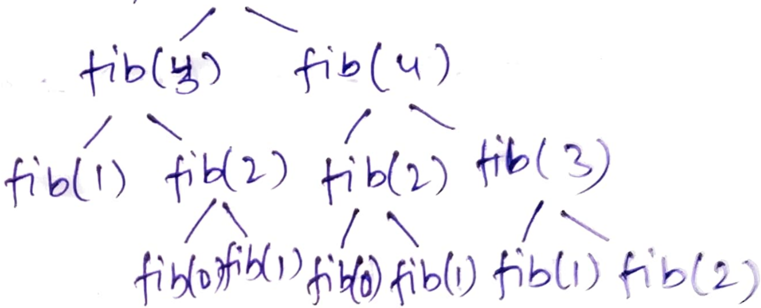
$\boxed{T(n) = 2^n\,T(0) = 2^n}$ for upper bound

$\because$ big oh represents upper bound of the func

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

for space complexity of recursive fibonacci series, it is proportional to the max depth of the recursion tree.

for eg: for n = 5, fib(5)



∵ max depth ≈ 5 which is n

∴ $\boxed{SC = O(n)}$

③ n(log n) :

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

n3

```
void fun (int n) {
    for(int i=0; i<n; i++){
        for( int j=0; j<n; j++){
            for(int k=0; k<n ; k++){
                cout << "Algorithm" <<endl;
            }
        }
    }
}
int main(){
    int n;
    cin >>n;
    fun (n);
}
```

log(log n):

```
int main(){
    int n;
    cin >>n;
    for (int i =2; i<=n ; i =i*i){
            cout <<"*" <<endl;
    }
}
```

(4) $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + cn^2$

$\because \quad T\left(\frac{n}{2}\right) > T\left(\frac{n}{4}\right)$

$\therefore \quad T(n) = 2T\left(\frac{n}{2}\right) + cn^2$

$a = 2, \quad b = 2, \quad f(n) = cn^2$

$\Rightarrow$ using master's method,

$c = \log_b a$

$c = \log_2 2 = 1$

$\Rightarrow \quad n^c = n^1$

$\because \quad f(n) > n^c$

$\therefore \quad \boxed{TC = \Theta(f(n)) = \Theta(n^2)}$

(5)

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

for $i=1$, j runs n times

for $i=2$, j runs $n/2$ times

for $i=3$, j runs $n/3$ times

⋮

for $i=n$, j runs n/n times

$$Sum = n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n}$$

$$Sum = n\left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}\right]$$

∵ the series is in HP, we will find its upper bound

$$1 + \underbrace{\frac{1}{2} + \frac{1}{3}}_{I} + \underbrace{\frac{1}{4} + \frac{1}{5} + \frac{1}{6}}_{II} + \cdots + \frac{1}{n}$$

upper bound of I $= \frac{1}{2}$

" " " II $= \frac{1}{4}$

replacing all the elements with their respective upper bounds

$$Sum = n\left[1 + \left(\frac{1}{2} + \frac{1}{2}\right) + \left(\frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4}\right) + \cdots + \frac{1}{n}\right]$$

$$= n[1 + 1 + 1 + \cdots + \log n \text{ times}]$$

$$Sum = n \log n$$

$$\Rightarrow \boxed{TC = O(n \log n)}$$

⑥

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

for $i = 2$, $\Rightarrow$ $2, 2^k, 2^{k^2}, 2^{k^3}, \ldots$

Since, the value of i is increasing exponentially by a constant value (k),

therefore, $\boxed{TC = O(\log(\log n))}$

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

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

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