- Problem set
- Strategies (Approaths)
  - Algorithms
- Analysis
- Code

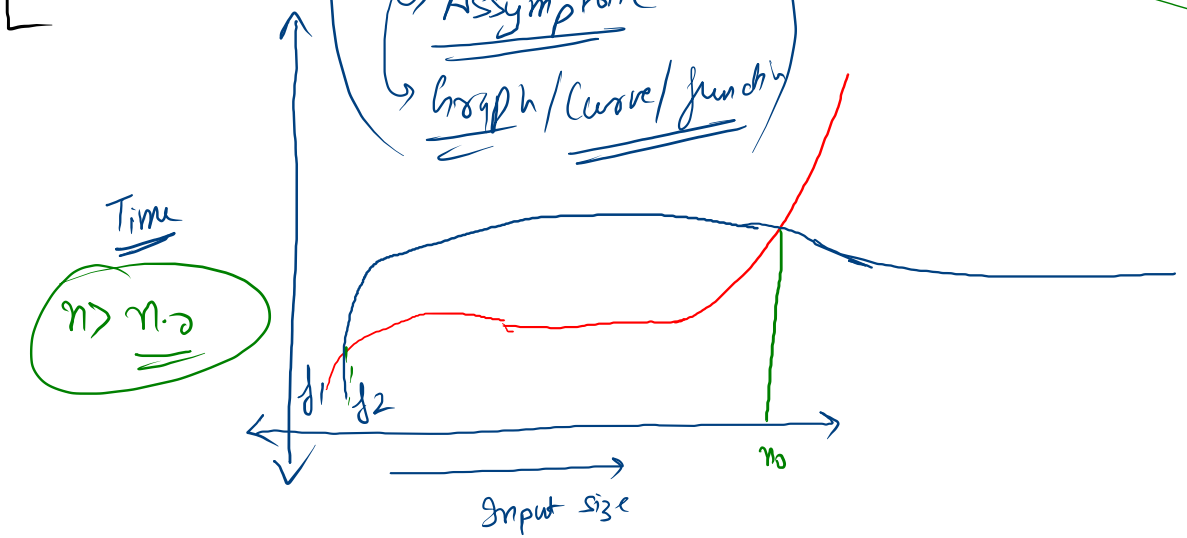
(Experimental) → Code → Execute → Time, space, CPU consumption

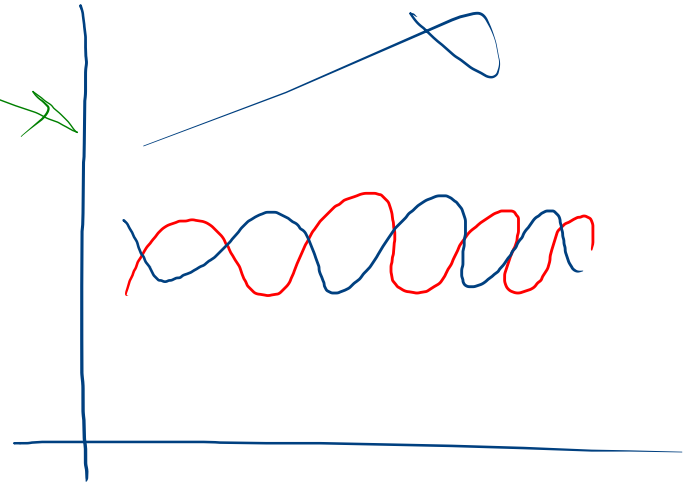
Problem
- → Language dependent
- → H/W

C/C++ → O.S.M/c

Java → JVM → O.S.

(Theoritical)
- Assymptotic
- Graph/Curve/Function





Asymptotic notations → Symbols

↳ O → W.C.

↳ Ω → B.C.

↳ Θ → Average

W.C

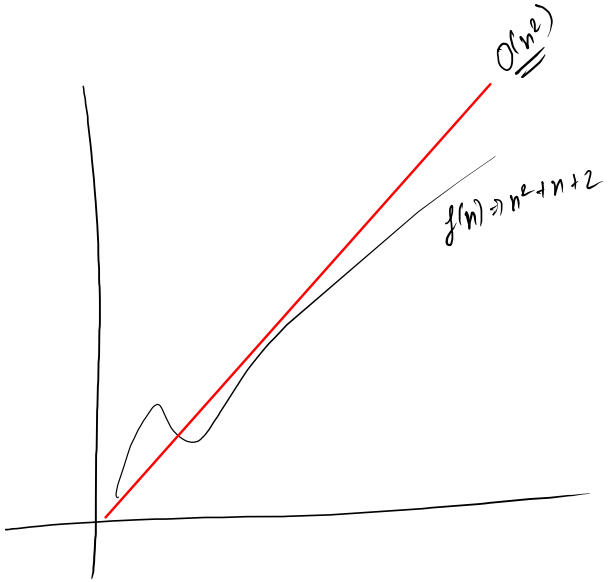
h(n)

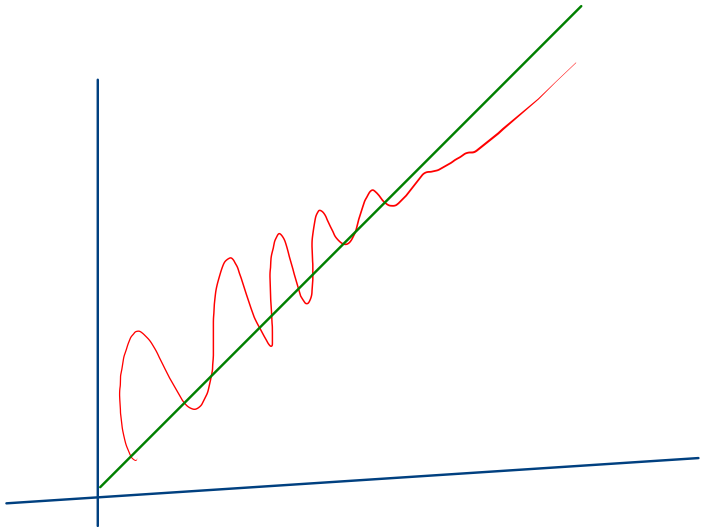
g(n) → B.C

Time

n_0

Input Size

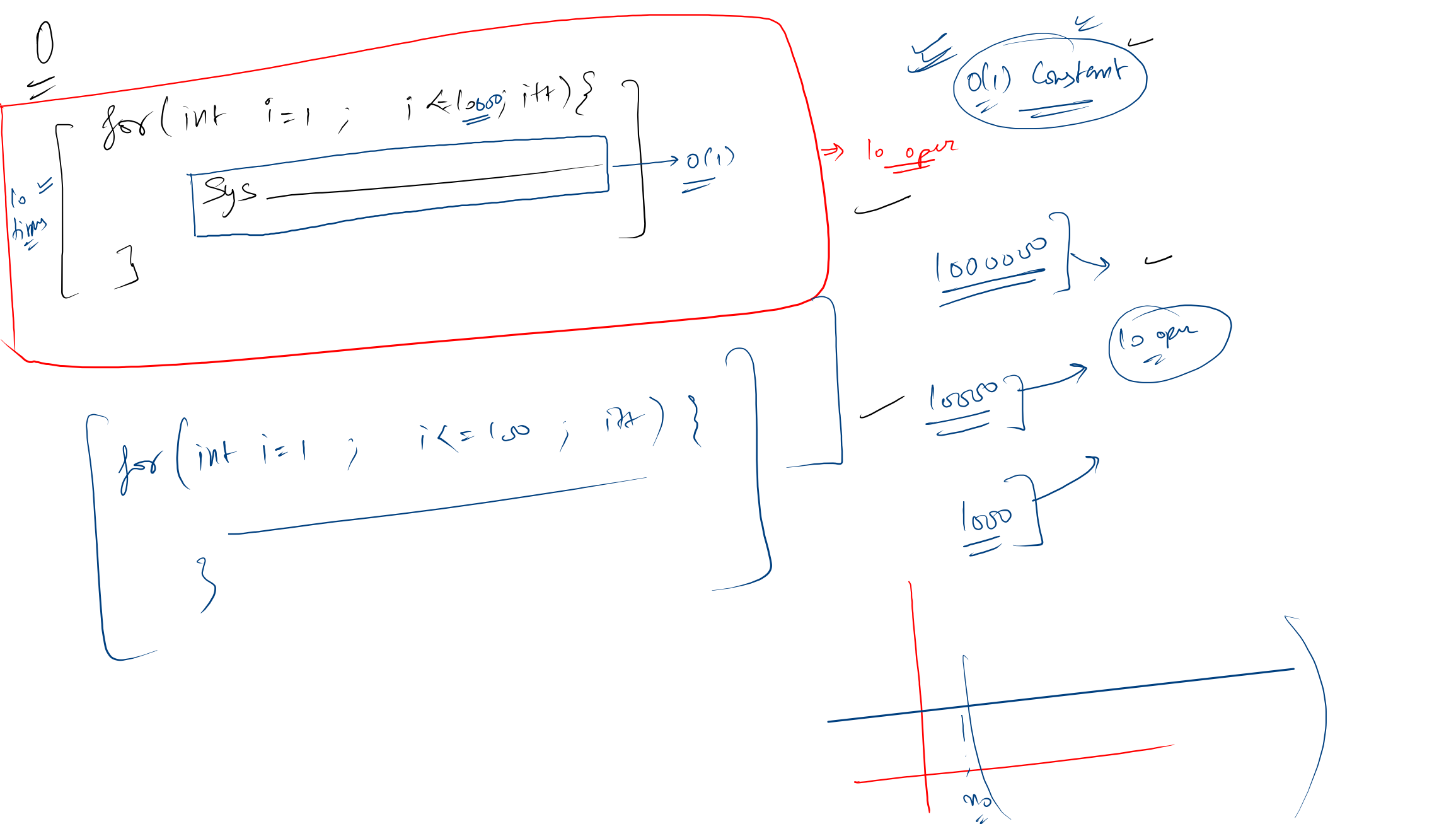




```
for(int i = 0; i < n; i = i * i) {
    System.out.println("___");
}
```

T.L.F.

infinite loop


O
for(int i=1 ; i<=1000; i++){
Sys
O(1)
10 times
}
O(1) Constant
10 opn
1000000
for(int i=1 ; i<=100 ; i++){
}
10000
10 opn
1000
n0

```
for (int i = 0 ; i <= n ; i = i + 3) {
    ( ------------------------------> O(1)
}
```

$a_n = a + (n-1)d$

K	1	2	3	4	5	.....	
i =	0	3	6	9	12	15	<= n
	↓	↓	↓	↓	↓	↓	
	1	1	1	1	1	1	

$0 + (K-1)3$

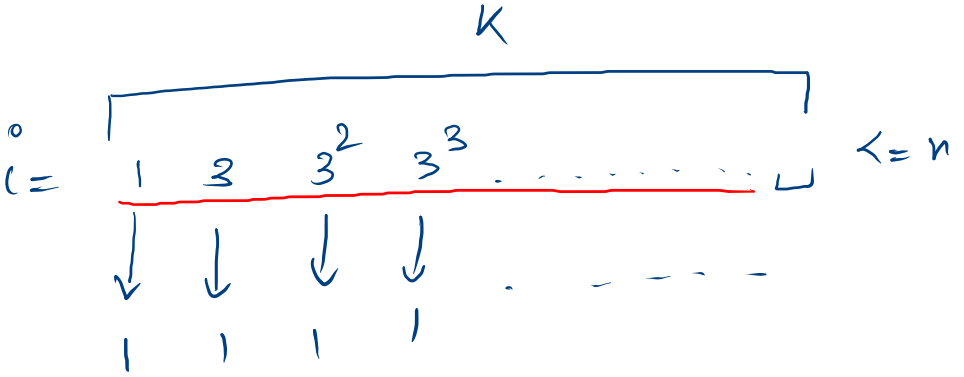
$(K-1)3 <= n$

$K - 1 <= n/3$

$K <= (n/3 + 1)$

$O(n)$

```
for(int i=1 ; i<=n ; i=i*3) {
        (1)
}
```



$\Rightarrow K$

$\Rightarrow (\log_3 n + 1)$

$= O(\log_3 n)$

$a_K = a\, r^{K-1}$

$\Rightarrow 1\,(3)^{K-1} <= n$

$K-1 <= \log_3 n$

$(K <= \log_3 n + 1)$

```
int s=0, i=1;
while ( true ) {
    s = s+i;
    if (s > n) {
        break;
    }
    i++;
}
```

i =	1	2	3	4	5	...	K
S =	1	3	6	10	15	...	$\frac{K(K+1)}{2}$ <= n
	↓	↓	↓	↓	↓		
	1	1	1	1	1		

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

T.O. => 1+1+1+1+.....+1

K times

K terms

① - T.O. => K

T.O. => $\sqrt{2}\sqrt{n}$

T.C. => $O(\sqrt{n})$

$K \cdot K < K \cdot (K+1) <= 2n$

$K^2 <= 2n$

$K <= \sqrt{2n}$

$K < \sqrt{2}\sqrt{n}$

n^2

n → input

```
int S = 1; i = 1;
while (true) {
    if (S <= n) {
        ——— O(1)
    } else {
        break;
    }
    i = i + 2
    S = S + i;
}
```

$a + (n-1)d$

$a + (k-1)d$

$1 + (k-1)2$

S =	1	4	9	16	25	. . . .			$k^2 <= n$
i =	1	3	5	7	9	. . . .			k
T =	1	2	3	4	5	. . . .			k
O ⇒	1	1	1	1	1	1	1	1	1

k times

$T.O. \Rightarrow 1+1+1+1+\dots+1$

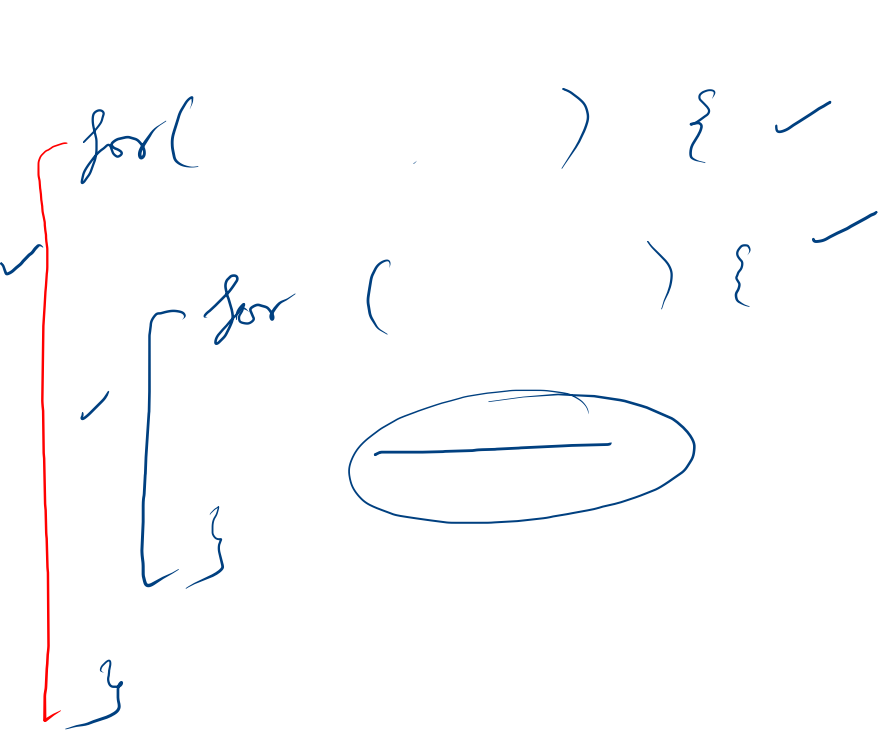
$T.O. \Rightarrow k$

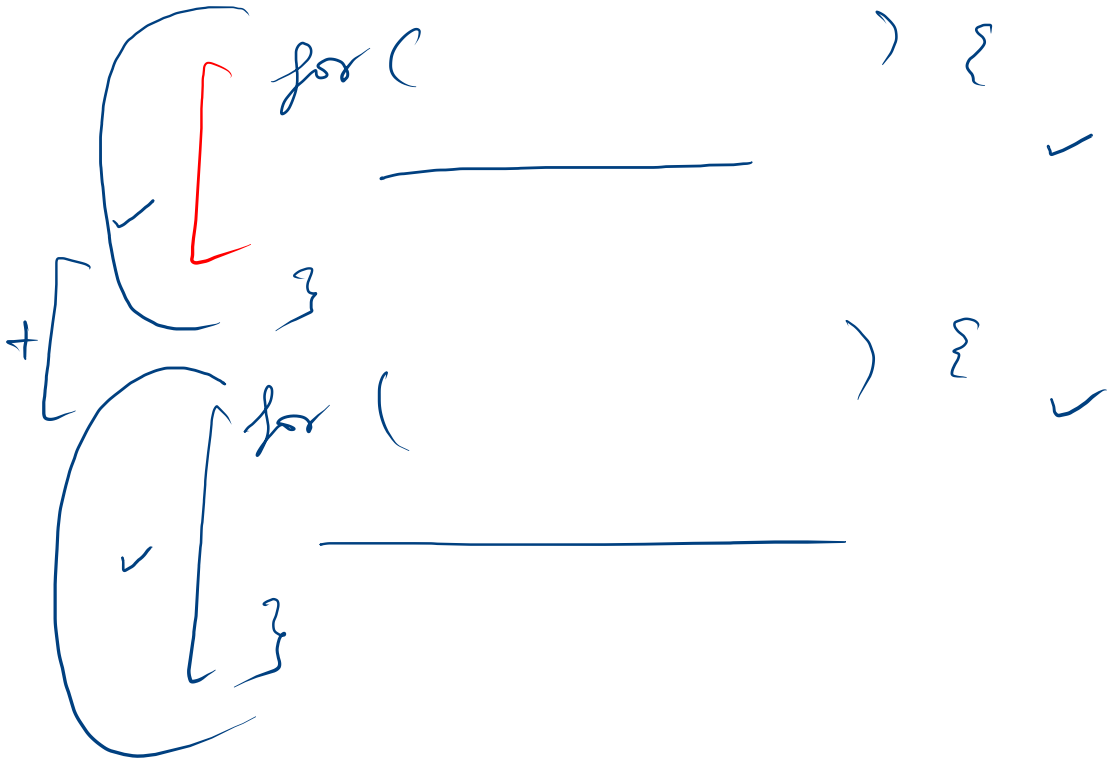
$k <= \sqrt{n}$

$T.O. <= \sqrt{n}$

$T.C. \Rightarrow O(\sqrt{n})$

$1+3+5+7+9+\dots \Rightarrow$


for(
for (


for (
for (

I

```
for (int i=0 ; i<n ; i++) {
    for(int j=0 ; j<n ; j++) {
        ——————— (O(1))
    }
}
```

T.O. => n^2

T.C. => $O(n^2)$

II

```
for (int i=0 ; i<n ; i++) {
    for(int j=0 ; j<=i ; j++) {
        ——————— (O(1))
    }
}
```

i	j	Ops
0	0	1
1	0,1	2
2	0,1,2	3
3	0,1,2,3	4
⋮	⋮	⋮
n-1	0,.....,n-1	n

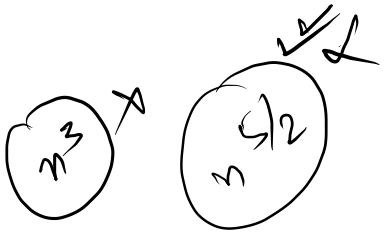
$1+2+3+4\ldots n \Rightarrow \frac{n(n+1)}{2}$

T.O. => $\frac{n(n+1)}{2}$

T.C. => $O(n^2)$

```
① for (i=0 ; i<=n ; i++) {
②     for (j=1 ; j*j<=n ; j++) {
          ____________
      }
③     for (k=1 ; k<=i ; k++){
          ____________
      }
  }
```

i	j	k	
1	$\sqrt{n}$	1	(1) $\Rightarrow \sqrt{n}+1$
2	$\sqrt{n}$	1,2	(2) $\Rightarrow \sqrt{n}+2$
3	$\sqrt{n}$	1,2,3	(3) $\Rightarrow \sqrt{n}+3$
⋮			
n	$\sqrt{n}$	1, ... n(n)	$\Rightarrow \sqrt{n}+n$



$$T.C. = n\sqrt{n} + \frac{n(n+1)}{2} \Rightarrow n\sqrt{n} + \frac{(n^2)+n}{2}$$

$$T.C. \Rightarrow O(n^2)$$

$n = 10$

i = 0 1 2 3 4 5 6 7 8 9 10

j = 0

```
for ( i=0, j=0; i <= n; i++ ) {
    if ( i == n ) {
        if ( j == n ) {
            break;
        }
        i = 0;
        j++;
    }
}
```

(O(1))

10 10

$T(0) \Rightarrow (n+1)^2$

$T(1) \Rightarrow O(n^2)$

$n+1$

j = 0	i = 0 1 2 3 4 ..... i (n)	[n+1]
j = 1	i = 0 1 2 3 4	n [n+1]
j = 2	i = 0	n [n+1]
j = 3	i = 0	n [n+1]
j = n-1	i = 0	n [n+1]
j = n	i = 0	n [n+1]

$k < n$

multiples $\longrightarrow O(n)$

$j = 0\ 3\ 6\ 9 \ldots <= n$

m

$k = 3$

$0 + (m-1)3 <= n$

```
for (i = 1 ; i <= k ; i++) {
    for (j = 0 ; j <= n ; j = j + k) {
        System.out.println(-) ;
    }
}
```

$(\frac{n}{k} + 1)$ times

$j = 0 \quad k \quad 2k \quad 3k \quad (m-1)k <= n$

m

$m = (\frac{n}{k} + 1)(\frac{n}{3} + 1)$

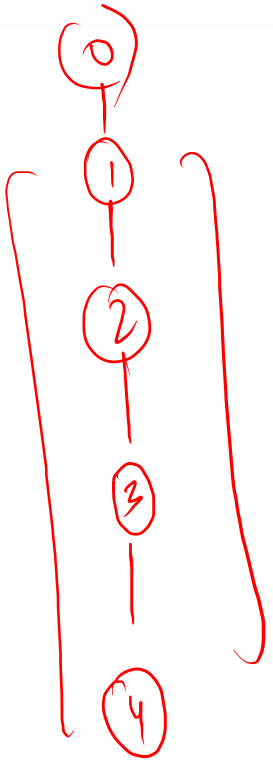
$T.O. \Rightarrow k[\frac{n}{k} + 1] \Rightarrow [n + k]$

$\max(n, k)$

$T.C. \Rightarrow O(n)$

$k \begin{bmatrix} i=1, & (n/k+1) \\ i=2, & (n/2+1) \\ \vdots \\ k \end{bmatrix}$

```java
public static void printIncreasing(int n){
    if(n == 0){
        return;
    }
    printIncreasing(n-1);
    System.out.println(n);
}
```



n terms

① Recurrence

$$T(n) \Rightarrow T(n-1) + 1$$

② Solving

$$T(n) = T(n-1) + 1$$
$$T(n-1) = T(n-2) + 1$$
$$T(n-2) = T(n-3) + 1$$
$$\vdots$$
$$T(1) = +1$$

$$T(n) = 1 + 1 + 1 + \cdots + 1 \Rightarrow n, \quad T.C. \Rightarrow O(n)$$

```java
public static int power(int x, int n){
    if(n == 0 || x == 1){
        return 1;
    }
    if(n == 1){
        return x;
    }
    if(x == 0){
        return 0;
    }
    int xPowNm1 = power(x,n-1);
    int xPowN = x * xPowNm1;

    return xPowN;
}
```

O(1)

O(1)

X

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

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

$n-1$

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

$T(1)$

T.C. => $1 + 1 + 1 + \dots + 1$ => $n-1$

$n-1$

T.C. => O(n)

```java
public static int powerFake(int x, int n){
    if(n == 0){
        return 1;
    }
    if(n%2 == 0){
        return powerFake(x,n/2) * powerFake(x,n/2);
    }else{
        return powerFake(x,n/2) * powerFake(x,n/2) * x;
    }
}
```

$$T(n) = 2T\left(\frac{n}{2}\right) + 1$$

$T(n) = 2T\left(\frac{n}{2}\right) + 1$

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

$4T\left(\frac{n}{4}\right) = 8T\left(\frac{n}{8}\right) + 4$

$8T\left(\frac{n}{8}\right) = 16T\left(\frac{n}{16}\right) + 8$

$2^k T(1) =$ 2^k

(k+1 terms)

①

T.O. $\Rightarrow$ $1 + 2 + 4 + 8 + \cdots$ (k+1 terms)

$n = n/2^0$

$n/2 = n/2^1$

$n/4 = n/2^2$

$n/8 = n/2^3$

$1 = n/2^k$

$1 = \frac{n}{2^k}$

$2^k = n$

taking log both sides

$k = \log_2 n$

T.O. $\Rightarrow$ $\frac{1 + 2 + 4 + 8 + \cdots}{\log_2 n + 1}$

$\Rightarrow 1\left[\frac{2^{\log_2 n + 1} - 1}{2 - 1}\right] = \left(\frac{(n \cdot 2) - 1}{1}\right)$

$\Rightarrow 2n + 1$, T.C. $\Rightarrow O(n)$

```java
public static int powerSmart(int x, int n){
    if(n == 0){
        return 1;
    }
    int xPowNb2 = powerSmart(x,n/2);
    if(n%2 == 0){
        return  xPowNb2 * xPowNb2;
    }else{
        return xPowNb2 * xPowNb2 * x;
    }
}
```

$\log_2 n + 1$

$T(n) \Rightarrow T(n/2) + 1$

$T(\frac{n}{2}) = T(\frac{n}{4}) + 1$

$T(\frac{n}{4}) = T(\frac{n}{8}) + 1$

$T(\frac{n}{8}) = T(\frac{n}{16}) + 1$

$\vdots$

$T(1) = +1$

$T(n) \Rightarrow 1 + 1 + 1 + \dots$

$T(n) \Rightarrow \log_2 n + 1$

$T.C. \Rightarrow O(\log_2 n)$

```
public static int fib(int n){
    if(n==0) return 0;
    if(n==1) return 1;

    return fib(n-1) + fib(n-2);
}
```

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

$$fib(n-1) > fib(n-2)$$

$$fib(n-1) + fib(n-1) > fib(n-2) + fib(n-1)$$

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

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

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

$$\boxed{T(n) < 2T(n-1) + 1} \quad -①$$

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

$2 \cdot T(n-1) < 4 \cdot T(n-2) + 2$

$4 \cdot T(n-2) < 8 \; T(n-3) + 4$

$T(2) <$ $+1$

$$T(n) < 1 + 2 + 4 + 8 + \ldots \quad (n-1 \text{ terms})$$

$$T(n) < 1\left[\frac{2^{n-1} - 1}{2-1}\right]$$

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

$$T(n) < 2^n - 2$$

$T.C. \Rightarrow O(2^n)$