Sprint-2 [Day-3]

Time and Space Complexity-3

ToC $n \geq 1$ → less time, Better ($n \geq 0$)

$1 \leq N \leq 10^6$ ← Size of i/p

① $f(n) = n^2$; $g(n) = n^3$

f(n) : better

② $f(n) = 2^n$; $g(n) = n^2$

2^n , n^2

$\log_2^n$, $\log_2 n^2$

$n \cdot \log_2^2$ (log₂² = 1) , $2 \cdot \log_2^n$

n , $2 \cdot \log_2^n$

let $n = 2^{1024}$

2^{1024} > $2 \cdot \log_2^{1024}$ (log₂² = 1) = 2048

g(n) : best

③ $f(n) = 2^n$; $g(n) = 3^n$

~~n = -ve~~

$2^n < 3^n$

↳ best

④ $f(n) = n \cdot \log_2^n$

$g(n) = n \cdot \sqrt{n}$

$n \cdot \log_2^n$ $\quad$ $n \cdot \sqrt{n}$

$$\frac{\log_2^n}{f} < \frac{\sqrt{n}}{g}$$

$\therefore f(n)$: best.

(5) $f(n) = n^{\sqrt{n}}$ ✓ $g(n) = n^{\log_2^n}$ ✓

$\sqrt{n} > \log_2^n \quad (n>0)$

$\Rightarrow$ g(n): best

⑥ $f(n) = n$. $g(n) = (\log_2^n)^{100}$

n $(\log_2^n)^{100}$

f

$\log_2(\log_2^n)^{100}$

$100 \cdot \log_2 \cdot \log_2^n$

g

let $n = 2^{1024}$

1024

$100 \cdot \log_2 \cdot \log_2^{2^{1024}}$

10

1000

$\therefore g(n)$: bust

(f) $f(n) = n \cdot \log n$ $\quad n > 0$ $\quad g(n) = n^{\log n}$

let $a = 1000$, $b = 10$

$a * b$ $\quad a^b$

$1000 * 10$ $\quad 1000^{10}$

$1000 * 1000 * \cdots$

$f < g$

↳ better

⑧ $f(n) = \sqrt{\log n}$ $\quad$ $g(n) = \log \cdot \log n$

$(\log n)^{1/2}$ $\quad$ $\log \cdot \log n$

$\sqrt{n} > \log n$

let $n = 2^{1024}$

$(\log 2^{1024})^{1/2}$ $\quad$ $= \log \log 2^{1024}$

$(1024)^{1/2}$

$(2^{10})^{1/2}$

$= 32$ $\quad$ $= 10$ ✓

$f > g$

↳ small ⇒ bust

Σ, Π, $, Σ

Big-Oh [O( )] : Order of

T.C

↳ Unit: Big-oh / order of

B = 130 kcal

O( )

O( )

↳ Mathematical notation, used to represent T.C

.)

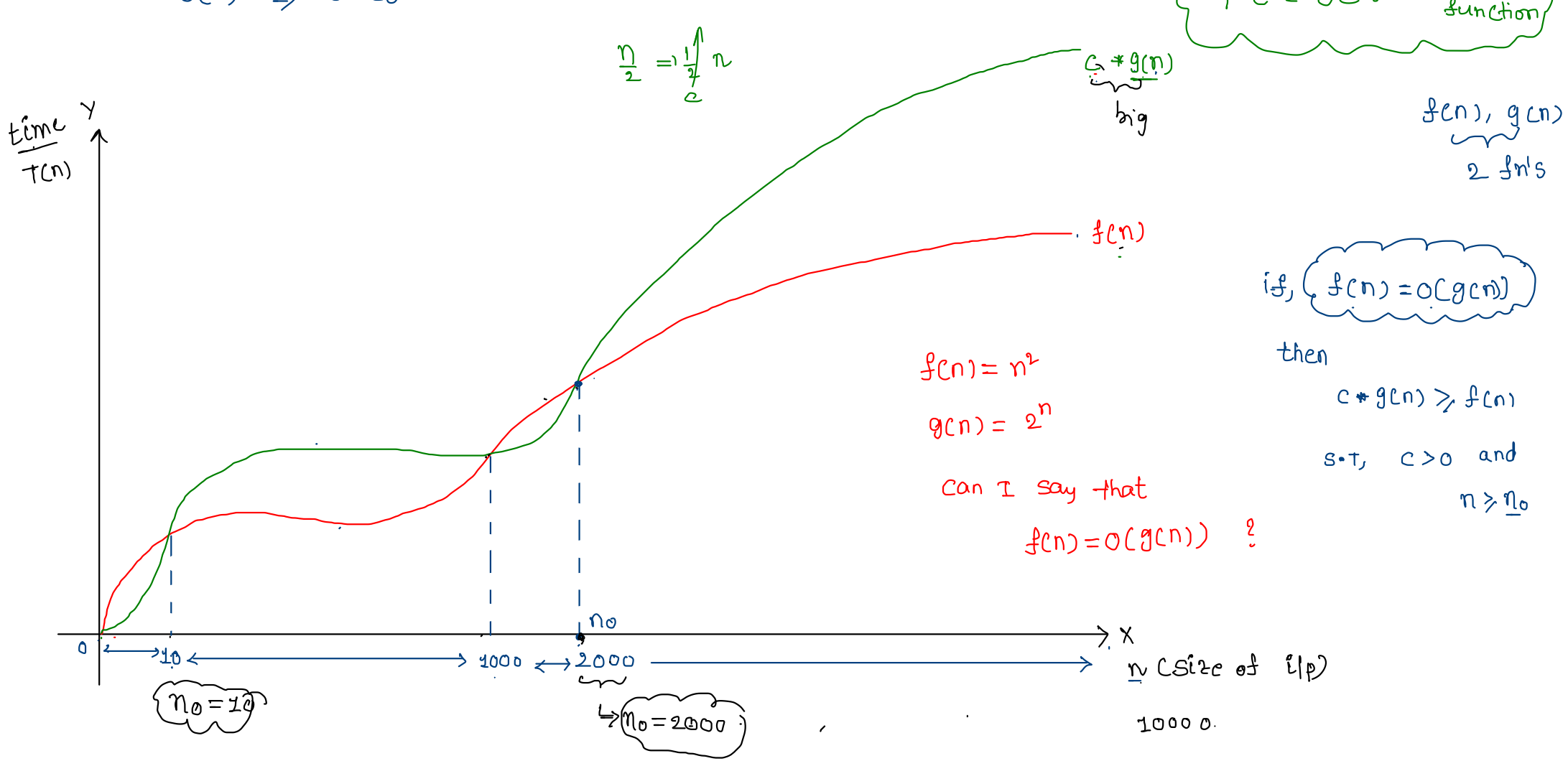
O(n) : order of n

O(√n) : order of √n

O(n²) : order of n²

O( ) ⇒ used to rep T.C, used bw 2 fn's

T.C = O( ) + maths function



f(n), g(n)
2 fn's

if, f(n) = O(g(n))

then

c * g(n) ≥ f(n)

s.t, c > 0 and

n ≥ n0

f(n) = n²

g(n) = 2ⁿ

Can I say that

f(n) = O(g(n)) ?

$f(n) = n^2$

$g(n) = 2^n$

Can I say that

$f(n) = O(g(n))$?

if, $f(n) = O(g(n))$

then

$\therefore c * g(n) \geq f(n)$ $\Rightarrow$ $f(n) \leq c * g(n)$

s.t, $c > 0$ and $n \geq n_0$

some +ve const

(1)

$f(n) = O(g(n))$

$n^2 \leq c * 2^n$

$c = 1$; $n^2 \leq 2^n$

some n:

$1 \Rightarrow 1 \leq 2$ ✓

$2 \Rightarrow 4 \leq 4$ ✓

$3 \Rightarrow 9 \leq 8$ ✗

$\rightarrow 4 \Rightarrow 16 \leq 16$ ✓

$5 \Rightarrow 25 \leq 32$ ✓

$6 \Rightarrow 36 \leq 64$ ✓

$n_0 = 1$ ✗

≥ 1 ✓

2 ✓

3 ✗

4

$n_0 = 4$ $\Rightarrow$ 5 6 7 . . .

$\therefore c = 1, \quad n_0 = 4.$

$\therefore f(n) = O(g(n))$ ✓

$f(n) = O(g(n))$

$f(n) \le c \cdot g(n)$, such that

$c > 0$, $n \ge n_0$

$f(n) = 2n^2 + 3n + 1 \Rightarrow O(n^2)$ ✓

$f(n) = 2n^2 + 3n + 1$, $g(n) = n^2$

$2n^2 + 3n + 1 \le c \cdot n^2$

let $c = 1$

$2n^2 + 3n + 1 \le n^2$

$n = 1$: ×

$n = 2$: ×

$n = 3$: ×

$c = 2$

$2n^2 + 3n + 1 \le 2n^2$

$n = 1$

2

3

⋮

$c = 3$

f g

$2n^2 + 3n + 1 \le 3n^2$

$n = 1$: × → f ✓

$n = 2$: × → f

$n = 3$: × → f

$n = 4$: $45 \le 48$ ✓ → g

$n = 5$: $66 \le 75$ → g

$n = 6$: $91 \le 108$ → g

⋮

$c = 1000$

$= 10,000$

$\therefore c = 3$, $n_0 = 4$

$\therefore f(n) = O(n^2)$

let

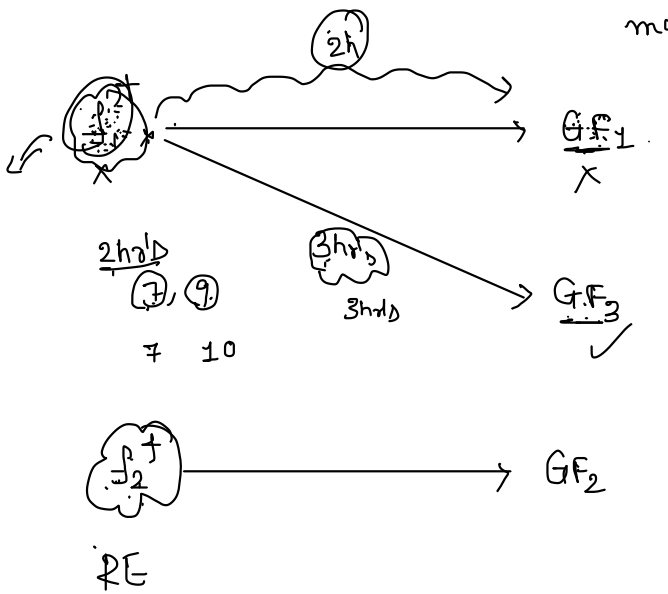
$$f(n) = 2n^2 + n^1 + 3 \cdot n^0 \Rightarrow 2n^2 \Rightarrow O(n^2)$$

Highest degree nth term

why?

$\frac{n(n+1)}{2} \Rightarrow \frac{n^2+n}{2} \Rightarrow O(n^2)$

$\frac{n(n+1)(2n+1)}{6} \Rightarrow O(n^3)$



max(2h, 3h) = 3h

$f(n) = 3n^4 + 2n^2 + \log_2 n \Rightarrow O(n^4)$

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

$\Rightarrow \frac{n^2(n^2+1+2n)}{4} \Rightarrow O(n^4)$

Code

$f(n) = O(\underbrace{\sqrt{n}}_{P_1} + \underbrace{\log_2 n}_{P_2}) \Rightarrow \underbrace{O(\sqrt{n})}_{\text{max}}$

max / Heighest

$= O(\sqrt{n})$

$\Rightarrow O(\log_2 n) \Rightarrow$ code

O( )
↳ to represent

Approximately

How many times loop running

T.C

```
⇒ for(i=1;i<=n;i++)   ⇒ O(n)
  {
      ↳ print(*)  ⇒ O(1)  constant
  }
```

+1

n times

break;
continue;
return
} O(1)

$\frac{n}{2} \Rightarrow \frac{1}{2} \cdot n$

O(n)

```
for(i=1;i<=n;i=i+2)  ⇒ n/2
{
      print(*) : O(1)
}
```

$\Rightarrow \frac{n}{2} = \frac{1}{2} \cdot n$

c

```
i for(i=1; i<=n;i++) ⇒ n
  {
    j for(j=1;j<=n/2;j++) ⇒ n/2
      {
          c=c+1 : O(1)
      }
  }
```

$n * \frac{n}{2} = \frac{n^2}{2}$

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

$\therefore O(n^2)$

Nested Loop (loopc)

↳ Multiply

```
{
loopc )
{
}
```

# of times O.L will run * # of times I.L will run

```
for(i=1;i<=n;i++)
{
    for(j=1;j<=n/4;j++)
    {
        for(k=1;k<=n;k++)
        {
            print("*")
        }
    }
}
```

i → n

j → n/4

k → n

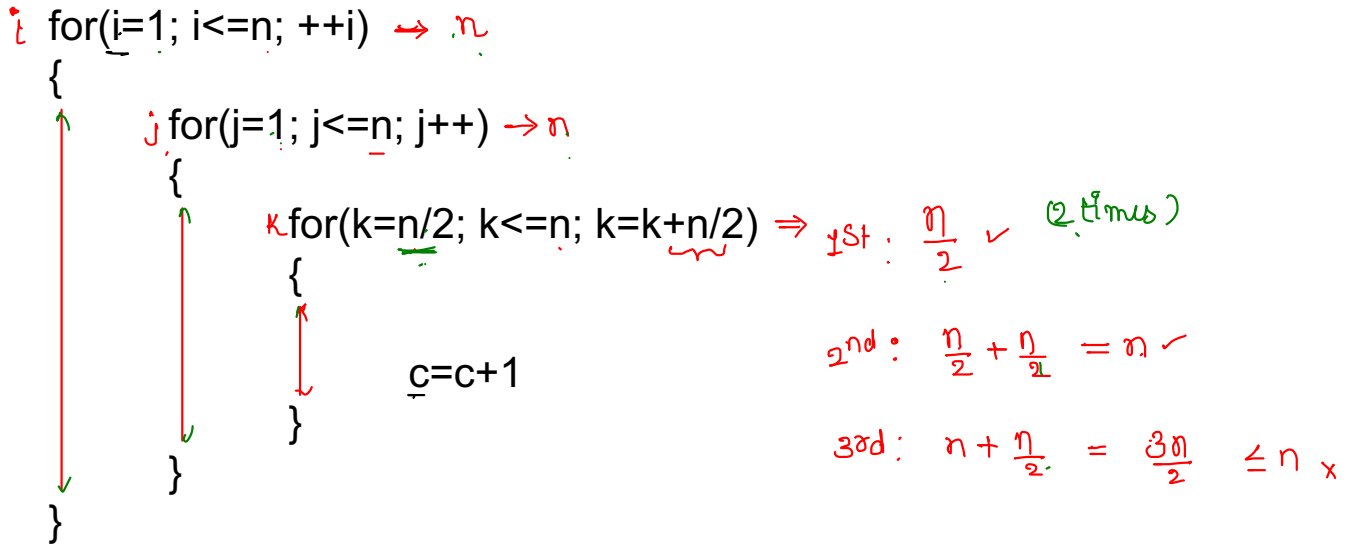
$n * \frac{n}{4} * n = \frac{n^3}{4}$

$= \frac{1}{4} \cdot n^3$

$= O(n^3)$

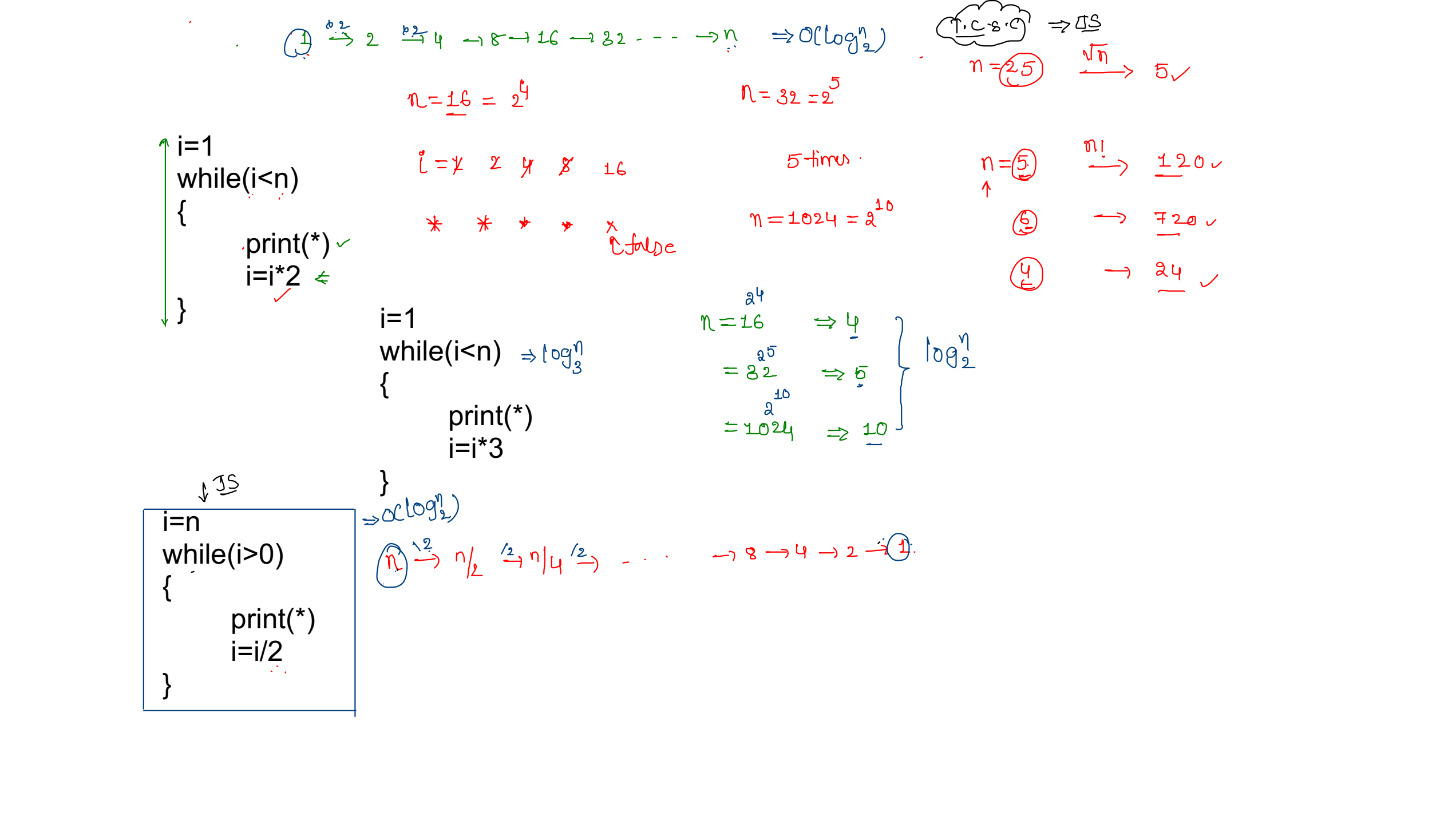
```
for(i=1;i<=n;i++)
{
    for(j=1;j<=n/4;j++)
    {
        for(k=1;k<=n;k++)
        {
            print("*")
            break;
        }
    }
}
```

$n * \frac{n}{4} * 1 \Rightarrow O(n^2)$


i for(i=1; i<=n; ++i) → n
{
j for(j=1; j<=n; j++) → n
{
k for(k=n/2; k<=n; k=k+n/2) ⇒ 1st : n/2 ✓ (2 times)
{
c=c+1
}
}
}
2nd : n/2 + n/2 = n ✓
3rd : n + n/2 = 3n/2 ≤ n ×

$$n * n * 2 = 2n^2$$

$$= O(n^2)$$


1 *2 → 2 *2 → 4 → 8 → 16 → 32 - - - → n ⇒ O(log2n)
T.C.S.C ⇒ DS
n = 25 √n → 5
n = 16 = 2^4
n = 32 = 2^5
i=1
while(i<n)
{
print(*)
i=i*2
}
i = 1 2 4 8 16
* * * * ×
false
5 times
n = 1024 = 2^10
n = 5 n! → 120
5 → 720
4 → 24
i=1
while(i<n) ⇒ log3n
{
print(*)
i=i*3
}
n = 16 = 2^4 ⇒ 4
= 32 = 2^5 ⇒ 5
= 1024 = 2^10 ⇒ 10
log2n
DS
i=n
while(i>0)
{
print(*)
i=i/2
}
⇒ O(log2n)
n *1/2 → n/2 *1/2 → n/4 *1/2 → - - - → 8 → 4 → 2 → 1

n

n^n, $\log_2^n$, $n!$, $n * \log_2^n$, ~~n~~

```
for(i=1; i<=n; i++)
{
    for(j=1; j<n; j=2*j)
    {
        c=c+1
    }
}
```

O(1) × n × $\log_2^n$

$\Rightarrow O(n * \log_2^n)$

for(j=1; j

for(j=1; j

```
for(j=1; j<n; j=2*j)
{
    c=c+1
}
```

j = 1 2 4 8 ... n

$\log_2^n$

```
for(i=1; i<=n; i++) ⇒n
{
    for(j=1; j<=n; j++) ⇒n
    {
        c=c+1
    }
}
```

$O(n^2)$

dependent loops

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=i; j++)   ⇒depends on outer loop variable
    {
        c=c+1
        print("*")
    }
}
```

n=4

i=1	i=2	i=3	i=4	i=5
j=1 2	j=1 2 3			
*	* *	* * *	* * * *	
1	2	3	4	

n=4

1 + 2 + 3 + 4 ✓

n=5

1 + 2 + 3 + 4 + 5

n

$1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2} \Rightarrow O(n^2)$

4ep

```
for(i=1; i<=n; i++)
{
    for(j=i; j<=i; j++)
    {
        c=c+1
        print("*")
    }
}
```

n=5

i=1, i=2, i=3, i=4, i=5

j=1 2 j=2 3

*

n=6

n ⇒ O(n)

```
for(i=1; i<=n; i++)
{
    for(j=i; j<=n; j++)
    {
        c=c+1
        print("*")
    }
}
```

n=4

⇒ O(n²)

i=1

j=1 . . 4 ⇒ 4

i=2

j=2 . . 4 ⇒ 3

i=3

j=3 . . 4 ⇒ 2

1

GATE

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=n; j=j+i)
    {
        c=c+1
        print("*")
    }
}
```

$n=10$

$i=1$ (+1)	$i=2$ (+2)	$i=3$	$i=10$
$j=1\ 2\ 3\ \cdots\ 10$	$j=1\ 3\ 5\ 7\ 9\ 11,$		$j=1\ 11$
$*\ *\ *\ \cdots\ *$	$*\ *\ *\ *\ *$		$*$
$10(*)$	$5(*)$		
n	$\frac{n}{2}$	$\frac{n}{3}$	$\frac{n}{4}\ \rightarrow\ \cdots\ 1$
$\frac{n}{1}$			

$$= \frac{n}{1} + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots + 1$$

$$= n\left[1 + 1/2 + 1/3 + 1/4 + \cdots + 1/n\right]$$

$\log_{10}^{n}$

$$= O(n \cdot \log_{10}^{n})$$

$O(n!)$

$n \times n-1 \times n-2 \times \cdots 1$

```
i=n
while(i>=0)
{
    j=1
    while(j<=n)
    {
        j=j*2
    }

    i=i/2
}
```

$\log_2^n$ → $\log_2^n$

$\log_2^n * \log_2^n \Rightarrow O((\log_2^n)^2)$

```
L1 for(i=1;i<n;i++)  ⇒ O(n)
   {
      L2 for(j=1;j<k;j++)  ⇒ k
        {
            print("*")
        }
      ---------------------------
      L3 for(j=1;j<p;j++)  ⇒ p
        {
            print("*")
        }
   }
```

$n * [k + p]$ — max

2h, 3h

$O(n \cdot \max(k, p))$

```
L1()
{
   L2()
}
```

$L_1 * L_2$

L1() ✓

L2() ✓

$L_1 + L_2$

```
for(i=1; i<=n; i++)  ⇒ n
{
        j=1

        while(j<=n)  log2n
        {
                j=2*j
        }

        for(k=1;k<=n;k++)  n
        {
                c=c+1
        }
}
```

2:40

$n * (\underbrace{\log_2 n}_{\max} + n) \Rightarrow O(n^2)$

GATE

```
function fun(n)
{
        q=0
    L1  for(i=1;i<=n;i++) ⇒ n
        {
            L2  p=0
                for(j=n; j>1; j=j/2) ⇒ log2 n
                {                       p
                        ++p
                }

            L3  for(k=1; k<p; k=k*2 ) ⇒ log2 p
                {
                        q++
                }
        }
}
```

$L_1 * (L_2 + L_3)$

$n * (\log_2 n + \log_2 p)$

$n * (\underset{①}{\log_2 n} + \underset{②}{\log_2 \log_2 n})$

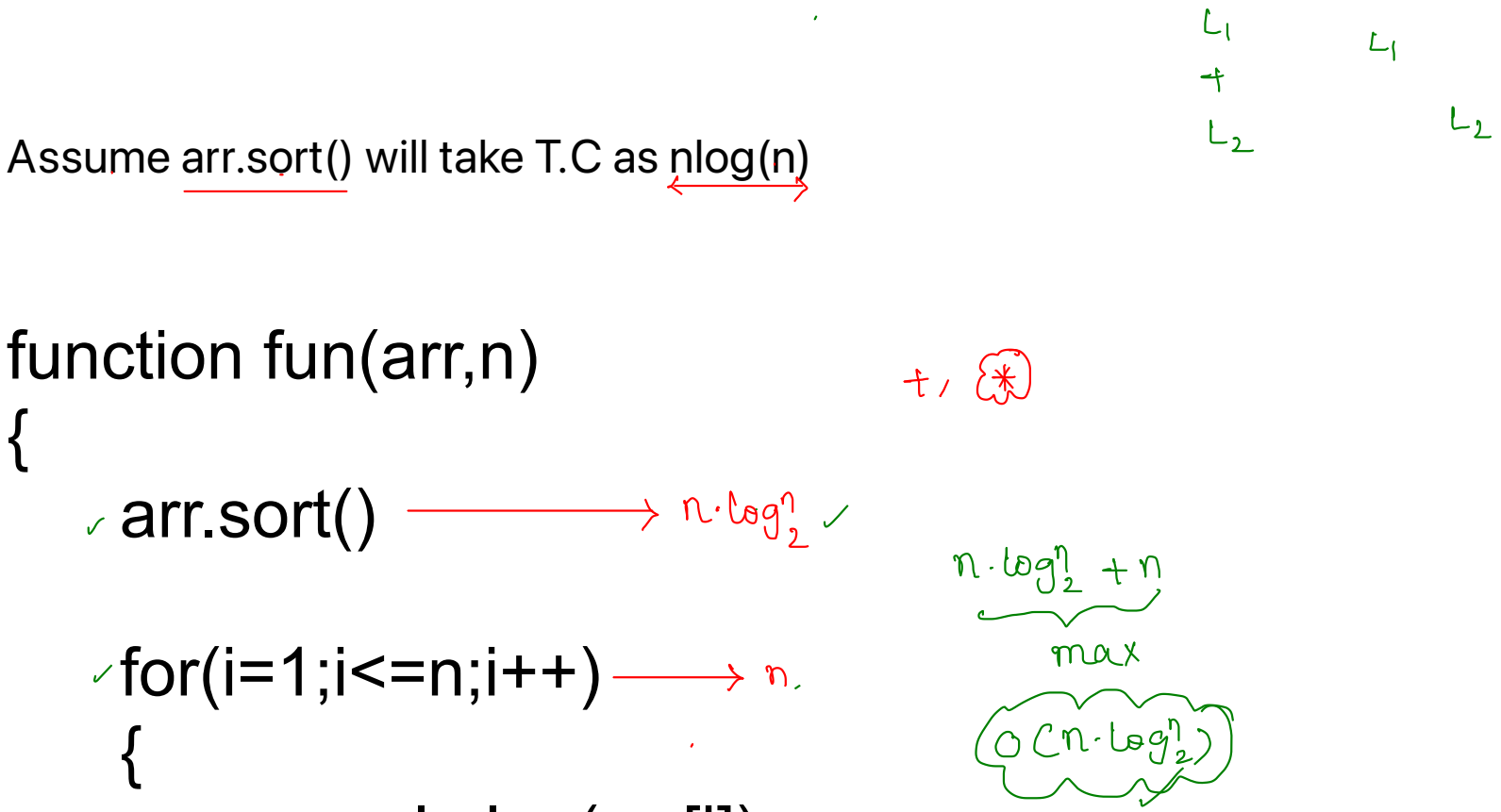
max : ①

$O(n * \log_2 n)$

Assume arr.sort() will take T.C as nlog(n)

```
function fun(arr,n)
{
    arr.sort()

    for(i=1;i<=n;i++)
    {
        console.log(arr[i])
    }
}
```



Let T: be the number of test cases

```
while(T>0)
{
        for(i=1;i<=n;i++)
        {
                arr.sort()
                j=1
                while(j<=n)
                {
                        j=j*2
                }
        }
        T--
}
```

Consider the program

```
void function(int n) {
int i, j, count=0;
for (i=n/2; i <= n; i++)
        for (j = 1; j <= n; j = j*2)
                count++;
}
```

The complexity of the program is

1. O(log n)
2. $O(n^2)$
3. $O(n^2 \log n)$
4. O(n log n)

What is the complexity of the following code?

```
i = n
while (i>=1){
    for j = 1 to n
        x=x +1
    i = i/2
}
```

a. $\Theta(n)$
b. $\Theta(\log_2 n)$
c. $\Theta(n/\log_2 n)$
d. $\Theta(n \log_2 n)$

```
function bubbleSort( arr, n)
{
var i, j;
for (i = 0; i < n-1; i++)
{
    for (j = 0; j < n-i-1; j++)
    {
        if (arr[j] > arr[j+1])
        {
        swap(arr,j,j+1);

        }
    }

}
}
```