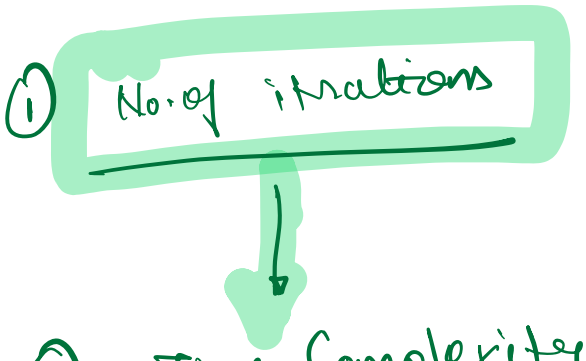
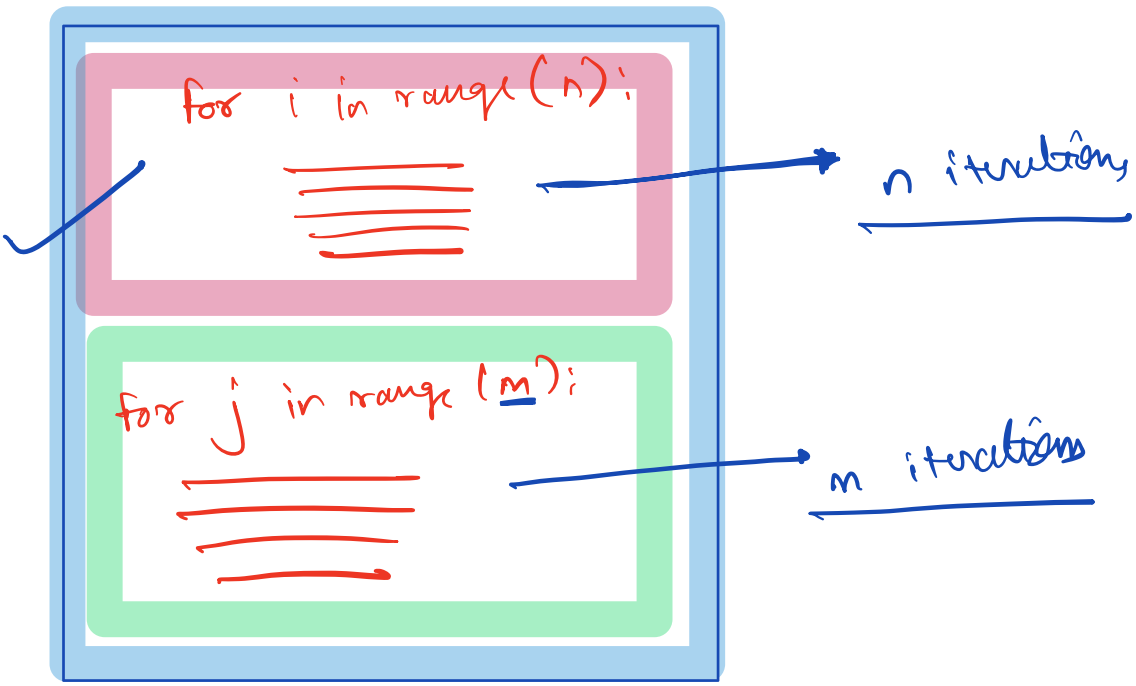
"Hello Everyone!"

Time Complexity of a Code.



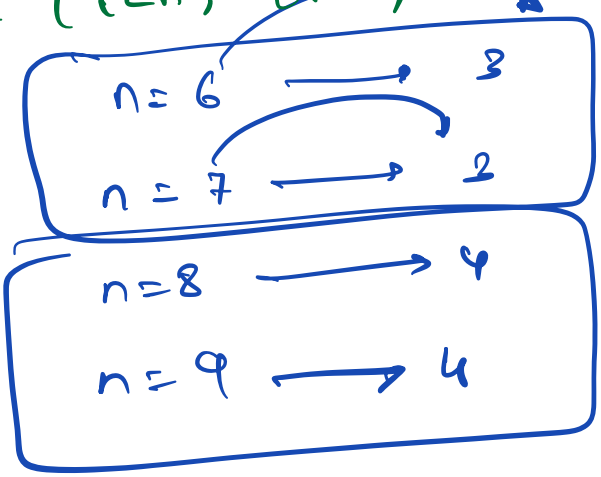
* ② Time Complexity → Big-O

O( )



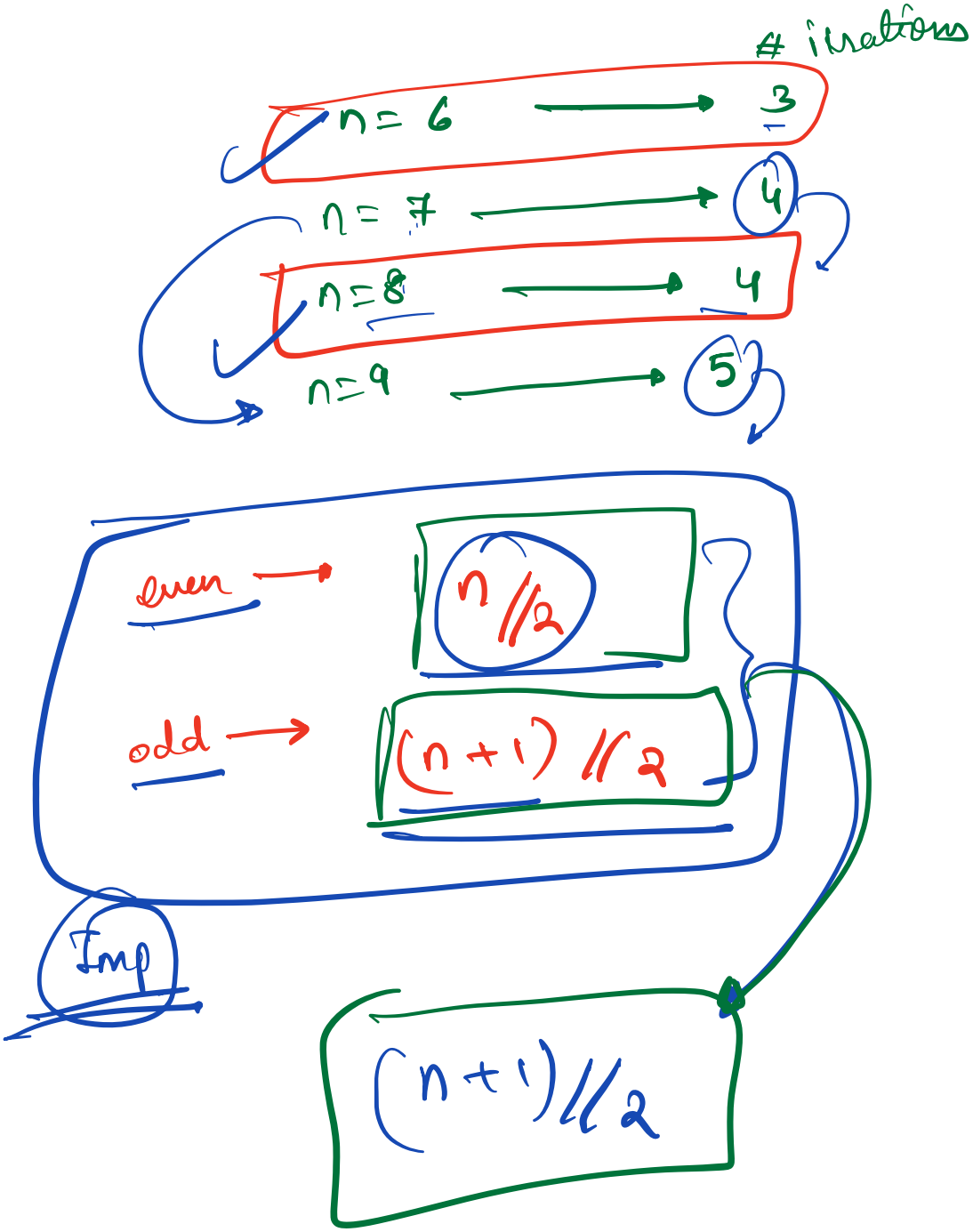
Total no. of iterations = $n + m$

Case 1 :- (i<n, i+=2)



7//2

Obs :- Case 2 (i <= n)



odd → $(n//2 + 1)$ ✓

even → $(n//2 + 1)$

$n = 6 \longrightarrow 6//2 + 1$

$= 4$ ✗

---

generalize

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

$n = 6 \longrightarrow (6+1)//2$

$\longrightarrow 7//2 \rightarrow 3$ ✓

$n = 7 \longrightarrow (7+1)//2$

$= 8//2$

$= 4$ ✓

```
i = 1
while i * i <= n:
    # task
    i += 1
```

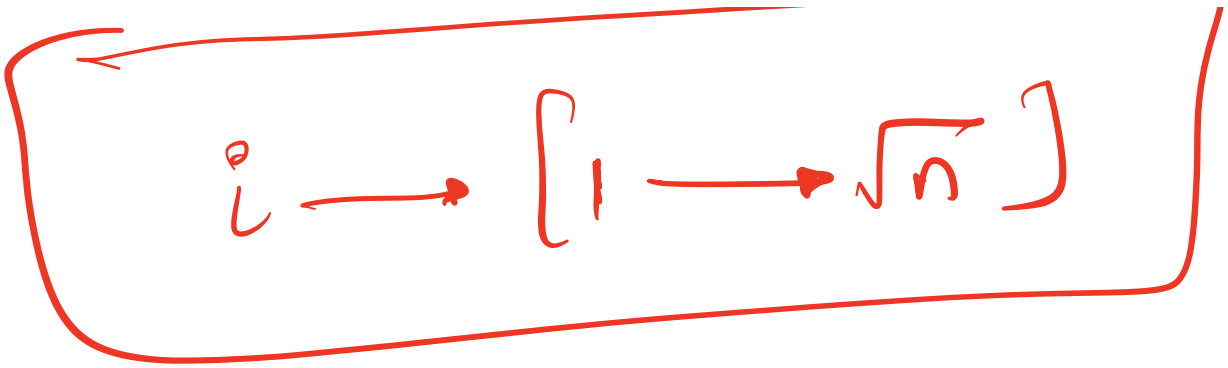
$i * i <= n$

initial $\longrightarrow$ $i * i$ $\longrightarrow$ 1

final $\longrightarrow$ $i * i$ $\longrightarrow$ n

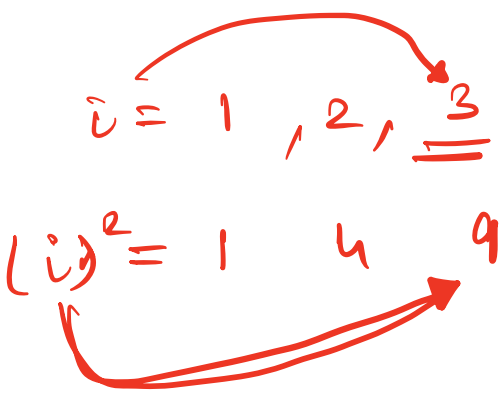
$i * i \longrightarrow [1 \longrightarrow n]$

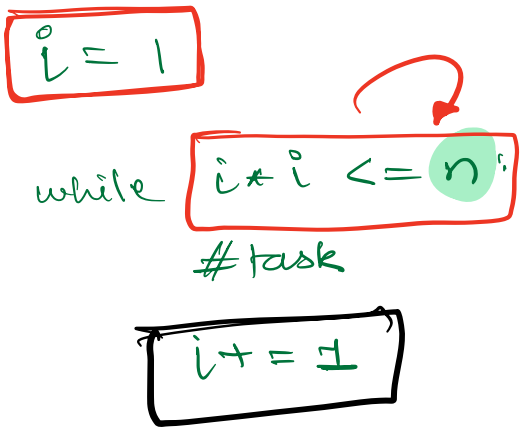
$(i)^2 \longrightarrow [1 \longrightarrow n]$



[1 5] → 5

Opp





n = 5

	i	i * i	Loop ?
✓ ①	1	1	1 <= 5 → ✓
✓ ②	2	4	4 <= 5 → ✓
③	3	9	9 <= 5 ✗

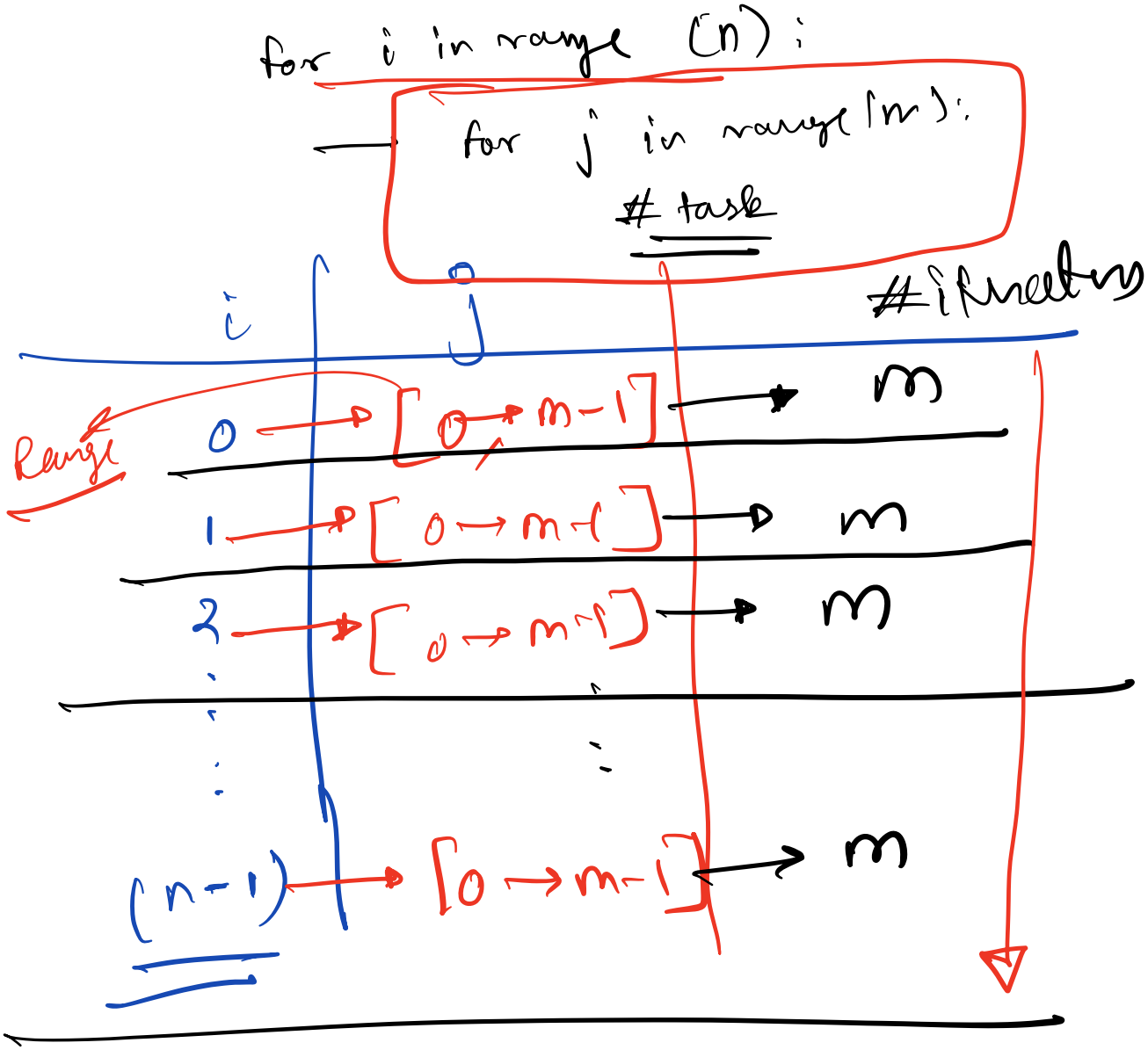
$n = 16$

	i	i * i	Loop ?
①	1	1	$1 <= 16$ ✓
②	2	4	$4 <= 16$ ✓
③	3	9	$9 <= 16$ ✓
④	4	16	$16 <= 16$ ✓
⑤	5	25	$25 <= 16$ ✗

not entered

no. of iterations = $int(\sqrt{n})$

10:01 sharp


for i in range (n):
for j in range (m):
# task
i
j
iterations
Range
0 → [0 → m-1] → m
1 → [0 → m-1] → m
2 → [0 → m-1] → m
(n-1) → [0 → m-1] → m

Total iterations

$m + m + m \cdots$ n times

$n * m$

[0 $m-1$]

m

* Time Complexity

Scenario :- Task

C1: Logic1 (Rupam)

C2

Logic2 (Sujit)

15 sec

10 sec

execution time

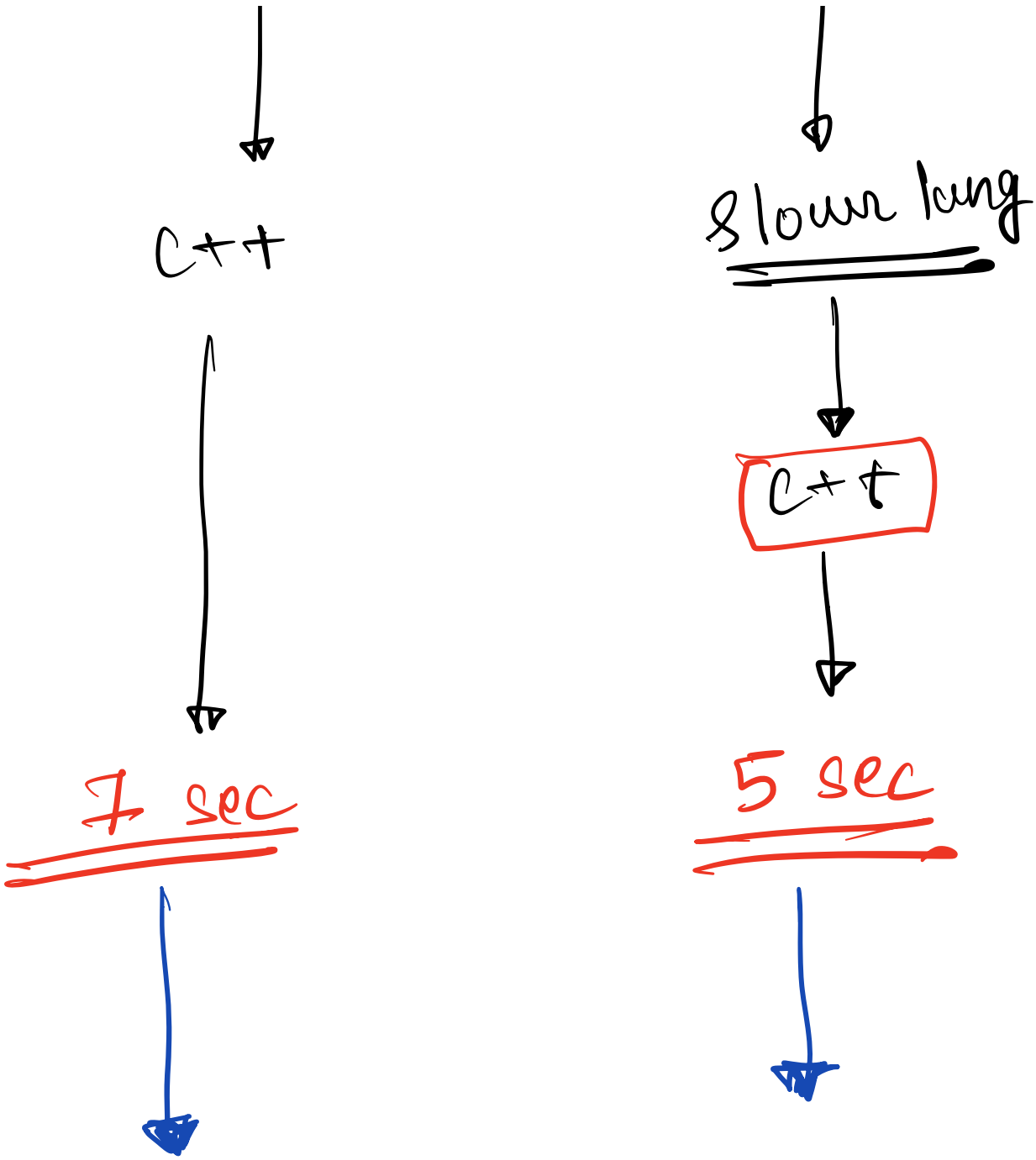
(windows XP)
2005 yr

(macbook M3)

(macbook M3)

7 sec

10 sec


C++
7 sec
slow lang
C++
5 sec

Execution time

↓

depends on so many external factors, hence we generally don't compare execution time between 2 Algorithm (logic)

Conclusion :-

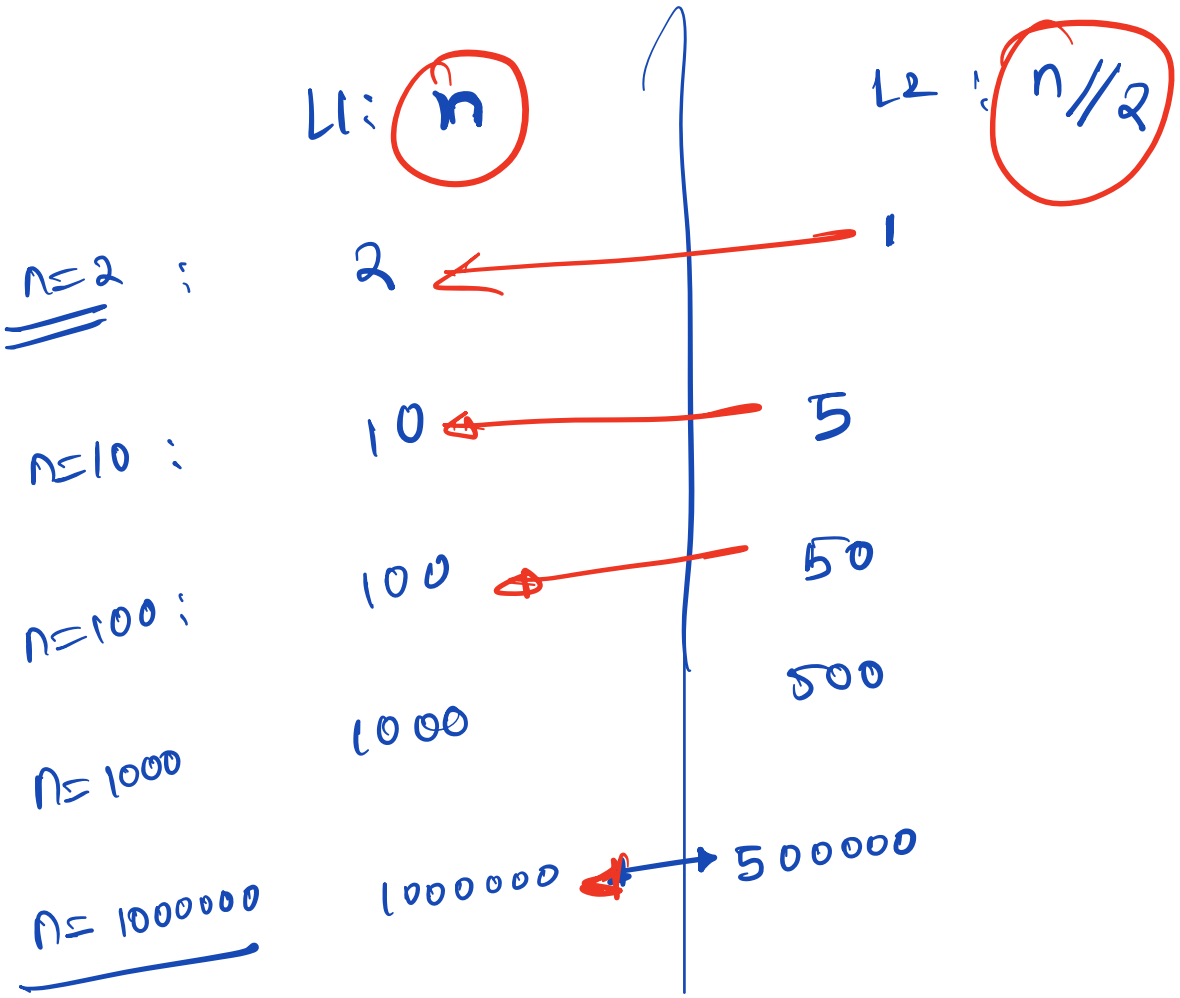
No. of iterations of the code as a measure of performance

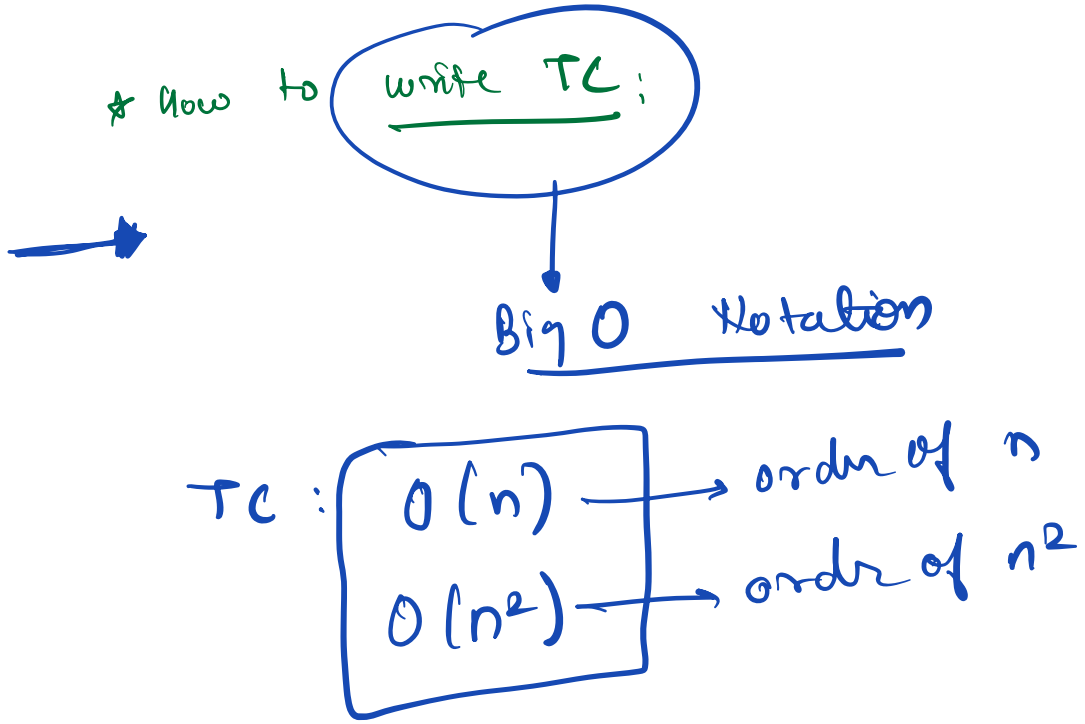
100 customers ?

1000

Million / Billion

Analyse the performance of an Algorithm (logic) for a very large input.


L1: n
L2: n//2
n=2 :
2
1
n=10 :
10
5
n=100 :
100
50
500
n= 1000
1000
n= 1000000
1000000
500000


* How to write TC;
Big O Notation
TC :
O(n)
O(n²)
order of n
order of n²

TC ⟶ # iterations

① Calculate no. of iterations

~~①~~ Ignore lower order terms

② Ignore constant Coefficients

(1) no. of iterations = $(n^2 + 2n)$

highest order term

TC = $O(n^2)$

(2) #iterations = $5n^2 + 2n$

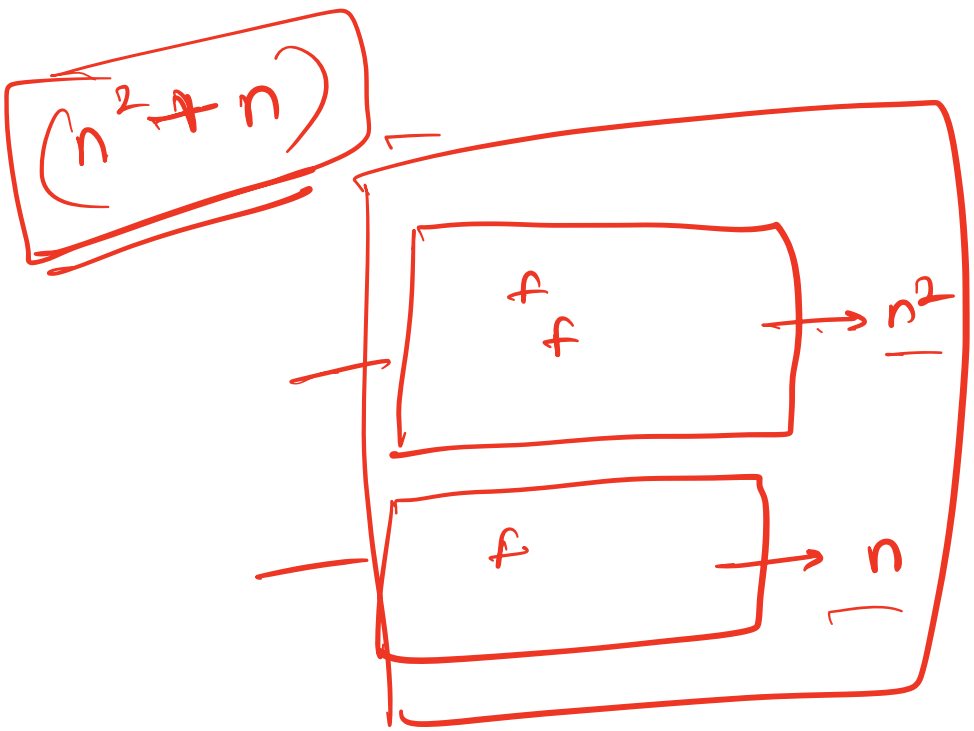
$5n^2$

TC = $O(n^2)$

eg:-

① → $(n^2 + 5n^3 + 3n + 9)$

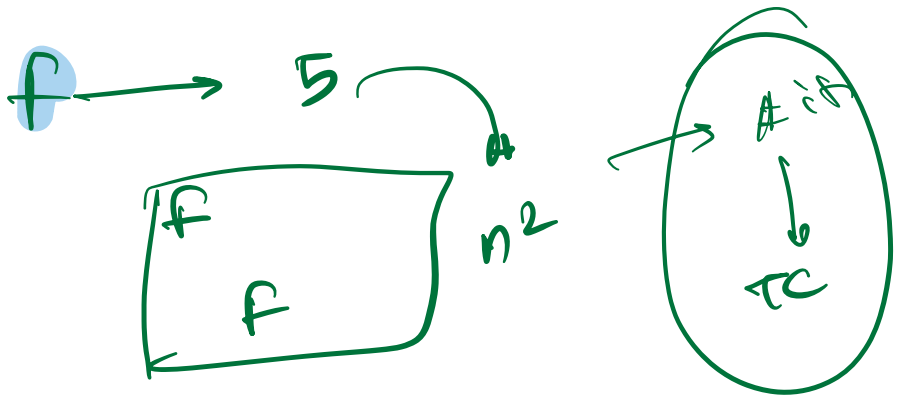
TC : $O(n^3)$

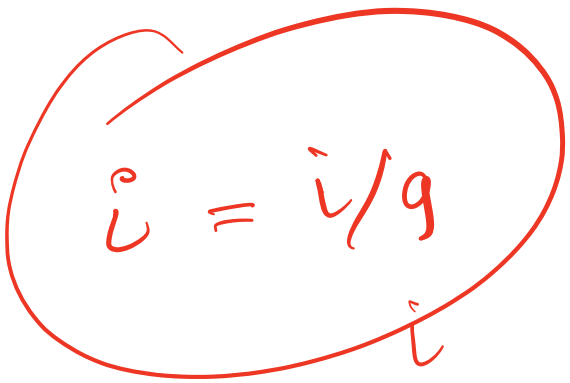


eg 3 :- $\frac{n^2}{2} + 5n + \frac{n^4}{9} + 5$

TC : $O(n^4)$

H.W → Try to find TC of all the codes done in Notebook.





v = i+1

i = i+2