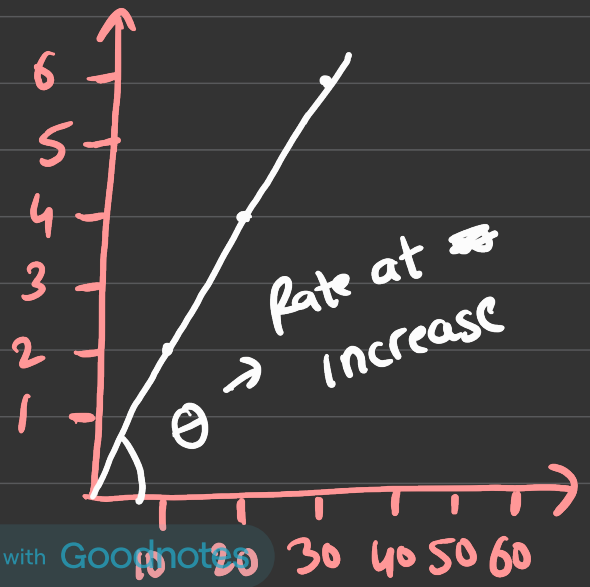
Time Complexity → TC != Time taken to run a code

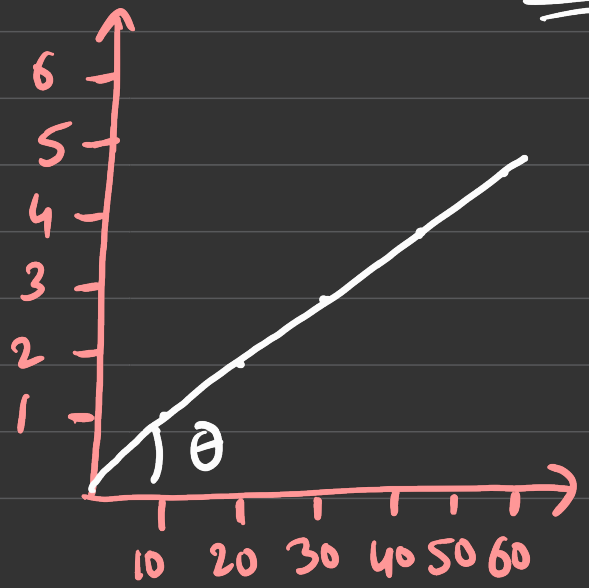
↳ The rate at which time taken increases with respect to input size

W8

CODE

Mac





How is it calculated?

Big-oh Notation

$O(3N+1)$

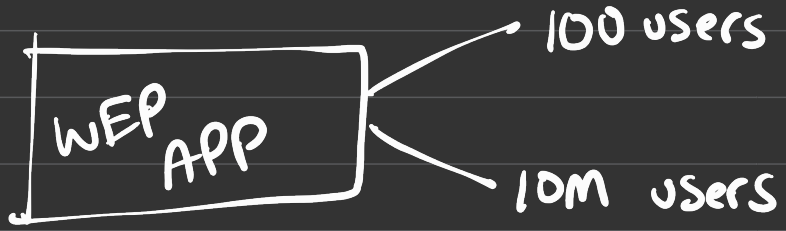
↑

```
i = 1
while i <= N:        → 3N + 1
    print(i)
    i = i + 1
```

$5 \times 3 + 1$

3 rules :

1) Always go for worst-case
2) Avoid / Remove constants
3) Avoid / Remove lower bounds



2) Ignore constants

$O(10N^5 + 5N^2 + 15)$ $N = 10^5$

$O(10 \times 10^{25} + 5 \times 10^{10} + 15)$ 100000

3) Avoid Lower bounds

$O(10 \times 10^{25} + 5 \times 10^{10})$

$\leq O(10N^5) \leq O(N^5)$

Big-Oh (O) → Worst-case [upper bound]

Theta (θ) → Average-case

Omega (Ω) → Best-case [Lower-bound]

marks = 55 <60

```
if marks >90: print ("A")
elif marks >80: print("B")
elif marks >70: C
elif marks >60: D
else:
    F
```

Fix 1

```
for i in range(1, n+1):
    for j in range(1, n+1):
        Block
```

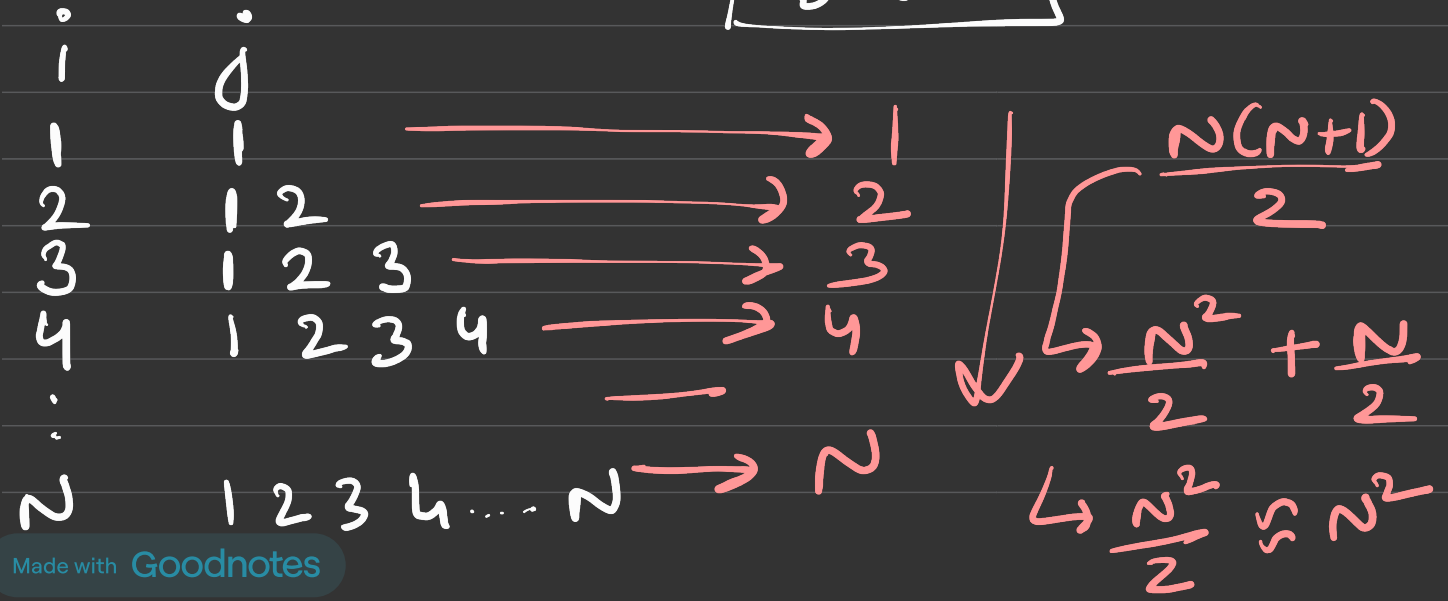
i	j
1	1 2 3 .... N
2	1 2 3 .... N
3	1 2 3 .... N
:	
N	1 2 3 .... N

N N N → N × N

TC → $O(N^2)$

Ex2

```
for i in range (1, n+1):
    for j in range(1, i+1):
        Block
```



Space Complexity Big Oh

SC → Auxiliary Space + Input Space

Auxiliary Space → It is the extra space used to solve the algorithm

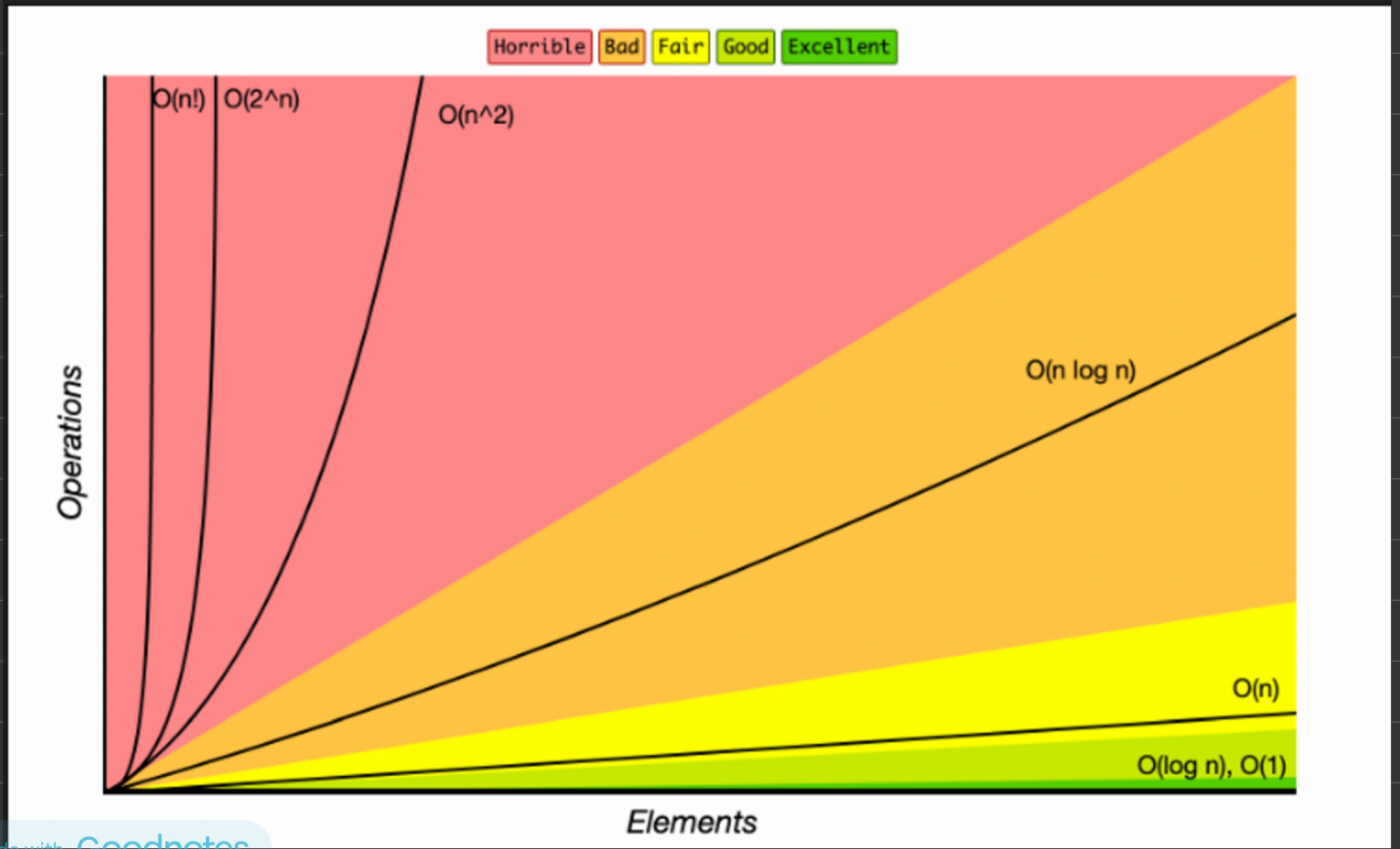
Input Space → It is the space to store the input

Input Space

a = 5 , b = 10 , C = 12

```
total = a+b+C
print (total)
```

→ Auxiliary Space



Operation	Average Case	Amortized Worst Case
Copy	O(n)	O(n)
Append[1]	O(1)	O(1)
Pop last	O(1)	O(1)
Pop intermediate[2]	O(n)	O(n)
Insert	O(n)	O(n)
Get Item	O(1)	O(1)
Set Item	O(1)	O(1)
Delete Item	O(n)	O(n)
Iteration	O(n)	O(n)
Get Slice	O(k)	O(k)
Del Slice	O(n)	O(n)
Set Slice	O(k+n)	O(k+n)
Extend[1]	O(k)	O(k)
Sort	O(n log n)	O(n log n)
Multiply	O(nk)	O(nk)
x in s	O(n)	
min(s), max(s)	O(n)	
Get Length	O(1)	O(1)

arr=[1,5,3,2,6,7]

100

[1,5,3,2,6,7,100]

0 1 2 3 4 5 6

a[3]

if 10 in arr: