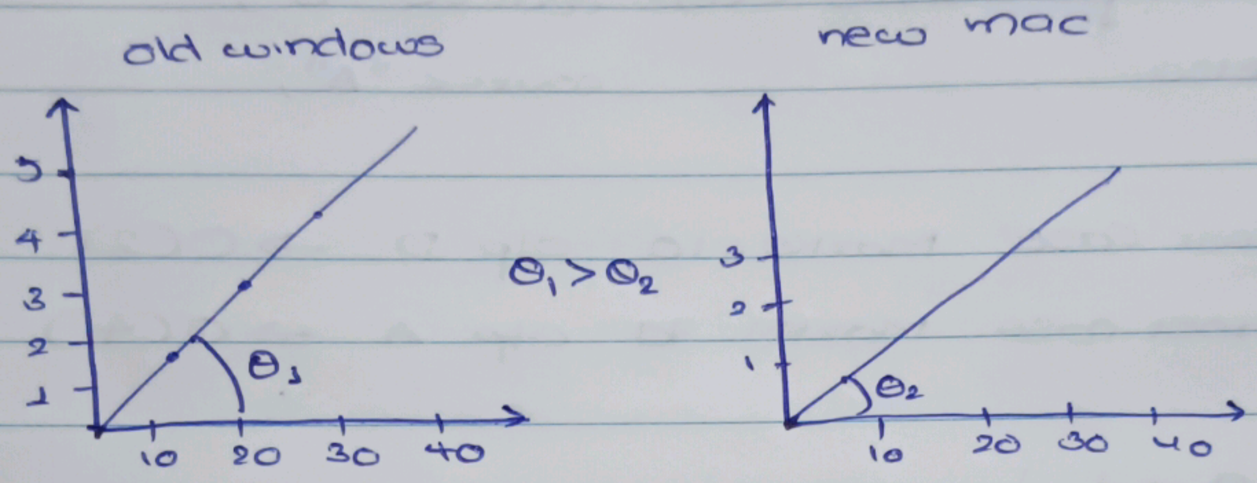
Time Complexity?

TC != time taken

Rate at which the time taken increases with respect to input size



TC → Big-O-Notation O(1)
↓
time taken

eg:
```
for (i=1; i<=5; i++)
{
    cout<< "Raja";
}
```

→ O(3×5)

→ O(15)

Rules:

→ TC should be computed on worst scenarios.

→ avoid constants

→ avoid lower values.

① Rule 1: Always consider wrost case

Best Case   Average Case   Wrost case

```
eg: if (marks < 25) cout<< "D";
 else if (marks < 45) cout << "C";
 else if (marks < 65) cout << "B")
 else                 cout<< "A";
```

Best case: Marks = 10   o/p: D $\rightarrow O(2)$

Wrost case: Marks = 70   o/p: A $\rightarrow O(4)$

② Avoid lower values and constants

$O(4N^3 + 3N^2 + 8)$

let $N = 2\times10^5$

$\approx O(4\times(2\times10^5)^3 + 3(2\times10^5)^2 + 8)$

$\approx O(N^3)$

other TC: Big-O-Notation → wrost case (upper bound)

Theta Θ Notation → average

Omega (Ω) → Lowest bound

Q|A:

(1)

```
for(int i=0; i<N; i++)
{
 for(int j=0; j<N; j++)
 {
      □ → constant time
                code
  }
}
```

$O(N^2)$.

(2)

```
for(i=0; i<N; i++)
{
 for(j=0; j<=i; j++)
 {
      □ → constant time
             code
  }
}
```

$1+2+3+\cdots+n$

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

$= \frac{n^2}{2} + \frac{n}{2}.$

$\approx O(n^2)$

Space Complexity: → memory space
→ Big-O-Notation.

→ Auxillary Space + Input space

auxillary Space: space that you take to solve the problem

Input space → space that you take to store the problem

→ input space.

a , b

$$\left.\begin{array}{l} c = a + b \end{array}\right\} O(3).$$

auxillary space

Eg: int a[N]; O(N)

Note: a, b

[Never do anything to The input data]

b = a + b ×

1s = 10^8 operations.

2s = 2×10^8 operations.

5s = 5×10^8 operations

TC → 1s ⇒ $O(10^8)$ operations.