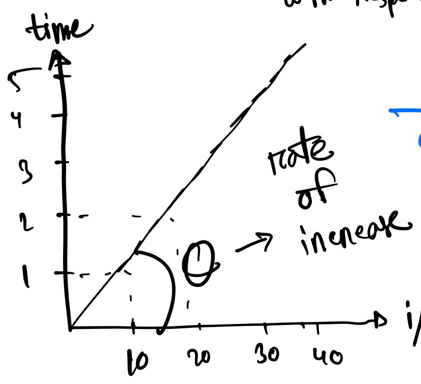
Time & Space Complexity

what is time complexity?

- → TC != time taken
- → Dependent on system
- → The rate of which the time taken increases with respect to the input size.



} time complexity

example:

```
for( i=0 ; i<5 ; i++)
{
    cout << " ucchash" ;
}
```

#Number of steps this code takes

Assign, Compare, Print, Iteration......

- → TC is computed in terms of worst case scenario
- → Avoid constants
- → Avoid lower values

here, 5 × 3 ← increment check print

∴ TC → $\boxed{O(N \times 3)}$

How do we define TC?

→ Big-O-Notation → O( ) ← time taken

Best vs Average vs Worst

```
if ( marks <25 )        cout << " grade D" ;
else if ( marks <45 )   cout << " gabe c" ;
elseif ( marks <65 )    cout << " grade B" ;
else                    cout << " grade A" ;
```

marks = 10 → O(2) ← Best Case

marks = 70 → O(4) ← Worst Case

→ **Always compute TC in terms of worst case** for example:

$O(4N^2 + 3N + 8)$ let, $N = 10^5$

$\rightarrow O(4(10^5)^2 + 3(10^5) + 8)$

↑ very low significance

∴ Avoid Constants

for example:

$O(100000 + 10)$

$\rightsquigarrow O(100000)$

↑ very low significance

∴ Avoid lower values

Big-O (O)	Theta θ	Omega (Ω)
Worst Case	Average	Best Case
{upper bound}	{Avg Complexity}	{lower bound}

Questions:

```
for(i=0; i<N; i++) {
    for(j=0; j<M; j++) {
        Const block of Code }}
```

Ans: $TC = O(N \times M)$

→ Const avoided

→ Iterations : $N \times M$

```
for(i=0; i<N; i++){
    for(j=0; j<=i; j++){
        Const block of code }}
```

Ans:

$i=0$ $[j=0]$

$i=1$ $[j=0,1]$

$i=2$ $[j=0,1,2]$

$i=n-1$ $[j=0,1,2 \ldots n-1]$

$[1+2+3+4+\ldots n]$

$$\boxed{\begin{aligned} &\rightarrow \frac{N \times (N+1)}{2} \\ &= \frac{N^2}{2} + N/2 \\ &O\left(\frac{N^2}{2}\right) \approx O(N^2) \end{aligned}}$$

What is Space Complexity :

- Memory space that your program takes
- Varies machine to machine
- Auxiliary Space + Input Space

Auxiliary Space ↓ Space that you take to solve the problem

Input Space ↓ Space that you take to store the input

for example:

input; a, b ← input space

output c = a+b

↑ Auxiliary space

So, space : O(3) ← In terms of Big-O Notation

array: int a[N] ← O(N)