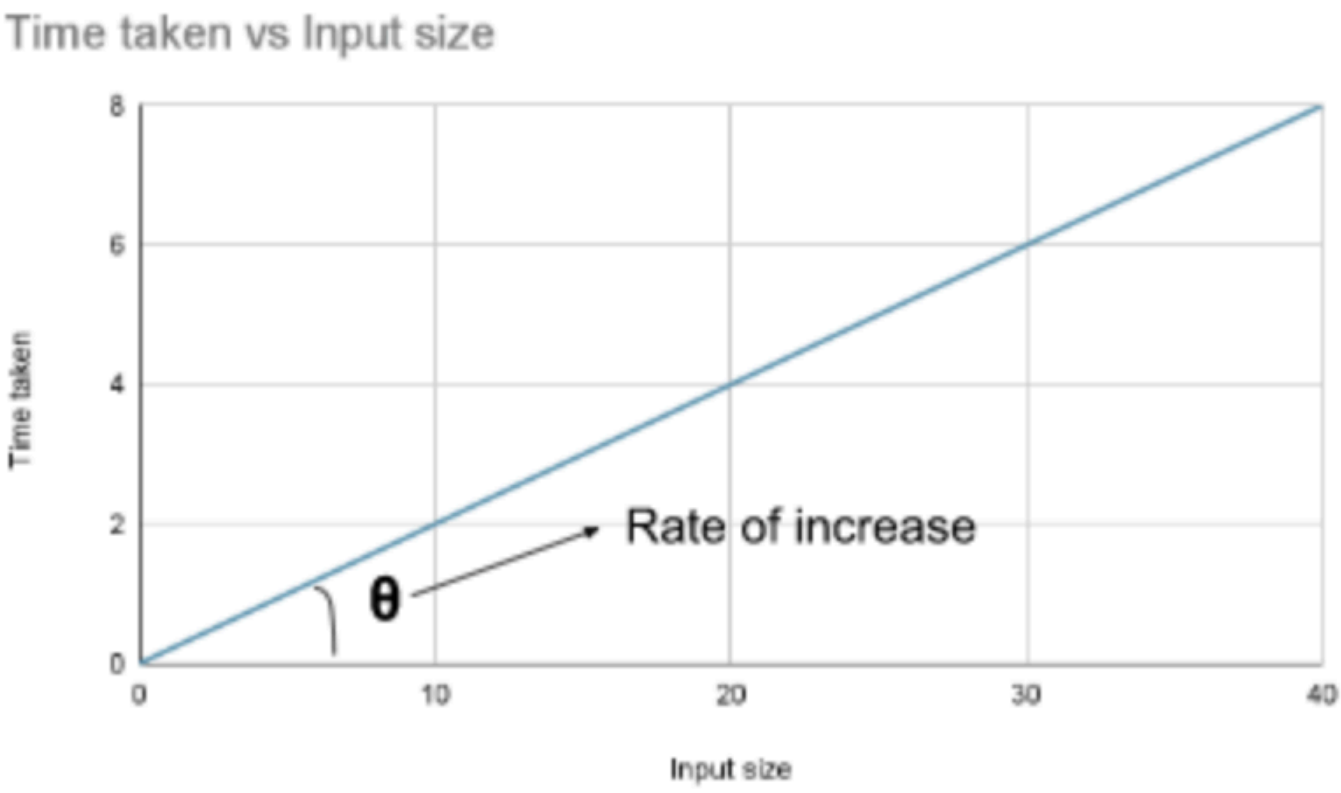
Ksama Arora

> ***Time Complexity***

Time complexity does not refer to the time taken by the machine to execute a particular code as different machines may take different time based on their configurations.

"The rate at which the time taken increases with respect to the input size is called Time Complexity."

Basically, the time complexity of a particular code depends on the given input size, not on the machine used to run the code. Time complexity is calculated in terms of Big-Oh Notation.



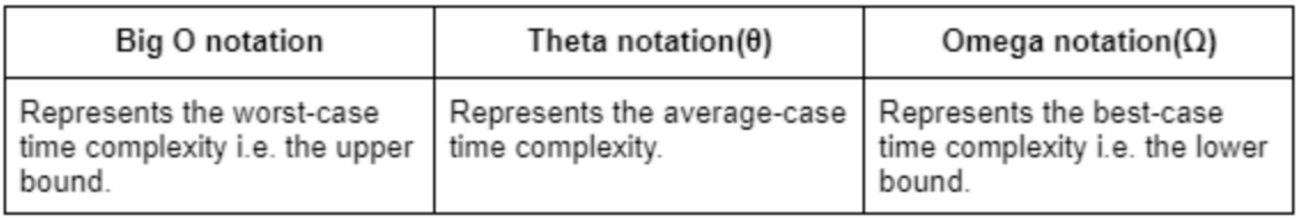
Rules for calculating Time Complexity:

- Time complexity to be computed in terms of worst case scenario
- Avoid constants

- Avoid lower values

Best Case: This term refers to the case where the code takes the least amount of time to get executed. **Worst Case:** This term refers to the case where the code takes the maximum amount of time to get executed. **Average Case:** This term is pretty self-explanatory. This is basically the case between the best and the worst.

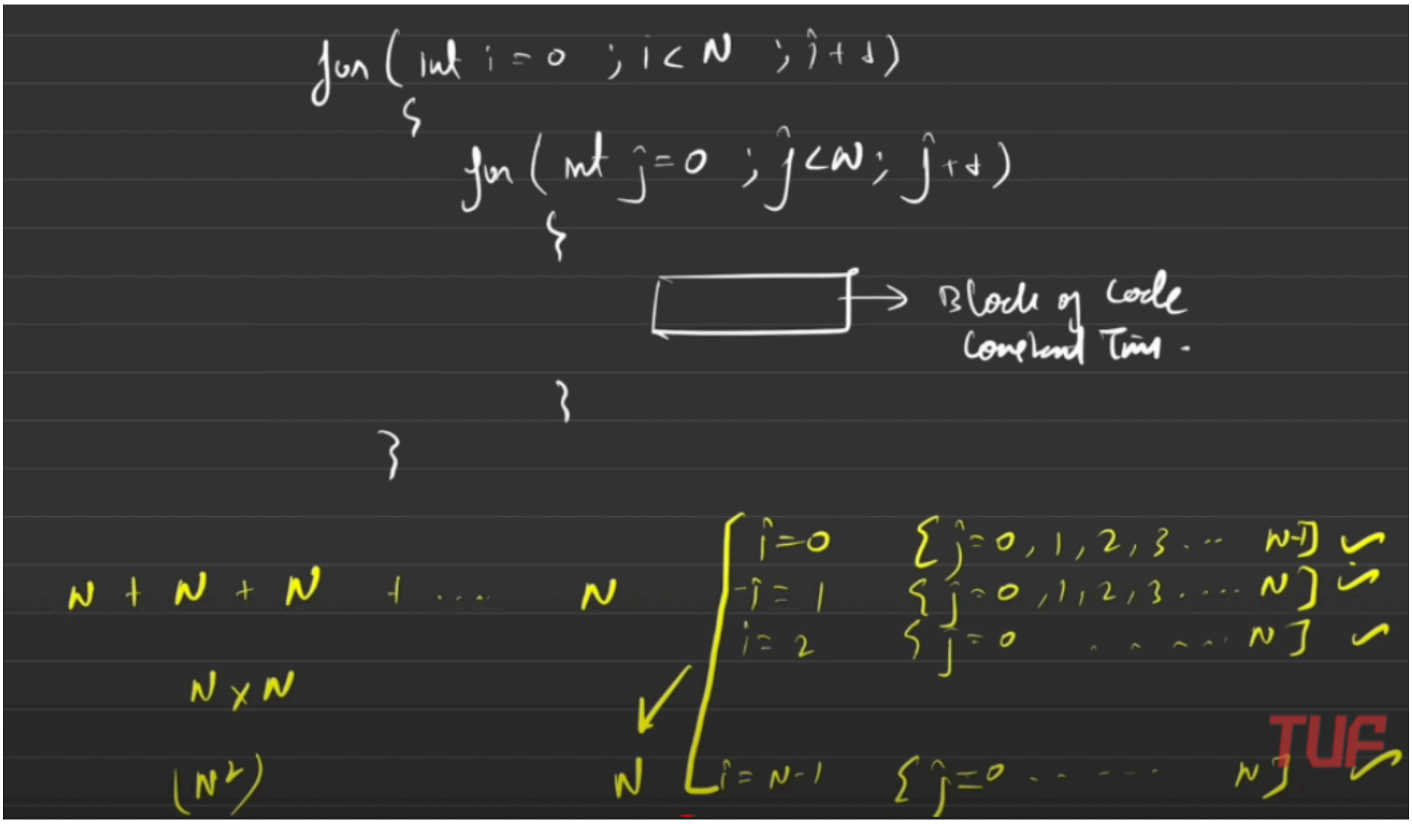
- Notations:



Big O notation	Theta notation(θ)	Omega notation(Ω)
Represents the worst-case time complexity i.e. the upper bound.	Represents the average-case time complexity.	Represents the best-case time complexity i.e. the lower bound.

Refer to Some Examples Here

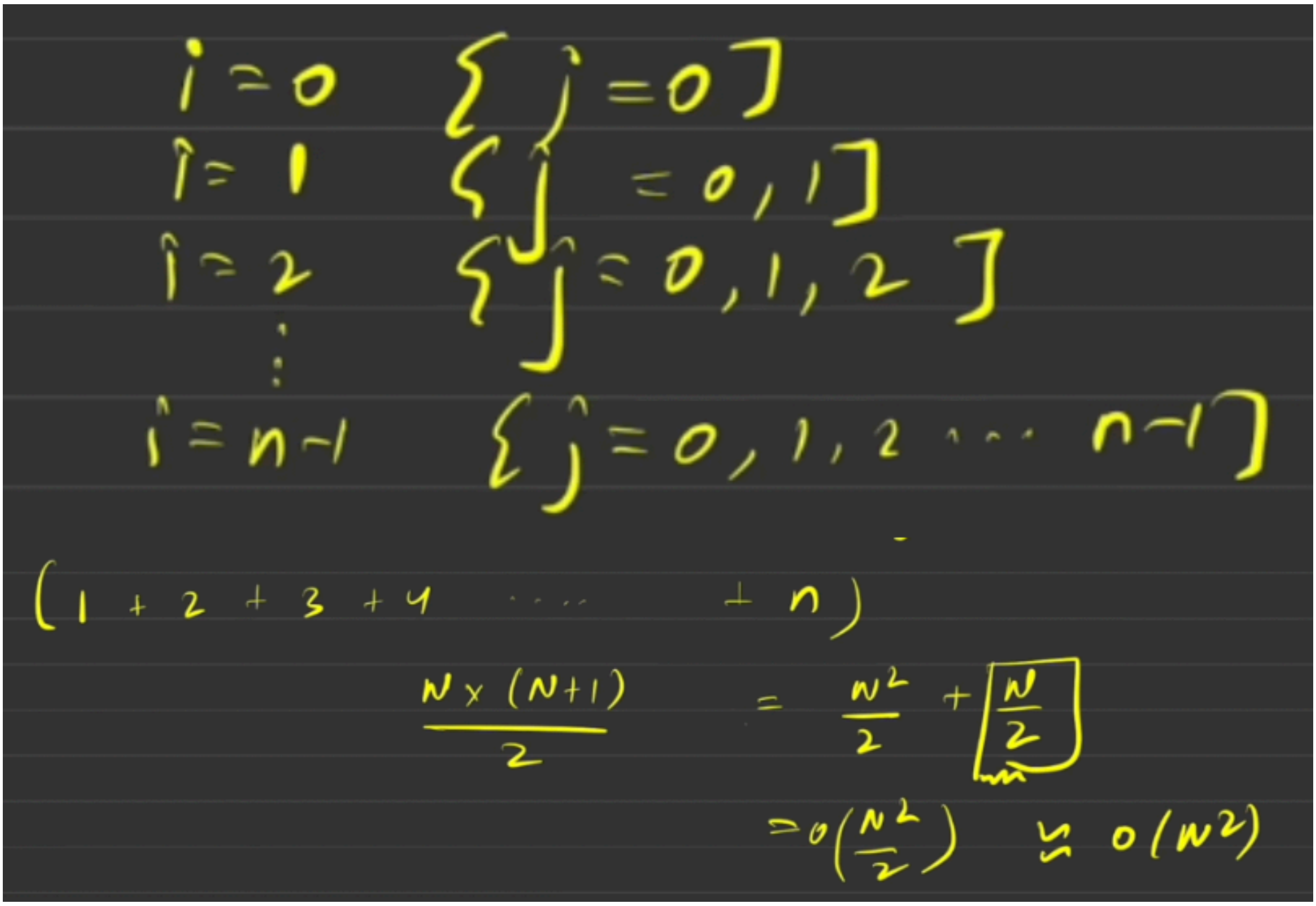
```
// Example 1:
for(int i=0; i<N; i++){
    for(int j=0; j<N; j++){
        // Block of code
    }
}
```



```
for ( int i = 0 ; i < N ; i++)
{
    for ( int j = 0 ; j < N ; j++)
    {
        → Block of code
          Constant Time .
    }
}
```

N + N + N + . . . N
N x N
(N^2)
i=0 {j=0, 1, 2, 3 ... N-1] ✓
i=1 {j=0, 1, 2, 3 .... N] ✓
i=2 {j=0 . . . . . N] ✓
N
i=N-1 {j=0 . . . . . N] ✓
TUF

```
// Example 2:
for(int i=0; i<N; i++){
    for(int j=0; j<i; j++){
        // Block of code
    }
}
```



$i=0 \quad \{j=0\}$

$i=1 \quad \{j=0,1\}$

$i=2 \quad \{j=0,1,2\}$

$\vdots$

$i=n-1 \quad \{j=0,1,2 \cdots n-1\}$

$(1+2+3+4 \cdots + n)$

$$\frac{N \times (N+1)}{2} = \frac{N^2}{2} + \boxed{\frac{N}{2}}$$

$$= O\left(\frac{N^2}{2}\right) \simeq O(N^2)$$

> ### *Space Complexity:*

It is the memory space that your program takes. We use Big-Oh Notation.

```
 Auxiliary Space + Input Space = Space Complexity
// Auxiliary space refers to the space that we use additionally to solve a prol
// Input space refers to the space that we use to store the inputs.
```

- Example 1: int arr[N] Space complexity is O(N)
- Example 2:

```
 Input(a) // Input Space
Input(b) // Input Space
c=a+b // c is Auxiliary Space
// Space Complexity is O(3)
```

NOTE: Never do anything to the input because input data should not be touched i.e. don't do b=a+b.