Time & Space Complexity

↳ what?

slow → High processing → Algorithm A → Fast machine → Fast Algo
& same o/p

low processing → Algorithm B → slow machine → slow algo
&
Fast

Time Complexity

Time Complexity

→ It is the amount of time taken by an algorithm to run

→ as a function of length of the input

Why ?
- → For making better Programs
- → Comparision of Algo

→ Big O notation → Upper Bound

→ Theta Θ → for avg-case complexity

→ Omega Ω → Lower Bound

- → Constant time → $O(1)$
- → Linear time → $O(n)$

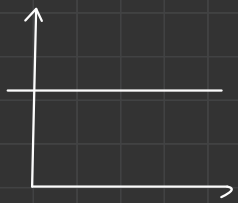
```
for (i=0 → n)
{
  cout << "Hello";
}
```

- → Logarithmic time → $O(\log n)$ → Binary Search
- → Quadratic time → $O(n^2)$
- → Cubic time → $O(n^3)$

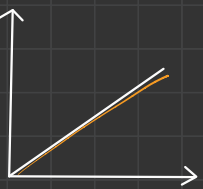
```
for (1 — n)
  for (1 — n)
    for (1 — n)
```

Big-O notation

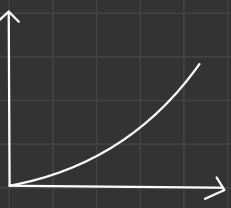
→ O(1)



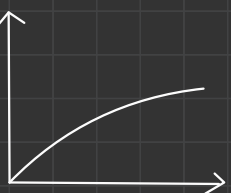
→ O(n)

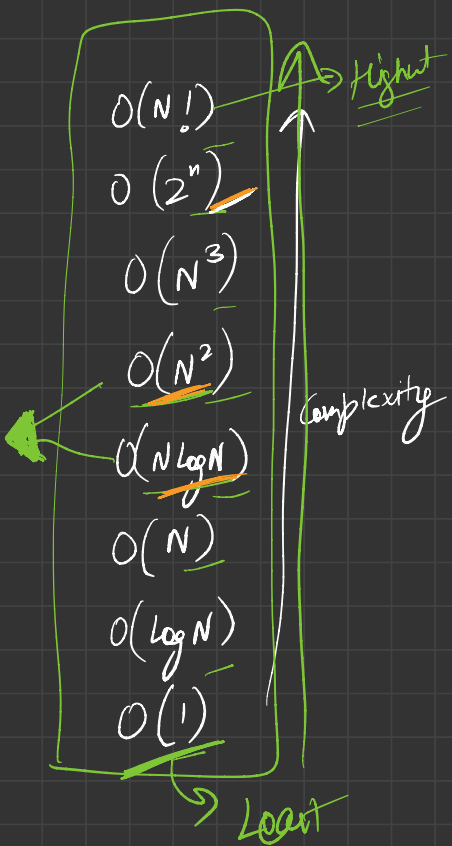


→ O(n²)



→ O(log n)





Question

By - O → Upper Bound

$f(n) \rightarrow 2n^2 + 3n \rightarrow O(n^2)$

$f(n) \rightarrow 4n^4 + 3n^3 \rightarrow O(n^4)$

$f(n) \Rightarrow N^2 + \log N \rightarrow O(N^2)$

$f(n) \Rightarrow 12001 \rightarrow O(1)$

$f(n) \Rightarrow 3n^3 + 2n^2 + 5 \rightarrow O(n^3)$

$f(n) \Rightarrow \frac{n^3}{300} \rightarrow O(n^3)$

$f(n) \Rightarrow 5n^2 + \log n \rightarrow O(n^2)$

$f(n) \Rightarrow \frac{n}{4} \Rightarrow O(n)$

$f(n) = \frac{n+4}{4} \rightarrow O(n)$

} linear time

We will solve recursive time complexities in future videos

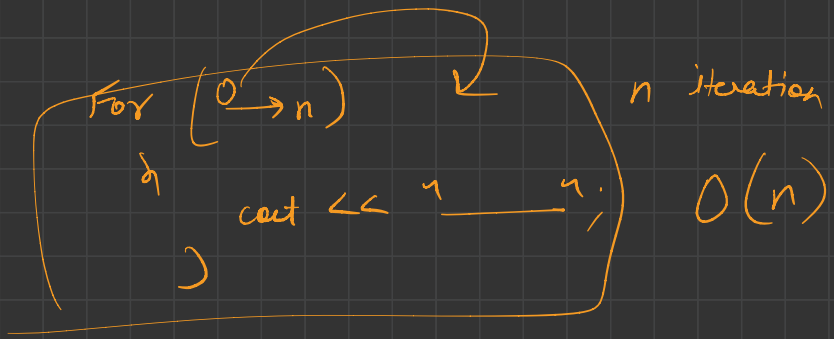
→ Stuck in TLE

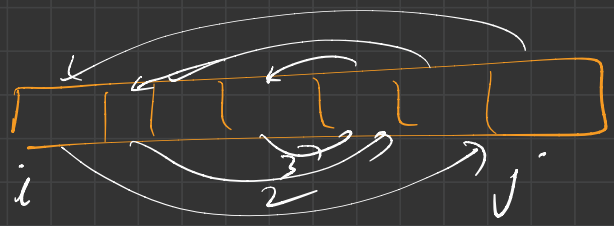
10^8 Operation rule → Most of the modern machine can perform 10^8 Operation / second

Constraints	Time Complexity
$\leq [10..11]$	$O(n!)$, $O(n^n)$
$< [15..18]$	$O(2^n * n^2)$
< 100	$O(n^4)$
< 400	$O(n^3)$
< 2000	$O(n^2 * \log n)$
$< 10^4$	$O(n^2)$
$< 10^6$	$O(n \log n)$
$< 10^8$	$O(n)$, $O(\log n)$

$1 < n < 10^6$

$1 < n < 1000$



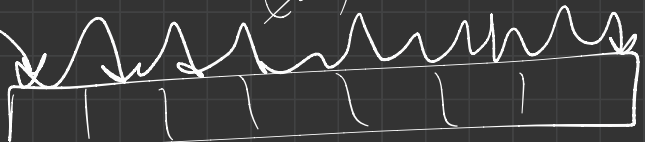


1

n — length array

$\frac{n}{2}$ swap

$O(\frac{n}{2}) \rightarrow O(n)$



n length array

Key

n — comparision

For (0 — n)

$O(n)$

For ( ) → n O(n×m)
{ For ( ) — m
}

O(n) +

For ( )
{ → O(n)
}
For ( ) → O(m)
{
}

O(n+m)

T.C
O(N) → O(N+M)
O(M)

(11) For (0 — N) — N } O(N×N)
{ For (0 — N) — N } O(N²)
}

O(N²) + O(N)

For (0 - N) → O(N)
{
}

O(N²)

For (0 → N) → N → O(N²)

{

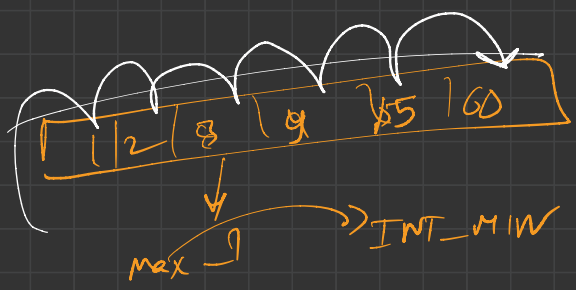
For (N → i) → N

N → 0

{

}

}



For (0 — n) O(n)

O(n-2)

O(n)

For ( )

{

}

For (0 — n) → O(n) ×

{

fun (arr, n);

}

O(1)

Space Complexity :-

↳ memory → Space

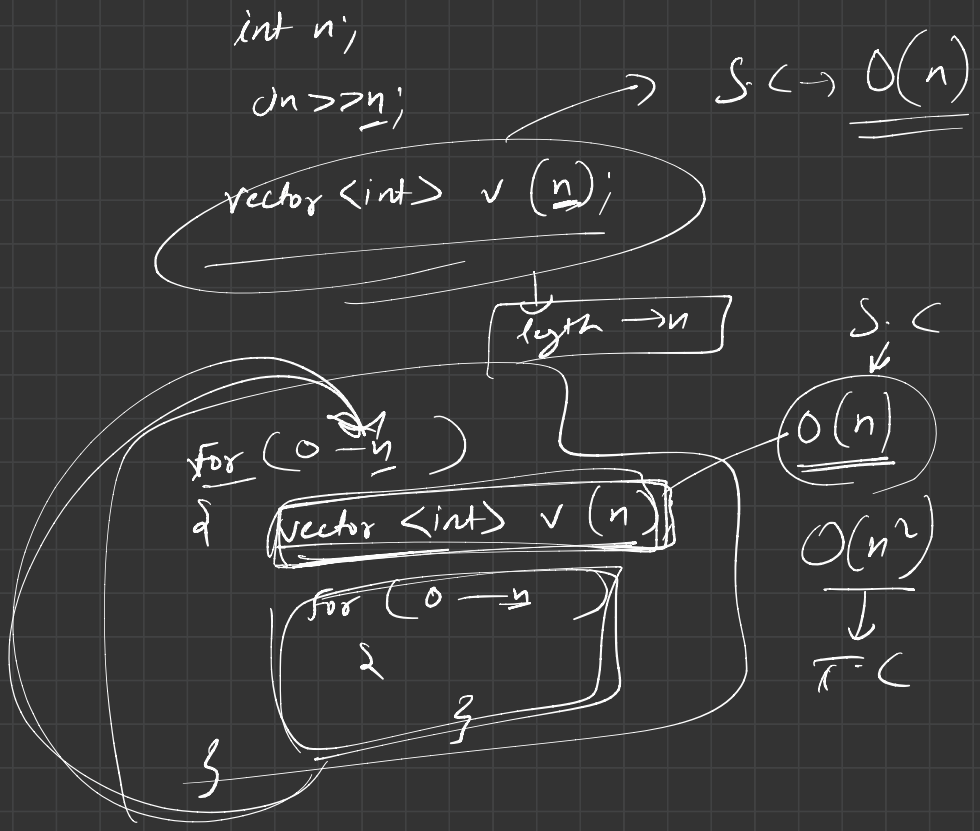
int a, int b ; → O(1)

O(1) =

S.C → O(1)

```
func ( )
{   int arr [5] = {1, 2, 3, 4, 5};
}
```

O(1)


int n;
cin >> n;
S.C → O(n)
vector <int> v (n);
log n → n
S.C
O(n)
For (0 — n)
{ vector <int> v (n)
O(n²)
for (0 — n
{
}
}
T.C

O(n) → Naa
aage bolo
or btao

O(log n) → Kya IQ
Archer