Time and Space Complexity

Time Complexity:

Amount of time taken by an algorithm to run as a function of the length of input.

Why? → for making better programs
→ comparison of algo

Big O notation → Theta Θ → Omega Ω
- Big O notation ↓ upper bound
- Theta Θ ↓ for avg case complexity
- Omega Ω ↓ lower bound

Constant time → O(1) → for (int i=1; i<11; i++){
cout << "Hello";
}

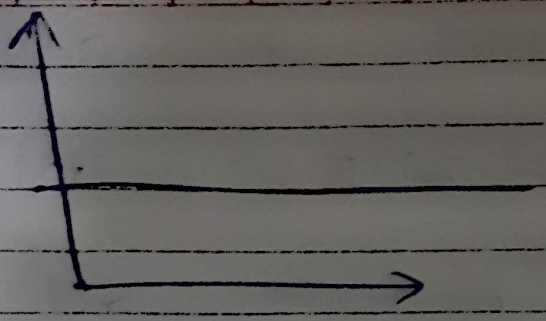
Linear time → O(n) → for (int i=1; i<n; i++){
cout << "Hello";
}.

Logarithmic time → O(logn) → Binary Search

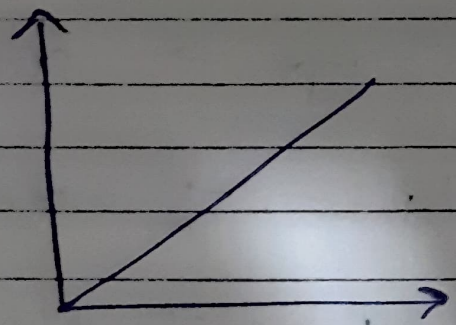
Quadratic time → O(n²) → for (1 → n)
↳ for (1 → n)

Cubic time → O(n3) → for (1 → n)
↳ for (1 → n)
↳ for (1 → n)

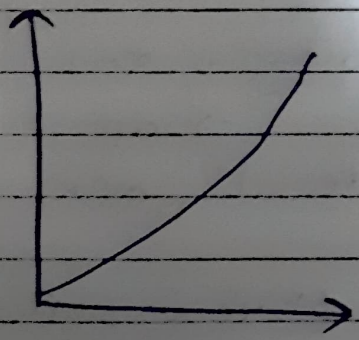
→ O(1)



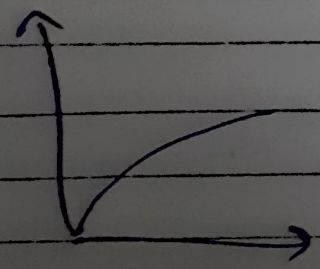
→ O(n)

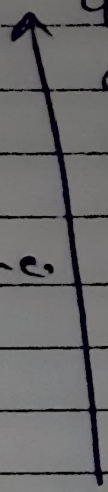


→ O(n²)



→ O(log n)





Questions:

(1) → $2n^2+3n$ → $O(n^2)$

→ $4n^4+3n^3$ → $O(n^4)$

→ $n^2+\log n$ → $O(n^2)$

→ 1200 → O(1)

→ $3n^3+2n+5$ → $O(n^3)$

→ $\frac{n^3}{350}$ → $O(n^3)$

→ $5n^2+\log n$ → $O(n^2)$

→ $n/4$ → $O(n)$

→ $\frac{n+4}{4}$ → $O(n)$

Eg: Print array → O(n)
Reverse → O(n)
Linear search O(n)

Stuck in TLE:

10^8 operation max: Most of modern machines can perform 10^8 operations/sec

Time Complexity

Constraints		
	< [10..11]	O(n!), O(n^6)
	< [15..18]	$O(2^n * n^2)$
$10 < n < 10^6$	< 100	$O(n^4)$
$1 < n < 100$	< 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)

Space complexity:

↳ memory taken by an algorithm.

```
int a, b;      → O(1).

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

int n;
cin >> n;                    → O(n)
vector <int> v(n);

for (0 → n)
 { vector <int> v(n);
      for (0 → n)                → O(n).
       {

       }
 }
```

Binary Search   cond^n → element should be in monotonic function

3	5	9	13	27

key = 13.

a = ~~8~~ = 13
↓
x

1371

3	4
13	27

mid = (start + end) / 2

= 7/2

= 3

13 == 13

→ return True → return index

↓

3

→ Steps:

1. Search mid
2. Compare with mid
3. == → return index

!= → search direction

L-search → 1000 values → array

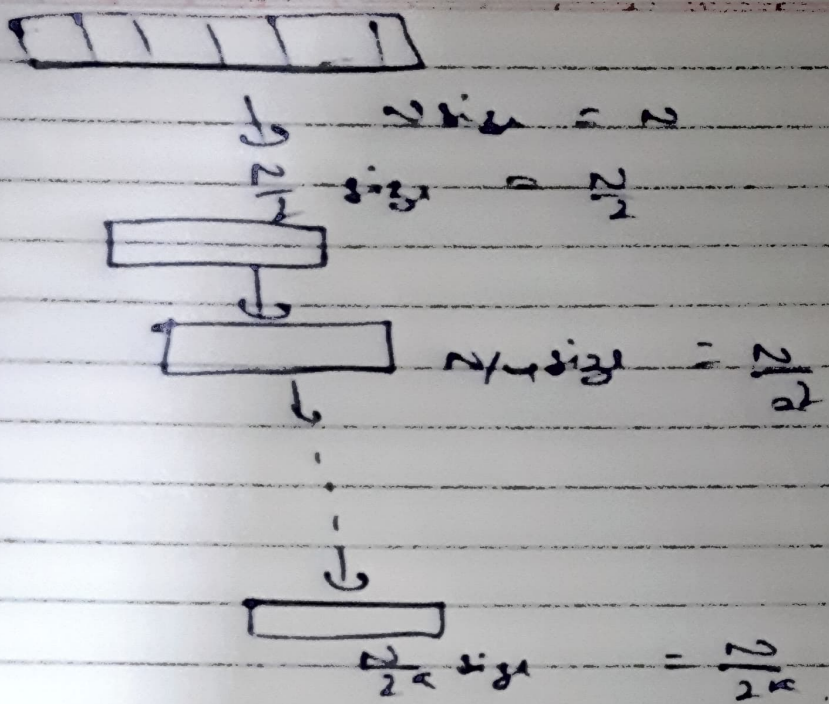
worst case → 1000 comparisons → O(n)

B.S → 1000 values → sorted array

1000 size
↓
500
↓
250
↓
125
↓
62
↓
31
↓
15
↓
7
↓
3
↓
1
↓
0

10 comparisons

→ O(log n)



$\frac{n}{2^k} = 1$

$n = 2^k$

$k = \log n$

so, $O(\log n)$.