Time and Space Complexity

eg. error opening file eg invalid input data
here → eg. element not found

What is time complexity?

reference ex. →

Old Computer	M1 macbook
data: 1,00,000 elements in array	"
Algorithm, linear search for target that doesn't exist in array	"
Time taken → 10 sec	Time: 1 sec.

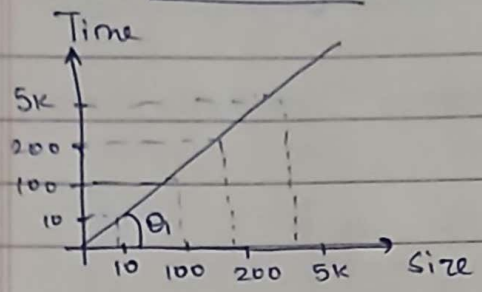
Both have same Time Complexity i.e. O(n)

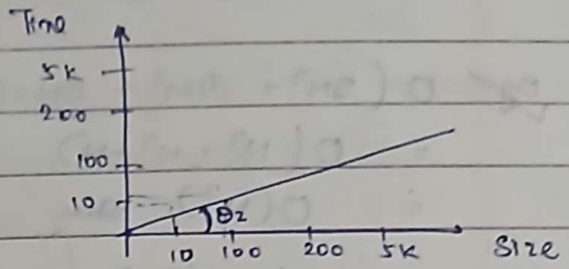
because [Time Complexity ≠ Time Taken]

It's basically gf a function that gives us the relationship about how the time will grow as input grows

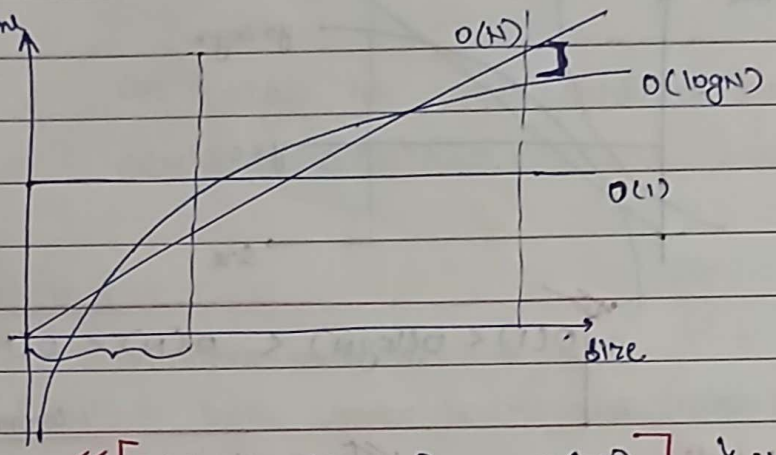
Space complexity = original space that input req. + auxiliary space (extra)

Old machine → O(N) ← M1 macbook





Why?



$O(1) < O(\log N) < O(N)$ } ∵ always considered after large time (worst case)

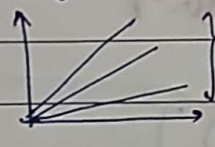
Smallest time complexity — largest

Q. What do we consider when thinking about complexity.

(1) Always consider worst case complexity.

(2) Always look at complexity for large/∞ data.

(3)



$y = x$
$y = 2x$
$y = 4x$

Even the value of actual time is diff. they are all growing ~~together~~ linearly.

We don't care about actual time.

This is why we ignore all constants.

(4) $O(N^3 + \log N)$

from pt. (2) considering 1 million

$1 \text{ mil} = (1\text{mil})^3 + \log(1\text{mil})$

$= (1\text{mil})^3 + 6$ sec.

very small Hence ignored

$\therefore O(N^3)$ ← Ans.

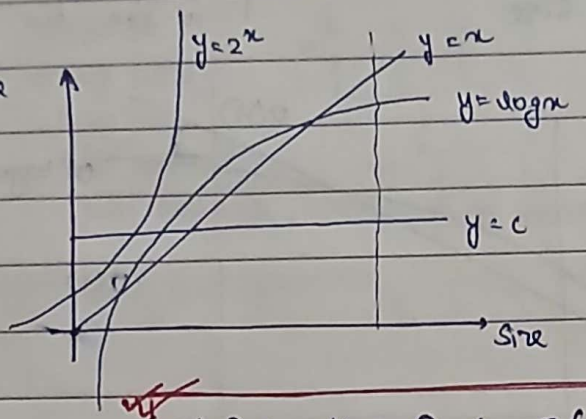
* Always ignore less dominating terms.

e.g. $O(3N^3 + 4N^2 + 5N + 6)$

$= O(N^3 + N^2 + N)$

$= O(N^3)$ ans.

* e.g.



$O(1) < O(\log N) < O(N) < O(N \log N) < O(2^N)$

↓ common sense

complexity $\propto \frac{1}{\text{optimization}}$

* Asymptotic Notations

Big-Oh Notation

word definition: e.g. $O(N^3)$ → upperbound

this means time complexity will never exceeds N^3. e.g. N^2 ✓, $N^3 \log N$ ✗

∴ time complexity $\leq N^3$

mathematical analysis:-

$$f(N) = O(g(N))$$

$$\lim_{N \to \infty} \frac{f(n)}{g(n)} < \infty$$

e.g. $O(\underbrace{N^3}_{g(N)}) = O(\underbrace{6N^3 + 3N + 5}_{f(N)})$

$$\lim_{N \to \infty} \frac{6N^3 + 3N + 5}{N^3} = \lim_{N \to \infty} 6 + \frac{3}{N^2} + \frac{5}{N^3} = \underbrace{6}_{\text{finite}} < \infty$$

Big Omega Notation

word definition: opposite of Big Oh. eg. $\Omega(N^3)$ → lower bound.
i.e. time complexity $\geq N^3$.

$$\therefore \left[\lim_{N \to \infty} \frac{f(N)}{g(N)} > 0 \right]$$

Now what if an algo. has upper bound and lower bound as (N^2).
= $O(N^2)$ & $\Omega(N^2)$.

Theta Notation

↳ combines both $O()$ & $\Omega()$.

e.g. $\Theta(N^2)$ = both upper bound and lower bound is N^2.

$$\left[0 < \lim_{N \to \infty} \frac{f(N)}{g(N)} < \infty \right]$$

little Oh. notation

↳ This gives strictly upper bound (no equality.)

i.e.	Big oh	little o
⇒	iff $f = O(g)$	$f = o(g)$
	$f \leq g$	$f < g$ strictly slower

$$\therefore \left[\lim_{N \to \infty} \frac{f(N)}{g(N)} = 0 \right]$$

e.g. $f = N^2$, $g = N^3$

$$\therefore \lim_{N \to \infty} \frac{f(N)}{g(N)} = \lim_{N \to \infty} \frac{N^2}{N^3} = 0 .$$

little omega

Big Ω	little ω
$f = \Omega(g)$	$f = \omega(g)$
$f \geq g$	$f > g$

$$\therefore \lim_{N \to \infty} \frac{f}{g} > 0$$

- Space Complexity and Auxiliary Space?

Auxiliary space is the extra space or temporary space used by an [algorithm].

Space complexity of an algo. is total space taken by an algo. w.r.t. input size. It includes both Auxiliary space and space used by input.

e.g. If we want to compare standard sorting algo. on the basis of space, then Aux. space would be a better criteria than space complexity.
Merge sort uses → O(n) aux. space,
Insertion & Heap sort uses → O(1) aux space
But space complexity of all these sorting algo is O(n).

```
for (int i=1 ; i<=N) {
        for (j=1 ; j<=K ; j++) {
                // some operation that takes t time
        }
        i = i+K ;
}
```

⇒ inner loop = $O(kt)$ time.

Answer = $O(kt *$ times outer loop is running)

∵ $i = i+k$ ∴ $i = 1, 1+k, 1+2k, 1+3k, 1+4k \ldots, 1+mk$.

$1+mk \leq N$

$mk \leq N-1$

$\left[m \leq \frac{N-1}{k} \right]$ → times the outer loop is running.

∴ Hence, $O\left(kt * \frac{(N-1)}{k} \right)$

$= O(Nt)$ ∵ 1 is constant.

Bubble sort → • Aux space $O(1)$.

• worst and avg. Time complexity $O(n*n)$.

↳ when array is reverse sorted.

• Best Time complexity $O(n)$.

Selection sort → • Worst complexity = $O(n^2)$ = avg. complexity

• Best " = $O(n^2)$

• Space " = $O(1)$.

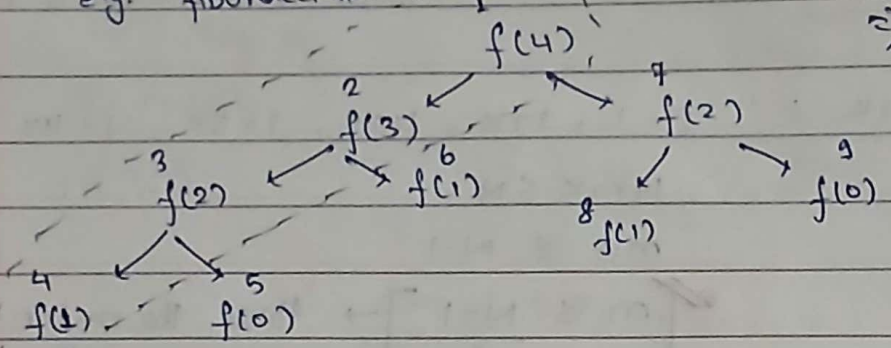
Insertion sort → • Time complexity = $O(n^2)$

• Aux. space = $O(1)$

• Recursion

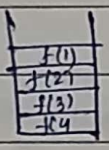
- **Recursive Algorithms :-**

e.g. fiboracci no.



$\Rightarrow O(N)$

Trick: Only calls that are interlinked will be in the stack in same time.



True ek baar m ek stack m ek se jyada func. nhi reh skta.

Space Complexity: Height of Tree.

- **Two types of recursions:-**

① linear

$f(N) = f(N-1) + f(N-2)$

② Divide & conquer

$f(N) = f(N/2) + O(1)$.

I. Divide and conquer recurrence :-

$$\text{Form: } T(x) = a_1 T(b_1 x + \Sigma_1(x)) + a_2 T(b_2 x + \Sigma_2(x)) + \ldots + a_k T(b_k x + \Sigma_k(x)) + g(x).$$

for $x \geq x_0$ (constant)

Ex of Binary search $\Rightarrow$ e.g. $T(N) = T(\frac{N}{2}) + C$.

$\therefore a_1 = 1 \quad C = g(N)$.

$b_1 = 1/2$

$\Sigma_1(N) = 0$

$$\left[T(x) = \left[\sum_{i=1}^{k} a_i T(b_i x + \Sigma_i(x)) \right] + g(x) \right]$$

$\lim_{x \to 0} (\frac{1}{2})^p = 1$

$p = 1 \quad b_2 \neq 1$

$\Rightarrow p = 0$

$p = 0$

$\therefore T(x) = \frac{1}{2} x^p + x^p \left(\int_1^x \frac{g(u)}{u^{p+1}} du \right)$

$= x^p + C x^p (\ln x)^x$

$= x^p + C x^p \ln x \Rightarrow 1 + C \ln x$

$\Rightarrow \log(n)$.

e.g. $T(N) = 9T\left(\frac{N}{3}\right) + \frac{4}{3}T\left(\frac{5N}{6}\right) + 4N^3$

(a_1, b_1, a_2, b_2, $g(N)$)

e.g. $T(N) = 2T\left(\frac{N}{2}\right) + (N-1)$

(a_1, b_1, $g(N)$)

e.g. Merge sort → 1/2 sorted + 2nd half sorted + together adjusted.

$\therefore T(N) = T(N/2) + T\left(\frac{N}{2}\right) + (N-1)$.

$= 2T\left(\frac{N}{2}\right) + \underbrace{(N-1)}_{g(N)}$ → recurrence rel^n of merge sort.

$a_1 = 2,\ b_1 = 1/2,\ g(N) = N-1$

How to solve? to get complexity.

1. Plug and chug

$F(N) = F(N/2) + C$

2. Master's Theorem

Akra Bazzi (1996).

Akra Bazzi :-

$$T(x) = \Theta\left(x^p + x^p \int_1^x \left(\frac{g(u)}{u^{p+1}}\right) du\right)$$

and $a_1 b_1^p + a_2 b_2^p + \dots + a_n b_n^p = 1$

i.e. $\sum_{i=1}^{k} a_i b_i^p = 1$

get 'p' with hit and trial

e.g. $T(N) = 2T\left(\frac{N}{2}\right) + N - 1$ i.e. Time Complexity of Merge Sort?

Sol^n: $g(n) = n-1$, $a_1 = 2$, $b_1 = 1/2$

$\therefore a_1 b_1^p = 1 = 2 \times \left(\frac{1}{2}\right)^p = 1$

by Hit & Trial (easy) $\Rightarrow p = 1$.

Putting P in formula :-

$$T(x) = \Theta\left(x^1 + x^1 \int_1^x \frac{u-1}{u^2}\, du \right)$$

or $$T(n) = \Theta\left(n + n \int_1^n \frac{u-1}{u^2}\, du \right)$$

$$= \Theta\left(n + n \left[\int_1^n \frac{du}{u} - \int_1^n \frac{du}{u^2} \right] \right)$$

$$= \Theta\left(n + n \left[\log u + \frac{1}{u} \right]_1^n \right)$$

$$\boxed{T(n) = \Theta(n \log n)}$$ Ans.

Time Complexity.

Hence, for array of size N : Merge sort complexity $= O(N \log N)$ Ans.

Q. Khelte raho

$$T(N) = 2T\left(\frac{N}{2}\right) + \frac{8}{9} T\left(\frac{3N}{4}\right) + N^2$$

$$= 2 \times \left(\frac{1}{2}\right)^p + \frac{8}{9} \times \left(\frac{3}{4}\right)^p = 1$$

$p = 2$ $\therefore$ $2 \times 1/4 + \frac{8}{9} \times \frac{9}{16} = 1/2 + 1/2 = 1$ ✓

$$\therefore T(n) = \Theta\left(n^2 + n^2 \int_1^n \frac{v^2}{v^3}\, dv \right)$$

$$= \Theta\left(n^2 + n^2 \ln n \right)$$

↓ less dominating

$$= \Theta(n^2 \ln n).$$

• If you can't find value of p :-

e.g. $T(n) = 3T\left(\frac{n}{3}\right) + 4T\left(\frac{n}{4}\right) + n^2$.

method ★ :- put $p = 1$ $\quad 3 \times \frac{1}{3} + 4 \times \frac{1}{4} \neq 1$

find shortest range of p. and focus on power of g.

$2 > 1$

mtlb increase the denominator.

$\therefore \boxed{p > 1}$

now, $p = 2 \Rightarrow 3 \times 1/9 + 4 \times 1/16 = 7/12 < 1$

$\therefore \boxed{p < 2}$

★★★ note :- when $p <$ power of $(g(n))$

then answer $= g(n)$.

here $g(n) = n^2$ & $p < 2$

$\therefore$ [ans. $= \Theta(n^2) = \Theta(g(n))$.]

Proof :-

$$T(n) = \Theta\left(n^p + n^p \int_1^n \frac{u^2}{u^{p+1}} du\right)$$

$$= \Theta\left(n^p + n^p \int_1^n u^{1-p} du\right)$$

$$= \Theta(n^p + n^2) \quad \therefore \Theta(n^2) \text{ Ans.}$$

($\because p < 2$ $\therefore$ less dominating)

II. Solving linear recurrences :-

Homogenous eq^n's.

eg. $F(N) = F(N-1) + F(N-2)$.

form :- $f(x) = a_1 f(x-1) + a_2 f(x-2) + \dots + a_n f(x-n)$.

$$\boxed{f(x) = \sum_{i=1}^{n} a_i f(x-i)}$$

$a_i, n \rightarrow$ fixed

n = order of recurrence

no 'b' here (ofc.)

↓

straight div & conq. bo jata.

Solution for ∄ Fibonacci no. :-

$$f(n) = f(n-1) + f(n-2).$$

Steps :-

1. Put $\boxed{f(n) = \alpha^n}$ for some constant α.

$$\therefore \alpha^n = \alpha^{n-1} + \alpha^{n-2}$$

also called Characteristic Eq^n : ⟵ $\alpha^n - \alpha^{n-1} + \alpha^{n-2} = 0$

$$\Rightarrow \underbrace{\alpha^2 - \alpha - 1 = 0}_{\text{roots of this quad. eq}^n \text{ using } \Rightarrow \text{SriDharacharya rule.}} \quad \because \text{Divided by } \alpha^{n-2}.$$

$$\alpha = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-1 \pm \sqrt{1+4}}{2}$$

$$\therefore \alpha_1 = \frac{1+\sqrt{5}}{2}, \quad \alpha_2 = \frac{1-\sqrt{5}}{2}$$

2. $\boxed{f(n) = c_1 \alpha_1^n + c_2 \alpha_2^n}$ is eq^n for fibonacci.

$$\Rightarrow f(n) = c_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + c_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$$

use given values to find constants

3. Fact: (we know that in case of Fibonacci no.)
* [no. of roots = no. of ans. you have already]

Here, we have 2 roots α_1 & α_2

Hence, we should have 2 ans. already.

i.e. $F(0) = 0$, $F(1) = 1$ ∵ fibo nos.

$\therefore f(0) = 0 = C_1 + C_2$ ∵ put $n = 0$

$\therefore C_1 = -C_2$

$f(1) = 1 = C_1\left(\frac{1+\sqrt{5}}{2}\right) + C_2\left(\frac{1-\sqrt{5}}{2}\right)$

∴ On solving,

$$C_1 = \frac{1}{\sqrt{5}}, \quad C_2 = -\frac{1}{\sqrt{5}}.$$

Putting this in $f(n)$, we get :-

$$\left[f(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

↳ [Formula for n^{th} fibonacci no.]

$$\Rightarrow f(n) = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

↳ as $n \to \infty$ this will be zero ∴ less dominating

also formula same gives same result now also → $\left[f(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n\right]$.

```
(Math.pow((1+Math.sqrt(5))/2,n)
/Math.sqrt(5));
```

Hence,

Time complexity = $O\left(\left(\frac{1+\sqrt{5}}{2}\right)^N\right)$.

↓ also called

[Golden ratio in mathematics]

$T(N) = O(1.6180)^n$.

In case of equal roots :-

e.g. $f(n) = 2f(n-1) + f(n-2)$.

1. $f(n) = \alpha^n$

$\Rightarrow \alpha^n = 2\alpha^{n-1} + \alpha^{n-2}$

$\Rightarrow \alpha^2 - 2\alpha + 1 = 0 \Rightarrow (\alpha-1)^2 = 0$

$\therefore (\alpha = 1, 1)$

General case : If α is repeated r times then $\alpha^n, n\alpha^n, n^2\alpha^n, \ldots, n^{r-1}\alpha^n$ are all sol^n of the recurrence.

i.e. If 3 roots $\Rightarrow \alpha^n, n\alpha^n, n^2\alpha^n$

Hence, here we will take 2 roots as $n\alpha^n$.

$\therefore \Rightarrow [f(n) = c_1\alpha^n + c_2 n\alpha^n]$

$\because \alpha = 1$

$\Rightarrow f(n) = c_1 + nc_2$

if not, then this will leads to $c_1 + c_2 = 0$ and $c_1 + c_2 = 1$ which is not possible!

$f(0) = 0 \quad \& \quad f(1) = 1$

$\therefore c_1 = 0, \quad c_1 = 1$

Ans. $f(n) = n \Rightarrow$ Time complexity $= O(N)$.

- **Non-Homogenous linear recurrences**

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \ldots + a_d f(n-d) + g(n)$$

when this extra function is there then it is non-homogenous linear eqn.

- How to solve?

Steps:-

(1) Replace $g(n)$ by 0 & solve usually.

e.g.

$f(n) = 4f(n-1) + \underbrace{3^n}_{0}$, $f(1) = 1$.

$\therefore f(n) = 4f(n-1)$

$\alpha^n = 4\alpha^{n-1}$

$\alpha - 4 = 0$

$[\alpha = 4]$

Homogenous sol^n. $\Rightarrow$ $[f(n) = C_1\alpha^n = C_1 4^n]$

(2) Take $g(n)$ on one side & find particular sol^n.

$f(n) - 4f(n-1) = 3^n$.

Guess something that is similar to $g(n)$.

if $g(n) = n^2$, guess a poly. of degree 2. $\rightarrow \underbrace{an^2}_{f(n)}$

$\therefore$ if $g(n) = 3^n$ $\therefore \rightarrow f(n) = c3^n$

$\therefore c3^n - 4c3^{n-1} = 3^n \Rightarrow \underline{c = -3}$.

$\therefore$ Particular sol^n. $\Rightarrow$ $[f(n) = -3^{n+1}]$

(3) Add both sol^n together.

$f(n) = C_1 4^n + (-3^{n+1})$

$\because (f(1) = 1) \Rightarrow C_1 4 - 3^2 = 1$

$C_1 = \frac{5}{2}$

$\therefore \left[f(n) = \frac{5}{2}(4^n) - 3^{n+1}\right]$ Ans.

How do we guess particular soln:- ?

↓

→ If $g(n)$ is exponential, guess of
ie. e.g. $[g(n) = 2^n + 3^n]$

↓

∴ guess :→ $[f(n) = a2^n + b3^n]$

→ If n → poly. of some degree

e.g. $g(n) = n^2 - 1$

guess → $an^2 + bn + c = f(n)$ // generalized

e.g. $g(n) = n^3 + 4$

guess → $an^3 + bn^2 + cn + d = f(n)$.

e.g. $g(n) = 2^n + n$

$f(n) = a2^n + (bn + c)$

Note :- if we guess $f(n) = a2^n$ & if it fails
then try $(an+b)2^n$, if this also fails
increase the degree $(an^2+bn+c)2^n$. Ily.
follow for others.

Hinted next Q.

* Q. $f(n) = 2f(n-1) + 2^n$, $f(0) = 1$.

Soln → ① Put $2^n = 0$

$f(n) = 2f(n-1)$

$\alpha^n = 2\alpha^{n-1}$

$\therefore \alpha = 2$ ∴ vvvv easy

② $g(n) = 2^n$

guess → $f(n) = a2^n$

Tip:- Revision is very important!
+
Khud se practice nhi kiya impact hoga ye semester exam jitna aasan nhi hai!

$\therefore \quad a2^n = 2a\,2^{n-1} + 2^n$

$a = a+1 \quad \times \therefore$ Wrong

$\therefore$ Guess another $\Rightarrow f(n) = (an+b)2^n$

$\Rightarrow (an+b)2^n = 2\,(a(n-1)+b)2^{n-1} + 2^n$

$\Rightarrow \boxed{a = 1}$ and Discard $\to b$

$\therefore f(n) = n2^n \rightarrow$ Particular sol^n.

(3) General answer :-

$f(n) = C_1 2^n + n2^n$

$f(0) = 1 = C_1$ we given value in general ans.

$\therefore \boxed{f(n) = 2^n + n2^n}$ Ans.
(less dominant)

$\boxed{\text{Complexity} = O(n2^n)}$ Ans.

#BitManipulation

$n \& 1 = n.$

XOR = exclusive or = if and only if.

$a \wedge 1 = \bar{a}$
$a \wedge 0 = a$
$a \wedge a = 0$
} ***

$(21)_8 = (?)_{10}$

$2 \times 8^1 + 1 \times 8^0 = (17)_{10}$ Ans