Space & Time Complexity

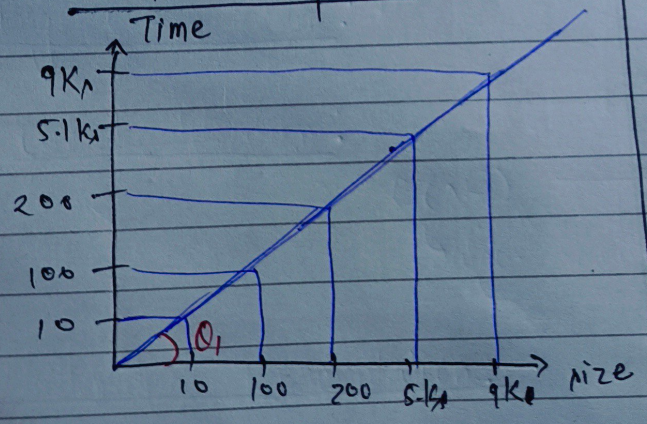
→ What is Time Complexity?

Old Computer	M1 Macbook (very fast).
data - 1,000,000 elements in an array	"
Algorithm, linear search for target that does not exist in array	"
Time ~~Complexity~~ taken 10 sec	Time taken: 1 sec

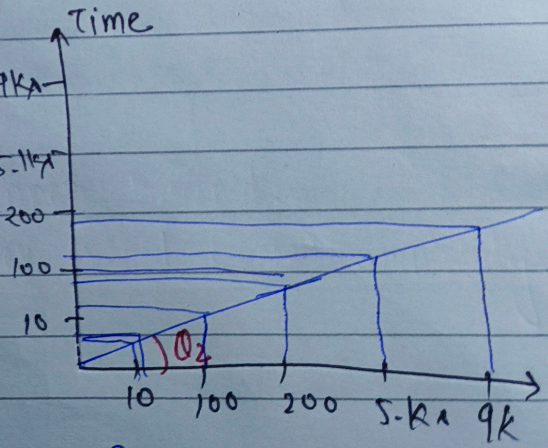
* Both machines have same time Complexity

Conclusion: Time Complexity != Time taken.

Old Computer

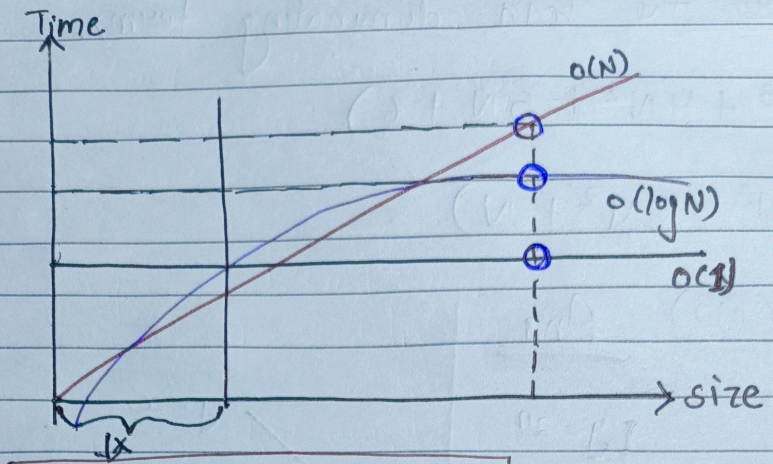


M1 Macbook.



* Function that gives us the relationship about how the time will grow as the input grows.

Why?



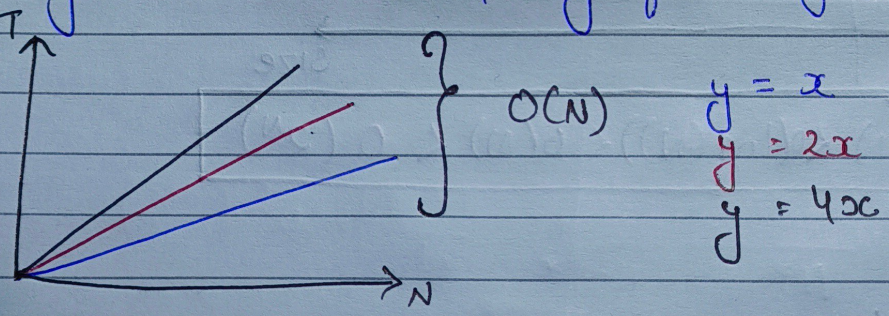
$O(1) < O(\log N) < O(N)$

> What do we need to consider when thinking about complexity.

① Always look after worst case Complexity

② Always look at complexity for large / ∞ data.

③



★ Even though value of actual time is diff. they all are growing linearly.

★ we don't care about actual time

★ This is why, we ignore all constants.

④ $O(N^3 + \log N)$

1 million of array element = $((1\text{ million})^3 + \log(1\text{ million}))$

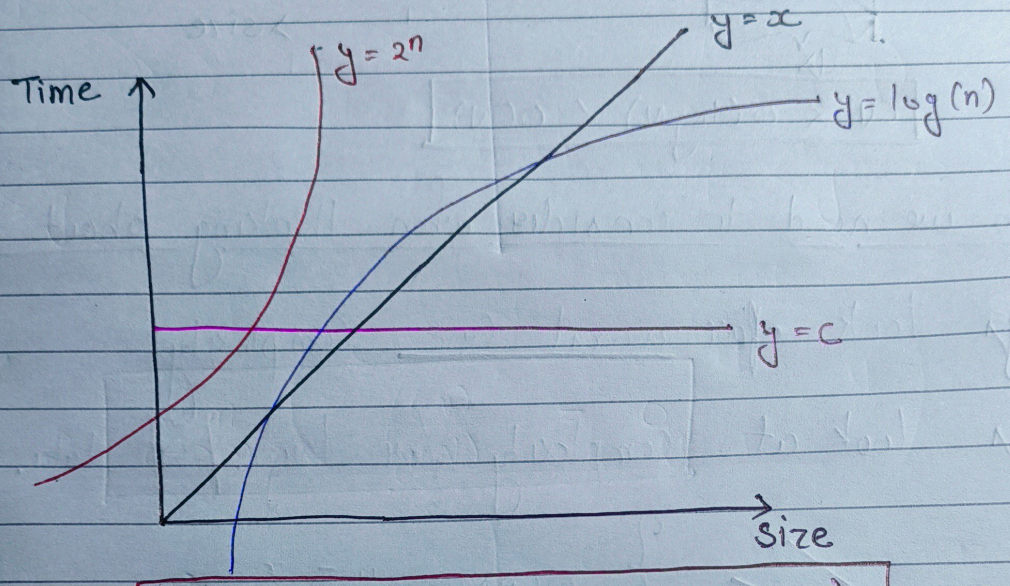
$= (1\text{ million})^3 + \underbrace{6 \text{ sec}}_{\text{ignore}}$

Always ignore the less dominating terms.

Ex:- $O(3N^3 + 4N^2 + 5N + 6)$

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

$= O(N^3)$ Ans



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

Big - O Notation:

Word defination:

$O(N^3)$ ⟶ Upper Bound.

↳ algo. should never exceed $O(N^3)$ complexity.

Mathematical Representation:

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

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

Ex:- $O(N^3) = O(6N^3 + 3N + 5)$
$g(N)$ $f(N)$

$$= \lim_{N \to \infty} \frac{6N^3 + 3N + 5}{N^3}$$

$$= \lim_{N \to \infty} 6 + \frac{3}{N^2} + \frac{5}{N^3} = 6 + \frac{3}{\infty} + \frac{5}{\infty}$$

$$= 6 + 0 + 0$$

$$= 6 < \infty$$ ✓

Big Omega (Ω): opposite of Big O

words: $\Omega(N^3)$ lower Bound

maths: $\lim_{x \to \infty} \frac{f(N)}{g(N)} > 0$

Q: what if an algo. has lb & ub as (N^2)

$= O(N^2)$ & $\Omega(N^2)$

Theta Notation: Combining both

$O(N^2) \Rightarrow$ words: Both ub & lb = N^2

$$0 < \lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty$$

Little O Notation

★ This is also giving Upperbound.

★ Loose Upperbound.

Big O	little O
$f = O(g)$	$f = o(g)$
$f \leq g$	$f < g$ (strictly slower).

★ Math

$$\lim_{n \to \infty} \frac{f(N)}{g(N)} = 0$$

Ex $f = N^2$ $g = N^3$

$$\lim_{N \to \infty} \frac{N^2}{N^3} = \lim_{N \to \infty} \frac{1}{N} = 0$$

little omega

★ This is also giving lowerbound

★ loosely lowerbound.

$f = \omega(g)$ → ~~math~~

word:

Big Ω

$f = \Omega(g)$

$f \geq g$

Math: → $\lim_{N \to \infty} \frac{f(N)}{g(N)} = \infty$

Ex:- $\lim_{n \to \infty} \frac{N^3}{N^2} = \lim_{N \to \infty} N = \infty$.

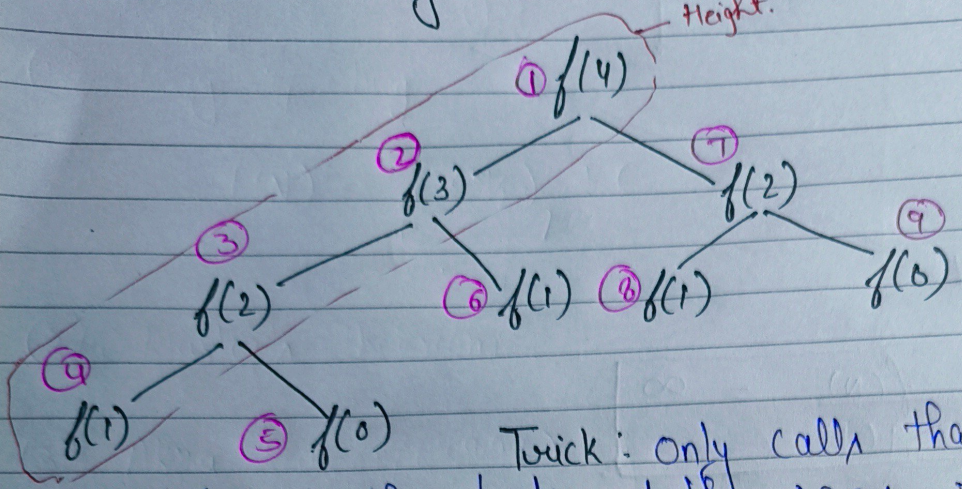
SPACE COMPLEXITY or Auxilary Space.

Auxilary Space is the extra space or temporary space used by an algo.

Space Complexity, ~~if we~~ of an algo. is total space taken by the algo. with respect to the input size. Space complexity includes both Auxilary space & space used by input.

Ex:- if we want to compare standard sorting algo. on the basis of space, then Auxilary space would be a better criteria than space complexity. Merge sort uses $O(n)$ auxilary space, Insertion sort and Heap sort use $O(1)$ auxilary space. Space complexity of all these sorting algo. is $O(n)$ though.

Recursive Algorithm



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

Space Complexity = Height of the tree (path)

2 Types of Recursion

① Linear

→ $f(N) = f(N-1) + f(N-2)$

→ Fibonacci

② Divide & Conquerer.

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

→ Divide & Conquerer.
↳ Binary Search.

① Divide & Conquer Recurrences

Form:

$$T(x) = a_1 T(b_1 x + \varepsilon_1(x)) + a_2 T(b_2 x + \varepsilon_2(x)) + \cdots + a_k T(b_k x + \varepsilon_k(x)) + g(x), \text{ for } x \geq x_0$$

(x_0: sum constant)

$$T(N) = T\left(\frac{N}{2}\right) + C$$

$a_1 = 1$

$b_1 = 1/2$

$\varepsilon_1(n) = 0$

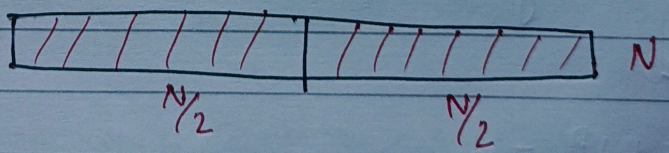
$g(n) = C$

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

(a_1, b_1, a_2, b_2)

$$T(N) = \underbrace{2T\left(\frac{N}{2}\right)}_{a_1,\ b_1} + \underbrace{(N-1)}_{g(N)}$$

↳ when you get answer from this + what you are doing with that answer takes how much time.



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

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

* How to actually solve to get Complexity

① Plug & chug

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

② Master's Theorem

③ Akra Bazzi (1996)

Akra Bazzi formula

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

what is p?

$$a_1 b_1^p + a_2 b^{2p} + \dots = 1$$

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

Ex: $T(N) = 2T(N/2) + (N-1)$

$a_1 = 2$ $\quad g(x) = x - 1$

$b_1 = 1/2$

$$2 * \left(\frac{1}{2}\right)^p = 1$$

$p = 1$

$2 \times \frac{1}{2} = 1$ ✓

put p in formula.

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

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

$$= O\left(x + x \int_1^x \frac{du}{u} - \int_1^x \frac{du}{u^2}\right)$$

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

$$= O\left(x + x\left[\log x - \cancel{\log 1}^{0} + \frac{1}{x} - 1\right]\right)$$

$$= O\left(\cancel{x} + x\log x + 1 - \cancel{x}\right)$$

$$= O(x\log x + 1)$$

$$= O(x\log x)$$ // Time Complexity.

For array of size N: Merge sort Complexity = $O(N \log N)$.

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

(a_1, b_2, a_2, b_2)

finding p

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

↳ put p = 2 (~~find~~ Hit & trial)

$$2 \times \frac{1}{4_2} + \frac{8}{8} \times \frac{8}{16_2} = 1$$

$$\frac{1}{2} + \frac{1}{2} = 1$$

$$1 = 1$$

Hence $p = 2$.

Put p in the formula.

$$T(x) = O\left(x^2 + x^2 \int_1^x \frac{u^2}{u^3} du\right)$$

$$T(x) = O\left(\underbrace{x^2}_{\text{ignore less dominating term.}} + x^2(\log x - 0)\right)$$

$$= O(x^2 \log x) \text{ // Time Complexity.}$$

If we can't find value of P.

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

Let's try $p = 1$

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

$$2 \neq 1$$

$2 > 1 \rightarrow$ Increase the denominator

$\therefore p > 1$

$p = 2$

$$2 \times \frac{1}{9_3} + 4 \times \frac{1}{16_4} = \frac{1}{3} + \frac{1}{4} = \frac{4+3}{12} = \frac{7}{12} < 1$$

$\therefore p < 2$

Note: when $p <$ power of $(g(x))$ then ans $= g(x)$.

Here $g(x) = x^2$

$p < 2$ (i.e. power of $g(x)$)

Hence, ans $= O(g(x))$

↳ x^2 Ans

Solving Linear Recurrences

Ex:- $F(N) = F(N-1) + F(N-2)$

Form:

$$f(x) = a_1 f(x-1) + a_2 f(x-2) + a_3 f(x-3) + \cdots + a_n f(x-n)$$

$$f(x) = \sum_{i=1}^{x} a_i f(x-i), \text{ for } a_i, n \text{ is fixed}$$

n = order of recurrence

Solution: for Fibonacci no.

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

① Put $f(x) = \alpha^n$ for some constant α

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

$\alpha^n - \alpha^{n-1} - \alpha^{n-2} = 0$

$\alpha^2 - \alpha - 1 = 0$

↳ divided by α^{n-2}

$$\frac{\alpha^n}{\alpha^{n-2}} = \frac{\alpha^n \times \alpha^2}{\alpha^n} = \alpha^2$$

$$x^2 - x - 1 = 0$$

roots of this quadratic eqn

$$\alpha = \frac{1 \pm \sqrt{5}}{2}$$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\alpha_1 = \frac{1+\sqrt{5}}{2} \qquad \alpha_2 = \frac{1-\sqrt{5}}{2}$$

(2) $f(n) = c_1\alpha_1 + c_2\alpha_2$ is a soln for fibonacci.
$= f(n-1) + f(n-2)$.

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

(3) No. of roots that we have = no. of ans we already have.

Here we have 2 roots α_1 & α_2.
Here, we should have 2 ans already.

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

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

from (3)

$$1 = c_1\left(\frac{1+\sqrt{5}}{2}\right) - c_1\left(\frac{1-\sqrt{5}}{2}\right)$$

$$c_1 = \frac{1}{\sqrt{5}} \qquad c_2 = -\frac{1}{\sqrt{5}}$$

Putting this in eqn number 2.

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

↳ formula for nth Fibonacci number.

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

↳ as $n \to \infty$ this will be close to 0. Hence this is less dominating term. Ignore this term.

Time Complexity: $O\left(\frac{1+\sqrt{5}}{2}\right)^N$

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

↓ Golden ratio

Code

```
public class Fibofromula{
    public static void main (String [] args ){
        for(int i = 0; i <= 12; i++){
            Sysо (fibo(i));
    }

static int fibo (int n ){
    double phi = (1 + Math.sqrt(5)) /2;
    double phi = (1 - Math.sqrt(5)) /2;

return (int) Math.round((Math.pow (phi,n) - Mathpow
                                (psi,n )) /Math.sqrt(5));
}
}
```

Q Equal Roots

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

(i) $f(x) = \alpha$

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

$$\frac{\alpha^n - 2\alpha^{n-1} - \alpha^{n-2}}{\alpha^{n-2}} = 0$$ both LHS & RHS

$\Rightarrow \alpha^2 - 2\alpha + 1 = 0 \Rightarrow \boxed{\alpha = 1}$ double root

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

Hence I can take two roots $\Rightarrow 1, n\alpha^n \rightarrow 1, n$.

$$f(n) = c_1(1)^n + c_2 n\alpha^n$$

$$= c_1 + c_2 n$$

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

$f(0) = \boxed{0 = c_1}$ | $f(1) = 1 = c_1 + c_2$

$\boxed{c_2 = 1}$

Ans $= f(N) = n \rightarrow$ Time complexity of above relation is $O(N)$.

Non Homogenous Linear Recurrence

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + a_3 f(n-3) + \cdots + a_d f(n-d) + g(n)$$

when this extra f is there, it is non-h.g.l.rec.c.

How to solve

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

$$f(n) = 4f(n-1) + 3^n, \quad f(1) = 4$$

$3^n \to 0$

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

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

$$\alpha^n - 4\alpha^{n-1} = 0$$

$$\alpha - 4 = 0 \qquad \boxed{\alpha = 4}$$

H.g. soln $\Rightarrow$ $f(n) = c_1 \alpha^n$

$\boxed{f(n) = c_1 4^n}$

② Take $g(n)$ on one side & find particular soln:

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

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

If $g(n) = n^2$, guess a polynomial of degree 2.

My guess: $f(x) = c3^n$

$$c3^n - 4c3^{n-1} = 3^4 \Rightarrow -3$$

Particular soln $\Rightarrow f(n) = -3^{n+1}$

③ Add both soln together.

$$f(n) = c_1 4^n + (-3^{n+1})$$

$$f(1) = 1 \Longrightarrow c_1 4 - 3^2 = 1$$

$$\Rightarrow \boxed{c_1 = \frac{5}{2}}$$

$$\boxed{f(x) = \frac{5}{2} 4^n - 3^{n+1}}$$

How do we guess particular soln?

★ If g(n) is exponential, guess of same type.

Ex:- $g(n) = 2^n + 3^n$

guess: $f(n) = a2^n + b3^n$

★ If it is polynomial (g(n)), guess of same degree.

Ex: $g(n) = n^2 - 1 \Rightarrow$ guess of same degree → 2 degree

$$\boxed{ax^2 + bx + c = f(x)}$$

Ex: $g(n) = 2^n + n$

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

Let say you guessed, $f(n) = a2^n$ and if it fail, then try $(an + b)2^n$, if this also fail increase the degree. $(an^2 + bn + c)2^n$.