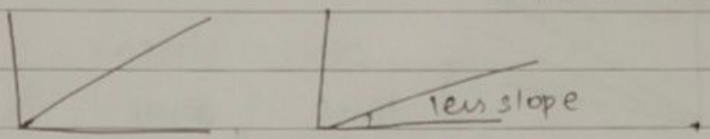
TIME & SPACE COMPLEXITY.

ex: fibonacci numbers.

① What is time complexity ?

linear time complexity.



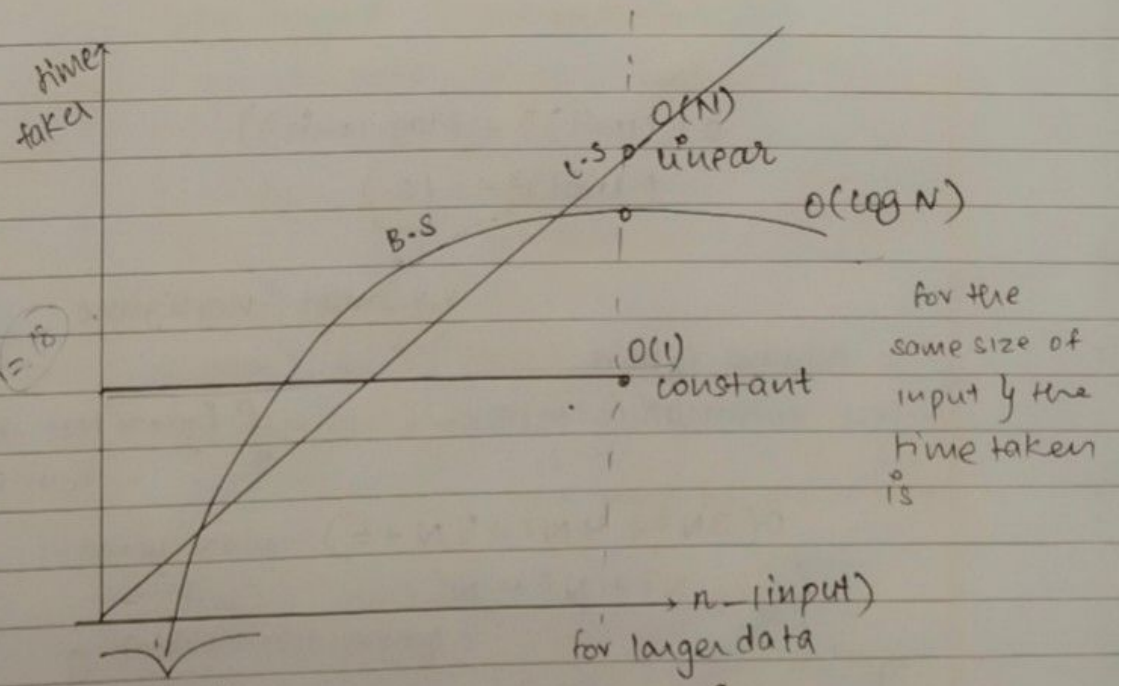
Time complexity ≠ Time taken.

mathematical func = that shows how the time grows ~~as~~ when the input grows

⇒ linear = $y = kx$.

> ★ func that gives us the relationship about how the time grows as input grows

comparing time complexities

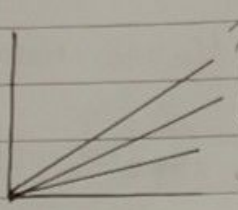


$O(1) < O(\log N) < O(N)$

fastest least time — less time — takes more time

what do we consider when thinking about complexity.

1. Always look for worst case complexity
2. Always look at complexity for large / ∞ data
3.



grows in linear fashion

O(N) ⇒ linear

time complexity } same.

$y = x$

$y = 2x$

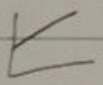
$y = 4x$

★ Even tho values of actual time is different, they are all growing linearly
★ we don't care about actual time
★ this is why we ignore all const.

$x = 0$ y

$y = 3x + 5$

$y = 5$



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

from pt ②

↳ consider large/∞ data

$N = 1$ mil.

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

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

v.v. small / negligible

{ ∴ Always ignore less dominating terms. }

{ ignore less dominating terms

$O(3N^3 + 4N^2 + 5N + 6)$ ignore const.

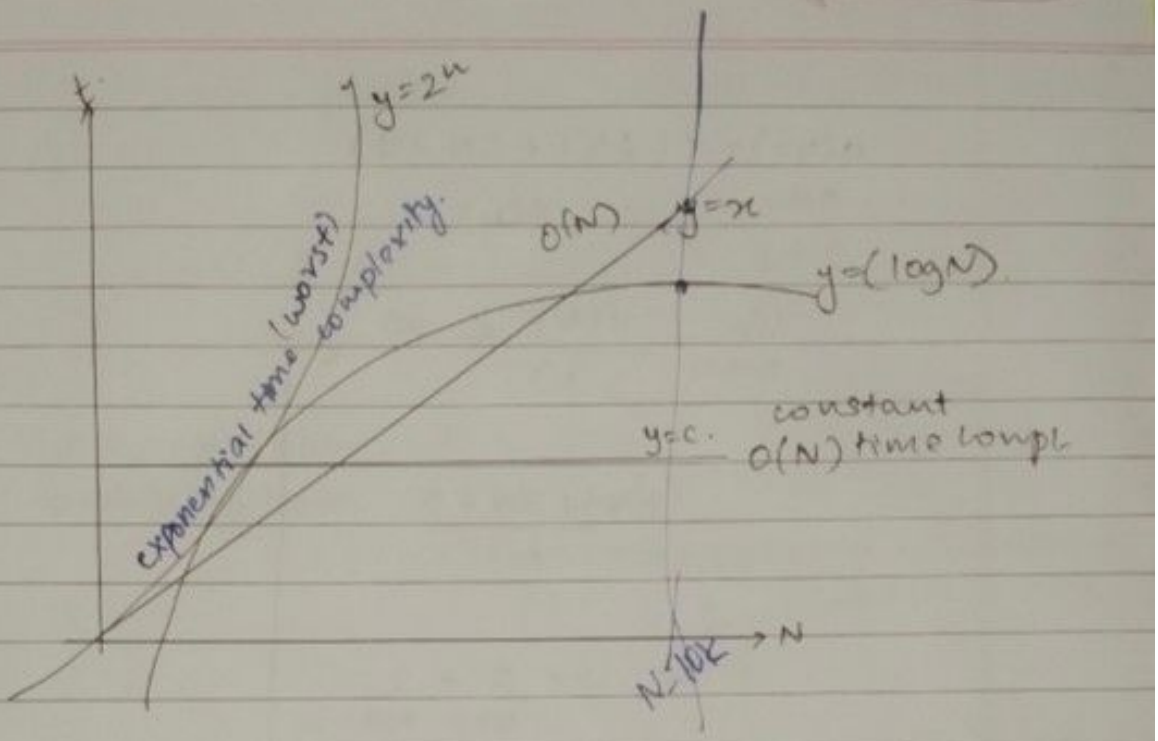
$N^3 + N^2 + N$

ignore less dominating.

equal

$O(N^3)$

↳ ans.



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

worst time complexity
↳ exponential

exponential time complexity worst

=) Big-oh notation

* word definition:

upper bound (not exceed this).

you can never exceed this complexity.

$O(N^3)$ → time grows in N^3 fashion.

can be (N), (N^2)

$O(\log N)$, $O(N^2 \log N)$

$(N^2 \log N \log N) \geq O(N^3)$

can never be more than N^3.

* maths:

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

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

$O(N^3) = O(6N^3 + 3N + 9)$

$g(N)$ of $f(N)$

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

$$\frac{6N^3 + 3N + 9}{N^3}$$

$$6 + \frac{3}{N^2} + \frac{9}{N^3}$$

$N \to \infty$

$$6 + \frac{3}{\infty} + \frac{9}{\infty}$$

$\frac{3}{\infty} \to 0$, $\frac{9}{\infty} \to 0$

$6 < \infty$

↳ finite value

↳ showing upper bound.

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

⇒ Big-omega notation

↳ opposite of Big-Oh

$\Omega(N^3)$

↳ lower bound.

↳ it takes atleast this time complexity

it can be $N^3, N^4, N^5, N^4 \log N$ --

but can never be less than N^3

$$\lim_{N \to \infty} \frac{f(N)}{g(N)} > \infty \; 0$$

a what if an algorithm has lowerbound and upperbound of $O(N^2)$.

why repitions

$O(N^2)$ & $\Omega(N^2)$ ↙

↳

Thetha notation :

combining both $O()$ & $\Omega()$

$\Theta(N^2)$ → has both upperbound & lowerbound $= N^2$.

↳ combining both

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

⇒ little oh notation

★ This is also giving upper bound.

words: loose upperbound (not strictly ub)

Big-oh	little-oh
$O()$	$o()$
↳ $f = O(g)$	$f = o(g)$
growth of f can be no faster than g.	$f < g$ ↳ strictly slower than g.
$f \leq g$.	

★ both are upper bound.

★ Maths :

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

$f = N^2$

$g = N^3$.

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

⇒ little omega ω()

$f = \Omega(g)$

$f : \Omega(g)$

$$\lim_{N \to \infty} \frac{f(N)}{g(N)} = \infty \qquad f \geq g$$

little ω()

$f = \omega(g)$

$f > g$ strictly greater than (g)

a.

```
for( int i=1 , i≤N ){
    for( int j=1 , j≤K ; j++ ){
        // some operation (t)
    }
    i = i + K ;
}
```

takes time (t).

i = 1

$1+K, 1+2K, 1+3K, 1+4K \ldots 1+xK$

$1 + xK \leq N.$

$xK \leq N-1$

$x \leq \frac{N-1}{K}$

$$x < \frac{N-1}{K}$$

$O(kt * \text{how many times outer loop runs})$

$O(kt * \frac{N-1}{k})$

$O(t * N-1)$

time complexity $O(N \times t)$

$O(N) / O(Nt)$

Bubble sort

⇒ worst and avg-case time complexity: $O(n*n) \Rightarrow O(N^2)$
worst case occurs when array is in reverse ordered

⇒ Best case time complexity : $O(n)$. Best case occurs when array is already sorted.

⇒ Auxilary space : $O(1)$

⇒ Boundary cases : bubble sort takes minimum time (order of n) when elements are already sorted.

⇒ sorting in place : Yes

⇒ Stable : Yes

selection sort

⇒ worst case complexity : n^2

⇒ Avg complexity : n^2

⇒ Best complexity : n^2

⇒ stable NO

⇒ space complexity : 1

The good thing about selection sort is it never makes more than O(N) swaps and can be useful when memory write is costly operation

insertion sort

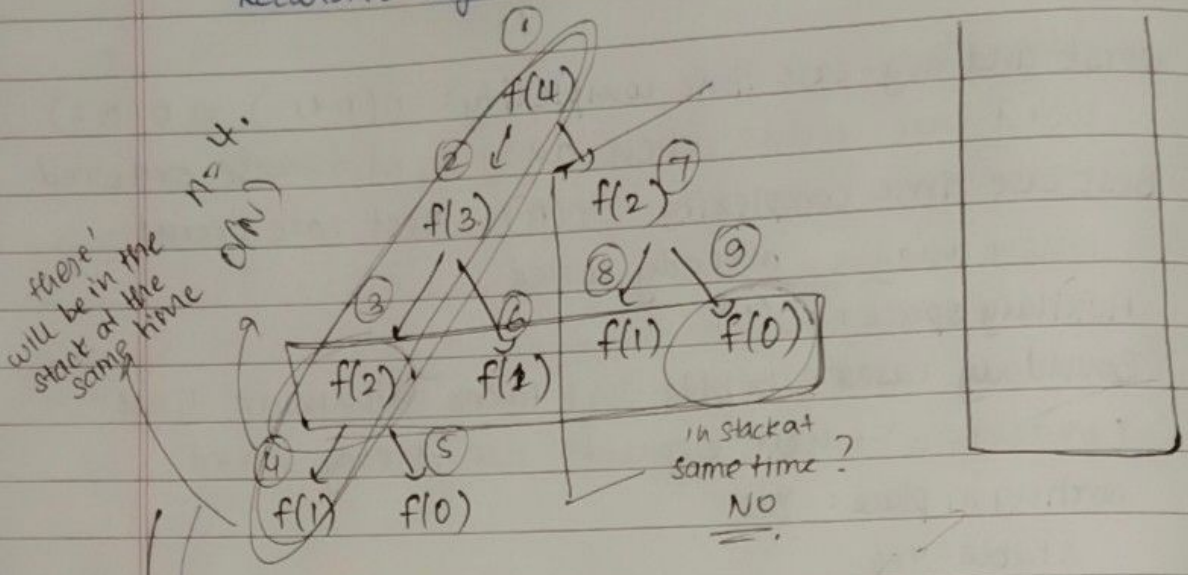
⇒ Time complexity : $O(n*2)$

⇒ Auxilary space : $O(1)$

⇒ Boundary cases : insertion sort takes max. time to sort if elements are sorted in reverse order. And takes min time (order of n) when elements are already sorted

⇒ sorting in place : Yes

Recursive Algorithms.



do not have constant space complexity.

"at any particular point of time, no two function calls at the same level of recursion, will be in the stack at the same time."

Trick: only calls that are interlinked w/ each other be in stack at same time

∴ space complexity = longest chain = root → branch.
= height of the tree

space complexity = height of tree (path)

↳ O(N).

Types of Recurrence Relations

① Linear

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

② divide and conquer

$f(N) = f\left(\frac{N}{2}\right) + O(1)$

⇒ # divide and conquer Recurrence.
solving divide and conquer Recurrence relations //.
form:

$$T(x) = a_1 T(b_1 x + \varepsilon_1(x)) + a_2 T(b_2 x + \varepsilon_2(x) + \ldots + a_k T(b_k x + \varepsilon_k(x)) + g(x)$$

for $x \geq x_0$ ↓ some constant

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

$a_1 = 1$ $\quad g(x) = C$ (const.)

$b_1 = \frac{1}{2}$

$\varepsilon(x) = 0.$

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

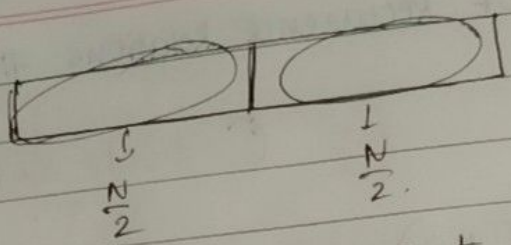
a_1 b_1 a_2 b_2 $g(N)$.

$g(N)$?

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

a_1 b_1 $g(N)$

↳ when you get ans from this + what u's doing with that ans - takes how much time.



search in both half of arrays + merge these arrays

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

$= \boxed{2T\left(\frac{N}{2}\right) + (N-1)}$ Recurrence relation (merge sort)

*
**

how to actually solve to get ~~plug~~ a time complexity?

(1) Plug & chug.

$$f(N) = f\left(\frac{N}{2}\right) + c$$

(2) Master's theorem.

~~(3)~~ Akra-Bazzi formula (1996)

Akra-Bazzi theorem

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

what is P?

$a_1 B_1^p + a_2 B_2^p \ldots = 1$

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

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

merge sort.

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

$a_1 = 2.$ $\quad g(u) = (N-1)$

$b = \frac{1}{2}.$

$k = 1.$

$$a_i b_i^p = 1 \qquad (a \cdot b)^p = 1.$$

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

$$\therefore P = 1 \quad \text{---} (1)$$

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

subs (1) in akra bazzi formula!

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

$$= \theta\left(n + n \int \frac{1}{u} - \frac{1}{u^2}\, du\right)$$

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

u^{-2} $\quad \frac{-1}{u}$

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

$$= \theta\left(n + n\left[\log n + \frac{1}{n} - 1\right]\right)$$

$$= \theta\left(n + n \log n + 1 - n\right)$$

$$= \theta\left(n \log n + 1\right)$$

$\theta(n \log n)$ $\longrightarrow$ Time complexity.

For array of size n. $\rightarrow$ Merge sort complexity
$= O(N \log N)$

2)

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

a_1 b_1 a_1 b_1

$\Rightarrow$ (P)

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

$P = 2$.

$$\frac{2 \times 1}{4\ 2} + \frac{8}{9} \times \frac{9}{16\ 2} = 1.$$

$\boxed{P = 2}$. ✓

$$T(x) = \Theta\left[x^2 + x^2 \int_1^x \frac{u^2}{u^3}\, du\right]$$

$$\Rightarrow \Theta\left(x^2 + x^2 \log x\right)$$

$$\Rightarrow \underline{\Theta\left(x^2 \log x\right)}$$

what if you don't get the value of P. ?.

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

$\Rightarrow$ let's try $P = 1$.

try 1.

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

$2 > 1 \Rightarrow$ Increase the denominator.

conc. $P > 1$

try 2. $\Rightarrow$ lets try $P = 2$

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

$g(x) = x^{(2)} = x^n$

$n = 2$

$\therefore p < 2$

conclusion $1 < p < 2$.

when $p < n$ i.e $p < 2$.

aus $= O(g(x)) \to O(x^2)$

★.

⇒ note: when P is less than the power of $g(x)$ then aus $= g(x)$

$g(x) = x^{(2)}$.

$p < 2$ (ie powe of $g(x)$)

aus $= O(g(x))$

$= O(x^2)$

$= O(N^2)$. → time complexity.

Reason.

$$T(x) = \theta\left(x^p + x^p \int_1^x \frac{u^2}{u^{p+1}}\, dx\right)$$

$$= \theta\left(x^p + x^p \int_1^x u^{-p}\, du\right)$$

$$= \theta\left(x^p + x^2\right) \quad \therefore p < 2.$$

x^p → less dominating term.

$\theta(x^2)$ = aus $= g(x)$.

Linear Recurrence Relations.

↳ How to solve?.

↳ a single formula. → no recursion / no loop.

solving linear recurrence.

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

form:

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

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

for a_i, n is fixed.
$n \rightarrow$ order of recurrence.

solution: for fibonacci nos. relation

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

steps

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

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

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

(divide by α^{n-2} on both side)

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

$$\hookrightarrow \frac{\alpha^n}{\alpha^n} \cdot \alpha^2 = \alpha^2$$

charecteristic eqⁿ of recurrence.

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

roots.

$$\Rightarrow \text{roots of quadratic equation: } \frac{b^2 \pm 4ac}{2a}.$$

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

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

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

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

$a=1, b=-1, c=-1$

$$\frac{-1 \pm \sqrt{1-4(1)(-1)}}{2(1)} = \frac{-1+\sqrt{5}}{2}.$$

$$\boxed{\alpha = \frac{-1 \mp \sqrt{5}}{2}}$$ this can be split into 2 roots

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

(2) $f(n) = c_1\alpha_1^n + c_2\alpha_2^n$ is a sol^n for fibonacci

↓ constants. → roots. — no. of roots

this is equal to

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

$= c_1\alpha_1^n + c_2\alpha_2^n$

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

fact: (3) no. of roots = no. of answers you have already.

here we have 2 roots α_1 and α_2 hence we should have 2 ans already.

$\therefore f(0)=0$ and $f(1)=1$.

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

$f(0)$ in eq 2.

$f(0) = C_1 + C_2 \Rightarrow \boxed{C_1 = -C_2}$

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

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

from (3)

$\Rightarrow$ & $\Rightarrow \boxed{C_1 = \frac{1}{\sqrt{5}}}$, $\boxed{C_2 = \frac{-1}{\sqrt{5}}}$ → here we get the values of constants

* putting these in eq (2)

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

→ formula for nth fibo. number.

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

$\left(\frac{1+\sqrt{5}}{2}\right)$ → value. = 1.6180

$\left(\frac{1-\sqrt{5}}{2}\right)$ → value = 0.6180

as $n \to \infty$

→ this will be equal to 0.

hence, this is less dominating term. ∴ ignore it.

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

thus. $O\left(\frac{1+\sqrt{5}}{2}\right)^n$

$\boxed{T(N) = O(1.6180)^n \Rightarrow \text{Golden Ratio}}$

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

↳ golden ratio

↳ exponential T(N) = worst.

using formula:

```
for(int i=0, i<11, i++)
{
    S.O.P(fiboformula(i));
}

static int fiboformula(int n){
    // just for demo, use long instead
    return (int)(Math.pow(((1+Math.sqrt(5))/2), n)
                 /Math.sqrt(5));
}
```

fibo(80)

↳ aus ✓

Q.: Equal roots.

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

① $f(n) = \alpha^n$

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

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

divide by α^{n-2} LHS & RHS

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

$$= \alpha^2 - 2\alpha + 1 = 0 \quad (\alpha = 1)$$ → double root.

★ In general case, if α is repeated 'r' times then $\alpha^n, n\alpha^n, n^2\alpha^n \cdots n^{r-1}\alpha^n$ are all soln to the recurrence.

अगर हमें इतने सारे ans नहीं मिलते then we can consider the general solutions as well

⇒ Hence, we can take 2 roots as

$1, n\alpha^n \quad \because \alpha = 1$

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

$$= c_1 + c_2 n \quad \text{—(eq.)}.$$

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

$f(n) = c_1 + c_2 n.$

$f(0) = 0 = c_1$

$\boxed{c_1 = 0}$

$f(1) = c_1 + c_2 n.$

$\boxed{1 = c_1 + c_2}$

$c_1 + c_2 = 1$

$\because c_1 = 0$

$\boxed{c_2 = 1}$

$c_1 = 0$. $c_2 = 1$.

$f(n) = c_1 + c_2 n$

$f(n) = 0 + (1) n$

$f(n) = n$.

↳ Time complexity = $O(N)$.

homogeneous Recurrence Relation.

↳ $f(n) = f(n-1) + f(n-2)$

form ⇒ we don't have other func. $g(x)$

no separate func.

Non-homogeneous " "

↳,

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

↓

this extra func. is present. it is Non-homogeneous linear rec.

How to solve:

① Replace $g(n) = 0$ and solve usually.

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

↓

$g(n) = 0$.

consider.

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

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

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

$\alpha - 4 = 0$. $\quad \boxed{\alpha = 4}$.

homog sol^n $\Rightarrow f(n) = c_1 \alpha^n$

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

(2) take g(n) on one side and find particular sol^n.

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

guess something that is similar to g(n)

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

my guess: $f(n) = c\,3^n$ (equal) — put c over here — put this.

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

$$\boxed{\text{particular sol}^n \Rightarrow f(n) = -3^{n+1}}$$

(3) Add both solutions

(homogeneous) + particular sol^n = General

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

$\because f(1) = 1$. ans already provided.

$= c_1 4 - 3^2 = 1$

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

$c_1 \cdot 4 - 9 = 1$.

$c_1 \times 4 = 10$

$c_1 = \frac{10}{4} = \frac{5}{2}$.

$f(n) = c_1 4^n$

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

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

How to guess a particular solution.

* If $g(n)$ is exponential, guess of same type

ex. $g(n) = 2^n + 3^n$

guess : $f(n) = a \cdot 2^n + b \cdot 3^n$

↳ particular soln.

* If $g(n)$ is polynomial → then guess of same degree

ex.

$g(n) = n^2 - n$

↳ degree 2

$an^2 + bn + c \rightarrow f(n) = \text{guess.}$

If $g(n) = \underbrace{2^n}_{I} + n$

guess → $f(n) = a \cdot 2^n + (bn + c)$

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

ex.

$f(n) = 2f(n-1) + 2^n$ $\qquad f(0) = 1$

① putting $2^n = 0$

$f(n) = 2f(n-1) + 0$

$\Rightarrow f(n) = \alpha^n$

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

$\alpha = 2$

guess particular sol.

we know $g(n) = 2^n$

guess: $a \cdot 2^n$

↓ given $f(x) = a \cdot 2^n$

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

$a = a + 1$ ✗ wrong.

guess: $(an+b)2^n$

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

$an + b = an - a + b + 1$

$a = 1$ →

discard b —

$f(n) = a \cdot n \cdot 2^n$

$= (1)n \cdot 2^n$

$= n \cdot 2^n$

$\boxed{n \cdot 2^n}$ // discard b.

(3) general ans.

$f(n) = c_1 2^n + n \cdot 2^n$

$f(0) = 1 \quad = c_1 + 0 \quad = 1$

$\boxed{c_1 = 1}$

$\boxed{f(n) = 2^n + n \cdot 2^n}$

complexity $= O(n \cdot 2^n)$

⇒ N-P complete Problems ex- graph colouring

↳ certain problems cannot be solved in polynomial time? but can be verified in polynomial time

SP = NP or not →