Time and Space Complexity-1

* Why to studty Time Complexity?

best ?
↳ efficient (T, space)

DSA-qn

AP1 AP2 AP3 . . . . .

Slove u_1 → AP_1 → TLE

AP_2 → X

Analysis of algorithms is important for two reasons

1. To estimate the efficiency of the given algorithm
2. To find a framework for comparing the algorithms or solutions for the given problem.

PF. Dep.

pre = before
post = after

① "Apriori analysis" V/S Posteriori analysis

Test1:

code

product

System

MAC	Win/Len	Linux/HP
	8GB i3	16GB i7
1sec	10sec	5sec

PF. Dep

⇒ release

PF. Indep

post

Test2:-

Run x

Without running, your code you should say whether it is efficient or not?

How?

1. **Apriori Analysis (Mathematical Analysis)**: Apriori analysis is conducted before the algorithm is translated into a program. It involves estimating the running time by counting the number of executions of the "dominant operations" within the algorithm. This analysis helps in understanding the efficiency of the algorithm in terms of time complexity without the need to run it on actual data.

2. **Posteriori Analysis**: Posteriori analysis, on the other hand, is performed after the algorithm has been implemented as a program. This involves executing the program using standard datasets to empirically estimate its running time and space requirements. Posteriori analysis provides practical insights into the performance of the program in real-world scenarios.

$f(x) = \frac{1}{x}$

$\lim_{x \to \infty} f(x) = ?$

① "Asymptotic" Notations

adjective. of or referring to an asymptote. (of a function, series, formula, etc) approaching a given value or condition, as a variable or an expression containing a variable approaches a limit, usually infinity.

approaching a given value or condition, as a variable or an expression containing a variable approaches a limit, usually infinity.

a given value or condition, as a variable or an expression containing a variable approaches a limit, usually infinity.

$f(x) = \frac{1}{x}$

$\frac{1}{0} = \infty$

$\lim_{x \to \infty} f(x) = \lim_{x \to \infty} \frac{1}{x} = \frac{1}{\infty} = \frac{1}{\frac{1}{0}} = \frac{0}{1} = 0$

when x approaches to ∞

DSA:-

Very very small

✓Asymptotic Analysis:-

Means, we should consider input as ~~infinite~~ size as large as possible

praticaly infinite cannot exist

✓ size of the input :-

Constraints:-

① 1<=n<=10^9 ✓ ← refer 'n' value

② 1<=arr[i]<=10^5

eles's value

it means we have to analyse by taking larger inpu
in a problem?

Finding T.C of code :-

code

① cannot run & check the time

② for that we do "Asymptotic analysis"

↳ i.e our i/p should be as large as possible.

* Goal : is to Simplify analysis of running time by getting rid of "details"

* A technique that focuses analysis on the " significant term "

*Capturing the essence : how the running time of an algorithm increase with the size of input

significant term.

-ve value ?

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

$$f(n) = 2n + \log_2 n \Rightarrow 2n.$$

30-mins

1)There are some symbols are there, which are used to represent the time complexity of a program

2) Δ, δ, τ, ϕ, τ, θ, β, α, λ, μ, . .

✓ 3) O [Big - Oh]

* Definition we will see it in next class

* Big-Oh Notation [O( )] : read it "Order of"

→ O(n) : order of n (or) Big-oh of n

O(n^2) : order of n^2 (or) Big-oh of n^2

sig. Term

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

Big-oh gives or tells upper bound ✓

↳ Heighest n^{th} degree term / Dominated Term

U_1

* $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} = \frac{n^2+n}{2} \Rightarrow O(n^2)$

* $1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} \Rightarrow O(n^3)$

$$f(n) = n^2 + n \cdot \log n + 2n \Rightarrow O(\quad)$$

1. Algorithms T.C is very much related to functions in maths

2. The following functions are commonly used in Algorithms

99%

Sno	Function Name	Function Expression
1	Constant	1 → O(1)
2	Logarithmic	log(n)
3	Square root	√n
4	Linear	n
5	Linearithmic	n.log(n)
6	Quadratic	n^2
7	Cubic	n^3
8	Exponential	2^n
9	Factorial	n!

best

worst

One thing that is sure :-

Time complexity of any program cannot be in particular units

like : sec, nano sec, msec, etc....

Why? because we are not running the program to know the time

but still we should say which program is efficient?

by using Asymptotic analysis

Summary :-

By doing AS. Analysis

$C_1 \xrightarrow[\text{(How)}]{} TC_1 : O(n)$

<Next class>

$C_2 \longrightarrow TC_2 : O(n^2)$

problem

$C_3 \longrightarrow TC_3 : O(\sqrt{n})$

$C_4 \longrightarrow TC_4 : O(\log_2^n)$

** compare and tell which is best?

$O(1)$, $O(\sqrt{n})$

$O(n)$

$O(n \log n)$ $O(\log n)$

$O(n^2)$

Comparision among various functions :

$f(n) = \sqrt{n}$ ✓ $g(n) = \underline{n}$ ✓

-> see if any common terms are there, if yes cancel them

-> * substitute very large values of n

$f(n)$	$g(n)$
$\cancel{\sqrt{n}}$	$\cancel{\sqrt{n}} \cdot \sqrt{n}$
1 ✓	$\sqrt{n}$ ✓

$\therefore$ $g(n)$ bigger

Hence : $g(n)$ is worst

Q1)

n : size of input

↳ -ve

$f(n) = n^2$	$g(n) = n^3$
$\cancel{n^2}$	$\cancel{n^2} * n$
1	n

$\therefore$ $g(n)$ is big, Hence it is worst.

Q2)

$f(n) = n \log_2 n$	$g(n) = n^2$
$\cancel{n} \cdot \log_2 n$	$\cancel{n} \cdot n$
$\log_2 n$	n

Remember:

n power very very small value v/s logn power very very very large value

$$n^{\underbrace{0.00001}} > (\log_2^n)^{\underbrace{1000000000}}$$

bigger

best.

Q) $\sqrt{n}$ v/s $\log_2^n$

$$n^{0.5} > \log_2^n$$

best.

Q) $f(n) = \sqrt{n}$, $g(n) = \log_2^n$, $h(n) = n$

Arange above fn's best to worst.

a) f, g, h

✓ b) g, f, h

c) h, f, g

d) h, f, g

$$\log_2^n < \sqrt{n} < n$$

best $\rightarrow$ worst

Q) $f_1(n) = \sqrt{n}$ $\quad$ $f_2(n) = \log_2^n$ $\quad$ $f_3(n) = n \cdot \log_2^n$

Always best

$\log_2^n < \sqrt{n} < n \cdot \log_2^n$

$\sqrt{n}$ $\quad$ $\cancel{n \cdot \log_2^n}$

$\sqrt{n}$ $\quad$ $n \cdot$ $\quad$ $*\ \log_2^n$

better

Whenver logn is there, that does not mean that it is always BEST

best

$\sqrt{n}$ $\quad$ $\log_2^n$

Consider the following three functions.

$$f_1 = 10^n \quad f_2 = n^{\log n} \quad f_3 = n^{\sqrt{n}}$$

Which one of the following options arranges the functions in the increasing order of asymptotic growth rate?

↳ best to worst

A. f_3, f_2, f_1

B. f_2, f_1, f_3

C. f_1, f_2, f_3

D. f_2, f_3, f_1

$\sqrt{n} \quad \sqrt{n} \qquad \log_{10}^{n} \quad \log_{10}^{n}$

$f_2 < f_1$

$f_1 \qquad f_2 \qquad f_3$

$10^n \qquad n^{\log_{10}^{n}} \qquad n^{\sqrt{n}}$

$f_3 < f_1$

$f_2 < f_3$

$10^n \qquad n^{\log_{10}^{n}}$

$f_1 \qquad f_3$

$\log_{10} 10^n \qquad \log_{10} n^{\log_{10}^{n}} \qquad \log_{10} n^{\sqrt{n}}$

$n \cdot \cancel{\log_{10}10} \to 1 \qquad \log_{10}^{n} \cdot \log_{10}^{n} \qquad \sqrt{n} \cdot \log_{10}^{n}$

$n \;\; f_1 \qquad f_2 \;\; (\log_{10}^{n})^2 \qquad \sqrt{n} \cdot \log_{10}^{n}$

$n \qquad \sqrt{n} \cdot \log_{10}^{n}$

$\sqrt{n} \cdot \sqrt{n} \qquad \log_{10}^{n}$

$\sqrt{n}$

$\sqrt{n}$ $\qquad$ $\log n * \log n$

$n^{0.5} > (\log n)^2$

best

H/W best $\rightarrow$ worst

Consider the following functions:

- $f(n) = 2^n$
- $g(n) = n!$
- $h(n) = n^{\log n}$