DSA

Data Structure & Algorithm

Time Complexity
&
Space Complexity

SWIPE

@KRISHAN KUMAR

DSA

Data structure & Algorithm

Data Structure :- Data Structure is like organizing and storing the data so that we can use it efficiently.

↳ think of it as a method that helps a computer arrange and manage information, making it easy to work with.

→ It's all about structuring data in a way that makes sense for solving problems on the computer.

① Storage of the data

② Computation of the data

Company → Google, Amazon becames → Data Hungry → They have Huge Amount of Data

- Storage
- Computation

↳ Becames → Very Complex

Q. Is every data important to collect.?

→ For ex→

I'm Building IRCTC website

[we need to collect data from costomer in many form]

like

→ How much the clicktime.

→ No. of features that they have intrested in.

→ How much the intraction time.

→ How much the transtion time.

we need different Data for the Different Use case.

Now a Days Company collect all the Data which is related to the costomer they actually storing their each single information.

Data Structure

↳ Structuring the Data

So that

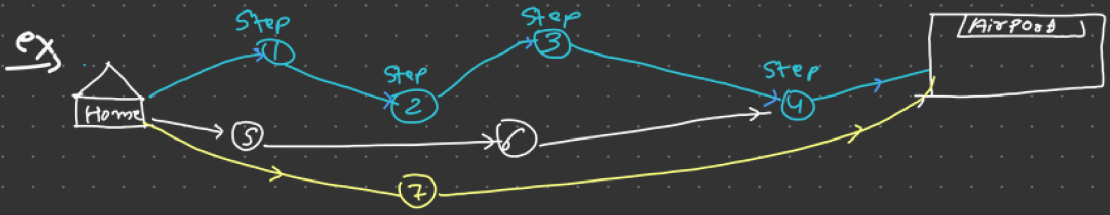
Efficiently

→ Storage Data

→ Computation Data

Algorithm :-

let's suppose i have a Problem. then the logical Steps that I follow to solve those Problems will be called as algorithm.



Prob. → To Reach the airport.

1 problem → There are multiple algorithm or Solution to solve that spacific Problem.

↳ We Pick the best solution so that → get speed

How do we choose the best algorithm

↳ to choose the best algorithm we deploy a technique called as "Time Complexity". to get the best algorithm

"Time Complexity"

Prob: Can i say A_1 is better than A_2, while

A_1 taking $\rightarrow t_1$ Second

A_2 taking $\rightarrow t_2$ Second

$\{t_1 < t_2\}$

(A) Yes

(B) NO

(C) it Depends

Why ⟹ Time taken by a algorithm is depends upon

1. Logic
2. Programing langauge
3. Operating System that we are using

Note: Just masaring the absoute time is not good enough only masaring time will not give us the best result.

Means.

Camparing algorithm

↓

Based on execution time

↳ is not only the right way.

➤ If a algorithm good in python then it also good in C++, Java, HTML

So Choose which algorithm is best we apply and go with ⟶ Time complexity Technique.

Time Complexity :- When i choose algorithm it should be independent of operating System / Programming Langauge / processor.

Algorithm ⟶ Logic

↓

work well in → Py.
→ C++
→ Java
→ HTML

Defination ⟶ What is time complexity ?

→ Number of operation = input Size

Q.

```
def printName(n):
    print('Hello MP')
```

if n = 0, 1, 2, 3
How many time print('Hello MP') exute.

Sol.

n	No. of execution
if n = 0	1
if n = 1	1
if n = 2	1

Here, No. of operation is independent of n,

So, the $T(1) = O(1)$ {constant}

Q.2

```
def Printname(n):
    for i in range(n):
        for j in range(n):
            print("Hellow world").
```

Nested loop

Sol,

n = 1 → How many time Hello world printed ⇒ 1

if n = 2 ⟹ $4 = 2^2$

n = 3 ⟹ $9 = 3^2$

n = 4 ⟹ $16 = 4^2$

No. of operation $\propto n^2$

$T(c) = O(n^2)$

time complexity

Q.3

```
def PrintName(n):
    i = 1
    while (i < n):
        i = i * 2
        Print("Hello World")
```

What is the T(c) = Time complexity

Ans

T(c) = No. of operation = input size

No. → How many time Hello world printed.

Note → Here **No. of operation < input size**

When $i \geq n$ ⇒ loop Break

Lets →

$n = 5$, then No. of operation →
- 1st → i = 1
- 2nd → i = 2
- 3rd → i = 4

⇒ 3 times

$n = 10$, then No. of operation →
- 1st → i = 1
- 2nd → i = 2
- 3rd → i = 4
- 4th → i = 8

⇒ 4 times

- 5th → i = 16 (Not accepted)

```
def printName(n):
    i = 1
    while (i < n):
        i = i * 2
        print("Hello World")
```

till $i<n$ | when $i \geq n$ / loop → Exit OR Break

No. of operation & input Size = $T(c)$

No. of operation $< n$ (Here)

(How many time while loop will execute)

So $T(c) \neq O(n)$

Right way to finding time complexity

Loof Run

1st → $i = 1 = 2^0 = 2^{1-1}$

2nd → $i = 2 = 2^1 = 2^{2-1}$

3rd → $i = 4 = 2^2 = 2^{3-1}$

4th → $i = 16 = 2^3 = 2^{4-1}$

Let loof Runs for the k times then

$$k = i = 2^{k-1}$$

Next → $(k+1)^{th} = i = 2^k$

And loop Boalces the i values becames

$$i \geq n$$

When loop Brake,

$i \geq n$

{ i Becomes → 2^K

$2^k \geq n$

what is k

$2^k \approx n$

→ No. of time loop Ran ⇒ No. of operation

$2^k \approx n$

$\log_2 2^k \approx \log n$

$k \approx \log n$ → Input size

No. of operation

$$T(c) = O(\log n)$$

Q.4

```
def PrintName(n):
    for i in range(n):
        J = 1
        while (J<n):
            J = Jx2
            Print ("Hello")
```

What is the T(c) = ?

Ans →

```
def PrintName(n):
    for i in range(n):          ← T(c) → O(n)
        J = 1
        while (J<n):            ← T(c) = (logn)
            J = Jx2
            Print ("Hello")
```

For this the $T(c) = O(n \log n)$

Rule

* if we have nested loop like,

Loop → T_1
↳ Loop → T_2
↳ Loop → T_3

then time complexity ↓

$$T(n) = T_1 \times T_2 \times T_3$$

* if we have loop like,

Loop 1 → T_1
Loop 2 → T_2
Loop 3 → T_3

then time complexity ↓

$$T(n) = \max(T_1, T_2, T_3)$$

Q.5

```
def printName(n):
    for i in range(n):
        for j in range(n):
            print("Hello")
```

break

↳ Break → Break the for loop, as we know their are 2 Loop. 1st loop runs → n times
2nd loop runs → 1 time {Because of Break}

So $T(n) = n \times 1$

$$T(n) = O(n)$$

Q.6

```
def PrintName(n):
    i=s=1
    while (s<n)
    i += 1
    s += i
    Print("Hello")
```

Time complexity ⇒ ??

Ans ⇒

```
def PrintName(n):
    i=s=1
    while (s<n)
    i += 1
    s += i
    Print("Hello")
```

T() = No. of opera. ∝ Input Size (n)

No. of time while loop runs

Let → k

finding the Relation

4 ① i=1, s=1

② i=2, s=1+2

③ i=3, s=1+2+3

④ i=4, s=1+2+3+4

k times ⑤ i=k, s=1+2+3+4+5+.....+k $\left\{\frac{n(n+1)}{2}\right\}$

$S_k - \frac{k(k+1)}{2}$

$S_{k+1} = \frac{(k+1)(k+2)}{2}$ ⇒ then loop Break

$\therefore S_{k+1} \approx n$

$$\frac{(k+1)(k+2)}{2} \approx n$$

$$(k+1)(k+2) \propto n$$

$$k^2 + 2k + k + 2 \propto n$$

$$k^2 + 3k + 2 \propto n$$

Ignore

$$k^2 \propto n$$

$$k \propto \sqrt{n}$$ → Input Size

No. of operation

$$T(n) = O(\sqrt{n})$$

Types of time complexity

- Best case
- Average Case
- Worst case

↳ Let's understand the best case, Average case and worst case using a small example. ↴

Suppose you have to find Rahul in the class.

Best case → if you find Rahul in first try that will be my Best case.

Worst case → if you find Rahul at the last try that will be our worst case.

When we have dealing with computer
↓
prepare for the
↓
worst Case

Because when we prepare for the worst we already Cover for the Best or Average Case.

T(1) ⇒ Worst case time complexity.

OR

also called → Big O

* Space Complexity *

The extra space that you need to solve the problem respect to the input size.

ex → I have a list = [1,2,3,4,5], Please reverse list?

Ans → list = [1,2,3,4,5]

New list = [5,4,3,2,1]

For reversing I am take new list that it the extra space that I used to solving the problem statement.

↳ Now more poople aware about time complexity Not for the space complexity.

In any algorithm

if we are using space to ssolving problem → that will be in Space complexity problem.

No. Extra space → Called → Constant Space complexity

OR

Inplace algorithm

These types of algorithm is Really Good algorithm.