Time Complexity

Hello Everyone

A very special Good Evening to All of you

We will start the session at 9:05 pm

~~~~~~~~
Agenda
~~~~~~~~

1. Introductions
2. What we have to learn in this module?
3. Factor Count
4. Count of Iterations
5. Time Complexity in Big-O notation
6. Space Complexity
7. TLE : Time Limit Exceed
8. Bitwise Operators

Request:

Please do not talk about a topic which is not yet discussed in class in public chat.

* Please do not create panic in others.

To: ~~Everyone~~ Shreesh Tripathi

Session flow:

9:06 Pm ⟶ 10 min break ⟶ 11:30pm ±15 min for buffer ⟶ doubt session

As Instructor what I need from all of you?

1. Support each other
2. Control your excitement
3. Show energy in every class by participating
4. Solve assignment and home work problems.

My Introduction:

→ Shreesh Tripathi

→ FT. Instructor and Software Eng. from last 2 yrs (Scaler)

→ previous Org. → Pep Coding → Edtech → acquired by Scaler.
Core team, Team lead for CBSE vertical, DSA Instr + Content creator.

→ More than 400 Session in Scaler

→ teaching DSA from last 3 years

There are few terms that you shall hear throughout the course :

1. PSP(Problem Solving Percentage) = Solved Assignment Problems / Total Unlocked Assignment
   * There are two type of section : Assignment and Additional, Assignment Problems consists of implementation of the problems done in the class. PSP calculation is there on Assignment Problems only.

   * Additional Problems are slight modification of Assignment Problems, They are not part of PSP but once you are done with Assignment, we will highly recommend you to solve it.

   * Try to keep PSP at-least 85% no matter what, it will really help you to stay focused and have seen in tha past that learner with PSP >= 85% perform well.

2. Attendance :
   * Try to maintain at-lest 75% attendance by live or by recording.

   * My recommendation is to join session live for better interactions

~~~~
FAQs
~~~~
* Notes will be uploaded after class.
* Assignment will unlock after the class ends., Addition problem unlocks After 1.5 Hrs of class end
* There is no deadline for Assignment.
* If Asking a question, ask it in public.
* If Answering a question, answer it in private.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

What we have to learn in this module?

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

* Arrays
* Bit Manipulation
* Recursion and Backtracking Basics

Contest 1 : Arrays and Bit Manipulation

* Maths
* OOPs Basic oops, essential for DSA
* Hashing
* Language Advance Concept

Contest 2 : Recursion, Maths and OOPs

* Sorting
* Searching
* Two Pointers
* Linked List

Contest 3 : Hashing, Sorting, Searching & 2 Pointers

* Stacks
* Queues
* Trees

Contest 4 : Linked List, stack and Queues

* Hashmap implementation
* Heaps
* Greedy
* DP

Contest 5 : Trees, Heaps and Greedy

* Graphs
* Interview Problems

Contest 6 : Full Syllabus

[This is mandatory Contest for eligible for placement]

Quiz1:

What is the sum of the first (n) natural numbers?

$$1+2+3+4+\cdots\cdots+n = \frac{n(n+1)}{2}$$

$$S = 1 + 2 + 3 + 4 + \cdots + n \longrightarrow ①$$
$$S = n + (n-1) + (n-2) + \cdots + 1 \longrightarrow ②$$

$$2S = \underbrace{(n+1) + (n+1) + (n+1) + \cdots + (n+1)}_{n \text{ times}}$$

$$\Rightarrow 2S = n*(n+1)$$

$$\Rightarrow S = \frac{n*(n+1)}{2}$$

Brackets → Inclusive → [ ]   [1 7] → 1, 2, 3, 4, 5, 6, 7

Exclusive → ( )   (2 5) → 3, 4,

Quiz2:

How many numbers are there in the range [3, 8]?

[3 8] → 3, 4, 5, 6, 7, 8

= 6 numbers are there

[0, 9] → 9−0+1   ⇒ [a b] = b−a+1

count of element in Range

[a b] → b−a+1

(a b) → b−a−1

[a b) → b−a

(a b] → b−a

Count of factor of N?

What is factor?

Suppose we have two numbers, a & b.

$b \% a == 0 \longrightarrow$ a can completely divids b

→ a is factor of b.

→ b is multiple of a.

for eg. $b=12$, $a=4$

$b \% a \Rightarrow 12 \% 4 == 0$

→ 4 is factor of 12

→ 12 is multiple of 4.

Problem: Given N, Count no. of factors.

Note: $N > 0$

Quiz 3:

Count of factors of the number 24.

$N=24 \longrightarrow 1, 2, 3, 4, 6, 8, 12, 24$

8 factors are there

Quiz 4:

Count of factors of the number 10.

$N=10 \longrightarrow 1, 2, 5, 10$

4 factors are there

for $N=10$

if a number can divide N, then that number is factor.

Smallest factor for N → 1

largest factor for N → N

all factors exist b/w smallest & largest

```
function factorCount(N) {
    factorCount = 0
    for( i = 1 to i <= N) {
        if( N % i == 0) {
            // i is factor of N
            factorCount++;
        }
    }
    return factorCount;
```

Code ⟶ compiler ⟶ Running on server

$\underbrace{\text{server}}$ It allows 1 sec to every user

Assumption

1 sec ⟶ 10^8 Iterations in code

N		Iteration		time
10^8	⟶	10^8	⟶	1 sec
10^9	⟶	10^9	⟶	10 sec
10^{18}	⟶	10^{18}	⟶	$\underbrace{10^{10}\ \text{sec.}}_{317\ \text{years.}}$

10^8 itr ≡ 1 sec

$\Rightarrow 1\ \text{Itr} = \frac{1}{10^8}$ sec

$\Rightarrow 10^9\ \text{itr} = \frac{1}{\cancel{10^8}} \times \cancel{10^9}$ sec.

$= 10$ sec.

Optimisation of factor Count:

$i * j = N \longrightarrow$ {i and j both are factor of N} for eg → $3*4=12$
3 & 4 both are factor

If we know first factor ⇒ i

Second factor ⇒ $j = \frac{N}{i}$

N=24.

i		$\frac{N}{i}$
1	<	24
2	<	12
3	<	8
4	<	6
6	$\cancel{<}$	4
8		3
12		2
24		1

factors are Repeated.

$i < \frac{N}{i}$ → keep iteration

⇒ $i * i < N$

N=100

i		$\frac{N}{i}$
1	<	$\frac{100}{1} = 100$
2	<	$100/2 = 50$
4	<	$100/4 = 25$
5	<	$100/5 = 20$
$\boxed{10}$	$\boxed{<=}$	$\boxed{100/10 = 10}$
20		$100/20 = 5$
25		$100/25 = 4$
$\boxed{50}$		$100/50 = 2$
100		$100/100 = 1$

→ Not repeating.

* Iterate until

$i * i <= N$

→ if $i * i == N$

→ It is not repeated factor

```
function solvefactorCount(N) {
    factCount = 0;
    for(int i=1; i*i<=N; i++) {
        if(N % i == 0) {
            // i is first factor
            if(i*i == N) {
                factCount++;
            } else {
                factCount += 2;
            }
        }
    }
    return factCount;
}
```

$i*i <= N$

$\Rightarrow i <= \sqrt{N}$

Iteration Count $\longrightarrow$ root(n)

(with optimised approach)

1 itr = $\frac{1}{10^8}$ sec

1sec $\longrightarrow$ 10^8 Iterations in Code

N	Iteration = $\sqrt{N}$	time
10^8	$\sqrt{10^8} = 10^4$	$\frac{10^4}{10^8}$ sec = $\frac{1}{10^4}$ sec
10^{18}	$\sqrt{10^{18}} = 10^9$	$\frac{10^9}{10^8}$ sec = 10 sec.

We Moved from 317 years $\longrightarrow$ 10 sec.

Prime Number:

If a number have exactly 2 factor then no. is prime number.

divisibly by 1 and itself ~~X~~

→ eg, n=1

→ count factors

if factorCount == 2

→ N is prime.

Iteration Count for loops:

What is the number of iterations in the following code:

```
int i = N;      // N > 0
while(i > 1) {
    i /= 2;
}
```

$$\underbrace{N \longrightarrow \frac{N}{2} \longrightarrow \frac{N}{2^2} \longrightarrow \frac{N}{2^3} \longrightarrow \cdots\cdots \longrightarrow 1}_{K \text{ Iterations.}}$$

$$\frac{N}{2^0} \longrightarrow \frac{N}{2^1} \longrightarrow \frac{N}{2^2} \longrightarrow \frac{N}{2^3} \longrightarrow \cdots\cdots \longrightarrow \frac{N}{2^K}$$

$$\frac{N}{2^K} <= 1$$

$$\Rightarrow \quad N <= 2^K$$

take $\log_2$ both side

$$\log_2 N <= \log_2 2^K$$

$$\Rightarrow \boxed{K >= \log_2 N}$$

No. of Itr = $\log_2 n$

Quiz6:

What is the number of iterations in the following code:

```
for(int i = 1; i <= N; i++) {
  for(int j = 1; j <= N; j++) {
    print(i + j);
  }
}
```

i	j loop	Iteration through J
1	[1 N]	$N - \cancel{1} + \cancel{1} = N$
2	[1 N]	N
3	[1 N]	N
⋮		⋮
N	[1, N]	N

N times.

Total Itr: $\underbrace{n + n + n + \cdots + n}_{n \text{ times}}$

$= n^2$

Quiz 7:

What is the number of iterations in the following code:

```
for(int i = 0; i < N; i++) {
    for(int j = 0; j <= i; j++) {
        print(i + j);
    }
}
```

[0 to i]

i	j loop	Iteration through J
0	[0, 0]	$0 - 0 + 1 = 1$
1	[0, 1]	$1 - 0 + 1 = 2$
2	[0, 2]	$2 - 0 + 1 = 3$
⋮		⋮
n-1	[0, n-1]	$n - \cancel{1} - 0 + \cancel{1} = n$

total Itr = $1 + 2 + 3 + 4 + \cdots + n$

$= \frac{n(n+1)}{2}$

$\Rightarrow O(n^2)$

10:26 – 10:36 pm

Big-O notation:

The concept of Big-O notation is, the rate of growth of function with respect to n and provides us the estimated line.

Big-O notation:

* Represented in form of O(f(n))
→ It always gives us upper bound, OR worst-case time complexity,

Steps to Calculate Big-O notation:

1. Calculate no. of Itrs
2. Neglect lower order terms
3. Ignore constant coefficient.

for eg ①: $f(n) = 3n^3 + \underbrace{4n^2 + 2}_{\text{lower order term.}}$

$f(n) = \underbrace{3}_{\text{constant coefficient}}n^3$

Big-O ⇒ O(f(n)) = O(n^3)

eg②: $f(n) = \underbrace{3}_{\text{constant}}n^2 + \underbrace{4 \log n}_{\text{lower order}}$

Big-O ⇒ O(n^2)

eg③ $f(n) = \sqrt{n} - \underbrace{2\log(n)}_{\text{lower order}}$, Big-O ?

Big-O = O($\sqrt{n}$)

```
boolean search (int [] arr, int k) {
    for(int i=1; i< arraySize ) {
        if( arr[i] == k) {
            return true;
        }
    }
    return false;
}
```

Iteration

Best case — element present at 0^{th} index → Single iteration, 1 itr

worst case. — element not present → N Iteration.

Itr = n ⟶ worst case

Big-O = $O(n)$

Space Complexity:

```
int add(int a, int b) {
    return a + b;
}
```

→ Inputs

function of space

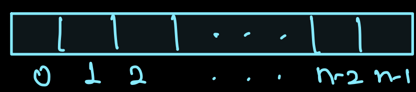
$f(n)$ = Constant

Big-O ⇒ $O(\text{constant}) = O(1)$ constant

→ This code is not creating any memory apart from input.
Input remain constant.

S.C: $O(1)$
T.C: $O(1)$

```
func( int n) {
    int[] arr = new int[n];
}
```



suppose, n = 3 ⟶ Space = 4+4+4 = 12 byte

n=4 ⟶ space = 4*4 = 16 byte.

n=10 ⟶ space = 4*10 = 40 bytes.

n → n ⟶ space = 4*n = 4n

Space = 4n

in Big-O ⇒ $O(n)$

```
fun(int n, int m){
    int[][] arr = new int[n][m];
}
```

Space = $n * m * 4$ byte

In Big-O ⇒ S.C: $O(n*m)$

What if $m=n$

in that case S.C: $O(n^2)$

TLE: time limit Exceed:

if itr > 10^8 → TLE

Code → Compiler → Server → 1sec per user.
↳ 10^8 iteration.

Value of N	Time complexity.	Iteration Count
$N = 10^5$ →	$O(n^2)$ →	$10^5 * 10^5 = 10^{10}$ itr. Require more than one sec → TLE
$N = 10^{10}$ →	$O(\sqrt{n})$ →	$\sqrt{10^{10}} = 10^5$ → Not give TLE

Bitwise Operator:

AND, OR, XOR, NOT, left Shift, Right Shift

→ Bitwise operator works on bits of data.

Same Some puppy Shame

A	B	A & B (AND)	A | B (OR)	A ^ B (XOR)	~A (NOT)
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

AND →

```
int a=5;
int b=3;
int res = a & b
print(res);  → 1 Ans
```

```
  0 1 0 1
& 0 0 1 1
---------
  0 0 0 1
```

OR →

```
int a=5;
int b=3;
int res= a|b;
print (res); → 7 Ans
```

```
  0 1 0 1
| 0 0 1 1
---------
  0 1 1 1
```

XOR →

```
int a=5;
int b=3;
int res= a|b;
print (res); → 6 Ans
```

```
  0 1 0 1
^ 0 0 1 1
---------
  0 1 1 0
```

NOT(~)

int a=5; → 0101 binary.

int res=~a; Assumption: 32 bits.

print(res); → some negative number.

left shift (<<)

shift every bit by left side & discard left most bit

int a=5; → 0101

int res= a<<2; 010100 → 10100

→ 20 ans

$a*2^2$

$5*4=20$

discard two bits from left most

int a=7; 00111000

int res= a<<3; ⇒ 111000

print(res); $a*2^3$

$\Rightarrow res = a*2^3$

$=7*8$

$=56$ ans

res = a<<n

$res = a*2^n$, what if a=1

$res = 2^n$

$1<<1 \to 2^1$

$1<<2 \to 2^2$

$1<<3 \to 2^3$

$1<<n \to 2^n$

calculate 2^x without loop.

int res= $\underbrace{(1<<x)}_{O(1)\ T.C.}$;

Right Shift (>>)

→ We will discard right most bit & insert 0 bit from left most side.

int a=16; → 1 0 0 0 0

int res = a>>2;

print(res);

└→ 4 ans

$00 \ldots 1 0 0 \underbrace{0\ 0}_{\text{discard}}$

→ 100

16>>2

$\frac{16}{2^2} = \frac{\cancel{16}}{\cancel{4}} = 4$

int a= 19; → $1 0 0 \underbrace{1\ 1}_{\text{discard}}$

int res= a>>3

print(res);

└→ 2 ans

⇒ 10

a = 19

a>>3 → $\frac{19}{2^3} = \frac{19}{8} = 2$

$$\boxed{a >> n \longrightarrow \frac{a}{2^n}}$$

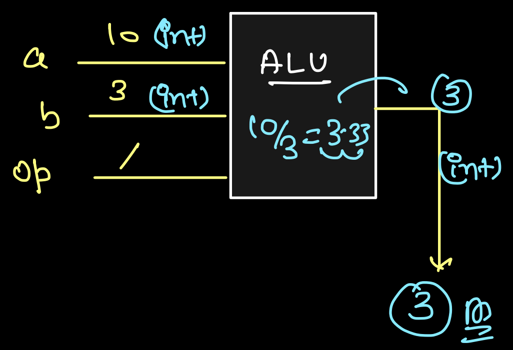
what if a=1

$$\boxed{1 >> n \rightarrow \frac{1}{2^n}}$$

int a=10;

int b=3;

int c= a/b;



Summary:

* sum of n- natural no = $\frac{n(n+1)}{2}$

* Range of number

  [a b] → b-a+1
  (a b) → b-a-1
  [a b) → b-a
  (a b] → b-a

* 10^8 itr = 1 sec.
  10^9 itr = 10sec
  10^{18} itr = 317 year.

* Iteration count

* Big-O nation → Time Complexity
  → Space Complexity

* Bitwise operator
  → AND, OR, XOR, NOT, <<, >>