Agenda

1) count of factors
2) Basic maths revision
3) Big O
4) TC & SC

- Intermediate DSA: Time & Space Complexity
- Intermediate DSA: Introduction to Arrays
- Intermediate DSA: Lab Session on TC, SC, Output & Debugging
- Intermediate DSA: Arrays - Prefix Sum & Carry Forward
- Intermediate DSA: Lab Session on Prefix Sum & Carry Forward
- Intermediate DSA: Arrays: Sliding Window & Contribution Technique
- Intermediate DSA: Lab Session on Memory Management & Sorting Basics
- Intermediate DSA: Bit Manipulations Basics
- Intermediate DSA: Lab Session on 2D Matrices
- Intermediate DSA: Strings [Including String Immutability]
- Intermediate DSA: Lab Session on 2D Matrices & Strings - 2
- Introductory DSA Contest

OOPS (DSA 2)

→ ch: 1 Classes & Objects

Q. Given N, count factors of N.

↳ factor can divide N completely.

N = 18, factor → 1 2 3 6 9 18 ans = 6

N = 15, factor → 1 3 5 15 ans = 4

N = 24, factor → 1 2 3 4 6 8 12 24 ans = 8

N = 10, factor → 1 2 5 10 ans = 4

1) Brute force : N factors → 1 to N

```
int countFactor(int N) {
    int count = 0;
    for(int i=1; i<=N; i++) {
        if (N % i == 0) {
            count++;
        }
    }
    return count;
}
```

N = 10

i → 1 2 3 4 5 6 7 8 9 10

count = $\cancel{0}$ $\cancel{1}$ $\cancel{2}$ $\cancel{3}$ 4

itr : N

2) optimised logic

$i * j = N$ (both i & j are factors of N)

$j = \frac{N}{i}$ (both i & $\frac{N}{i}$ are factors of N)

N = 24

i	N/i
1	24
2	12
3	8
4	6
6	4
8	3
12	2
24	1

above part :

$\rightarrow i <= \frac{n}{i}$

$\rightarrow \boxed{i * i <= n}$

(take sqrt on both sides)

$\Rightarrow \sqrt{i^2} <= \sqrt{n}$

$\boxed{i <= \sqrt{n}}$

N = 36

i	N/i
1	36
2	18
3	12
4	9
6 *	6
9	4
12	3
18	2
36	1

check i is factor or not

N = 27

i = 1	✓	(1, 27)
i = 2	X	
i = 3	✓	(3, 9)
i = 4	X	
i = 5	X	

count = ~~2~~ 4

```
int countFactors(int N) {
    int count = 0;
    for (int i=1 ; i*i <= N ; i++) {     // i <= √N
        if (N%i == 0) {
            if (i != N/i) {
                // both i & N/i needs to be considered
                count += 2;
            }
            else {
                count++;
            }
        }
    }
    return count;
}
```

N = 27

i	if (N%i == 0)		count
1	✓	(1, 27)	~~0~~ ~~2~~ 4
2	X		
3	✓	(3, 9)	
4	X		
5	X		

N = 36

i	if (N%i == 0)		count
1	✓	1, 36	~~2~~
2	✓	2, 18	~~4~~
3	✓	3, 12	~~6~~
4	✓	4, 9	~~8~~
5	X		
→ 6	✓	6, 6	9

itr count → $\sqrt{n}$ times

n = 10000

brute force	optimised logic
itr → n	itr → $\sqrt{n}$
10000	$\sqrt{10000} = 100$

prime no. are those no. for which factor count = 2

How many $\boxed{\text{prime numbers}}$ are there?

10, 11, 23, 2, 25, 27, 31

ans = 4

Some Basic math properties

1) Range :

$[a, b] \rightarrow b - a + 1$

$[3, 10] = 10 - 3 + 1 = 8$

2) Sum of N natural no.

$1 + 2 + 3 + 4 + \dots\dots\dots + N$

$$\boxed{S_N = \frac{N(N+1)}{2}}$$

$1 + 2 + 3 + 4 + \dots\dots + 100$

$$\frac{\overset{50}{\cancel{100}}(100+1)}{\cancel{2}} = 50 \times 101 = 5050$$

3) GP (Geometric progression)

a	ar	ar^2	ar^3	ar^4	....	ar^{n-1}
1	2	3	4	5		n

$a \rightarrow$ first term

$r \rightarrow$ common ratio

$n \rightarrow$ total terms

$$\boxed{S_n = \frac{a(r^n - 1)}{r - 1}}$$

$S_n \downarrow$ Sum of n terms of GP

2 6 18 54 162

$a = 2$

$r = 3$

$n = 5$

Sum of GP terms $= 2 + 6 + 18 + 54 + 162$

$$\frac{a(r^n - 1)}{r - 1} = \frac{\cancel{2}(3^5 - 1)}{\cancel{2}} = 242$$

Iteration count

How many times will the below loop run ?

```
for(i -> 1 to N)
{
    if(i == N) break;
}
```

$i \rightarrow 1$ to N

itr = N

$[a,b] \rightarrow b-a+1$

How many iterations will be there in this loop ?

```
for(i -> 0 to 100){
    s = s + i + i^2;
}
```

$i \rightarrow 0$ to 100

itr = 101

How many iterations will be there in this loop?

```
func(){
    for(i -> 1 to N){
        if(i % 2 == 0){
            print(i);
        }
    }
    for(j -> 1 to M){
        if(j % 2 == 0){
            print(j);
        }
    }
}
```

N itr

M itr

total itr = N+M

* How to compare two algo's ?

→ sorting ques , local machine

S1

↓ Macbook pro, M2

10 sec

↓

10 sec (python)

↓ C++

7 sec

S2

↓ Windows XP

15 sec

↓ Macbook pro, M2

7 sec (C++)

↓

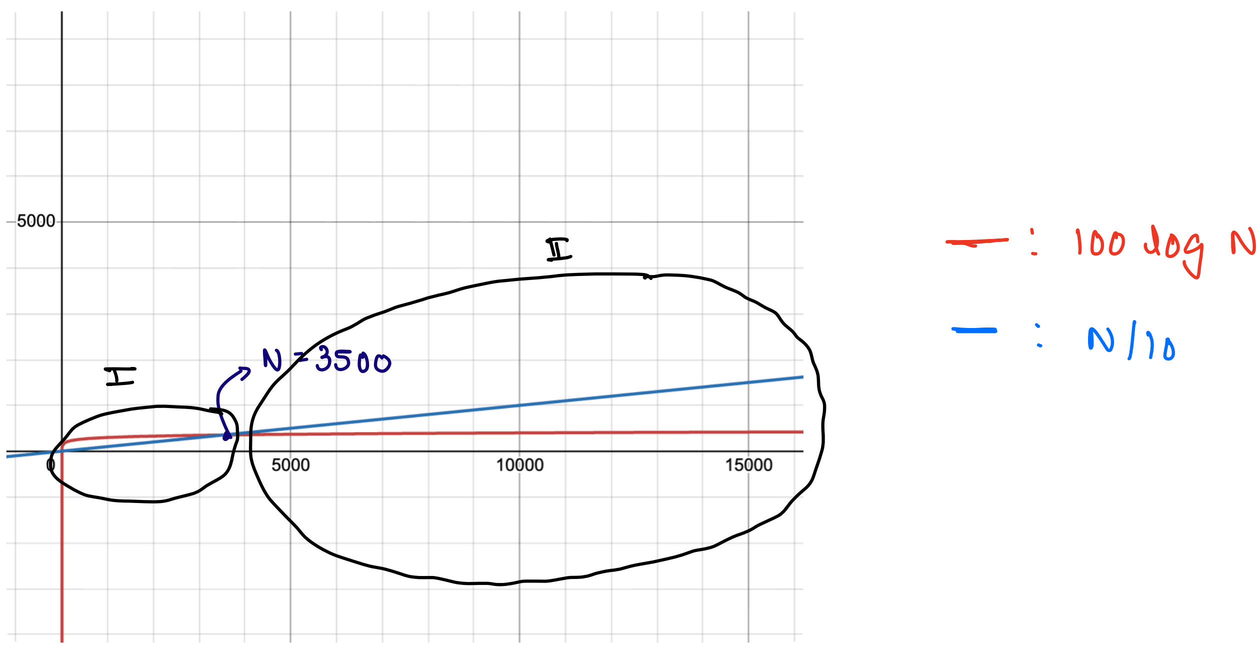
7 sec

→ Execution time is not a good factor to compare two algo's because it depends on lot of external factor (eg. processor, language, temperature etc.)

$\boxed{\text{iteration count}}$ is an independent factor itr1 → 100 log N

itr2 → N / 10

which code is better ?



N <= 3500 (I)	N > 3500
100 log N is taking more itr than N / 10	N/10 is taking more itr than 100 log N
→ winner : N / 10	→ winner : 100 log N

→ to analyse performance of code of very large input size.

Assymptotic analysis

How → Big O

What → it analyses the performance of algo for $\boxed{\text{large input size}}$

Steps to calculate Big O

1) find itr count
2) ignore the lower order terms
3) ignore the constant coefficient

→ Big O (itr count) or Time complexity

eg. itr → $\cancel{4}N^2 + \cancel{3N} + \cancel{\sqrt{N}}$ TC : $O(N^2)$

Comparsion Order:

log(N) < sqrt(N) < N < N log(N) < N sqrt(N) < N^2 < N^3 < 2^(N) < N! < N^N

N! → N factorial

itr : $\cancel{4 N \log N} + \cancel{3}N^2 + \cancel{92N}$

TC : $O(N^2)$

1) find itr count
2) ignore the lower order terms
3) ignore the constant coefficient

itr: $\cancel{4N} + \cancel{3}N * \log(N) \cancel{+ 1}$

Big O $\rightarrow$ $O(N * \log N)$

itr: $\cancel{4N\log(N)} + \cancel{3}N * Sqrt(N) \cancel{+ 10^6}$.

Big O $\rightarrow$ $O(N * sqrt\, N)$

itr : $\cancel{4N\log N} + \cancel{3}N^2 + \cancel{92N}$

Big O $\rightarrow$ $O(N^2)$

1) find itr count
2) ignore the lower order terms
3) ignore the constant coefficient

itr : $\cancel{4N\log(N)} + \cancel{3}N * \mathrm{Sqrt}(N) + \cancel{10^6}.$

Big O $\rightarrow$ $O(N * \mathrm{sqrt}\, N)$