-> Factors and Divisors

-> Brute Force Approach to find Divisors, Divisors count and Divisors Sum.

-> Time Complexity O(n)

```
VOID BRUTEDIVISOR (INT n)
{
    INT CNT = 0, SUM = 0;
    FOR (INT i = 1; i <= n; i++)
    {
        IF (n % i == 0)
        {
            COUT << i << "\n";
            CNT++;
            SUM += i;
        }
    }
    COUT << CNT << " " << SUM << "\n";
}
```

-> But we can observe a pattern in divisors:

Scanned with CamScanner

For n = 24,

1 ⟶ 24/1 = 24
2 ⟶ 24/2 = 12
3 ⟶ 24/3 = 8
4 ⟶ 24/4 = 6

→ We can see we can get those lower part of divisors from the upper once as well.
→ So, we just need to iterate our loop till SQRT(n) ($\sqrt{n}$).

6
8
12
24

→ OPTIMIZED APPROACH

→ Square Root Method to find divisors, divisors count and divisors sum.
→ It is faster than Brute Force as we have to iterate our loop till SQRT(n) only.
→ Time Complexity : $O(\sqrt{n})$

```
VOID SQRTDIVISOR (INT n)
{
  INT CNT = 0, SUM = 0;
  FOR (INT i = 1; i * i ≤ n; i++)
  → (i*i ≤ n) is same as (i ≤ SQRT(n))
  {
      IF (n % i == 0)
```

Scanned with CamScanner

```
{
    cout << i << " " << n/i << "\n";
    cnt++;
    sum += i;
```

→ Check in case if we don't consider a number twice.
→ eg:- In case of n = 36
→ 1 36
2 18
3 12
4 9
6 6 → We don't want to consider this 6 twice.

```
    if (n/i != i)
    {
        cnt++;
        sum += n/i;
    }
  }
 }
 cout << cnt << " " << sum << "\n";
}
```

→ There is even more faster approach to find divisor count and divisor sum using Prime factorization.

Let x be a no., representing its Prime Factors like this :

Scanned with CamScanner

$x = p_1^{n_1} \times p_2^{n_2} \times p_3^{n_3}$

Suppose $x = 36$

$\therefore \quad 36 = 2^2 \times 3^2$

$= \underbrace{2 \times 2 \times 3 \times 3}$

The subsets of these will give divisors

$\therefore$ For, $x = p_1^{n_1} \times p_2^{n_2} \times p_3^{n_3}$

In how many way we can select p_1, p_2 and p_3

(n_1+1)

↓ ↘

We choose — We don't choose (case → 2)

(case → 1) p_1 $(1 - n_1)$ time — p_1

Similarly:

$$\boxed{(n_1+1)(n_2+1)(n_3+1) = \text{no. of divisors}}$$

Proof: if $x = 36 = 2^2 \times 3^2$

$n_1 = 2 \quad n_2 = 2$

$(2+1)(2+1) \Rightarrow$ 9 divisors

→ Now, For sum of divisors:

$(1 + p_1 + p_1^2 + p_1^3 \text{ ---} p_1^{n_1})$

$\times (1 + p_2 + p_2^2 + p_2^3 \text{ ---} p_2^{n_2})$

Scanned with CamScanner

Page No.
Date

Proof : $36 = 2^2 \times 3^2$

$\Rightarrow (1+2+4)$ ← for 2^2

$\times$

$(1+3+9)$ ← for 3^2

$\Rightarrow 7 \times 13 \Rightarrow$ (91)

→ Now general formula for sum:

$(1 + P_1 + P_1^2 + P_1^3 \text{ — — } P_1^{n_1})$

↳ We can see this is G.P

$\therefore$ Sum of divisors $= \left(\frac{P_1^{n_1+1} - 1}{P_1 - 1}\right) \times \left(\frac{P_2^{n_2+1} - 1}{P_2 - 1}\right) \times \left(\frac{P_3^{n_3+1} - 1}{P_3 - 1}\right)$

PROOF : $36 = 2^2 \times 3^2$

$\Rightarrow \left(\frac{2^3 - 1}{2-1}\right) \times \left(\frac{3^3 - 1}{3-1}\right)$

$\Rightarrow 7 \times \frac{26}{2} 13$

$\Rightarrow$ (91)

Scanned with CamScanner