PRIME Factorisation

If $n = 20$, then

$i = 2 \Rightarrow$ keep dividing n by i, until it cannot be divided further

$\frac{20}{2} = 10 \Rightarrow \frac{10}{2} \Rightarrow 5$ ← Now this cannot be divided by 2.

$i = 3 \Rightarrow 5$ is not divisible by 3

$i = 4 \Rightarrow$ " " " 4

$i = 5 \Rightarrow$ It gets divided. So stop (as $n = 1$ now);

Dry run on

25, 32, 28

same procedure;

Prime no that divides cannot be larger than $\sqrt{n}$;

```
void prime_factorisation( int n) {
      vector<int>v;
      for(int i = 2; i*i <=n ;i++){
            while ( n % i == 0 ) n/= i, v.push_back(i);
      }
      if ( n > 1 ) v.push_back(n);
      for(int i : v) cout << i << " ";
      cout << endl ;
}
```

keep dividing n by i; (until possible)

Store prime number that divides it.

If n could not be divided by any 'i' this means it's prime, and its prime fact^n itself.

Prime Factorization ($\log n$)

* We are looking forward to optimize our algo from $O(\sqrt{n})$ to $O(\log n)$.

* In earlier ex of $n=20$ we did

$i=2, \frac{20}{2} = \frac{10}{2} \Rightarrow 5$

then

$i=3$, $5 \not| 3$ $\frac{1}{0} = 0$

$i=4$ ✗

$i=5$ $5/5 = 1$ ✓

So prime factors: 2, 2, 5

* If we look closely we checked for $i=4$ in b/w, but it cannot devide n ever, as 2 have covered that case.

Continuing this idea, we only need to check whether n is divisible by primes. As primes will cover cases for all composites and this will reduce our complexity.

using sieve we can store prime till N. then follow the same procedure as before.

```
vector<int> primes;

void fn(int n) {
    vector<int> factors;
    for (int i =0; i< (int)primes.size(); i++) {
        if (primes[i] * primes[i] > n)
            break;
        while (n % primes[i] == 0) {
            factors.push_back(primes[i]);
            n /= primes[i];
        }
    }
    if (n > 1)
        factors.push_back(n);
  for (int i : factors) cout << i << " ";
}
```

No of divisors

$N = 54 = 2^1 \times 3^3$

Using P&C ⇒ The possible numbers we can form from these factors are →

From 2 ⇒ either we pick 0 2's
or " " 1 2's
⇒ we have 2 options

From 3 ⇒ either we pick (1) 3's
" " " 1 3's
" " " 2 3's
⇒ we have 3 options here

So we can form total = (2) x (3) = 6 Numbers out of this;

Now Generalizing it ⇒ $N = p_1^a \cdot p_2^b \cdot p_3^c$ (Prime factorization)

No of total divisors = $(a+1)(b+1)(c+1)$