Solved Problem - Factorization

In the lesson, we look at an efficient way to factor a number.

We'll cover the following

Problem statement

Given a number $N > 1$, count the number of factors of the number N.

Input format

A single line of input contains the number $1 \leq N \leq 10^{12}$.

Output format

Print a single integer equal to the number of factors of N.

Sample

Input

```
36
```

Output

```
9
```

Explanation

Factors of 36 : $1, 2, 3, 4, 6, 9, 12, 18, 36$

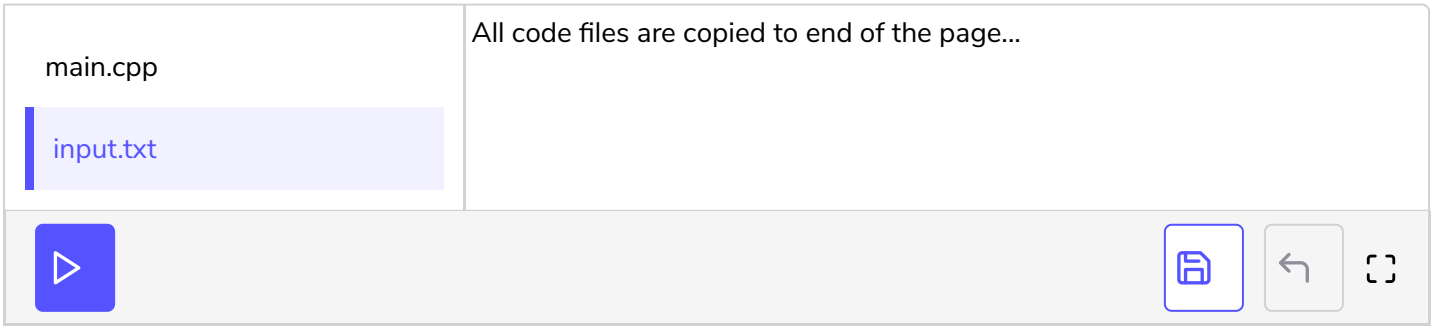
Count: 9

Brute force

The brute force solution would be to loop over all the numbers from `1` to `N` and check if it is a factor. If it is, you would then print it.

We can use the modulus operator to check if it's a factor or not.

Here is the code:



Since there is only one loop that runs N times, the runtime complexity is simply $O(N)$.

This is good enough for N up to 10^6 or even 10^8, but it will not work with the given constraints. *Typically, you have 1-3 seconds for the code to execute and print the results*.

Let's see how we can optimize it further.

Optimization

Let's take a number, `n`, and one of its factors, `a`. Then there must be another factor, `b`, such that

$a * b = n$

or $b = \frac{n}{a}$

Also, let's assume $a <= \sqrt{n}$

$\frac{1}{a} >= \frac{1}{\sqrt{n}}$

$\frac{n}{a} >= \frac{n}{\sqrt{n}}$

$\frac{n}{a} >= \frac{n}{\sqrt{n}}$

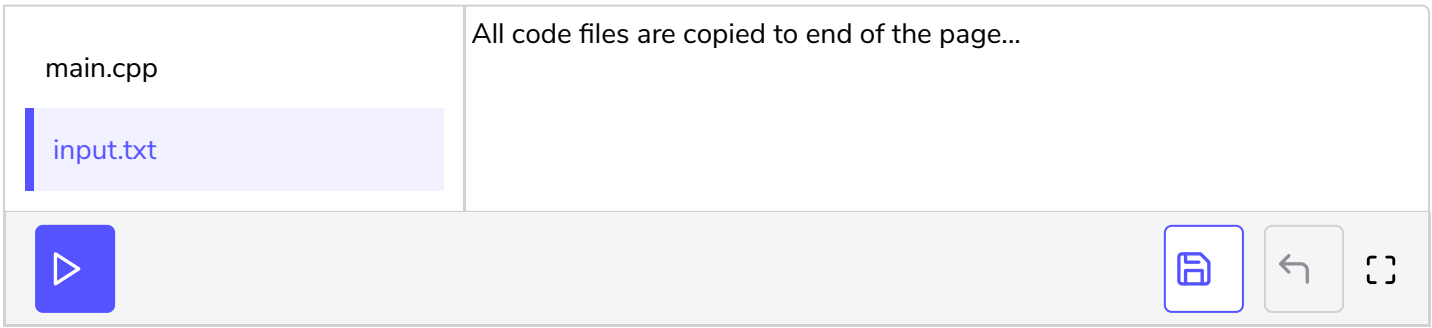
$b >= \sqrt{n}$

Which means if we find two factors, `a` and `b` of `n` such that `ab=n`, one is always less than or equal to $\sqrt{n}$ and the other one is greater than or equal to $\sqrt{n}$.

Factors always come in pairs, except when the number is a perfect square. In which case both factors are equal.

Based on this observation, we only need to iterate up to $\sqrt{N}$ times. The complexity is reduced to $O(\sqrt{N})$.

Below is the optimized code for the above problem.



In the next lesson, we'll see how we can use the same observation to speed up the primality test. *The primality test determines whether the input is prime or not.*

Code Files Content !!!

⇓ ⇓

```
------------------------------------------------------------------------
|  main.cpp [1]
------------------------------------------------------------------------


#include
#include
#define lli long long int
using namespace std;

int print_factors_count(lli N) {
```

```
  int cnt = 0;

  for (int i = 1; i <= N; i ++)
    if (N % i == 0)
      cnt ++;

  return cnt;
}

int main() {
  ifstream cin("input.txt");

  int N;
  cin >> N;
  cout << print_factors_count(N);

  return 0;
}
```

```
-------------------------------------------------------------------------
|  input.txt [1]
-------------------------------------------------------------------------

36
```

```
*************************************************************************
```

```
-------------------------------------------------------------------------
|  main.cpp [2]
-------------------------------------------------------------------------

#include
#include
#define lli long long int
using namespace std;

int print_factors_count(lli N) {
  int cnt = 0;

  for (int i = 1; i * i <= N; i ++)
    if (N % i == 0) {
      cnt ++;
      if (i != N/i)
        cnt ++;
    }

  return cnt;
}

int main() {
  ifstream cin("input.txt");

  int N;
  cin >> N;
```

```
  cout << print_factors_count(N);
  return 0;
}


------------------------------------------------------------------------------
|  input.txt [2]
------------------------------------------------------------------------------


36




********************************************************************************
```