</ CodeNCode >

Number Theory

L01 : Primality Test

What is Primality test?

Primality test is to determine whether the input integer is a prime number or not.

Example :

Input : 5	output : true
Input : 12	output : false

</ CodeNCode >

Naive Approach

```
bool isPrime(int n)
{
        if(n == 1)
                return false;

        for(int i=2;i<n;i++)
        {
                if(n % i == 0)
                        return false;
        }
        return true;
}
```

Time Complexity : O(n)

</ CodeNCode >

Better Approach

All divisors of a number N occur in pairs of (a , b) s.t. $a*b = N$

For example 12 has following divisors
d = 1 , 2 , 3 , 4 , 6 , 12.
Pairs are : (1 , 12) , (2 , 6) , (3 , 4)

</ CodeNCode >

Better Approach

Claim : for a divisor pair (a , b) one of them lies below sqrt(N) and other lies above sqrt(N).

Proof :

There would be 3 cases

Case 1 : both a and b are below sqrt(N)

Case 2 : both a and b are above sqrt(N)

Case 3 : one is below sqrt(N) , and above sqrt(N)

</ CodeNCode >

Better Approach

Case 2 : Both a and b are above sqrt(N).

Let's assume that this statement is true , hence
a > sqrt(N) b > sqrt(N)
But then a * b > N
Which contradicts the fact that a * b = N.
Hence , Case 2 is not true.

Better Approach

Case 3 : one is below sqrt(N) , and above sqrt(N)

a = sqrt(N) * sqrt(N) / b . . . eq(1)

subCase 1 : b < sqrt(N) gives 1 < sqrt(N) / b

a = sqrt(N) * (1 + x)

Hence a > sqrt(N)

subCase 2 : b > sqrt(N) gives 1 > sqrt(N) / b

a = sqrt(N) * (1 - x)

Hence a < sqrt(N)

Implementation

12 has pairs (1 , 12) (2 , 6) , (3 , 4)

```
bool isPrime(int n)
{
        if(n == 1) return false;

        for(int i=2;i*i<=n;i++)
        {
                if(n % i == 0)
                return false;
        }
        return true;
}
```

Time Complexity : O(sqrt(N))

</ CodeNCode >