fsinan Name changes fc91932 on Mar 10, 2018

1 contributor

34 lines (26 sloc) 1.52 KB Raw Blame History

Sieve of Eratosthenes

Sieve of Eratosthenes is an algorithm for finding all the prime numbers in a segment **[1 ; n]** using **O(n *log log* n)** operations.

The idea is simple: at the beginning we write down a row of numbers and eliminate all numbers divisible by 2, except number 2 itself, then divisible by 3, except number 3 itself, next by 7, 11, and all the remaining prime numbers till ***n***.

##Implementation

```
int n;
vector<char> prime (n + 1, true);
prime[0] = prime[1] = false;
for (int i = 2; i * i <= n; ++i)
    if (prime[i])
        if (i * 1ll * i <= n)
            for (int j = i * i; j <= n; j += i)
                prime[j] = false;
```

This code firstly marks all numbers except zero and number one as prime numbers, then it begins the process of sifting composite numbers. For this purpose we go through all numbers **2** to **n** in a cycle, and if the current number **i** is a prime number, then we mark all numbers that are multiple to it as composite numbers. In doing so, we are starting from **i^2** as all lesser numbers that are multiple to **i** necessary have prime factor which is less than **i**, and that means that all of them were sifted earlier. (Since **i^2** can easily overflow the type **int** the additional verification is done using type **long long** before the second nested cycle).

Using such implementation the algorithm consumes **O(n)** of the memory (obviously) and performs **O(n *log log* n)** (this is being proved in the next section).