



Lecture:

Linear Sieve & Factorization

Unit:

8 - Mathematics

Instructor:

Marcela Cruz :-)

## 8-2. Linear Sieve and Factorization

### Some facts and approximations

- Number of primes  $\leq n \approx \frac{n}{\log n}$ .
- $k$ -th prime  $\approx k \log k$ .
- Number of prime factors of  $n = O(\log n)$ .
- Number of different prime factors of  $n = O\left(\frac{\log n}{\log \log n}\right)$ .

### Linear Sieve

*Given a number  $n$ , find all prime numbers in a segment  $[2, n]$ .*

The standard way of solving such task is to use the sieve of Eratosthenes. This algorithm is very simple, but it has runtime  $O(n \log \log n)$ .

The next linear algorithm is interesting by its simplicity: it isn't any more complex than the classic sieve of Eratosthenes.

### Algorithm

Our goal is to calculate **minimum prime factor** `lp[i]` for every number `i` in the segment `[2, n]`. We need to store the list of all the found prime numbers - let's call it `pr[]`.

1. We'll initialize the values `lp[i]` with zeros.
2. Now we'll go through the numbers from 2 to  $n$ . We have two cases for the current number `i`:
  - `lp[i] == 0` : that means that `i` **is prime**. We assign `lp[i] = i` and add `i` to the end of the list `pr[]`.
  - `lp[i] != 0` : that means that `i` **is composite**, and its minimum prime factor is `lp[i]`.

In both cases we update values of `lp[]` for the numbers that are divisible by `i`.

We want to set a value `lp[]` at most once for every number. We can do it as follows:

Let's consider numbers `x_j = i * p_j`, where `p_j` are all prime numbers less than or equal to `lp[i]` (this is why we need to store the list of all prime numbers). We'll set a new value `lp[x_j] = p_j` for all numbers of this form.

### Complexity

- **Time:**  $O(n)$ .
- **Space:**  $O(n)$ .

## Integer Factorization

### Trial division

- Most basic algorithm to find a prime factorization.
- We divide by each possible divisor  $d$ .
- We only need to test the divisors  $2 \leq d \leq \sqrt{n}$ .
- The smallest divisor is a prime number, we remove the factor from the number and repeat the process.

### Complexity

- **Time :**  $O(\sqrt{n})$ .
- **Space:**  $O\left(\frac{\log n}{\log \log n}\right)$ .

### Precomputed primes

Once we removed the factors  $p$  from the number  $n$ , there is no need to check if multiples of  $p$  are factors.

We precompute all prime numbers with the sieve until  $\sqrt{n}$  and test them individually.

### Complexity

- **Time:**  $O\left(\frac{\sqrt{n}}{\log \sqrt{n}}\right) = O\left(\frac{\sqrt{n}}{\log n}\right)$ .
- **Space:**  $O\left(\frac{\log n}{\log \log n}\right)$ .

### Using linear sieve

If we computed the linear sieve mentioned above then we already know the smallest divisor  $p$  of  $n$ .

We remove the factors  $p$  from the number and repeat the process.

### Complexity

- **Time:**  $O(\log n)$ .
- **Space:**  $O\left(\frac{\log n}{\log \log n}\right)$ .

## Practice problems

- [UVa 00406 - Prime Cuts](#)
- [UVa 00543 - Goldbach's Conjecture](#)
- [UVa 10394 - Twin Primes](#)
- [UVa 11466 - Largest Prime Divisor](#)
- [UVa 00993 - Product of digits](#)