



Signup and contribute to Notes.

[Get Started Now](#)[← Notes](#)

▲ Number Theory - III

2

CodeMonk

Euler's totient function

Sieve of Eratosthenes

Euler's totient function

Description: $\phi(N)$ counts the number of integers from 1 to N inclusive that are relatively prime to N.

Implementation: let me remind you that factorization is the way to represent given number as a product of primes. And it's easy to see that for every number such representation is unique. For example:

$$8 = 2^3$$

$$11 = 11$$

$$36 = 2^2 * 3^2$$

$$935 = 5 * 11 * 17$$

$$5136 = 2^4 * 3 * 107$$

So, implementation is based on factorization:

```
int phi(int n) {
```

```

int res = n;
for (int i = 2; i * i <= n; ++i) {
    if (n % i == 0) {
        while (n % i == 0) {
            n /= i;
        }
        res -= res / i;
    }
}
if (n != 1) {
    res -= res / n;
}
return res;
}

```

It works in $O(\sqrt{N})$ time. How to make it faster read further.

Modification of Sieve of Eratosthenes for fast factorization

Implementation: let's see how we can factorize N in $O(\sqrt{N})$ time

```

vector<int> factorize(int n) {
    vector<int> res;
    for (int i = 2; i * i <= n; ++i) {
        while (n % i == 0) {
            res.push_back(i);

```

```

        n /= i;
    }
}
if (n != 1) {
    res.push_back(n);
}
return res;
}

```

At every step we are looking for a minimal prime number that divides our current **N**. This is the main idea of Sieve's modification. Let's construct such array which in **O(1)** time will give us this number:

```

int minPrime[n + 1];
for (int i = 2; i * i <= n; ++i) {
    if (minPrime[i] == 0) { //if i is prime
        for (int j = i * i; j <= n; j += i) {
            minPrime[j] = i;
        }
    }
}
for (int i = 2; i <= n; ++i) {
    if (minPrime[i] == 0) {
        minPrime[i] = i;
    }
}

```

Now we can factorize N in $O(\log(N))$ time using this modification:

```
vector<int> factorize(int n) {
    vector<int> res;
    while (n != 1) {
        res.push_back(minPrime[n]);
        n /= minPrime[n];
    }
    return res;
}
```

Conditions: you can implement this modification only if you're allowed to create an array of integers with size N .

Advices: this approach is useful when you need to factorize a lot of times some not very large numbers. It's not necessary to build such modified Sieve in every problem where you need to factorize something. Moreover you can't build it for such large N like 10^9 or 10^{12} . So, use factorization in $O(\sqrt{N})$ instead.

Cool fact: if factorization of N is $p_1^{q_1} * p_2^{q_2} * \dots * p_k^{q_k}$ then N has $(q_1 + 1) * (q_2 + 1) * \dots * (q_k + 1)$ distinct divisors.

Sieve of Eratosthenes on the segment

Sometimes you need to find all primes not in range $[1...N]$ but in range $[L...R]$, where R is large enough.

Conditions: you're allowed to create an array of integers with size $(R - L + 1)$.

Implementation:

```
bool isPrime[r - 1 + 1]; //filled by true
for (long long i = 2; i * i <= r; ++i) {
    for (long long j = max(i * i, (1 + (i - 1)) / i * i); j <= r; j += i) {
        isPrime[j - 1] = false;
    }
}
for (long long i = max(1, 2); i <= r; ++i) {
    if (isPrime[i - 1]) {
        //then i is prime
    }
}
```

The approximate complexity is $O(\sqrt{R}) * \text{const}$

Advices: again it's not necessary to build such Sieve if you need to check just several numbers for primality. Use the following function instead which works in $O(\sqrt{N})$ for every number:

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

```
}
```



0

Tweet




0

AUTHOR



Boris Sokolov

 C++ Developer at Module...

 Simferopol, Crimea, Russian Federation

 3 notes

TRENDING NOTES

[Number Theory - III](#)

written by Boris Sokolov

[Exact String Matching Algorithms](#)

written by Alei Reyes

[Binary Indexed Tree or Fenwick Tree](#)

written by Chandan Mittal

[Small tricks in for loop](#)

written by Rajagopal Rangeesh

[Strings And String Functions](#)

written by Vinay Singh

[more ...](#)

ABOUT US

[Blog](#)

[Engineering Blog](#)

[Updates & Releases](#)

[Team](#)

[Careers](#)

[In the Press](#)

HACKEREARTH

[API](#)

[Chrome Extension](#)

[CodeTable](#)

[HackerEarth](#)

[Academy](#)

[Developer Profile](#)

[Resume](#)

[Campus](#)

[Ambassadors](#)

[Get Me Hired](#)

[Privacy](#)

[Terms of Service](#)

DEVELOPERS

[AMA](#)

[Code Monk](#)

[Judge Environment](#)

[Solution Guide](#)

[Problem Setter](#)

[Guide](#)

[Practice Problems](#)

[HackerEarth](#)

[Challenges](#)

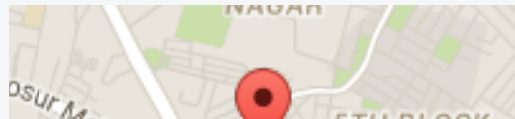
[College Challenges](#)

RECRUIT

[Developer Sourcing](#)

[Lateral Hiring](#)

REACH US



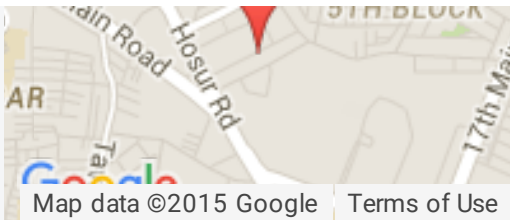
IIIrd Floor,
Salarpuria Business

Campus Hiring

FAQs

Customers

Annual Report



Center,
4th B Cross Road,
5th A Block,
Koramangala
Industrial Layout,
Bangalore,
Karnataka 560095,
India.



contact@hackerear



+91-80-4155-
4695



+1-650-461-
4192



© 2015 HackerEarth