- Competitions
- TopCoder Networks
- Events
- Statistics
- Tutorials
  - Overview
  - Algorithm Tutorials
  - Software Tutorials
  - Marathon Tutorials
  - Wiki
- Forums
- Surveys
- My TopCoder
- Help Center
- About TopCoder

UML TOOL

Member Search:
Handle: Go
Advanced Search

Algorithm Tutorials

Prime Numbers, Factorization and Euler Function

Archive
Printable view
Discuss this article
Write for TopCoder

By medv
TopCoder Member

In addition to being a TopCoder member, medv is a lecturer in Kiev National University's cybernetics faculty.

Prime numbers and their properties were extensively studied by the ancient Greek mathematicians. Thousands of years later, we commonly use the different properties of integers that they discovered to solve problems. In this article we'll review some definitions, well-known theorems, and number properties, and look at some problems associated with them.

A prime number is a positive integer, which is divisible on 1 and itself. The other integers, greater than 1, are **composite**. **Coprime** integers are a set of integers that have no common divisor other than 1 or -1.

The fundamental theorem of arithmetic:
Any positive integer can be divided in primes in essentially only one way. The phrase 'essentially one way' means that we do not consider the order of the factors important.

One is neither a prime nor composite number. One is not composite

One is neither a prime nor composite number. One is not composite because it doesn't have two distinct divisors. If one is prime, then number 6, for example, has two different representations as a product of prime numbers: 6 = 2 * 3 and 6 = 1 * 2 * 3. This would contradict the fundamental theorem of arithmetic.

Euclid's theorem:
There is no largest prime number.

To prove this, let's consider only n prime numbers: $p_1, p_2, \ldots, p_n$. But no prime p_i divides the number

$N = p_1 * p_2 * \ldots * p_n + 1,$

so N cannot be composite. This contradicts the fact that the set of primes is finite.

Exercise 1. Sequence a_n is defined recursively:

$$a_1 = 2,\ a_{n+1} = a_n^2 - a_n + 1$$

Prove that a_i and a_j, $i \neq j$ are relatively prime.

Hint: Prove that $a_{n+1} = a_1 a_2 \ldots a_n + 1$ and use Euclid's theorem.

Exercise 2. Ferma numbers F_n ($n \geq 0$) are positive integers of the form

$$F_n = 2^{2^n} + 1$$

Prove that F_i and F_j, $i \neq j$ are relatively prime.

Hint: Prove that $F_{n+1} = F_0 F_1 F_2 \ldots F_n + 2$ and use Euclid's theorem.

Dirichlet's theorem about arithmetic progressions:

For any two positive coprime integers *a* and *b* there are infinitely many primes of the form *a* + *n***b*, where *n* > 0.

Trial division:
Trial division is the simplest of all factorization techniques. It represents a brute-force method, in which we are trying to divide *n* by every number *i* not greater than the square root of *n*. (Why don't we need to test values larger than the square root of *n*?) The procedure *factor* prints the factorization of number *n*. The factors will be printed in a line, separated with one space. The number *n* can contain no more than one factor, greater than *n*.

```
void factor(int n)
{
  int i;
  for(i=2;i<=(int)sqrt(n);i++)
  {
    while(n % i == 0)
    {
      printf("%d ",i);
      n /= i;
    }
  }
  if (n > 1) printf("%d",n);
  printf("\n");
}
```

Consider a problem that asks you to find the factorization of integer g($-2^{31} < g < 2^{31}$) in the form

$g = f_1 \times f_2 \times \ldots \times f_n$ or $g = -1 \times f_1 \times f_2 \times \ldots \times f_n$

where f_i is a prime greater than 1 and $f_i \le f_j$ for $i < j$.

For example, for g = -192 the answer is -192 = -1 x 2 x 2 x 2 x 2 x 2 x 2 x 3.

To solve the problem, it is enough to use trial division as shown in function

factor.

Sieve of Eratosthenes:
The most efficient way to find all small primes was proposed by the Greek mathematician Eratosthenes. His idea was to make a list of positive integers not greater than *n* and sequentially strike out the multiples of primes less than or equal to the square root of *n*. After this procedure only primes are left in the list.

The procedure of finding prime numbers *gen_primes* will use an array primes[MAX] as a list of integers. The elements of this array will be filled so that

$$\text{primes}[i] = \begin{cases} 1, \text{if } i \text{ is prime} \\ 0, \text{if } i \text{ is composite} \end{cases}$$

At the beginning we mark all numbers as prime. Then for each prime number *i* ($i \geq 2$), not greater than $\sqrt{MAX}$, we mark all numbers $i*i$, $i*(i+1)$, … as composite.

```
void gen_primes()
{
  int i,j;
  for(i=0;i<MAX;i++) primes[i] = 1;
  for(i=2;i<=(int)sqrt(MAX);i++)
    if (primes[i])
      for(j=i;j*i<MAX;j++) primes[i*j] = 0;
}
```

For example, if MAX = 16, then after calling *gen_primes*, the array 'primes' will contain next values:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
primes[*i*]	1	1	1	1	0	1	0	1	0	0	0	1	0	1	0	0

Goldbach's Conjecture:
For any integer n ($n \geq 4$) there exist two prime numbers p_1 and p_2 such that $p_1 + p_2 = n$. In a problem we might need to find the number of essentially different pairs (p_1, p_2), satisfying the condition in the conjecture for a given even number n ($4 \leq n \leq 2\ 15$). (The word 'essentially' means that for each pair (p_1, p_2) we have $p_1 \leq p_2$.)

For example, for $n = 10$ we have two such pairs: $10 = 5 + 5$ and $10 = 3 + 7$.

To solve this,as $n \leq 2^{15} = 32768$, we'll fill an array primes[32768] using function *gen_primes*. We are interested in primes, not greater than 32768.

The function *FindSol*(n) finds the number of different pairs (p_1, p_2), for which $n = p_1 + p_2$. As $p_1 \leq p_2$, we have $p_1 \leq n/2$. So to solve the problem we need to find the number of pairs $(i, n - i)$, such that i and $n - i$ are prime numbers and $2 \leq i \leq n/2$.

```
int FindSol(int n)
{
  int i,res=0;
  for(i=2;i<=n/2;i++)
    if (primes[i] && primes[n-i]) res++;
  return res;
}
```

Euler's totient function
The number of positive integers, not greater than n, and relatively prime with n, equals to Euler's totient function φ (n). In symbols we can state that

φ (n) ={a Î N: $1 \leq a \leq n$, gcd(a, n) = 1}

This function has the following properties:

1. If p is prime, then φ (p) = $p - 1$ and φ (p^a) = p a * (1 – 1/p) for any a.
2. If m and n are coprime, then φ (m * n) = φ (m) * φ (n).

3. If $n = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$, then Euler function can be found using formula:

$\varphi(n) = n * (1 - 1/p\,1) * (1 - 1/p\,2) * ... * (1 - 1/p\,k)$

The function fi(*n*) finds the value of φ(*n*):

```
int fi(int n)
{
  int result = n;
  for(int i=2;i*i <= n;i++)
  {
    if (n % i == 0) result -= result / i;
    while (n % i == 0) n /= i;
  }
  if (n > 1) result -= result / n;
  return result;
}
```

For example, to find φ(616) we need to factorize the argument: $616 = 2^3 * 7 * 11$. Then, using the formula, we'll get:

$\varphi(616) = 616 * (1 - 1/2) * (1 - 1/7) * (1 - 1/11) = 616 * 1/2 * 6/7 * 10/11 = 240.$

Say you've got a problem that, for a given integer *n* ($0 < n \le 10^9$), asks you to find the number of positive integers less than *n* and relatively prime to *n*. For example, for *n* = 12 we have 4 such numbers: 1, 5, 7 and 11.

The solution: The number of positive integers less than *n* and relatively prime to *n* equals to φ(*n*). In this problem, then, we need do nothing more than to evaluate Euler's totient function.

Or consider a scenario where you are asked to calculate a function Answer(*x*, *y*), with *x* and *y* both integers in the range [1, *n*], $1 \le n \le 50000$. If you know Answer(*x*, *y*), then you can easily derive Answer(*k***x*, *k***y*) for any integer *k*. In this situation you want to know how many values of Answer(*x*, *y*) you need to precalculate. The function Answer is not symmetric.

For example, if $n = 4$, you need to precalculate 11 values: Answer(1, 1), Answer(1, 2), Answer(2, 1), Answer(1, 3), Answer(2, 3), Answer(3, 2), Answer(3, 1), Answer(1, 4), Answer(3, 4), Answer(4, 3) and Answer(4, 1).

The solution here is to let res(i) be the minimum number of Answer(x, y) to precalculate, where $x, y \in \{1, \ldots, i\}$. It is obvious that res(1) = 1, because if n = 1, it is enough to know Answer(1, 1). Let we know res(i). So for $n = i + 1$ we need to find Answer(1, i + 1), Answer(2, i + 1), … , Answer(i + 1, i + 1), Answer(i + 1, 1), Answer(i + 1, 2), … , Answer(i + 1, i).

The values Answer(j, i + 1) and Answer(i + 1, j), $j \in \{1, \ldots, i + 1\}$, can be found from known values if GCD(j, i + 1) > 1, i.e. if the numbers j and i + 1 are not common primes. So we must know all the values Answer(j, i + 1) and Answer(i + 1, j) for which j and i + 1 are coprime. The number of such values equals to 2 * φ (i + 1), where φ is an Euler's totient function. So we have a recursion to solve a problem:

$$\text{res}(1) = 1,$$
$$\text{res}(i + 1) = \text{res}(i) + 2 * \text{j}\ (i + 1),\ i > 1$$

Euler's totient theorem:

If n is a positive integer and a is coprime to n, then $a^{\varphi(n)} \circ 1 \pmod{n}$.

Fermat's little theorem:

If p is a prime number, then for any integer a that is coprime to n, we have

$$a^{p} \equiv a \pmod{p}$$

This theorem can also be stated as: If p is a prime number and a is coprime to p, then

$$a^{p-1} \equiv 1 \pmod{p}$$

Fermat's little theorem is a special case of Euler's totient theorem when n is prime.

The number of divisors:

If $n = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$, then the number of its positive divisors equals to

$(a_1 + 1) * (a_2 + 1) * \ldots * (a_k + 1)$

For a proof, let A i be the set of divisors $\{1, p_i, p_i^2, \ldots, p_i^{a_i}\}, 1 \le i \le k$. Any divisor of number n can be represented as a product $x_1 * x_2 * \ldots * x_k$, where x_i Î A_i. As $|A_i| = a_i + 1$, we have

$(a_1 + 1) * (a_2 + 1) * \ldots * (a_k + 1)$

possibilities to get different products $x_1 * x_2 * \ldots * x_k$.

For example, to find the number of divisors for 36, we need to factorize it first: $36 = 2^2 * 3^2$. Using the formula above, we'll get the divisors amount for 36. It equals to $(2 + 1) * (2 + 1) = 3 * 3 = 9$. There are 9 divisors for 36: 1, 2, 3, 4, 6, 9, 12, 18 and 36.

Here's another problem to think about: For a given positive integer n ($0 < n < 2^{31}$) we need to find the number of such m that $1 \le m \le n$, $GCD(m, n) \ne 1$ and $GCD(m, n) \ne m$. For example, for $n = 6$ we have only one such number $m = 4$.

The solution is to subtract from n the amount of numbers, coprime with it (its amount equals to $\varphi(n)$) and the amount of its divisors. But the number 1 simultaneously is coprime with n and is a divisor of n. So to obtain the difference we must add 1. If $n = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$ is a factorization of n, the number n has $(a_1 + 1) * (a_2 + 1) * \ldots * (a_k + 1)$ divisors. So the answer to the problem for a given n equals to

$n - \varphi(n) - (a_1 + 1) * (a_2 + 1) * \ldots * (a_k + 1) + 1$

Practice Room:
Want to put some of these theories into practice? Try out these problems, from the TopCoder Archive:

- Refactoring (SRM 216)
- PrimeAnagrams (SRM 223)
- DivisibilityCriteria (SRM 239)
- PrimePolynom (SRM 259)
- DivisorInc (SRM 302)
- PrimePalindromic (SRM 303)
- RugSizes (SRM 304)
- PowerCollector (SRM 305)
- PreprimeNumbers (SRM 307)
- EngineersPrimes (SRM 181)
- SquareFree (SRM 190)