INFO1056
Aula 11/12
Teoria dos Números

Prof. João Comba

Baseado no Livro Programming Challenges

Outline

- Prime Numbers
- Divisibility
- Modular Arithmetic
- Congruences

Prime Numbers

- Prime number: Integer p > 1 such that
  - only divisible by 1
  - p = a * b
  - 2, 3, 5, 7, 11, 13, 17, 19, 23, 27, ...
- Fundamental Theorem of Arithmetic
  - Every integer can be expressed in only one way as the product of primes
  - 105 = 3 x 5 x 7 32 = 2 x 2 x 2 x 2 x 2

Finding Primes

- Only even prime is 2
- test only odd numbers up to $\sqrt{n}$.

Finding Primes

- Only even prime is 2
- test only odd numbers up to $\sqrt{n}$
- Proof (By contradiction):
  - suppose n is composite with a factor p larger than $\sqrt{n}$
  - n / p must divide n, and also be > $\sqrt{n}$
  - but then $p * n / p > n$ CONTRADICTION

Finding Prime Factors

```
prime_factorization(long x) {
   long i;      /* counter */
   long c;      /* remaining product to factor */

   c = x;
   while ((c % 2) == 0) {
      printf("%ld\n",2);
      c = c / 2;
   }

   i = 3;
   while (i <= (sqrt(c)+1)) {
      if ((c % i) == 0) {
         printf("%ld\n",i);
          c = c / i;
      }
      else
         i = i + 2;
          }

   if (c > 1) printf("%ld\n",c);
}
```

Counting Primes

- How many primes are there ?

Counting Primes

- How many primes are there ?
  - Suppose finite number of primes $p_1, p_2, \ldots, p_n$
  - Let $m = 1 + \prod_{i=1}^{n} p_i$, be a composite number
  - Which prime divide it ?
    - m leaves a reminder of 1 for all primes
    - Therefore, m is also prime CONTRADICTION

The Number of Prime Numbers is Infinite

Outline

- Prime Numbers
- Divisibility
- Modular Arithmetic
- Congruences

Divisibility

- Integer divisibility

$$b \text{ divides } a \qquad a = bk \text{ for some integer } k$$

- How to find all divisors of a given integer ?
  - find prime factorization
  - form all subsets of prime factors:
    - 12: 2 * 2 * 3 (1,2,3,4,6,12)

Greatest Common Divisor (GCD or MDC)

- largest divisor shared by a pair of integers
- GCD(x,y) = 1, x and y are relatively prime
- Euclid's Algorithm:
  - If B divides A, A = BK, therefore:
    - GCD(BK,B) = B or GCD(A,B) = B
  - If A = BT + R for integers T and R
    - GCD(A, B) = GCD(B, R)

Greatest Common Divisor (GCD or MDC)

- Largest divisor shared by a pair of integers
- GCD(x,y) = 1, x and y are relatively prime
- Euclid's Algorithm:
  - If B divides A, A = BK, therefore:
    - GCD(BK,B) = B or GCD(A,B) = B
  - If A = BT + R for integers T and R
    - GCD(A, B) = GCD(B, R)
    - PROOF: GCD(A, B) = GCD(BT+R, B)
      - But BT is divisible by any divisor of B

Greatest Common Divisor (GCD or MDC)

Let $a = 34398$ and $b = 2132$.

$$\begin{aligned} gcd(34398, 2132) &= gcd(34398 \bmod 2132, 2132) = gcd(2132, 286) \\ gcd(2132, 286) &= gcd(2132 \bmod 286, 286) = gcd(286, 130) \\ gcd(286, 130) &= gcd(286 \bmod 130, 130) = gcd(130, 26) \\ gcd(130, 26) &= gcd(130 \bmod 26, 26) = gcd(26, 0) \end{aligned}$$

Therefore, $gcd(34398, 2132) = 26$.

Greatest Common Divisor (GCD or MDC)

$$a \cdot x + b \cdot y = gcd(a, b)$$

which will prove quite useful in solving linear congruences. We know that $gcd(a, b) = gcd(b, a')$, where $a' = a - b\lfloor a/b \rfloor$. Further, assume we know integers x' and y' such that

$$b \cdot x' + a' \cdot y' = gcd(a, b)$$

by recursion. Substituting our formula for a' into the above expression gives us

$$b \cdot x' + (a - b\lfloor a/b \rfloor) \cdot y' = gcd(a, b)$$

$$34398 \times 15 + 2132 \times -242 = 26.$$

Greatest Common Divisor (GCD or MDC)

```
long gcd1(long p, long q) {
  if (q > p) return(gcd1(q,p));
  if (q == 0) return(p);
  return( gcd1(q, p % q) );
}
```

Greatest Common Divisor (GCD or MDC)

```
long gcd(long p, long q, long *x, long *y) {
   long x1,y1;       /* previous coefficients */
   long g;         /* value of gcd(p,q) */

   if (q > p) return(gcd(q,p,y,x));

   if (q == 0) {
      *x = 1;
      *y = 0;
      return(p);
   }

   g = gcd(q, p%q, &x1, &y1);

   *x = y1;
   *y = (x1 - floor(p/q)*y1);

   return(g);
}
```

Least Common Multiple (LCM) - MMC

When is the next year (after 2000) that the presidencial election (which happens every 4 years) will coincide with census (which happens every 10 years) ? LCM(4,10)=20

Least Common Multiple (LCM or MMC)

$$lcm(x,y) \geq \max(x,y)$$

$$lcm(x,y) \leq xy$$

$$lcm(x,y) = xy/gcd(x,y)$$

Outline

- Prime Numbers
- Divisibility
- Modular Arithmetic
- Congruences

Addition

- *Addition* — What is $(x + y) \bmod n$? We can simplify this to

$$((x \bmod n) + (y \bmod n)) \bmod n$$

to avoid adding big numbers. How much small change will I have if given \$123.45 by my mother and \$94.67 by my father?

$$(12,345 \bmod 100) + (9,467 \bmod 100) = (45 + 67) \bmod 100 = 12 \bmod 100$$

Subtraction

- *Subtraction* — Subtraction is just addition with negative values. How much small change will I have after spending \$52.53?

$$(12 \bmod 100) - (53 \bmod 100) = -41 \bmod 100 = 59 \bmod 100$$

Notice how we can convert a negative number mod n to a positive number by adding a multiple of n to it. Further, this answer makes sense in this change example. It is usually best to keep the residue between 0 and $n - 1$ to ensure we are working with the smallest-magnitude numbers possible.

Multiplication

- *Multiplication* — Since multiplication is just repeated addition,

$$xy \bmod n = (x \bmod n)(y \bmod n) \bmod n$$

How much change will you have if you earn \$17.28 per hour for 2,143 hours?

$$(1,728 \times 2,143) \bmod 100 = (28 \bmod 100) \times (43 \bmod 100) = 4 \bmod 100$$

Further, since exponentiation is just repeated multiplication,

$$x^y \bmod n = (x \bmod n)^y \bmod n$$

Since exponentiation is the quickest way to produce really large integers, this is where modular arithmetic really proves its worth.

Finding the Last Digit

- *Finding the Last Digit* — What is the last digit of 2^{100}? Sure we can use infinite precision arithmetic and look at the last digit, but why? We can do this computation by hand. What we really want to know is what $2^{100} \bmod 10$ is. By doing repeated squaring, and taking the remainder $\bmod 10$ at each step we make progress very quickly:

$$\begin{aligned}
2^3 \bmod 10 &= 8 \\
2^6 \bmod 10 &= 8 \times 8 \bmod 10 \to 4 \\
2^{12} \bmod 10 &= 4 \times 4 \bmod 10 \to 6 \\
2^{24} \bmod 10 &= 6 \times 6 \bmod 10 \to 6 \\
2^{48} \bmod 10 &= 6 \times 6 \bmod 10 \to 6 \\
2^{96} \bmod 10 &= 6 \times 6 \bmod 10 \to 6 \\
2^{100} \bmod 10 &= 2^{96} \times 2^3 \times 2^1 \bmod 10 \to 6
\end{aligned}$$

Outline

- Prime Numbers
- Divisibility
- Modular Arithmetic
- Congruences

Congruences

Congruences are an alternate notation for representing modular arithmetic. We say that $a \equiv b (\text{mod } m)$ if $m|(a-b)$. By definition, if $a \bmod m$ is b, then $a \equiv b (\text{mod } m)$.

- *Addition and Subtraction* — Suppose $a \equiv b (\text{mod } n)$ and $c \equiv d (\text{mod } n)$. Then $a + c \equiv b + d (\text{mod } n)$. For example, suppose I know that $4x \equiv 7 (\text{mod } 9)$ and $3x \equiv 3 (\text{mod } 9)$. Then

$$4x - 3x \equiv 7 - 3 (\text{mod } 9) \rightarrow x \equiv 4 (\text{mod } 9)$$

- *Multiplication* — It is apparent that $a \equiv b (\text{mod } n)$ implies that $a \cdot d \equiv b \cdot d (\text{mod } n)$ by adding the reduced congruence to itself d times. In fact, general multiplication also holds, i.e., $a \equiv b (\text{mod } n)$ and $c \equiv d (\text{mod } n)$ implies $ac \equiv bd (\text{mod } n)$.

Congruences

- *Division* — However, we cannot cavalierly cancel common factors from congruences. Note that $6 \cdot 2 \equiv 6 \cdot 1 (\text{mod } 3)$, but clearly $2 \not\equiv 1 (\text{mod } 3)$. To see what the problem is, note that we can redefine division as multiplication by an inverse, so a/b is equivalent to ab^{-1}. Thus we can compute $a/b (\text{mod } n)$ if we can find the inverse b^{-1} such that $bb^{-1} \equiv 1 (\text{mod } n)$. This inverse does not always exist – try to find a solution to $2x \equiv 1 (\text{mod } 4)$.

  We *can* simplify a congruence $ad \equiv bd (\text{mod } dn)$ to $a \equiv b (\text{mod } n)$, so we can divide all three terms by a mutually common factor if one exists. Thus $170 \equiv 30 (\text{mod } 140)$ implies that $17 \equiv 3 (\text{mod } 14)$. However, the congruence $a \equiv b (\text{mod } n)$ must be false (i.e., has no solution) if $gcd(a, n)$ does not divide b.

Solving Linear Congruences

- **Congruences**

$$ax \equiv b (\mathrm{mod}\ n)$$

- **Example:** $ax \equiv 1 (\mathrm{mod}\ n)$ **has solution if**

$$gcd(a, n) = 1$$

$$ax \equiv 1 (\mathrm{mod}\ n) \rightarrow ax \equiv a \cdot x' + n \cdot y' (\mathrm{mod}\ n)$$

Clearly $n \cdot y' \equiv 0 (\mathrm{mod}\ n)$, so in fact this inverse is simply the x' from Euclid's algorithm.

Congruences

- $\gcd(a, b, n) > 1$ — Then we can divide all three terms by this divisor to get an equivalent congruence. This gives us a single solution mod the new base, or equivalently $\gcd(a, b, n)$ solutions $(\mathrm{mod}\ n)$.
- *$gcd(a, n)$ does not divide b* — Then, as described above, the congruence can have no solution.
- *$gcd(a, n) = 1$* — Then there is one solution $(\mathrm{mod}\ n)$. Further, $x = a^{-1}b$ works, since $aa^{-1}b \equiv b (\mathrm{mod}\ n)$. As shown above, this inverse exists and can be found using Euclid's algorithm.

Congruences

The *Chinese remainder theorem* gives us a tool for working with systems of congruences over different moduli. Suppose there is exists an integer x such that $x \equiv a_1 (\text{mod } m_1)$ and $x \equiv a_2 (\text{mod } m_2)$. Then x is uniquely determined (mod $m_1 m_2$) if m_1 and m_2 are relatively prime.

To find this x, and thus solve the system of two congruences, we begin by solving the linear congruences $m_2 b_1 \equiv 1 (\text{mod } m_1)$ and $m_1 b_1 \equiv 1 (\text{mod } m_2)$ to find b_1 and b_2 respectively. Then it can be readily verified that

$$x = a_1 b_1 m_2 + a_2 b_2 m_1$$

is a solution to both of the original congruences. Further, the theorem readily extends to systems of an arbitrary number of congruences whose moduli are all pairwise relatively prime.

Diophantine Equations

- **Variables are restricted to integers**

$$ax - ny = b$$

- **is equivalent to**

$$ax \equiv b (\text{mod } n)$$

$$a^n + b^n = c^n$$

Light, More Light

The Problem

There is man named "mabu" for switching on-off light in our University. He switches on-off the lights in a corridor. Every bulb has its own toggle switch. That is, if it is pressed then the bulb turns on. Another press will turn it off. To save power consumption (or may be he is mad or something else) he does a peculiar thing. If in a corridor there is `n' bulbs, he walks along the corridor back and forth `n' times and in i'th walk he toggles only the switches whose serial is divisable by i. He does not press any switch when coming back to his initial position. A i'th walk is defined as going down the corridor (while doing the peculiar thing) and coming back again.

Now you have to determine what is the final condition of the last bulb. Is it on or off?

The Input

The input will be an integer indicating the n'th bulb in a corridor. Which is less then or equals 2^32-1. A zero indicates the end of input. You should not process this input.

The Output

Output "`yes`" if the light is on otherwise "`no`" , in a single line.

Sample Input

```
3
6241
8191
0
```

Sample Output

```
no
yes
no
```

Light, More Light

```
#include <iostream>
#include <cmath>

using namespace std;

int main() {
  unsigned int n;
  double raiz;

  cin >> n;

  while (n != 0) {
        raiz = sqrt(n);
        if (floor(raiz) == ceil(raiz))
            cout << "yes" << endl;
        else
            cout << "no" << endl;
        cin.ignore();
        cin >> n;
  }
  return 0;
}
```