CS 91 USACO

Bronze Division

Unit 5: Number Algorithms

LECTURE 25: PRIME NUMBERS

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- Divisibility
- Prime Factorization
- GCD & LCM
- Practice: Super Prime Ribs (sprime)

Prime Factorization

SECTION 1

Prime Factorization

- A positive integer aaa is called a **divisor** or a **factor** of a non-negative integer b if b is divisible by a, which means that there exists some integer k such that $b = k\ a$. An integer $n>1$ is **prime** if its only divisors are 1 and n. Integers greater than 1 that are not prime are **composite**.
- Every positive integer has a unique **prime factorization**: a way of decomposing it into a product of primes, as follows:

$$n=p_1^{a1}\ p_2^{a2}\ \cdots\ p_k^{ak}$$

where the p_i are distinct primes and the a_i are positive integers.

Find All Factors

- Check x from 2 to n/2, if x can divide n, then, x is n's factor.
- Add 1, and n to n's factor list.

```java
public static ArrayList<Integer> factor(int n){
    ArrayList<Integer> p = new ArrayList<Integer>();
    for (int i=2; i<=n/2; i++){
        if (n%i==0) p.add(i);
     }
    p.add(0, 1);
    p.add(n);
    return p;
 }
```

Prime Factorization

- Find all prime numbers from 2 to sqrt(n), if the prime number x can divide n, then, put the x into the map of prime factors for n. Map is used here because it can be used to track the occurrence of the prime factors.

```java
public static Map<Integer, Integer> primeFactor(int n){
    Map<Integer, Integer> pf = new HashMap<Integer, Integer>();
    int x = 2;
    while (n!=1){
        if (x>Math.sqrt(n)) x = n;
        if (isPrime(x)) {
          while (n%x==0) {
              if (!pf.containsKey(x)){
                  pf.put(x, 0);
              }
              pf.put(x, pf.get(x)+1);
              n/=x;
          }
         }
        x++;
      }
    return pf;
 }
```

Divisibility

- This algorithm runs in O(sqrt(n)) time time, because the for loop checks divisibility for at most sqrt(n) values. Even though there is a while loopinside the for loop, dividing

  `n` by `i`

  quickly reduces the value of

  `n`,

  which means that the outer for loop runs less iterations, which actually speeds up the code.

Prime Factorization of a Number

- At this point, the for loop terminates, because i is already 3 which is greater than floor(sqrt(7)). In the last step, we add 7
- to the list of factors v, because it otherwise won't be added, for a final prime factorization of

{2, 2, 3, 3, 7}.

Solution - Counting Divisors

SECTION 2

Counting Divisors

- First, let's discuss an important property of the prime factorization. Consider:

  $x=p_1^{a1}p_2^{a2}\cdots p_k^{ak}$

- Then the number of divisors of x is simply $(a_1+1)\cdot(a_2+1)\cdots(a_k+1)$

- Why is this true? The exponent of pi in any divisor of x must be in the range $[0,a_i]$ and each different exponent results in a different set of divisors, so each pi contributes a_i+1 to the product.

  x can have $O(\log x)$ distinct prime factors, so if we can find the prime factorization of x efficiently, we can answer queries in $O(\log x)$ time instead of the previous $O(x)$ time.

Counting Divisors

Here's how we find the prime factorization of x in `O(logx)` time with `O(x logx)` preprocessing:

- For each $k \leq 10^6$, find any prime number that divides k. We can use the Sieve of Eratosthenes to find this efficiently.
- For each x, we can then find the prime factorization by repeatedly dividing x by a prime number that divides x until x=1.

GCD & LCD

SECTION 3

GCD

- The greatest common divisor (GCD) of two integers a and b is the largest integer that is a factor of both a and b.
- In order to find the GCD of two non-negative integers, we use the Euclidean Algorithm

```
a = p * f
b = q * f, where (p, q) = 1
```

LCM

- The least common multiple (LCM) of two integers a and b is the smallest integer divisible by both a and b.
- The LCM can easily be calculated from the following property with the GCD:
- In order to find the GCD of two non-negative integers, we use the Euclidean Algorithm

```
a = p * f
b = q * f, where (p, q) = 1
m = LCM(a, b) = p * q * f = a * b / f
f * m = a * b
```

Practice: Super Prime Ribs (sprime)

SECTION 4

Problem Statement

- Butchering Farmer John's cows always yields the best prime rib. You can tell prime ribs by looking at the digits lovingly stamped across them, one by one, by FJ and the USDA. Farmer John ensures that a purchaser of his prime ribs gets really prime ribs because when sliced from the right, the numbers on the ribs continue to stay prime right down to the last rib, e.g.:

```
7  3  3  1
```

- The set of ribs denoted by 7331 is prime; the three ribs 733 are prime; the two ribs 73 are prime, and, of course, the last rib, 7, is prime. The number 7331 is called a superprime of length 4.

Problem Statement

- Write a program that accepts a number N (1 <=N<=8) of ribs and prints all the superprimes of that length.
- The number 1 (by itself) is not a prime number.

INPUT FORMAT (file sprime.in):

- A single line with the number N.

SAMPLE INPUT

4

OUTPUT FORMAT (file sprime.out):

The superprime ribs of length N, printed in ascending order one per line.

SAMPLE OUTPUT

```
2333
2339
2393
2399
2939
3119
3137
3733
3739
3793
3797
5939
7193
7331
7333
7393
```

Nature of Problem

- Expansion of arraylists.
- 1 Digit Prime: 2 3 5 7.
- 2 Digit Prime: [2, 3, 5, 7] X [1, 3, 7, 9]
- 3 Digit Prime: [2, 3, 5, 7] X [1, 3, 7, 9] X [1, 3, 7, 9]

…

Keep doing up to N digits Primes