用递归算法求素数

宋敦波
(西昌学院 信息技术系,四川 西昌 615013)

【摘 要】本文根据素数的一些基本性质,提出使用一种新的算法——递归算法来解决素数判定及求给定范围内素数的问题。同时还对该算法进行了分析和改进,最后用JAVA语言实现其代码。

【关键词】素数;递归算法;试除法;算法分析
【中图分类号】TP312 【文献标识码】A 【文章编号】1673-1891(2009)02-0049-02

随着计算机技术的发展,特别是密码学的发展,判定所给自然数n是否是素数这一问题,不仅在理论上有重要意义,而且在实践中也具有很高的应用价值。本文将使用递归算法来解决素数判定问题及求给定范围内的素数,并对算法进行分析。

所谓素数,是指除了能被1和它本身整除而不能被其他任何数整除的数。根据素数的定义,只需用2到X-1去除X,如果都除不尽则X是素数,否则,只要其中有一个数能除尽则X不是素数。

1 算法实现及其分析

试除法是判别一个整数是否为素数的常用方法。假定任意给定一个整数X,用Y从2开始来试除它,如果能整除,则这个整数不是素数,否则,用下一个Y值来试除,直到Y大于X的平方根为止。

如果不仔细思考,会认为这样做已经是非常快了,但事实上它做了很多多余的工作。以下作简单分析:首先从奇偶性上看,所有的素数中只有2是偶数,也就是说几乎所有素数都是奇数,而在试除的过程中,还用许多偶数去试除。同样的道理,所得的奇数的倍数也能整除,那也不必试除这些数。使用这种算法的时间复杂度为:O(sqrt(n)/2)。

可以对试除法作一些改进。在寻找给定的数X的过程中只生成奇数,选Y时选3至X的平方根范围内的全体素数,也即是先判定Y是否是素数,如果是再用Y去试除。也就是说要判断X是否是素数只需要使用sqrt(X)/2以下的素数进行试除,如果所有这些素数都不能整除X,则X是素数。用程序实现,则可使用递归算法实现。以下是使用JAVA语言来实现其代码(为简单,此处只列出主要方法)。

```
private boolean isPrime(long i)
{
  if (i <= 2)
  {
    if (i==2)  return true;
    else
        return false;
  }
  If(i%2==0) return false;
  for (long t = (long) Math.sqrt(i); t >= 2; t--)
  {
    if (isPrime(t)) { if (i % t == 0) return false; }
  }
  return true;
}
```

使用此段代码可以判断任意X是否是素数,但程序运行后非常耗时。是什么原因呢?下面作一简单分析:此段代码在求素数时,由于事先不知道sqrt(X)/2以下的素数有哪些,因此需要找出这些素数,要找到这些素数又要进行递归。当找到并试除完后,还有(sqrt(X)/2)-1个数要重复做以上的步骤,从而增加时间复杂度。使用这种算法的时间复杂度约为:O(2n/2×sqrt(n)/2)。

2 改进后的递归算法

造成这么大的时间复杂度是由于要重复的求sqrt(X)/2以下的素数,如果能够用一个数组来存取每次求出的素数,当要求Y以下的素数时先判断Y以下的素数是否在数组中,如果在则直接从数组中取数进行试除,如果不在再进行递归。采用此法对上述算法进行改进后,时间复杂度为多少呢?从表1中可知,当判断X是否为素数的次数就是试除sqrt(X)以下的素数个数。而给定的某一段范围内的素数个数约为:ln(X)-3/2。所以使用这种算法的时间复杂度约为:O(ln(sqrt(n))-3/2)。

3 代码实现

下面是根据上面的算法用JAVA语言编写的一段完整的代码,在此段代码中求1至1000000内的所有素数大约用了10秒的时间。

```
import java.util.ArrayList;
```

收稿日期:2009-04-24
作者简介:宋敦波(1972-),男,硕士,讲师,主要从事计算机教学及研究。

表1 改进后X、Y和试除次数关系

X	Y=sqrt(X)	试除次数
100	10	4
1000	31	12
10000	100	25
100000	316	66
1000000	1000	168

```
public class Prime {
  ArrayList primeArray = new ArrayList();
  private boolean isPrime(long i) {
    if (i <= 2) {
      if (i == 2) {
        primeArray.add(i);
        return true;
      } else {
        return false;
      }
    }
    for (long t = (long) Math.sqrt(i); t >= 2; t--) {
      long value = (Long) (primeArray.get
(primeArray.size()-1));
      if (value >= t) {
        for (int a = 0; a < primeArray.size(); a++) {
          long k = (Long) primeArray.get(a);
          if (i % k == 0) {
            return false;
          }
        }
        return true;
      }
      if (isPrime(t)) {
        primeArray.add(t);
        if (i % t == 0) {
          return false;
        }
      }
    }
    return true;
  }
  public static void main(String args[])
  {
    Prime prime =new Prime();
    for(long i=1;i<1000000;i++)
    {
      if(prime.isPrime(i)) System.out.println(i+",");
    }
  }
}
```

4 小结

通过以上用递归算法求素数的探讨和分析，不难看出，缺少数学的基本知识很难设计好的算法，但是如果一味地只考虑数学原理，而忽略计算机的本质特征，也会有同样的问题。这一点在对改进后的递归算法分析中不难看出。使用递归算法求素数的方法，逻辑清晰且代码实现比较简单，对素数表的生成也是动态的，能充分节约计算机资源。但是此法不适合求大素数。

注释及参考文献：

[1] 于秀源，薛昭雄.密码学与数论基础[M].济南:山东科学技术出版社,1993.
[2] 李文卿.数论及其应用[M].北京:北京大学出版社,2000.
[3] 严蔚敏,吴伟民.数据结构[M].北京:清华大学出版社,1997.

Using the Method of Recursive Algorithm to Get Prime Number

SONG Dun-bo

(Department of Information Technology, Xichang College, Xichang, Sichuan 615013)

Abstract: This paper proposed a new algorithm: the recursive algorithm to judge the prime number and find prime number in given range based on the basic nature of prime number. We also analyzed and improved the algorithm in this paper. Finally, we used JAVA language to implement the algorithm.

Key words: Prime number; Recursive algorithm; Trial Division; Algorithm analysis