SHANDONG UNIVERSITY

密码学引论作业：素性检测和因子分解

网络空间安全学院 (研究院)
2021 级网安三班
谢子洋 202100460116

2023 年 5 月 1 日

目录

1 Miler-Rabin 素性检测

1.1 Miler-Rabin 算法代码实现

具体算法描述本文不再赘述, Python 实现如下:

```python
#Python实现MilerRabin算法
import gmpy2
import random
def isPrime_MilerRabin(n):
    m=n-1
    k=0
    while(m%2==0):
        k+=1
        m=m//2
    a=random.randint(1,n-1)
    b=gmpy2.powmod(a,m,n)
    if(b==1):return True
    for i in range(k):
        if b==n-1: return True
        b=gmpy2.powmod(b,2,n)
    return False
```

1.2 Miler-Rabin 算法效率测试

在每种 bit 长度下, 随机生成 1000 个该长度随机数, 测量一次素性检测的平均耗时.

得到结果如下表:

表 1: 素性检测时间开销

长度 (bit)	1024	2048	3072	4096
时间 (ms)	0.000391	0.003007	0.009900	0.025691

根据测试结果可估算,4096bit 长整数约为 2048bit 长整数素性检测耗时的 8.5 倍.

Miler-Rabin 素性检测算法的时间复杂度约为 $O((\log n)^3)$, 因此理论上两种长度的整数素性检测时间开销比值约为:

$$(\frac{4096}{2048})^3 = 8$$

.

实验所得数据与理论推导出结果差距较小, 符合实际.

2 利用完全平方数分解合数

2.1 证明完全平方数

设 n=pq, 且 p-q=2d>0, 证明 $n+d^2$ 为完全平方数.

$$\because p-q=2d>0$$

$$\therefore d=\frac{p-q}{2} \text{ 且 } d\in N^+$$

$$\begin{aligned}\therefore n+d^2 &= pq+(\frac{p-q}{2})^2\\ &= pq+\frac{p^2-2pq+q^2}{4}\\ &= (\frac{p+q}{2})^2\end{aligned}$$

$$\because p-q=2d$$

$$\therefore \text{pq 奇偶性相同}$$

$$\therefore \frac{p+q}{2}\in Z$$

$$\therefore n+d^2\text{是完全平方数}$$

$$\text{且}n+d^2=(\frac{p+q}{2})^2$$

2.2 算法描述

已知 n=pq 为两奇素因子乘积, 可提出如下算法计算 n 的素因子:

1. $x <= \sqrt{n}$

2. $x <= x+1$

3. 若 x^2-n 是完全平方数则令 $d=\sqrt{x^2-n}$ 并继续执行, 否则回到 2.

4. 若 $n+d^2$ 是完全平方数则令 $y=\sqrt{n+d^2}$ 并继续执行, 否则回到 2.

5. 计算 (y+d,y-d), 即 n 的两素因子 (p,q), 算法结束

2.3 算法正确性

若 d 满足 $p-q=2d$, 则 $n+d^2$ 是完全平方数, 那么一定存在整数 x 满足 $x^2=n+d^2$.

为找到 $d=\frac{p-q}{2}$, 可以从 $[\sqrt{n}]$ 开始遍历可能的 x. 当 x^2-n 为完全平方数时, 计算可能满足条件的 $d_2=\sqrt{x^2-n}$. 若 $n+d_2^2$ 也为完全平方数, 则 d_2 确实满足条件, 即为所求 d.

此时已知 d 且满足 $p-q=2d>0$, 可将整数分解问题转化为解一元二次方程 f(x)=0:

$$\because \begin{cases} p+q=2\sqrt{n+d^2} \\ p\cdot q=n \end{cases}$$

$$\begin{aligned} \therefore f(x) &= (x+p)(x+q) \\ &= x^2+(p+q)x+pq \\ &= x^2+2\sqrt{n+d^2}x+n \end{aligned}$$

方程 f(x) 的解

$$\begin{cases} x_1=\sqrt{n+d^2}+d \\ x_2=\sqrt{n+d^2}-d \end{cases}$$

即为 p 与 q.

2.4 算法实现

Python 代码实现如下:

```python
#python
import gmpy2
def factor(n):
    x=gmpy2.iroot(n,2)[0]
    while(True):
        #遍历x
        x+=1
        temp=x**2-n
        #计算可能满足条件的d
        d=gmpy2.iroot(temp,2)
        if(d[1]==True):
            d=d[0]
            temp2=n+d**2
            #检查d是否满足要求并对应返回结果
            y=gmpy2.iroot(temp2,2)
            if(y[1]==True):
                y=y[0]
                return(y+d,y-d)
n=2189284635403183
p,q=factor(n)
```

根据上文提出的算法对大整数

$$n=2189284635403183$$

进行因子分解, 得到两个素因子:

$$p=4678901 \quad q=46789783$$

3 因子分解

3.1 测试平台

表 2: 测试平台信息

CPU	INTEL CORE i5-11300H 3.1GHz
Memory	DDR4 3200Hz 16GB
OS	Windows10
Language	SageMath(Python)

3.2 整数因子分解效率测试

直接调用 SageMath 库中的因子分解函数分解给定大整数, 并记录平均耗时.

使用基于 Python 的 SageMath 实现, 代码如下:

```
#Sage Math脚本
import time
def test():
    loopTime=1000
    NumList=[0x1f3cbb99,0x59a38fdedb,0x25220f2a4ab77]
    for i in range(3):
        a=time.time()
        for j in range(loopTime):
            factorList=divisors(NumList[i])
        b=time.time()
        for ele in factorList:
            print(hex(ele),end=",")
        print(f"耗时 {(b-a)/loopTime} ms")
test()
```

对给定数据进行因子分解, 得到结果如下表:

表 3: 素因子分解结果记录

大整数 (hex)	分解耗时 (ms)	素因子 (hex)
0x1f3cbb99	0.000419	[0x4e67,0x65ff]
0x59a38fdedb	0.000579	[0x94e11,0x9a22b]
0x25220f2a4ab77	0.001304	[0x14638cf,0x1d23cd9]

随后本文测试了不同长度整数因子分解的平均耗时. 长度由 32 到 127, 在每个长度下选取 500 个随机数并进行因子分解, 计算一次因子分解的平均耗时. 得到因子分解平均耗时与整数长度的大致变化关系.

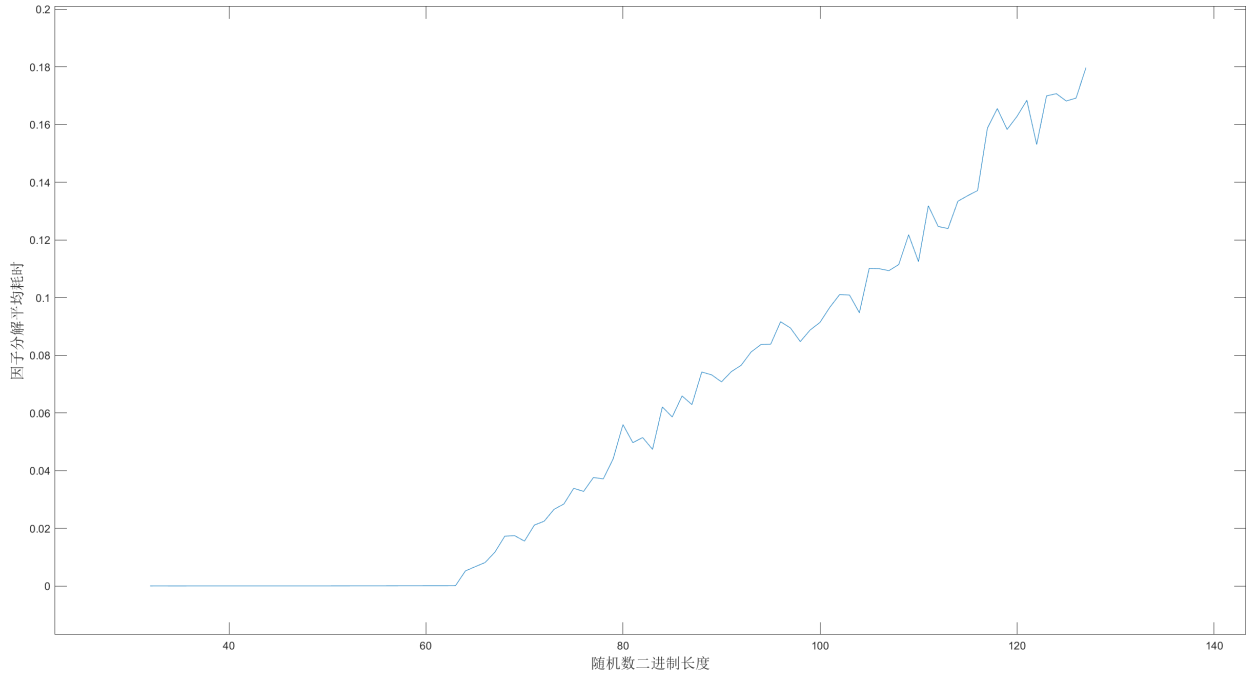

图 1: 不同 bit 长度整数因子分解耗时变化

参考文献

[1] Dougals R.Stinson. 密码学原理与实践: 第三版. 北京: 电子工业出版社,2009.7.