实验报告

【实验目的】

1、通过本次实验，熟悉编程环境，为后续实验做好铺垫。

2、回顾数论的基本算法，加深对其理解，为本学期密码学课程及实验课打好基础

【原理简介】

本次实验基于厄拉多塞筛法算法、欧几里得算法、快速幂取模算法、中国剩余定理、素性检测算法进行了实现，具体算法原理见下面详细介绍。

【实验环境】

VS2019 c++ / python3.6.8 ,内存8G的笔记本电脑；

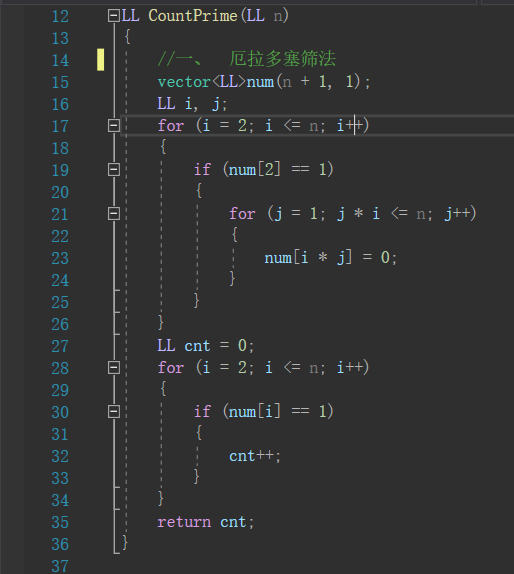
【实验内容】

# 厄拉多塞筛法

## 1、算法原理

## 厄拉多塞是一位古希腊数学家，他在寻找素数时，采用了一种与众不同的方法：先将2－N的各数放入表中，然后在2的上面画一个圆圈，然后划去2的其他倍数；第一个既未画圈又没有被划去的数是3，将它画圈，再划去3的其他倍数；现在既未画圈又没有被划去的第一个数 是5，将它画圈，并划去5的其他倍数……依次类推，一直到所有小于或等于N的各数都画了圈或划去为止。这时，表中画了圈的以及未划去的那些数正好就是小于 N的素数。

## 2、算法流程

本算法大致流程如下

针对10亿大数据的改进：

使用static关键字开辟两个数组，大小分别是a,n - a + 1类型使用bool类型来统计，如果当前数字x >= a, 则储存在b里面，否则a

## 3、测试样例及结果截图

## 1）2；

## 

## 2）103；

## 

## 3）10e4 ；

## 

## 4）10e6 ；

## 

## 5）4275117753

## 

4、总结

本算法的难点：

在于第五个测试样例需要开辟10亿大的数组，普通堆栈区能存放4M的数据，则开辟失败；解决方法：

可以使用bool类型，因为bool类型占一个字节，Int是4个字节，节省4倍，使用static关键字开辟两个数组，大小分别是a,n - a + 1类型使用bool类型来统计，如果当前数字x >= a, 则储存在b里面，否则a。

# 时间复杂度是O(log n)

# 空间复杂度是O(n)

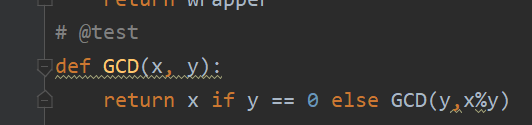
# 欧几里得算法

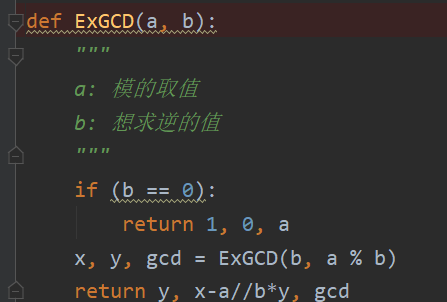
## 1、算法原理

## 辗转相除法， 又名欧几里德算法（Euclidean algorithm），是求最大公约数的一种方法。它的具体做法是：用较大数除以较小数，再用出现的余数（第一余数）去除除数，再用出现的余数（第二余数）去除第一余数，如此反复，直到最后余数是0为止。如果是求两个数的最大公约数，那么最后的除数就是这两个数的最大公约数。

2、算法流程

本算法大致流程如下……函数调用如下……





## 3、测试样例及结果截图

## 1）7,5；

## 

## 2）31,-13；

## 

## 3）24,36；

## 

## 4）

## 2461502723515673086658704256944912426065172925575，

## 1720876577542770214811199308823476528929542231719；

## 

## 5）

## 137096164691449488835122291235023051763859318102840889067550902

## 3843189897270890443917889846802171079840187598665712521108447262149

## 9595371254346390738382042，

## 192350399949876251675909634808997772559337752383120440971227732

## 5564753027680631763602672767980082537045932161772487151544214743242

## 0951257037823141069640181；

## 

## 6）

## 965578072786402991215194630452063779349788872980869942115905155

## 7171732595578592378315943243630787051274235487747679004689180215305

## 3719263845602618422474671707896136814707875793300040916757228826108

## 4994903112959425534780109130436805236126554005262552907029834903821

## 91419067057726624348815391509161304477322782，

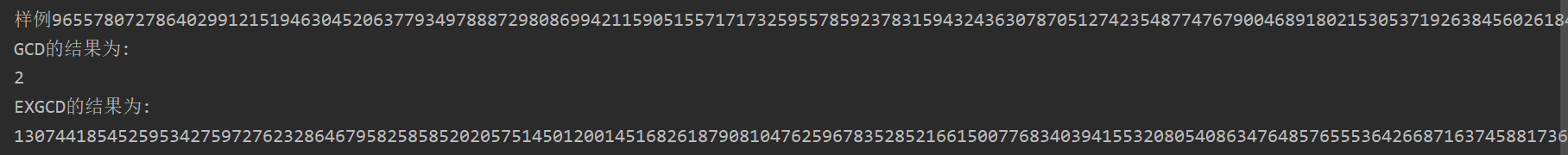
## 146116799305702219220540123503890666704710410600856387071776221

## 5924772567527599977981699318091564264712437997953740725104236453636

## 8053733781377426865890713096999414678345169283722277214494143490905

## 0652825715582967684984814095461041109999161468223272534833391335036

## 612863782740784573110824091866969655931097032；



4、总结

本算法难点：

主要是后面样例数字过大，c语言无法进行储存，可以使用str进行储存，然后模拟，但是难度较大

解决方法：

使用python语言，python可以很方便的进行大数运算。

时间复杂度：GCD（O(log n)）, EXGCD（log(max(a,b))）

# 快速幂取模

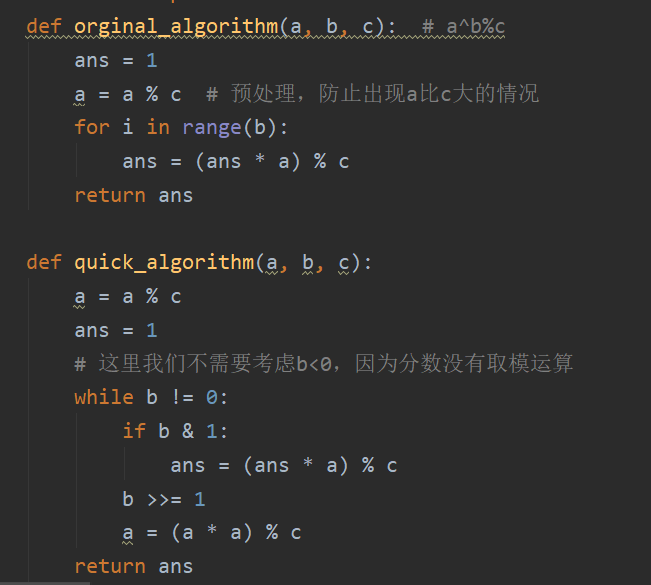
## 1、算法原理

 (a\*b)%c=(a%c)\*(b%c)%c

这个是成立的：也是我们实现快速幂的基础之后我们来看看快速幂的核心本质,将大数的幂运算拆解成了相对应的乘法运算，利用上面的式子，始终将我们的运算的数据量控制在c的范围以下，这样我们可以客服朴素的算法的缺点，我们将计算的数据量压缩了很大一部分，当指数非常大的时候这个优化是更加显著的.

2、算法流程

本算法大致流程如下……函数调用如下……



## 3、测试样例及结果截图

## 1）7,16,3;

## 

## 2）5,1003,31

## 

## 3）

## 1494462659429290047815067355171411187560751791530，

## 65537，

## 2268838711304724304304396119509416774597723292474；

## 

## 4）

## 224908128765398850463360530400433610227720622269057644143195314

## 1675262498296718145591252615303303022298577823031407083754914306802

## 1815197910334221004333099，

## 65537，

## 263810368062543912112558253300316259088954866354968201708113975

## 7611889270552615152613931291679885903024221918117851783792090402272

## 0459931859633170905729517；

## 

## 5）

## 237218075278892229535140238768762235405145645557640724744207466

## 3705448464576826633699763227989443924331042805955846358968212450487

## 3763728936189670330045479517548886172481332486745511912028461278587

## 1304351940501930714775024417724051440337510897547661217466354700893

## 011496892348407228806138461120064957907686566，

## 65537，

## 349972806688784936669965759420500287481274799328355633592840001

## 6613823405872472000557465228142759024303703309547256976487476100844

## 7791767622017920327336129109836828761283713597951090098204715426102

## 3406927515096043384562410643544643505195484211397819374480917731785

## 250826080723518532061522456937734714740424476；

## 

## 6）

## 448491664748214835887077572737743989471818983924746533195711112

## 7233709686801121455056758689058862976976518115359591533430192974458

## 1571878137080762225856531772968111014478142092370072319368083480854

## 9790079348059612669341617763791748262779560287414722951999863579554

## 8552263858419031745950115581439065662361137463558808154109351626536

## 15576832860612181499713446185302492149321184607038850277，

## 266848195381815818463717950266554236453862598799637312683425703

## 7333498609295735939964140208292798435740507497331418088263777339919

## 9243300029529038144493481715743519025117826811171308661166570875388

## 8640498769964215272509919856595016440707694201919276657450513307401

## 6424041697707456343064822862885626840633177568647767745489948120225

## 5227467773505313167510961788263581765823369837890930171124297082058

## 5209428037532351125028227556492657705501994644156977193457255573644

## 9875419903118346727670284395203781452229350828856233901927136651768

## 48108677291865357438200，

## 264496104806945826630879147982623492753075837057692083206159998

## 8796853354757804303239919347164946713039721233817228174089834448083

## 6053483980415141663259446884375373371451231004101622624801199838411

## 7072606363846922087540888426192580126275854056635599339955169813796

## 3133644631302014881764671798554905130111523852767747291485274278825

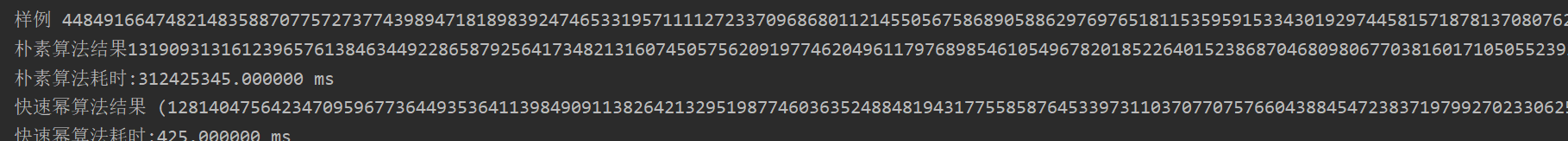
## 6890259402224899453419484216558327523122341749054612967901747155276

## 1001579139105477841364398884899527245085546136326414204870392428817

## 4323275616829270998998492543691126732288591795334806467302128382293

## 7158706678666372103627074163021260578078017304088904154859161289037

## 070912220207946945；



4、总结

本算法难点：

主要是后面样例数字过大，c语言无法进行储存，可以使用str进行储存，然后模拟，但是难度较大

解决方法：

使用python语言，python可以很方便的进行大数运算。

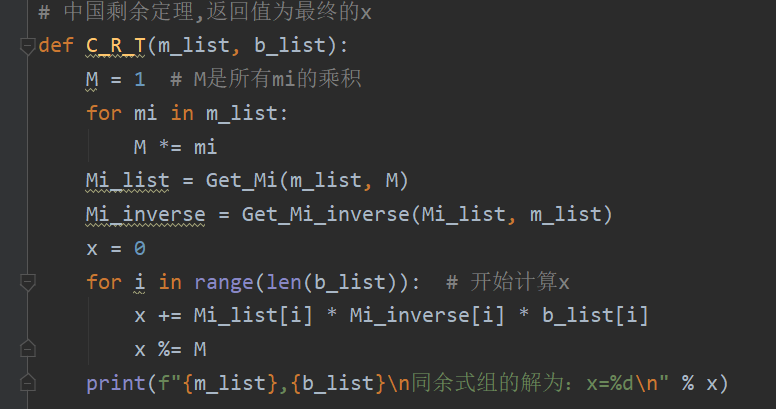
# 中国剩余定理

## 1、算法原理

## ①如果a%b=c，那么如果x%b=c/2，此时x=a/2；也就是说除数相等时，被除数和余数是成比例的。

## ②如果a%b=c，那么 (a + k\*b)%b=c，其中k为整数

2、算法流程

本算法大致流程如下……函数调用如下…… 

## 3、测试样例及结果截图

## 1）23,28,33; 0,0,0;

## 

## 2）23,28,33; 5,20,34;

## 

## 3）23,28,33; 283,102,23;

## 

## 4)

## 489808178709479466279507878773770708214878979673，

## 896234965496726578561614071442814700467907036641，

## 1213827005758305602466882992172310409456053868843；

## 802310684485241212312289432691586430708135062249，

## 961714109955647014172499578071923389425123540027，

## 1381194006087304024683552712488022595194097928701；

## 

## 5）

## 815796954028841163781843303955832318407477908610016550466853822

## 1920170369774913261335059602331627321130656458962980224196880533337

## 839226059601303464776145，

## 969961604431502119495357256107650299278313062321657422042604360

## 0142343504101508838526221359049417564415801914072315788919275792502

## 477693022853881785198116，

## 783269380225637166786651421311945219982191619366810690413581228

## 3217637737600922381702016472708855675649121271702977408217814917908

## 566132517503707494037556；

## 133923160816514208773088752761667728086018121220523714423390788

## 7774039956928167268382020619632095500586907200288384764652658410726

## 0355414977120453263391947，

## 973446693965828282334376020659328396876590484825002158021863438

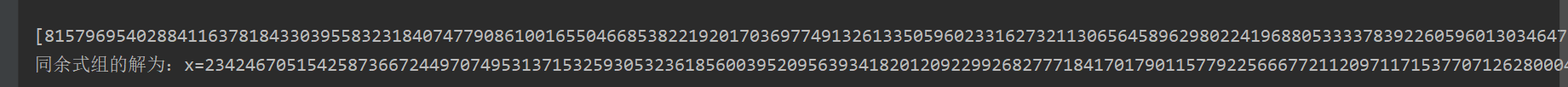
## 3869090913086348857668999272399075016287736914000854272239315769632

## 719896968098820774563511，

## 946020035779072839886291323266403603869452185841576593106450519

## 3755202156521446156499075450033429983317127589636591133111239548821

## 251790171694322930011927；



4、总结

中国剩余定理是数论中的一个定理。即如果一个人知道了一个数n被多个整数相除得到的余数，当这些除数两两互质的情况下，这个人就可以唯一的确定被这些个整数乘积除n所得的余数

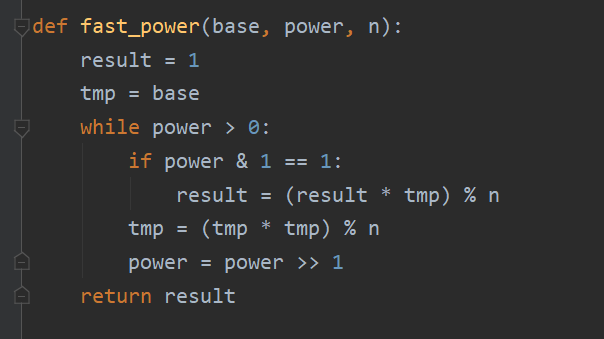
# 素性检测算法

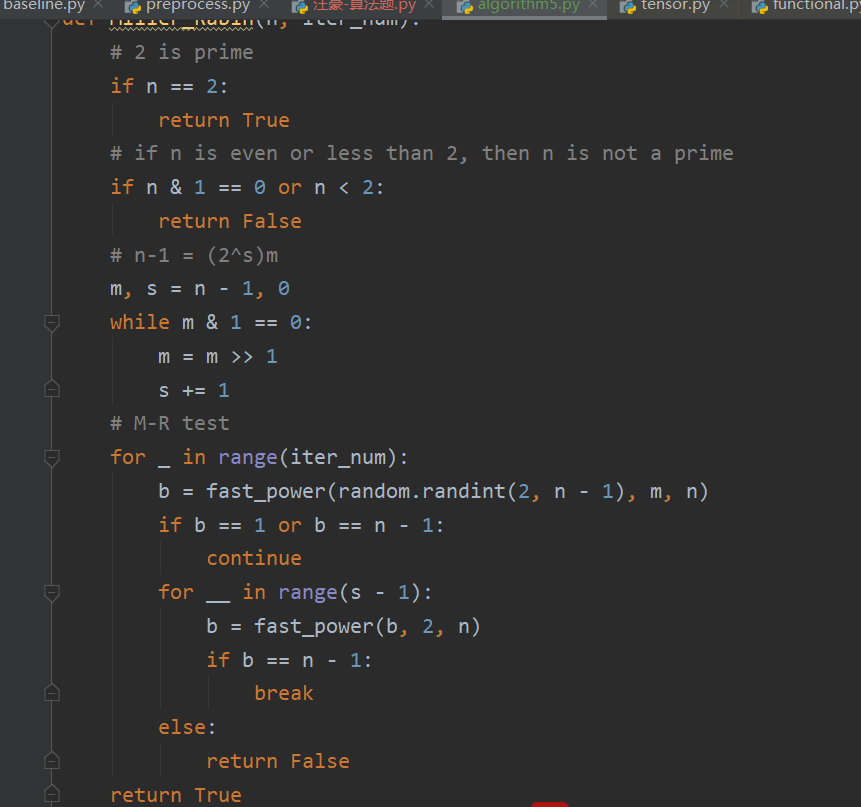
## 1、算法原理

## 由费马小定理可知,若p为质数,对于任意与p互质的整数a,有ap-1≡1(modp),假设我们要测试的数是x,然后在1到p-1内随机生成一个数作为底数a,然后测试它是否符合费马小定理,如果不符合则一定不是素数,符合则有可能是素数.单纯用费马小定理检验素数出错率很高(指满足费马小定理又不是素数的数-这些数称为Carmichael数,也称弱可能素数),因此需要一个更强的检定方法.

## 二次探测定理:p是质数,x2≡1(modp),x的解只能为1或p-1.利用这条定理就可以大大加强我们的素性检测.具体做法如下,对于被测数x,首先检验它是否能通过费马小定理,如果通过了,看p-1是否为偶数(只有p为2的情况不符合),如果是偶数这利用二次检测定理检测x(p-1)/2是否为1或p-1,如果是p-1的话就不能检测下去了,如果是1的话,则有x(p-1)/2≡1(modp),检查(p-1)/2是否为偶数,如果是偶数的话又能再次利用二次探测定理,检测x(p-1)/4的值是否为1或p-1.这样重复检测下去直到xn=p-1或n为奇数时才停止检测,在这期间任何一次检测不通过则可以断定x为合数.由于只用一个底数检测出错率还是不够低,因此一般是用多个底数进行判定,这样出错率才能降到令人接受的程度.由整个检测过程可知,我们刚开始生成的底数a的取值范围可以改进,想一下1n=1(modp),(p-1)n=1(modp)或(p-1)n=-1(modp)恒成立(p-1的用二项式拆开就证明).因此只要在2到p-2内随机生成一个数作为底数a即可.

2、算法流程

本算法大致流程如下……函数调用如下…… 



## 3、测试样例及结果截图

## 

4、总结

分析：

如果待测数是在1至1e15里面的话,用固定底数2,3,7,61,24251测速度会快很多,而且不会出错,这组底数能出错的第一个数是46856248255981.这是个RP算法,多次运行可以降低出错率,假设测的次数为n,出错率大约(1/4)^n

# 五、收获与建议

通过本次实验，对数论中的一些基本算法有了了解，通过实验，也发现了中国数学之美，巧妙的数学能够实现如此神奇的事情，在数据量比较大的时候，通过python语言能够更好的处理，也为后序学习密码学打下了一个基础。