

# 程序员讲数学

21工作室出品

# 书到用时方恨少

- 数学就像素数一样，是所有科学的理论表达基础，像三体中代际交替，只要环境合适就能积累出高段位的文明。
- 很多数学理论就像幻海遗珠，独自发展起来，但是在不被应用的时候就很难联系和成就一番事业。
- 也有很多理论，结果被传承了，但是发现过程被遗忘。人是懒惰的。但是这个发现过程有助于他人成长。我们虽然不需要了解所有失败和痛苦，但是能够让我们对理论领悟深刻的，永远是那些挫折，这也就是为什么记忆宫殿残暴却庞大的原因。

# 缘起

- 产生一个随机数可以用来抽奖。年会时烘托气氛、玩游戏时打怪掉落装备和碰到缘分的武器以及好看的皮肤等等。
- 手机验证码
- 数字化、信息化之后。抽签决定命运的时刻，远程会议决定运气的时刻，如何得到一个公认没有暗箱操作的随机数，是后续执行能力的基础。如果与会人员不服这个使命分配。那么将无法达成信息化决策。

# 研究已有流行语言

- java: 线性同余法生成随机数

$$\begin{cases} seed = (a * seed + c) \% m \\ return seed / float(m) \end{cases}$$

根据Hull-Dobell Theorem, 需要满足:

1.  $c$ 与 $m$ 互质
2.  $m$ 的所有质因数都能整除 $a-1$
3. 如果 $m$ 能被4整除, 那么要求 $a-1$ 也能被4整除

# 选取自己的参数做实验

- 实践是检验真理的唯一标准
- 很多知识不管来源是哪里，你获得之后需要考量的并不是权威与否，可信度可以节约你的时间，但是更重要的是复现结论。道理如果不能应用，那么讨论道理是否有用和道理真假就没什么意义了。
- 很多人仅仅处于第一阶段：哦，我找到了理论基础，立于不败之地。但是不知道自己如何用铠甲护住自己；嗯，我理解了，知道有这么回事，过程我不懂，但是结论我看懂了。这样的人也仅仅是会操作、会干活，但是绝不可能靠这样的人来创新。

# 印证设想是最兴奋的

- 兴趣是最好的老师。你想完成一件事情，你就需要去促成完成它的条件。去寻找完成它的步骤。去凝聚完成它的力量。当你完成一遍的时候，你就有信心完成很多遍。
- 人生能够完成很多遍的大道理的时候不多。有很多关键时刻掌握在别人手中。比如医生、命运。这些如果你在在选择之前能够有足够的积累，就可以不听别人的，就可以走自己的路。

# 实验一：

- 选取：

```
m = 985951 * 985951 * 985969
a = 985951 * 985969 + 1
c = 100
```

- 结果：

```
1.4199489644361577e-05 0.00012982389694259736 0.0003468732229234046
0.0029534936330660125 0.0037790924711422484 0.004706116227901056 0.
.010862608791933666 0.0123981820607879 0.01403518024832471 0.015773
```

- 分析：m数值过大，导致计算机整型无法正常运算。

```
>>> 13609640687331/(985951 * 985951 * 985969)
1.4199489644361577e-05
>>>
```

## 实验二：

- 选取：

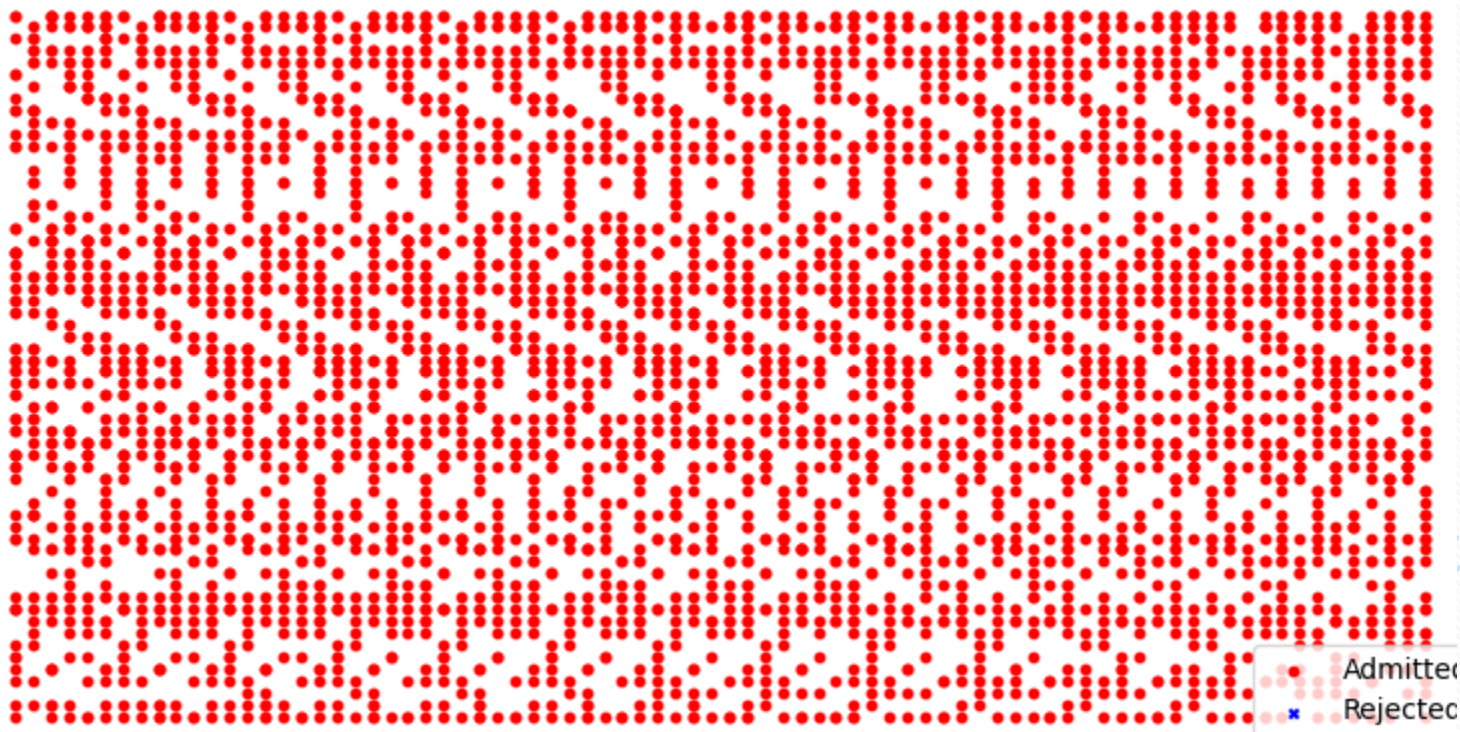
```
m = 199 * 199 * 49681
a = 199 * 49681 + 1
c = 12345
```

- 将m控制在0到 $2^{31}-1$ 范围内，c取5位数，打印4800个点分布到800\*600的画布上

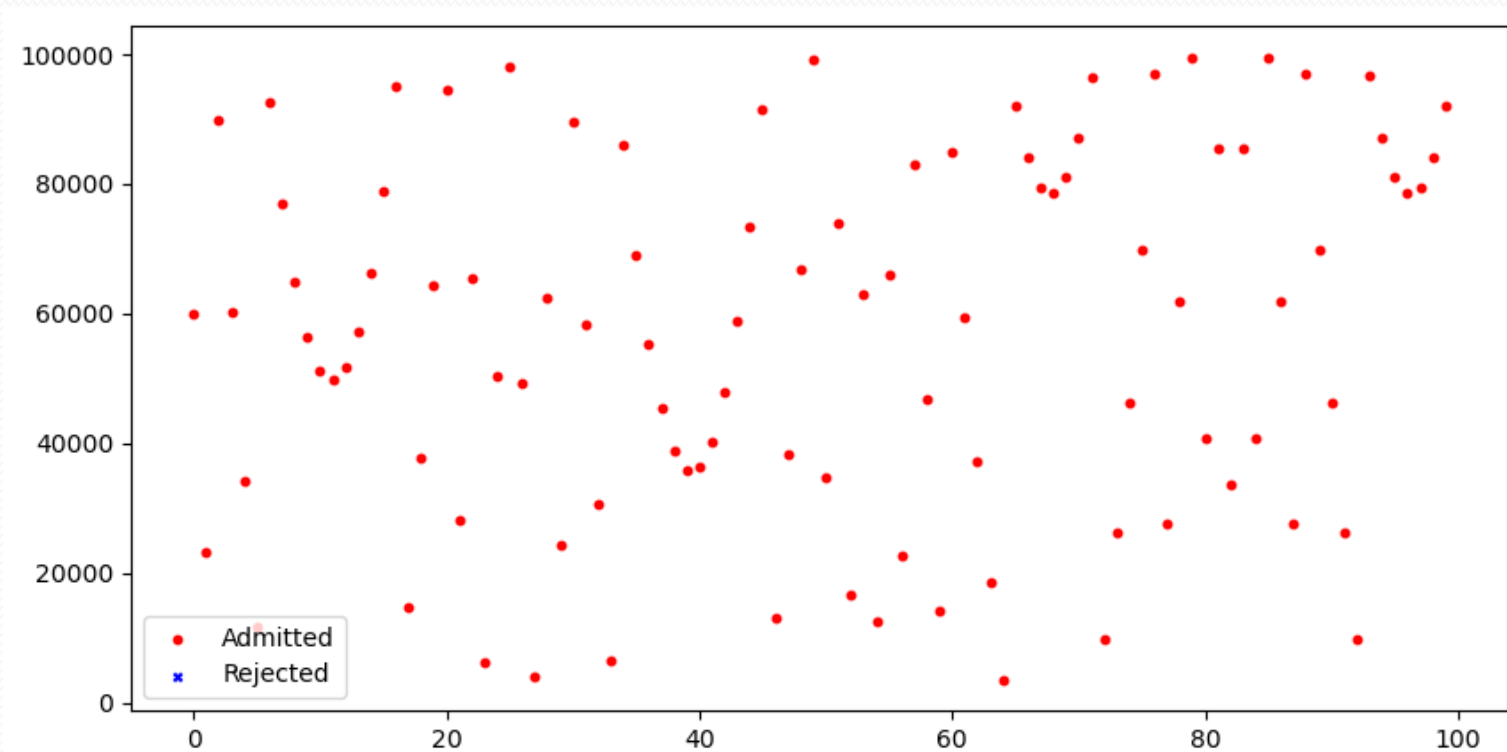


## 实验二：

- 可以看出虽然并不完全，但是循环明显



- 取100个数，查看分布
- 可以看出有类似低谷的V字出现



# 实验三：

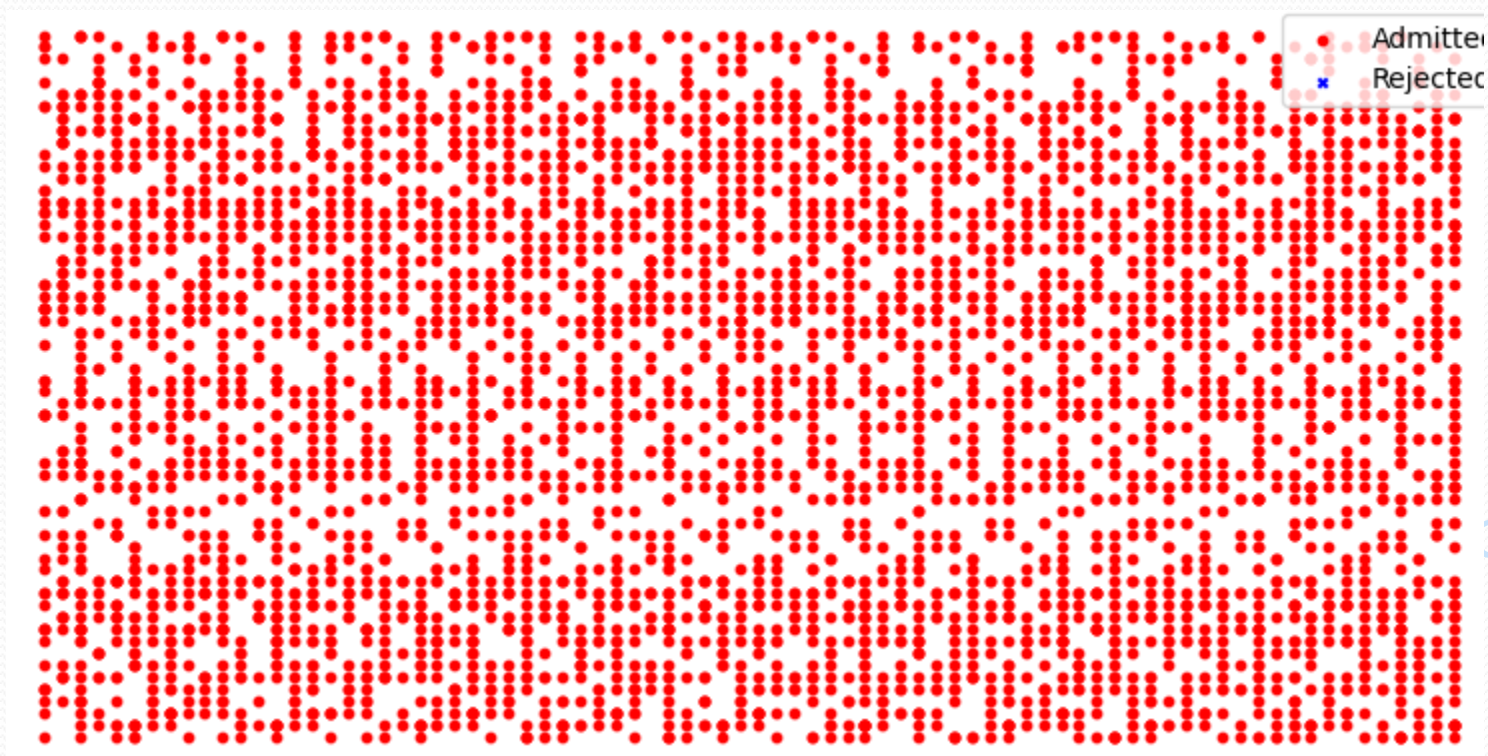
- 选取：

```
m = 199 * 199 * 49681 # 控制在40000*50000 < 2^31 = 21,474,836,480  
a = 199 * 49681 + 1  
c = 17017 # 7*11*13*17 避开800*600画布的质因数
```

- 避开画布尺寸的质因数

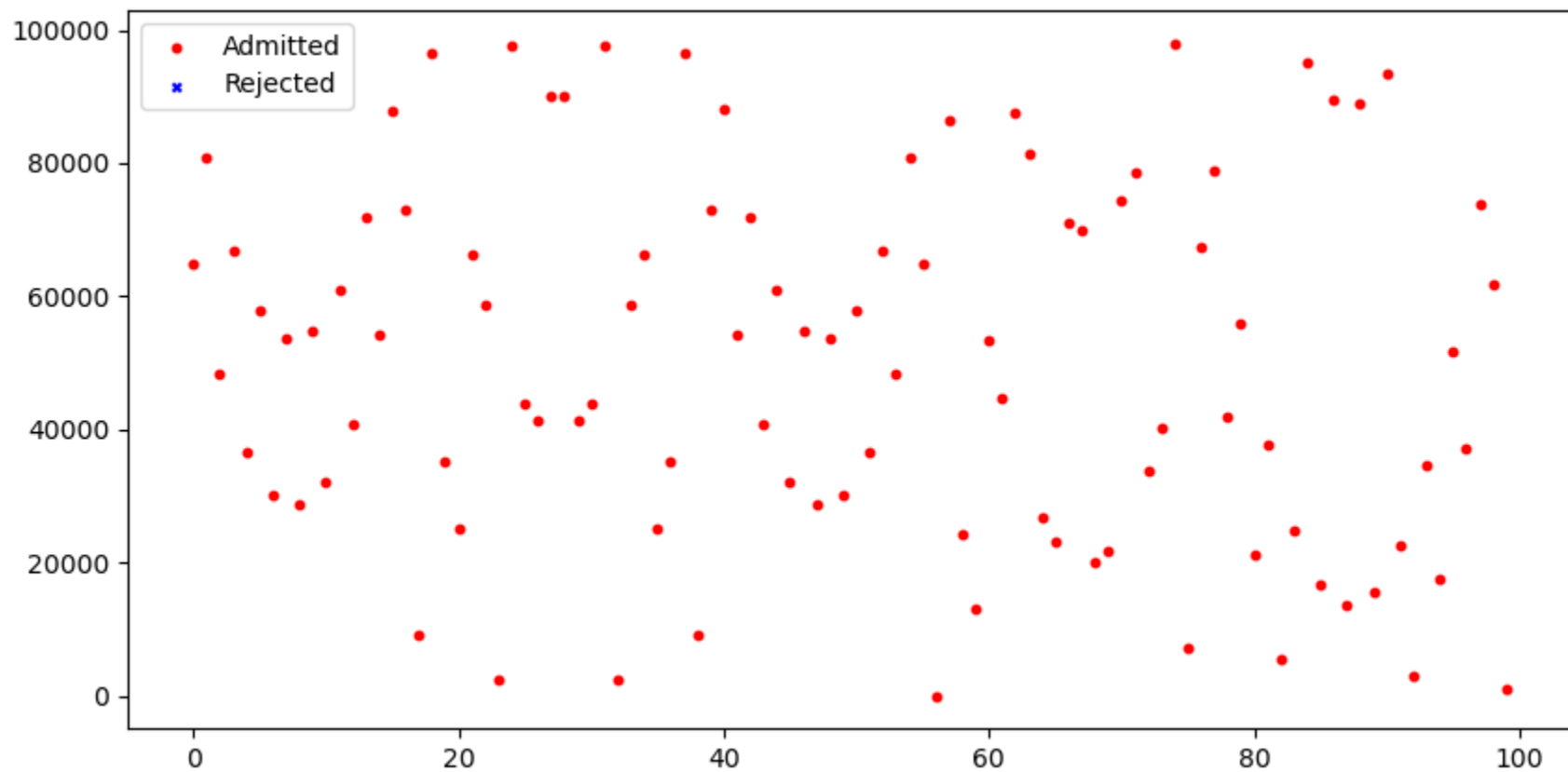
# 实验三：

- 可以看出虽然并不完全，但是循环明显



cn

- 可以看出有类似低谷的V字出现



# 实验四：

- 选用 $2^{31}-1$ 这个大素数的原根16807来做乘同余法

```
m = 2 ** 31 - 1  
a = 16807  
c = 0
```

- 很凑巧，计算机整型能装下的最大的数竟然是个素数
- 这个选取就很科学了，参考了文献，学习了欧拉函数、欧拉定理、原根、群、剩余系，知道了同余方程中一次（也就是线性）下的性质，不光有加同余、乘同余还有混合同余、除同余。而网上大多标题写的线性同余生成随机数并没有明确说明，实际上使用的是混合同余法。
- 也知道了原根和平方剩余这两个概念是数论的两颗明星

- 乘法是快速的加法，而自乘是升维的基石。
- 本来以为加法拥有最普遍的交换律、结合律、反身性、对称性、传递性
- 向上一维，乘法多了分配律，延伸到矩阵已经很了不起，延伸到多项式、延伸到微积分求导法则等等
- 再向上一维，幂乘又变成了加法
- 再向上N维，都可参照幂乘
- 之后便拓展了数域，实数域到复数域，以后还有四元数、八元数

# 实验五：

- 选取：

```
m = 100003
```

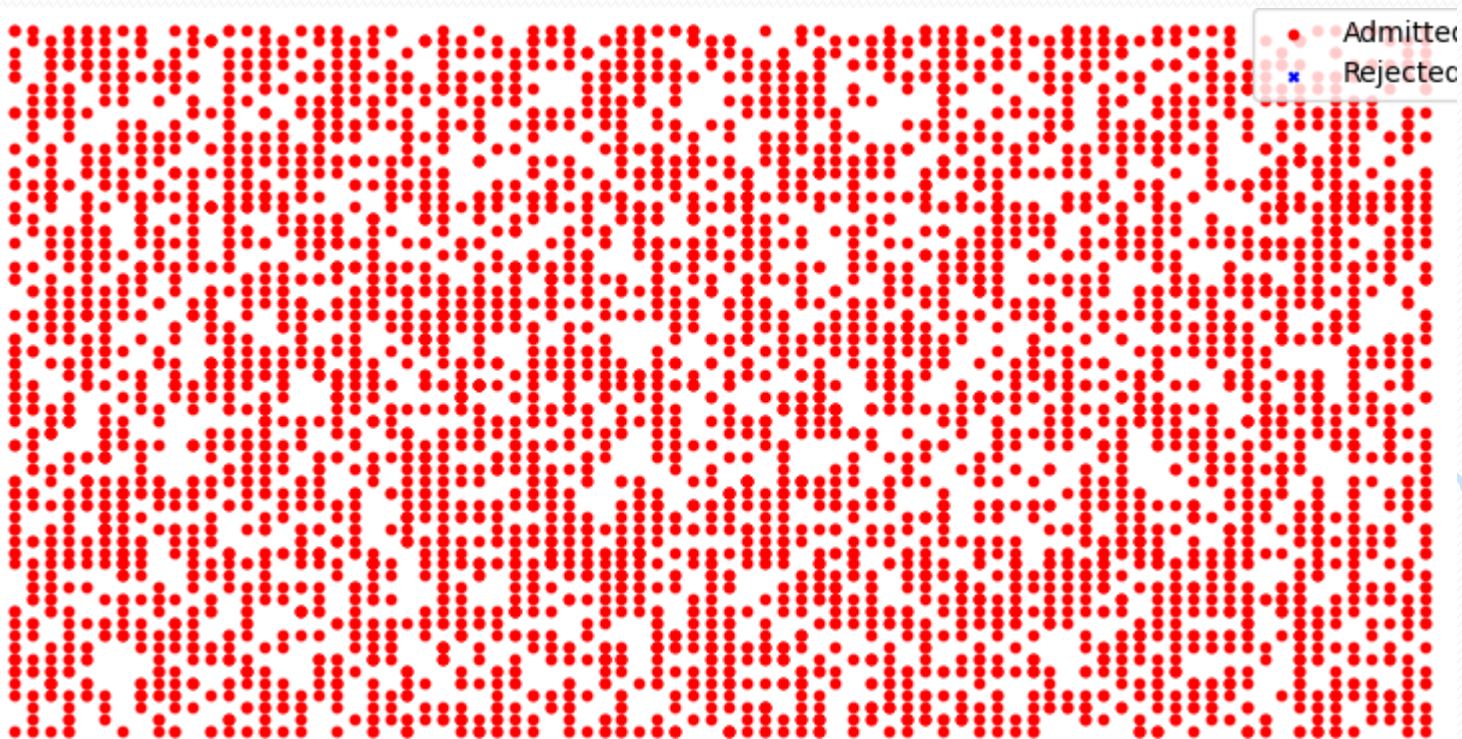
```
a = 16807
```

```
c = 0
```

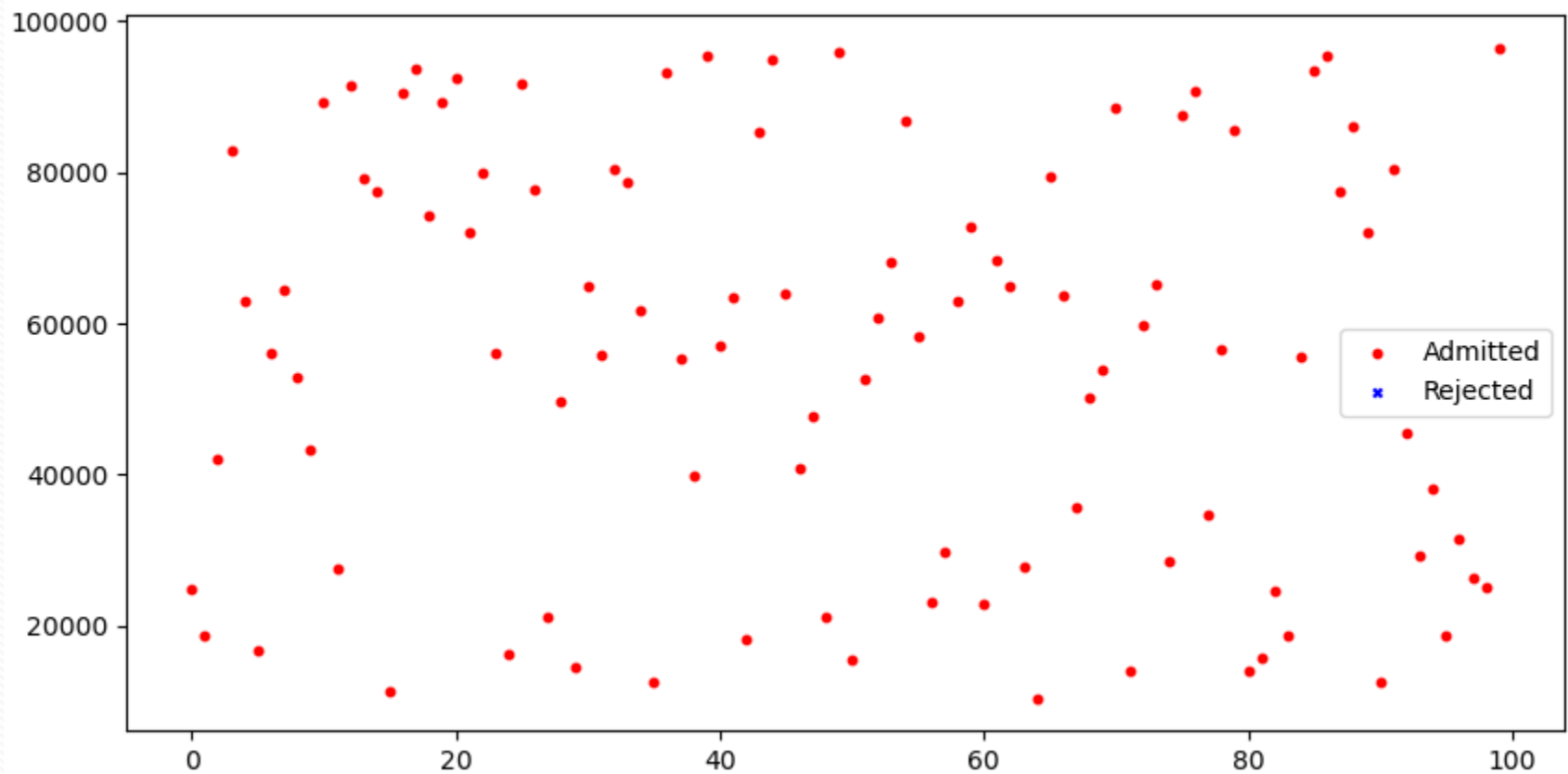


# 实验五：

- 肉眼已找不到规律



- 可以看出和系统产生的随机分布有的一拼



# 乘同余产生随机数原理

## 原根

阶：设  $a, m \in \mathbb{N}^+$ , 且  $a \perp m$ , 使  $a^x \equiv 1 \pmod{m}$  成立的最小正整数  $x$ , 称为  $a$  模  $m$  的阶, 记为  $\text{ord}_m a$

原根：设  $g, m \in \mathbb{N}^+$ , 且  $g \perp m$ ; 若  $\text{ord}_m g = \varphi(m)$ , 则称  $g$  是模  $m$  的原根。

定义原根的意义在于, 原根非常类似于单位根, 故有一些很好的性质, 比如说我们考虑当  $m$  是质数时, 原根有如下等价定义: 若  $g$  是模  $m$  的原根, 则  $g^1, g^2, \dots, g^{m-1}$  的值在模  $m$  的意义下两两不同, 注意到模  $m$  剩余系中恰有  $m$  个数, 而任意一个非零数的幂次都不可能等于 0, 故原根的幂次在模  $m$  的剩余系中是最稠密的, 换句话说, 仅用原根就能够充分表示模  $m$  剩余系的性质。

# 整数模 $n$ 乘法群

- 在同余理论中，模  $n$  的互质同余类组成一个乘法群，称为整数模  $n$  乘法群，也称为模  $n$  既约剩余类。在环理论中，一个抽象代数的分支，也称这个群为整数模  $n$  的环的单位群（单位是指乘法可逆元）。

# 原根的枚举找法

- 阶
  - 对于 $(a,n)=1$ 的整数，满足 $a^r \equiv 1 \pmod{n}$ 的最小整数 $r$ ,称为 $a$ 模 $n$ 的阶。
- 判断一个数 $a$ 是不是 $n$ 的原根，需要判断 $\varphi(n)$ 是否是 $a$ 的阶

- 这和加同余可以遍历模内数据的道理相同。但是不同的是，加同余肉眼可见的规律性。在乘同余中不容易显现
- 伪随机数之所以伪是因为其规律性可预知，周而复始循环往复。但是在人生苦短的这一刹那，周期大到你无力承受，种子选取、轮询的韵律已经对你短暂到失去了意义，那么这个随机就已经可以改变你的一生。

- 验证以上结论，需要工具PyCharm、Clion。网上有很多python源代码，需要画图numpy、matplotlib.pyplot支持库，需要扩展欧几里得算法解同余方程、需要欧拉函数、需要暴力算法求原根、需要筛法求素数、需要混合同余法、需要分解因数等等
- 其中最难的还是求原根，python就远不如C++效率高了，最后还好电脑上有C语言的环境。
- 另外大多数公式截图来源于Atom+latex截图。

# 线性同余方程求解

小圆滚滚

## 1 问题描述

解方程  $13x \equiv 6(\text{mod}34)$  (1)

对方程求解的步骤为先求特解，再乘以系数得到通解：

步骤1:  $13x + 34y = 1$ , 得到x的所有取值使得等式右端的值为1。

步骤2: 对上述方程的左右两端同时乘以6，然后系数13和34不变，用新的x代替原始的x，即得到方程的通解。



## 2 贝祖定理求特解

如果 $a$ 、 $b$ 是整数，那么一定存在整数 $x$ 、 $y$ 使得 $ax+by=\gcd(a,b)$

Greatest Common Divisor(GCD)

换句话说，如果 $ax+by=m$ 有解，那么 $m$ 一定是 $\gcd(a,b)$ 的若干倍。

拓展欧几里得就是求关于 $x,y$ 的方程  $ax + by = \gcd(a,b)$  的所有整数解。

我们知道，欧几里得公式可以由这个式子表示：

$$\gcd(a, b) = \gcd(b, a \% b)$$

不断往下连等，直到 $b=0$ ，此时 $a$ 即为最大公约数。

那么我们来讨论另一个问题，下面两个式子有没有关系呢？

$$a * x_1 + b * y_1 = g(a, b)$$

$$b * x_2 + (a \% b) * y_2 = g(b, a \% b) \quad (\text{让 } a, b = b, a \% b)$$

只要找出 $x_1$ 和 $x_2$ 的关系、 $y_1$ 和 $y_2$ 的关系，我们就能求出方程 $a * x + b * y = g$ 的一个特解。

回到刚才那个问题，很显然，两个等式右边就是欧几里得的公式

那么我们就得出 $a * x_1 + b * y_1 = b * x_2 + (a \% b) * y_2$

其中 $a \% b$ 可以换成 $a - (a // b) * b$  (地板除)

式子变成了

$$a * x_1 + b * y_1 = b * x_2 + (a - (a // b) * b) * y_2$$

因为我们要找 $x_1$ 、 $y_1$ 和 $x_2$ 、 $y_2$ 的关系，

我们可以用待定系数法，按照这种方法把右边化成 $b * (x_2 - (a//b) * y_2) + a * y_2$   
则等式变成了

$$a * x_1 + b * y_1 = a * y_2 + b * (x_2 - (a//b) * y_2)$$

现在得出了下面两个等式：

$$x_1 = y_2 \quad (\text{等式两边} a \text{的系数相同})$$

$$y_1 = x_2 - (a//b) * y_2 \quad (\text{等式两边} b \text{的系数相同})$$

也就是说，我们知道了方程 $b*x_2+(a\%b)*y_2 = g(b, a\%b)$ 的解 $x_2, y_2$ ，就可以得到方程 $a*x_1+b*y_1 = g(a, b)$ 的解 $x_1, y_1$ 了，这个小问题告一段落。

对于方程 $a*x+b*y = gcd(a, b)$ ，我们可以不断的往下变成 $b*x+(a\%b)y = gcd(a, b)$ ，按照欧几里得的过程，b会变成0，即此时方程 $a*x+b*y = gcd(a, b)$ 因为 $b=0$ ，变成了 $a*x = gcd(a, b)$ ，还是由欧几里得算法得到，此时a就等于 $gcd(a, b)$ ，所以得到由原方程往下推了不知道多少次的方程 $a*x = gcd(a, b)$ 来说，得到一个解 $x=1, y=0$ 。现在得到了这个方程的解，再回溯回去，就可以得出原方程 $a*x+b*y = gcd(a, b)$ 的一个特解。

### 3 具体实现

特解求取:

$$13x+34y=1$$

$$a=13,b=34$$

$\because a < b \therefore$  倒推的最后一步为:  $a = 34, b = 13$

$$x_1 = y_2 \quad (\text{等式两边}a\text{的系数相同})$$

$$y_1 = x_2 - (a//b) * y_2 \quad (\text{等式两边}b\text{的系数相同})$$

$\begin{cases} x_1 = -13 \\ y_1 = 5 \end{cases}$	$\begin{cases} x_1 = 5 \\ y_1 = -3 - 34//13 * 5 = -13 \end{cases}$	$\begin{cases} x_1 = -3 \\ y_1 = 2 - 13//8 * (-3) = 5 \end{cases}$
$(13,34) \Rightarrow$	$(34,13) \Rightarrow$	$(13,8) \Rightarrow$

$\begin{cases} x_1 = 2 \\ y_1 = -1 - 8//5 * 2 = -3 \end{cases}$	$\begin{cases} x_1 = -1 \\ y_1 = 1 - 5//3 * (-1) = 2 \end{cases}$	$\begin{cases} x_1 = 1 \\ y_1 = 0 - 3//2 * 1 = -1 \end{cases}$
$(8,5) \Rightarrow$	$(5,3) \Rightarrow$	$(3,2) \Rightarrow$

$\begin{cases} x_1 = y_2 = 0 \\ y_1 = x_2 - (a//b) * y_2 = 1 - 2//1 * 0 = 1 \end{cases}$	$\begin{cases} x_1 = 1 \\ y_1 = 0 \end{cases}$
$(2,1) \Rightarrow$	$(1,0)$

列表中，第二行向右推出1，0，然后第一行开始由最后向前反推。

通解

对方程 $13x+34y=1$ 的左右两端除以34取余可得：

$$13x \equiv 1(\text{mod}34)$$

即 $x=-13$ 是上述方程的一个特解，对上述方程的左右两端同时乘以6可得

$13 * (6x) \equiv 6(\text{mod}34)$ 的一个特解为-78，则 $13x \equiv 6(\text{mod} 34)$ 通解为：

$x=-78+34k$ ,其中 $k$ 是任意整数；

即通解为：

$x=-10+34k$ ,其中 $k$ 是任意整数

## 4 青蛙相遇问题：

两只青蛙在网上相识了，它们聊得很开心，于是觉得很有必要见一面。它们很高兴地发现它们住在同一条纬度线上，于是它们约定各自朝西跳，直到碰面为止。可是它们出发之前忘记了一件很重要的事情，既没有问清楚对方的特征，也没有约定见面的具体位置。不过青蛙们都是很乐观的，它们觉得只要一直朝着某个方向跳下去，总能碰到对方的。但是除非这两只青蛙在同一时间跳到同一点上，不然是永远都不可能碰面的。为了帮助这两只乐观的青蛙，你被要求写一个程序来判断这两只青蛙是否能够碰面，会在什么时候碰面。我们把这两只青蛙分别叫做青蛙A和青蛙B，并且规定纬度线上东经0度处为原点，由东往西为正方向，单位长度1米，这样我们就得到了一条首尾相接的数轴。设青蛙A的出发点坐标是x，青蛙B的出发点坐标是y。青蛙A一次能跳m米，青蛙B一次能跳n米，两只青蛙跳一次所花费的时间相同。纬度线总长L米。现在要你求出它们跳了几次以后才会碰面。

$$x + mt - (y + nt) = pL (p \in Z) \quad x, y \text{ 代表了两个不同的起点, } m, n \text{ 代表了不同的次数}$$

$$\text{即 } (n - m)t + Lp = x - y (L > 0)$$

即求一次（线性）**同余方程**  $At \equiv B \pmod{L}$  的最小正整数解，其中  $A=(n-m)$ ,  $B=x-y$ , A是系数，B是余数。

如同  $At + nL = B \quad n \in Z$  转为了二元一次**不定方程**的问题

1. 判断有根：二元一次不定方程 $ax + by = c$ 有解，当且 $\gcd(a,b)|c$  (翻译：a、b两数的最大公约数能整除c，请注意除和除以是有区别的。被除数除以除数等于商。除数除被除数等于商)。裴蜀定理

2. 求 $ax+by=\gcd(a,b)$ 的解, X,Y

$$Ax \equiv B(\text{mod}L) \quad (1)$$

若  $d = \gcd(A, B)$ ,  $d$  整除  $B$ , 那么  $B/d$  为整数。由裴蜀定理, 存在整数对  $(t,n)$  (可用辗转相除法求得) 使得  $At+nL=d$ ,  $\frac{AtB}{d} + \frac{nBL}{d} = 1 \times B$  因此  $x_0 = tB/d$  是方程 (1) 的一个解。其他的解都关于  $L/d$  与  $x_0$  同余。即  $x = x_0 + (B/d) \times r \pmod{L} (0 \leq r \leq d - 1)$ 。

3. 扩展欧几里得算法解得原方程提取公因数之后, 化简到等式 $ax + by = c$ , 当 $c=1$ 时有一组特解:

(这里很有意思, 是我先知道结果肯定是有, 然后再考虑如何去求取, 逆向思维) 
$$\begin{cases} X^* = \frac{X}{\frac{c}{\gcd(a,b)}} \\ Y^* = \frac{Y}{\frac{c}{\gcd(a,b)}} \end{cases}$$

4. 则其所有解为: 
$$\begin{cases} x = X^* \times \frac{c}{\gcd(a,b)} + \frac{b \cdot t}{\gcd(a,b)} \\ y = Y^* \times \frac{c}{\gcd(a,b)} - \frac{a \cdot t}{\gcd(a,b)} \end{cases} \quad t \in Z$$

例:

## 5 线性同余算法生成随机数

混合同余法:

4

$$\begin{cases} seed = (a * seed + c) \% m \\ return seed / float(m) \end{cases}$$

根据Hull-Dobell Theorem, 需要满足:

1.  $c$ 与 $m$ 互质
2.  $m$ 的所有质因数都能整除 $a-1$
3. 如果 $m$ 能被4整除, 那么要求 $a-1$ 也能被4整除



## 7 枚举法求原根，并加快

怎么找原根？

设 $m$ 是大于1的正整数，如果 $(a,m)=1$ ，则使同余式

$$a \equiv 1 \pmod{m}$$

成立的最小正整数 $d$ 称为 $a$ 对模 $m$ 的指数（或阶），记作 $ord_m(a)$ 。

如果 $a$ 对模 $m$ 的指数是 $\phi(m)$ ，则 $a$ 称为模 $m$ 的一个原根。

欧拉定理： $a$ 与 $m$ 互质时， $a^{\phi(m)} \equiv 1 \pmod{m}$ ，其中 $\phi(m)$ 称为对模 $m$ 缩系的元素个数。

判断一个数 $a$ 是不是 $m$ 的原根，需要判断 $\phi(m)$ 是否是 $a$ 的阶。

比如我们要判断 $a$ 模 $n=37$ 的阶是否为36，那么只需找出36的两个质因子2和3，判断 $36/2$ 和 $36/3$ 作为 $a$ 的幂的指数时， $(a^{\text{幂}}) \pmod{n}$ 是否为1。如果 $a$ 的阶为 $36/4$ ， $36/9$ ， $36/6$ ， $36/12$ ， $\dots$ ，其实情况已

5模18的阶是 $\phi(18) = 6$ , 1,5,7,11,13,17, 正是与18互质的那6个数啊,

$5^6 \equiv 1 \pmod{18}$ , 所以5是18的原根.18的欧拉函数值是6.

判断是否有原根:

若m有原根, 则m一定是下列形式: $2, 4, p^a, 2p^a$ , 这里p为奇素数, a为正整数.

欧拉函数通式:

$$\phi(m) = m \prod_{\substack{p \mid m \\ p \text{ 为质数}}} \left(1 - \frac{1}{p}\right) = m(1 - 1/p_1)(1 - 1/p_2)(1 - 1/p_3)(1 - 1/p_4) \cdots (1 - 1/p_n)$$

,其中 $p_1, p_2, \dots, p_n$ 为x的所有质因数, m是不为0的整数.  $\phi(1)=1$  (唯一和1互质的数(小于等于1)就是1本身)。(注意: 每种质因数只一个。比如 $12=2*2*3$ 那么  $\phi(12) = 12 * (1-1/2) * (1-1/3) = 4$

比如说求81的所有原根:

1.先算81的欧拉函数, 结果为54, 又54的素因数有2和3, 54除以这两个素因数得到18和27

2. 从2, 4, 5开始验算 $2^{18} \not\equiv 1 \pmod{81}, 2^{27} \not\equiv 1 \pmod{81}$ , 所以2是81的原根 (只要找到一个由素因数得出来的次数mod81不等于1的就可以停止了)

3.原根的个数就是就81算两次欧拉函数, 得到18, 那就不用18个原根, 54的简化剩余系的各个数字作为第二步找到的原根的次数就行

54的简化剩余系 (其实就是与54互素的) 为1,5,7,11,13,17,19,23,25,29,31,35,37,41,43,47,49,53

那么81的所有原根为  $\{2^1, 2^5, 2^7, 2^{11}, \dots, 2^{53}\}$ , 不要忘了, 里面的值还要mod81的哦

即  $\{2^1 \pmod{81}, 2^5 \pmod{81}, 2^7 \pmod{81}, 2^{11} \pmod{81}, \dots, 2^{53} \pmod{81}\}$

## 参考：

- HDU 4992 Primitive Roots（求出 $n$ 的所有原根）