

法律声明

□ 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，小象学院和主讲老师拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意及内容，我们保留一切通过法律手段追究违反者的权利。

□ 课程详情请咨询

■ 微信公众号：小象

■ 新浪微博：ChinaHadoop



概率组合



小象学院
ChinaHadoop.cn

邹博

主要内容

- ☐ 本福特定律
- ☐ 古典概型与几何概型
- ☐ 身边的概率：麻将
- ☐ 约会问题
- ☐ 男女比例问题
- ☐ 数据流随机采样方案
- ☐ Buffon's Needle
- ☐ 虚拟机可靠性分析
- ☐ 砵码问题
- ☐ Gale-Shapley 算法
- ☐ 猜数字游戏
- ☐ 2/3 的博弈
- ☐ 概率化商品推荐
- ☐ 圆内均匀取点/拒绝采样
- ☐ 带权推荐
- ☐ 金钗赠诗问题

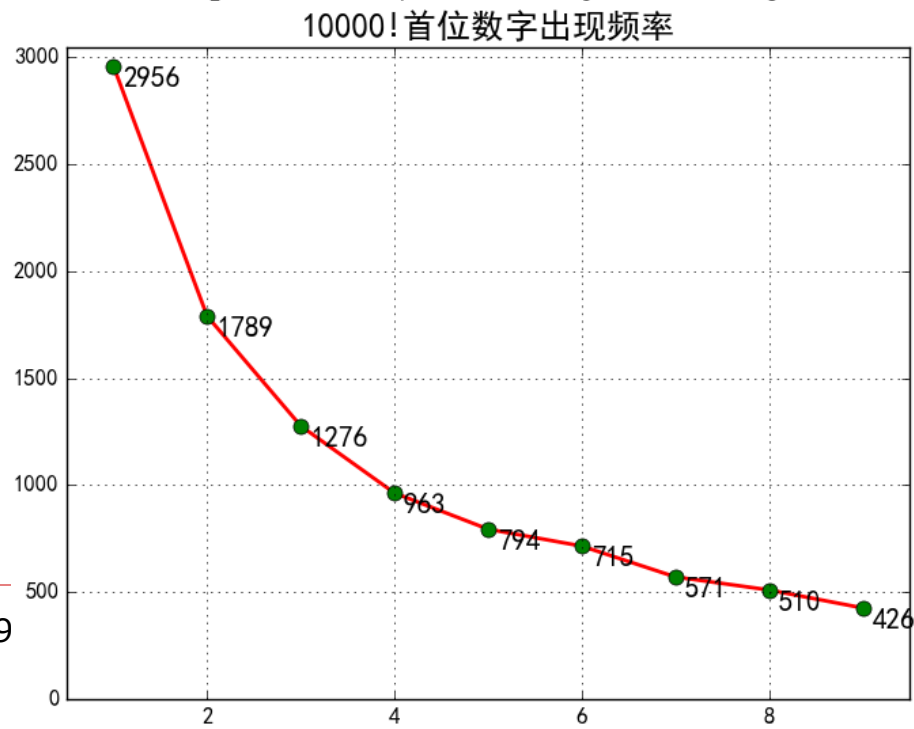
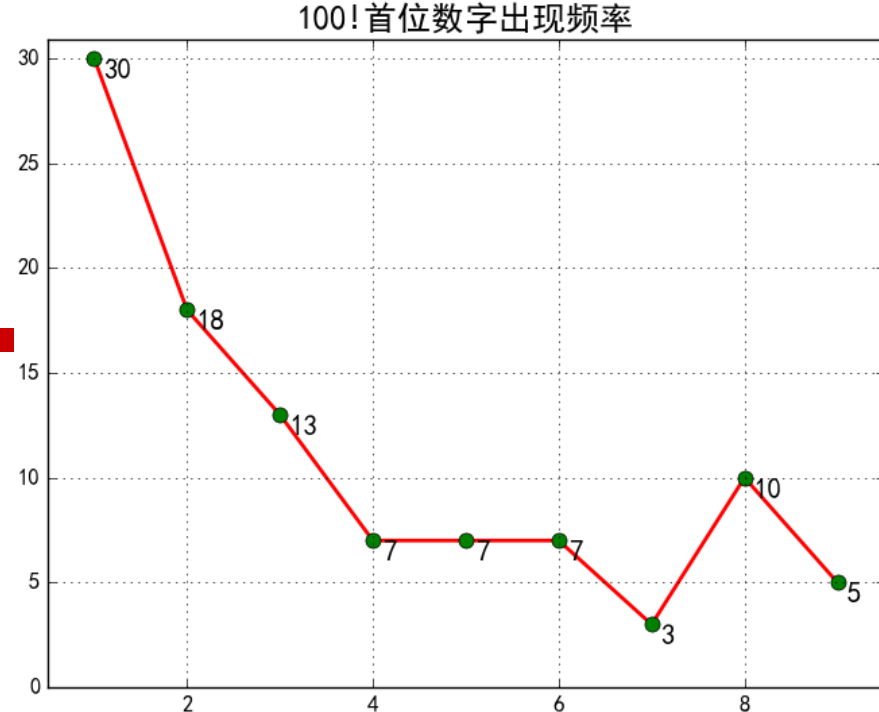
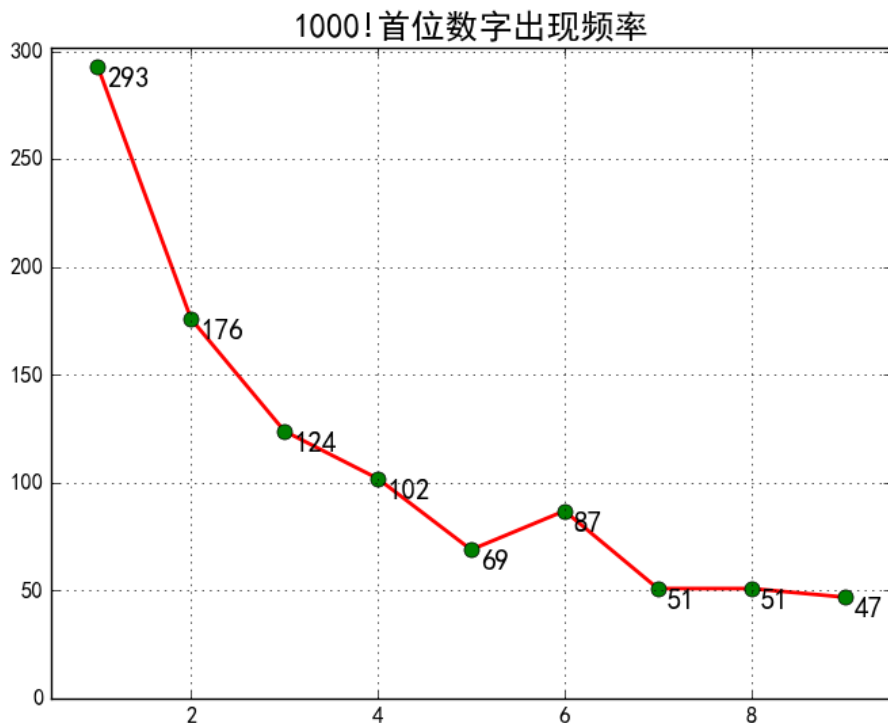
统计数字的概率

- 给定某正整数 N ，统计从1到 $N!$ 的所有数中，首位数字出现1的概率。
- 进而，可以计算首位数字是2的概率，是3的概率，从而得到一条“**九点分布**”。

Code

```
def first_digital(x):  
    while x >= 10:  
        x /= 10  
    return x  
  
if __name__ == "__main__":  
    n = 1  
    frequency = [0] * 9  
    for i in range(1, 1000):  
        n *= i  
        m = first_digital(n) - 1  
        frequency[m] += 1  
    print frequency  
    plt.plot(frequency, 'r-', linewidth=2)  
    plt.plot(frequency, 'go', markersize=8)  
    plt.grid(True)  
    plt.show()
```

数字的概率



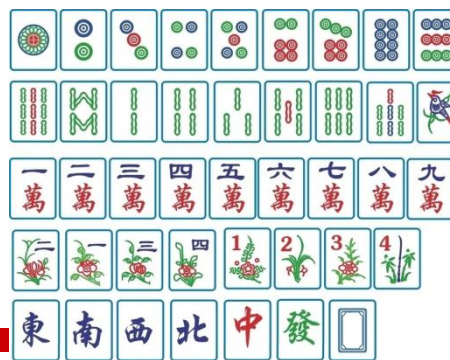
本福特定律

□ 本福特定律(本福德法则, Frank Benford), 又称第一数字定律, 是指在实际生活得出的一组数据中, 以1为首位数字出现的概率**约为总数的三成**; 是直观想象 $1/9$ 的三倍。

- 阶乘/素数数列/斐波那契数列首位
- 住宅地址号码
- 经济数据反欺诈
- 选举投票反欺诈

数字	出现概率
1	30.1%
2	17.6%
3	12.5%
4	9.7%
5	7.9%
6	6.7%
7	5.8%
8	5.1%
9	4.6%

身边的概率



- “国粹” 麻将是集技巧与运气的在我国开展广泛的娱乐项目。去除花牌后的标准麻将，由1到9的“万、条、饼”各4张，以及“东南西北中发白”各4张，共计136张组成。我们把两张内容一样的牌叫一幅“将”。
- 请问，庄家起手摸14张牌，则他起手没有“将”的概率是多少？
 - 此外，可以算下摸13张牌没有“将”的概率，摸13张牌没有“风”的概率。

问题分析——古典概型

□ 基本事件数目：

■ 一共136张牌，随意选14张，取法为： C_{136}^{14}

□ 有效事件数目：

■ 一共34组牌，选择某个组，然后在该组的4张中任选1张，取法为： $C_{34}^{14} \cdot 4^{14}$

□ “无将”的概率为：

$$p = \frac{C_{34}^{14} \cdot 4^{14}}{C_{136}^{14}} = \frac{[21 \sim 34] \cdot 4^{14}}{[123 \sim 136]} \approx 8.79\%$$

抽取纸币

- 现有一叠纸币，其中5元面值的纸币6张，10元面值的纸币5张，20元面值的纸币4张，从袋子中任意取4张纸币，则每种面值至少取到一张的概率是多少？

问题分析

- 现有一叠纸币，其中5元面值的纸币6张，10元面值的纸币5张，20元面值的纸币4张，从袋子中任意取4张纸币，则每种面值至少取到一张的概率是多少？
- 问题分析：基本事件总数为从15张纸币中取4张的所有取法，有效事件为满足题意的取法。

问题分析

□ 题干：6张/5张/4张 - 任取4张纸币

□ 基本事件总数： C_{15}^4

□ 有效事件分三种情况：从5元、10元和20元面值中选其中一种面值取2张，其他两种面值各取1张。总事件数目为：

$$C_6^2 \times 5 \times 4 + C_5^2 \times 6 \times 4 + C_4^2 \times 6 \times 5$$

□ 上面两式相除即为概率，化简后得到48/91。

计算概率

- A、B两国元首相约在首都机场晚20点至24点交换一份重要文件。如果A国的飞机先到，A会等待1个小时；如果B国的飞机先到了，B会等待2个小时。假设两架飞机在20点至24点降落机场的概率是均匀分布，试计算能够在20点至24点完成交换的概率。
- 假设交换文件本身不需要时间。

事件的形式化表达

- 假定A到达的时刻为 x ，B达到的时刻为 y ，则完成交接需满足 $0 < y - x < 1$ 或者 $0 < x - y < 2$ ；
- 同时要求 $20 < x < 24$ ， $20 < y < 24$ ；
- 由于 x ， y 系数都为1，为作图方便，可以将 $20 < x < 24$ ， $20 < y < 24$ 平移成 $0 < x < 4$ ， $0 < y < 4$ 。

计算面积

□ 三角形面积

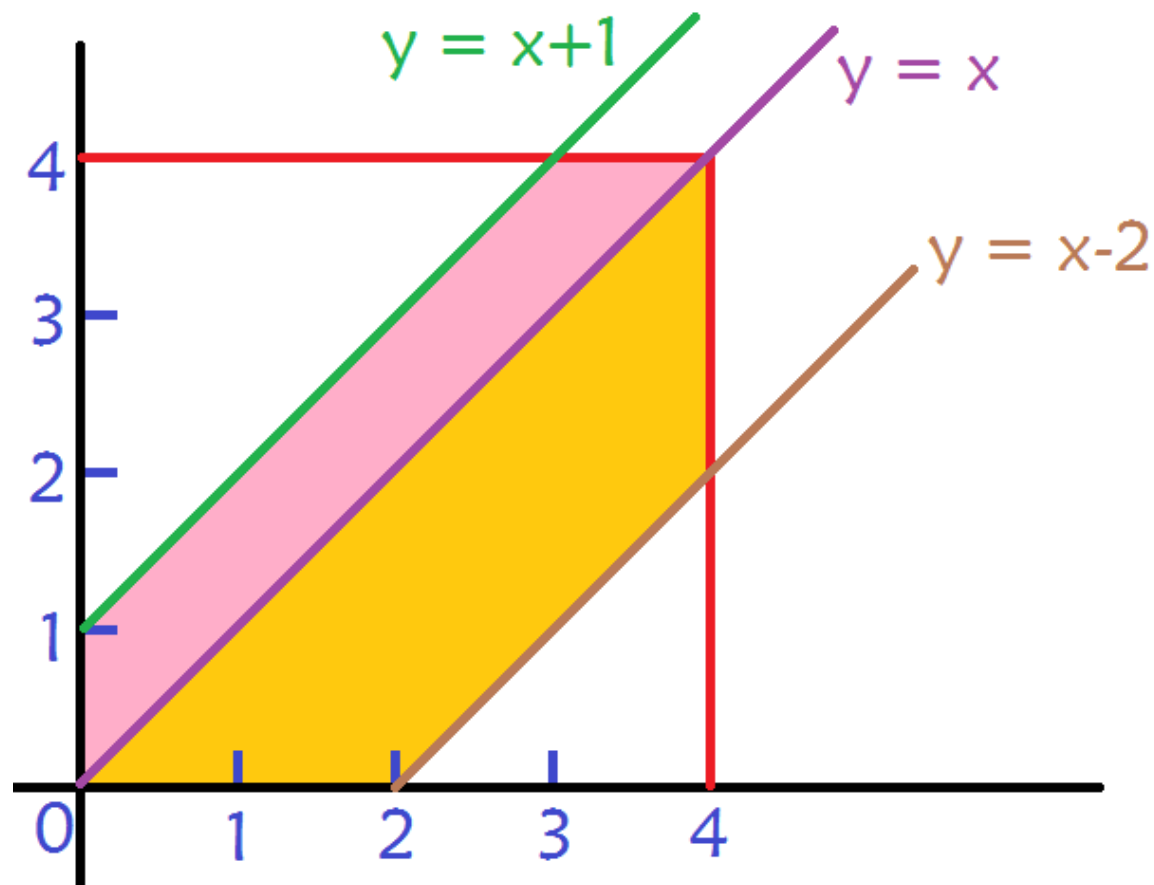
■ $9/2$ 、 2

□ 矩形面积

■ 16

□ 概率

■ $19/32$



男女比例问题

- 某国人民普遍重男轻女。大家遵守这样的习俗：一对夫妻若一胎没有生育儿子，则继续生孩子，**直到生育儿子为止**。假定该国当前男女比例是1:1，且子代中的所有男女都可婚配。试估算若干代后该国男女的比例

子代中男女孩数目公式

- 先验性的认为生男生女的自然概率近似相同，都是 $\frac{1}{2}$ ；由于生育儿子后则不再生育，所以，每个家庭有且只有一个儿子。假定家庭数目为1，则 $S(\text{男})=1$ 。
- 有 $\frac{1}{2}$ 的家庭一胎生男孩就不再生育；另外 $\frac{1}{2}$ 的家庭中，有 $\frac{1}{4}$ 的家庭二胎生男孩则不再生育，有 $\frac{1}{8}$ 的家庭三胎生男孩则不再生育……，从而，所有家庭的女孩总数为：

$$S(\text{女}) = \sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i (i-1)$$

子代中女孩的数目

□ 根据 $S(\text{女}) = \sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i (i-1)$

□ 即：
$$S(\text{女}) = \frac{1}{2} \cdot 0 + \frac{1}{4} \cdot 1 + \frac{1}{8} \cdot 2 + \frac{1}{16} \cdot 3 + \cdots + \left(\frac{1}{2}\right)^i (i-1) + \left(\frac{1}{2}\right)^{i+1} \cdot i + \cdots$$

□ 两边同时乘以

$$\frac{1}{2} S(\text{女}) = \frac{1}{4} \cdot 0 + \frac{1}{8} \cdot 1 + \frac{1}{16} \cdot 2 + \frac{1}{32} \cdot 3 + \cdots + \left(\frac{1}{2}\right)^{i+1} (i-1) + \left(\frac{1}{2}\right)^{i+2} \cdot i + \cdots$$

□ 错位相减，得：

$$\begin{aligned} \frac{1}{2} S(\text{女}) &= \frac{1}{2} \cdot 0 + \frac{1}{4} \cdot 1 + \frac{1}{8} \cdot 1 + \frac{1}{16} \cdot 1 + \cdots + \left(\frac{1}{2}\right)^{i+1} + \left(\frac{1}{2}\right)^{i+2} + \cdots \\ &\Rightarrow S(\text{女}) = 1 \end{aligned}$$

男女比例问题结论

- 由此得到，子代中男女数目仍然是1:1，即：维持性别比例的稳定。
- **思考**：如果原题增加一个条件：“在某胎生男孩后，**允许再生一胎**，无论性别男女都不再生育”，结论会是如何呢？
 - **答**：仍然1:1

男女比例问题总结

□ 原问题即标准的几何分布

■ 抛硬币，直到得到正面为止的次数。

□ 因为自然选择的概率近似1:1，所以，考察任何一次生育下的男女比例都是以1:1做概率选择，因此，只要服从自然选择，无论采取何种生育策略，人口比例都相同。

□ 人口增长率：原题目中，子代和亲代的人口总数不变；若按照修改版的规则，子代数目的是亲代的1.5倍，即自然增长率为50%。

■ 注：修改版家庭的孩子数目 $S_{\text{孩}} = \sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i \cdot (i+1) = 3$

数据流随机采样方案

- 随机选词：给定相当长的单词流(该数据流无法一次获取，且大于100)，试给出等概率的选择其中100个的方法。
 - 即：N个数据流中等概率选择k个。

算法思路

- 开辟长度为 k 的缓冲区 $a[0\dots k-1]$;
- 将前 k 个元素放置于 $a[0\dots k-1]$ 中;
- 对于第 $i(i > k)$ 个元素
 - 取随机数 $r \in [0, i-1]$, (如: $r = \text{rand}() \% i$)
 - 若 $r < k$, 则替换 $a[r] = a[i]$

算法的合理性

□ 第 i ($i \leq k$)个元素被选中的概率：

$$1 \times \frac{k}{k+1} \times \frac{k+1}{k+2} \times \cdots \times \frac{N-1}{N} = \frac{k}{N}$$

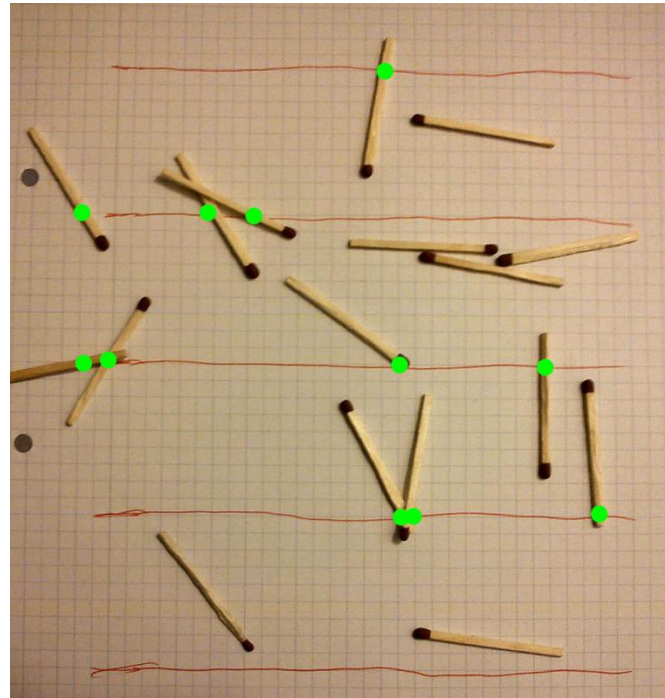
□ 第 i ($i > k$)个元素被选中的概率：

$$\frac{k}{i} \times \frac{i}{i+1} \times \frac{i+1}{i+2} \times \cdots \times \frac{N-1}{N} = \frac{k}{N}$$

Buffon's Needle

□ 桌面上有距离为 a 的若干平行线，将长度为 L 的针随机丢在桌面上，则这根针与平行线相交的概率是多少？

■ 假定 $L < a$



概率计算

- 记投针中点到最近横线的距离为 y ，则 $y \in [0, a/2]$ ，投针是随机的， y 为均匀分布：

$$f(y) = \begin{cases} \frac{2}{a} & 0 \leq y \leq \frac{a}{2} \\ 0 & \text{elsewhere} \end{cases}$$

- 假定横线向右为正向，记投针与横线正向的角为 θ ，则 $\theta \in [0, \pi]$ 为均匀分布。

$$f(\theta) = \begin{cases} \frac{1}{\pi} & 0 \leq \theta \leq \pi \\ 0 & \text{elsewhere} \end{cases}$$

- 投针与横线有交点，即： $y \leq \frac{L}{2} \sin \theta$

蒙特卡洛模拟

□ 有交点的概率：

$$\begin{aligned} P(X) &= \int_0^\pi \int_0^{\frac{L}{2}\sin\theta} f(y, \theta) dy d\theta = \int_0^\pi \int_0^{\frac{L}{2}\sin\theta} f(y) f(\theta) dy d\theta \\ &= \int_0^\pi \int_0^{\frac{L}{2}\sin\theta} \frac{2}{a} \cdot \frac{1}{\pi} dy d\theta = \frac{2L}{a\pi} \end{aligned}$$

□ 如果做n次试验，得到k次相交。则频率是 $\frac{k}{n}$

□ 从而：
$$\frac{2L}{a\pi} \approx \frac{k}{n} \Rightarrow \pi \approx \frac{2Ln}{ak}$$

Buffon's Needle

□ 公式: $\frac{2L}{a\pi} \approx \frac{k}{n} \Rightarrow \pi \approx \frac{2Ln}{ak}$

试验者	时间	投掷次数	相交次数	π 的试验值
Wolf	1850年	5000	2532	3.1596
Smith	1855年	3204	1218.5	3.1554
C.De Morgan	1860年	600	382.5	3.137
Fox	1884年	1030	489	3.1595
Lazzerini	1901年	3408	1808	3.141593
Reina	1925年	2520	859	3.1795

Code

```
double Buffon(double a, double L)    //横线之间的距离, 针长度
{
    double y;        //到最近的横线的距离
    double theta;    //针的倾角
    int c = 0;        //相交次数
    int n = 1000000;  //实验次数
    for(int i = 0; i < n; i++)
    {
        y = Rand(a/2);
        theta = Rand(PI);
        if(y < L*sin(theta)/2) //相交
            c++;
    }
    return 2 * (double) (n * L) / (double) (c * a);
}
```

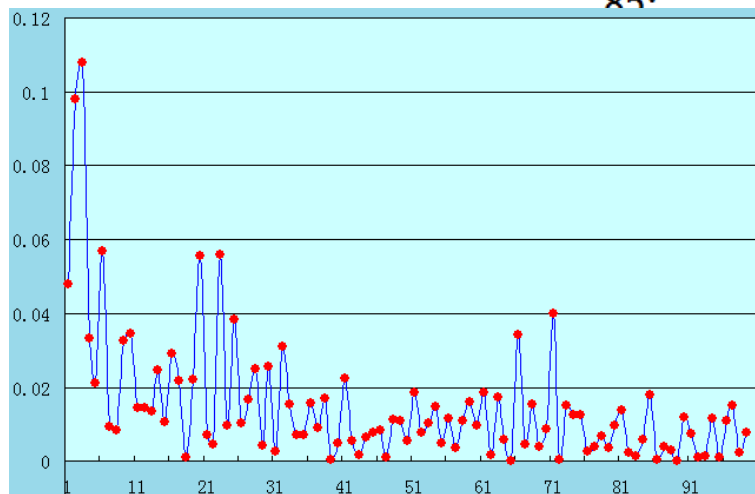
Code2

```
int _tmain(int argc, _TCHAR* argv[])
{
    double a = 100;      //横线的间隔
    double L;            //针的长度
    double pi;           //估算值
    double avg = 0;      //估算值的均值
    double count = 0;    //计算次数
    for(L = a; L > 1; L -= 1)
    {
        pi = Buffon(a, L);
        cout << pi << '\n';
        avg += pi;
        count++;
    }
    avg /= count;
    cout << avg << '\n';
    return 0;
}
```

```
double Buffon(double a, double L) //横线的间隔、针的长度
{
    double X = a * 1000; //取足够大的信纸
    double Y = a * 1000;
    int N = 100000;      //进行10万次投针试验
    int c = 0;
    double x1, x2, y1, y2;
    double d, y;
    for(int i = 0; i < N; i++)
    {
        x1 = Rand(X);
        y1 = Rand(Y);
        x2 = Rand(X);
        y2 = Rand(Y);
        d = sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
        y = (y2 - y1) * L / d + y1;
        if((int)(y1/a) != (int)(y/a))
            c++;
    }
    return 2 * L * N / (a * c);
}
```

效果/措施

100:	3.13386	0.0077297	27:	3.13134	0.0102502
99:	3.14405	0.00246222	26:	3.18004	0.0384465
98:	3.12665	0.0149476	25:	3.13205	0.00954555
97:	3.13065	0.0109446	24:	3.19744	0.0558494
96:	3.14054	0.00105353	23:	3.13693	0.00465867
95:	3.12994	0.0116506	22:	3.1487	0.00711208
94:	3.14308	0.00148422	21:	3.19708	0.0554843
93:	3.14274	0.00114863	20:	3.11964	0.0219545
92:	3.13426	0.0073304	19:	3.14257	0.000980949
91:	3.12973	0.0118637	18:	3.16344	0.021852
90:	3.14164	4.27422e-005	17:	3.17075	0.0291618
89:	3.14471	0.00311699	16:	3.13112	0.0104772
88:	3.14538	0.00379203	15:	3.11688	0.0247095
87:	3.14113	0.000458235	14:	3.15493	0.0133369
86:	3.12352	0.0180682	13:	3.15611	0.0145132
85:	3.14739	0.00579779	12:	3.12704	0.0145568
	3.14019	0.00140574	11:	3.10691	0.0346868
	3.13936	0.00223149	10:	3.10897	0.0326233
	3.12774	0.0138511	9:	3.13316	0.00843338
	3.13201	0.0095843	8:	3.15085	0.00925414
	3.13799	0.0035994	7:	3.19854	0.0569452
	3.1348	0.00679642	6:	3.16289	0.0212961
	3.14567	0.004076	5:	3.10849	0.0331065
	3.13895	0.00264318	4:	3.03375	0.107842
	3.12892	0.0126686	3:	3.23974	0.0981482
	3.15424	0.0126498	2:	3.09358	0.0480118
74:	3.15666	0.0150673	AVG:	3.14202	0.000430937



虚拟机可靠性问题

- 假定所有物理计算机出故障的概率都是 p ，每台物理机可以虚拟出若干台虚拟机；假设只有虚拟机所在的物理机发生故障，虚拟机才宕机。
 - 方案1：直接使用100台物理机。
 - 方案2：使用5个物理机，每个物理机虚拟出20个虚拟机
- 请问：哪种方案的可靠性高？

问题分析

- 单位时间内，一台物理机发生故障的概率服从**两点分布** $X \sim B(1, p)$ 。n台物理机发生故障的概率服从**二项分布** $F \sim B(n, p)$ 。单位时间内发生故障的物理机数目的期望和方差为：

$$\begin{cases} E(F_1) = np \\ Var(F_1) = npq \end{cases}$$

- 下面重点分析虚拟机方案：

- 一台虚拟机发生故障的概率记做**Y**，Y与该虚拟机所在的物理机发生故障的概率相同： $Y = X$ ，且 **$Y \sim B(1, p)$** 。

虚拟机方案的期望和方差

□ n台虚拟机，其中每m台位于相同的一台物理机上

□ 单位时间内故障机器数目的期望：

$$E[Y_1 + Y_2 + \cdots + Y_n] = E(Y_1) + E(Y_2) + \cdots + E(Y_n) = np$$

□ 单位时间内故障机器数目的方差：

$$\begin{aligned} & \text{Var}[Y_1 + Y_2 + \cdots + Y_n] \\ &= \text{Var}[(Y_1 + Y_2 + \cdots + Y_m) + (Y_{m+1} + Y_{m+2} + \cdots + Y_{2m}) + \cdots + (Y_{n-m+1} + Y_{n-m+2} + \cdots + Y_n)] \\ &= \text{Var}(Y_1 + Y_2 + \cdots + Y_m) + \text{Var}(Y_{m+1} + Y_{m+2} + \cdots + Y_{2m}) + \cdots + \text{Var}(Y_{n-m+1} + Y_{n-m+2} + \cdots + Y_n) \\ &= \frac{n}{m} \text{Var}(Y_1 + Y_2 + \cdots + Y_m) = \frac{n}{m} \text{Var}(X_1 + X_1 + \cdots + X_1) = \frac{n}{m} \text{Var}(mX_1) \\ &= \frac{n}{m} \cdot m^2 \text{Var}(X_1) = nmpq \end{aligned}$$

可靠性结论

$$\begin{cases} E(F_1) = np \\ E(F_2) = np \end{cases} \quad \begin{cases} \text{Var}(F_1) = npq \\ \text{Var}(F_2) = mnpq \end{cases}$$

- 可以看出，单位时间内发生宕机数目的期望相同
 - 仅从期望角度，可靠性相同。
- 但单位时间内宕机数目的方差，物理机方案是 npq ，而虚拟机方案是 $mnpq$ 。
 - 每个物理机上运行的虚拟机越多，方差越大。只有1台物理机只运行1台虚拟机，二者方差才相等。
 - 通俗的说：虚拟机要么大多数都能正常运行，要么大多数总是出问题。
 - 从方差意义上，物理机方案的可靠性更好。

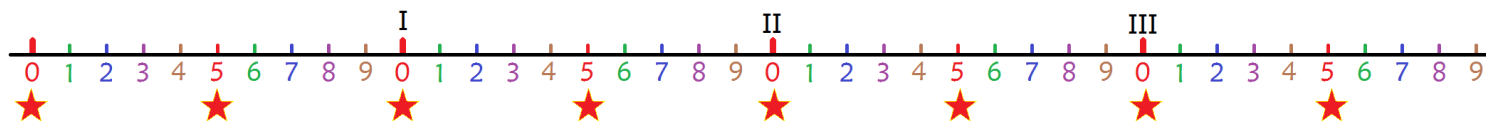
5-7砝码问题

□ 现有足够多的5克和7克的砝码，问：任何大于多少克的重量都能够用5克和7克的砝码组合出？

■ 注：不是用天平称量而是用砝码组合；即：只能将若干砝码重量相加，而不能相减。

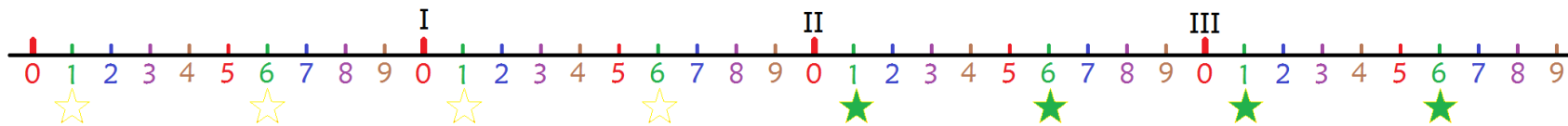
根据5的余数将整数分类

- 选择5和7的小者5来讨论(因为余数少, 思考步骤略少, 用7分类也没问题)
 - 用X表示能够表示的重量, 则根据被5除的余数, 分为 $5n$, $5n+1$, $5n+2$, $5n+3$, $5n+4$ ($n \geq 0$) 共5种情况:
 - 首先: 若 $X=5*n$, 可以用n个5克砝码来表示
-
- 下面分别讨论余数为1、2、3、4情况



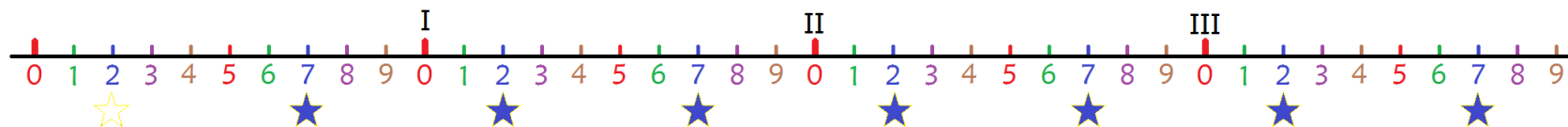
$$X=5n+1$$

- 必须将若干个5克砝码换成7克砝码，使得7克砝码的总重量比5克砝码的总重量大1，
- 即： $7a-5b=1$ ， $(a,b \geq 0, b \leq n)$
- 穷举得到 $a=3, b=4$ 是满足上式最小的——将4个5克换成3个7克，得到比原来重1克的重量
- 这显然需要在 $X=5n+1$ 中， n 至少存在4个5克砝码。即： $X=5n+1 \geq 5*4+1=21$ 。



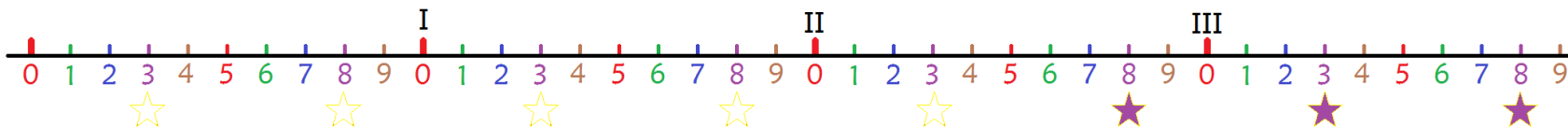
$$X=5n+2$$

- 必须将若干个5克砝码换成7克砝码，使得7克砝码的总重量比5克砝码的总重量大2，
- 即： $7a-5b=2$ ， $(a,b \geq 0, b \leq n)$
- 穷举得到 $a=1, b=1$ 是满足上式最小的——将1个5克换成1个7克，得到比原来重2克的重量
- 这显然需要在 $X=5n+2$ 中， n 至少存在1个5克砝码。即： $X=5n+2 \geq 5*1+2=7$ 。



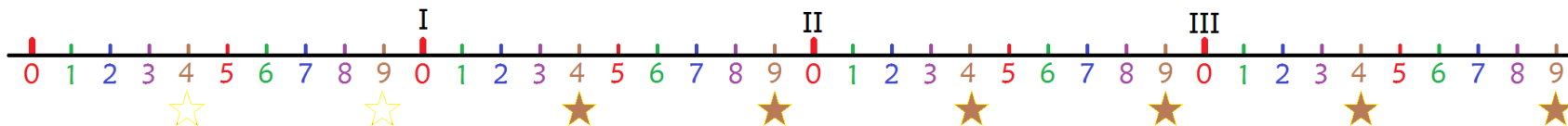
$$X=5n+3$$

- 必须将若干个5克砝码换成7克砝码，使得7克砝码的总重量比5克砝码的总重量大3，
- 即： $7a-5b=3$ ， $(a,b \geq 0, b \leq n)$
- 穷举得到 $a=4, b=5$ 是满足上式最小的——将5个5克换成4个7克，得到比原来重3克的重量
- 这显然需要在 $X=5n+3$ 中， n 至少存在5个5克砝码。即： $X=5n+3 \geq 5*5+3=28$ 。



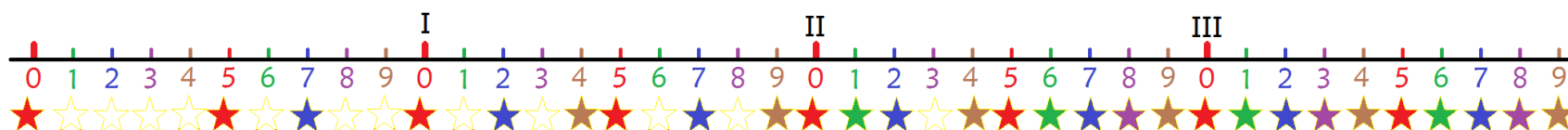
$$X=5n+4$$

- 必须将若干个5克砝码换成7克砝码，使得7克砝码的总重量比5克砝码的总重量大4，
- 即： $7a-5b=4$ ， $(a,b \geq 0, b \leq n)$
- 穷举得到 $a=2, b=2$ 是满足上式最小的——将2个5克换成4个7克，得到比原来重4克的重量
- 这显然需要在 $X=5n+4$ 中， n 至少存在2个5克砝码。即： $X=5n+4 \geq 5*2+4=14$ 。



综上

□ 将所有5种情况表示在统一的数轴上：



□ 显然，当 $X \geq 24$ 时，用足够大的5克和7克砝码能够组合出任意的整数克数。

花开堪折直须折

- 婚介所登记了N位男孩和N位女孩，每个男孩都对N个女孩的喜欢程度做了排序，每个女孩都对N个男孩的喜欢程度做了排序。你作为月老，能否给出稳定的牵手方案？
 - 分析：每个男孩都对N个女孩做了排序，N个男孩的钟情度形成 $N \times N$ 的矩阵A；同时，N个女孩的钟情度形成 $N \times N$ 的矩阵B；
 - 如果男孩i和女孩a牵手，但男孩i对女孩b更喜欢，而女孩b的男友j拼不过男孩i，则没有力量阻碍男孩i和女孩b的私奔，这即是不稳定的。

题目解析

- 采取贪心的思路：
- 所有男孩同时对各自最喜欢的女孩表白：
 - 若女孩a只有1个男孩i表白，接受男孩i；
 - 若女孩a有多位男孩表白，选择她最喜欢的男孩x
 - 若女孩a没有男孩表白，静静等待下一轮。
- 第k轮，所有单身男孩对各自第k喜欢的女孩表白：
 - 无论该女孩是否有男友；
 - 若女孩a收到了男孩i的表白：
 - 若女孩a没有男友，则接受男孩i；
 - 若相对于现任男友j，女孩a更喜欢男孩i，则接受男孩i；

正确性分析：匹配性和稳定性

- 在某个时刻，任何一个女孩要么已经有了男友，要么还会有男孩对其表白(她此刻没有男友，说明还有单身男孩，所以，该女孩不用着急，只需等待)，她总可以在表白的男孩中选择一个自己最喜欢的，因此，任何一个女孩必然都能找到男友。
- 因为“N-N”的比例关系，这也导致了每个男孩都能有女友。
- 不稳定情况：如果男孩i和女孩a牵手，但男孩i对女孩b更喜欢，而女孩b的男友j拼不过男孩i，则男孩i和女孩b的私奔。
 - 1、男孩i和女孩a牵手
 - 2、男孩i对女孩b更喜欢
 - 3、相对于男友j，女孩b的更喜欢男孩i
 - 打破上述三个前提中的一个：1和2说明男孩i曾经向女孩b表白却遭拒，这说明相对于男孩i，女孩b的更喜欢现任男友j。

Code

```
int _tmain(int argc, _TCHAR* argv[])
{
    int man[N][N] = {
        {2, 3, 1, 0},
        {2, 1, 3, 0},
        {0, 2, 3, 1},
        {1, 3, 2, 0},
    }; //男孩喜欢的女孩列表
    int woman[N][N] = {
        {0, 3, 2, 1},
        {0, 1, 2, 3},
        {0, 2, 3, 1},
        {1, 0, 3, 2},
    }; //女孩喜欢的男孩列表
    int match[N]; //男孩的女友
    GaleShapley(man, woman, match);
    Print(match, N);
    Validate(man, woman, match);
    return 0;
}
```

```
const int N = 4;
void GaleShapley(const int (&man)[N][N], const int (&woman)[N][N], int (&match)[N])
{
    int wm[N][N]; //wm[i][j]: 女孩i对男孩j的排名
    int choose[N]; //choose[i]: 女孩i当前的男朋友
    int manIndex[N]; //manIndex[i]: 男孩i被多少个女孩拒绝过
    int i, j;
    int w, m;
    for(i = 0; i < N; i++)
    {
        match[i] = -1; //所有男孩都初始化为光棍
        choose[i] = -1; //所有女孩都初始化为光棍
        manIndex[i] = 0; //为0, 则意味着从最喜欢的女孩开始选
        for(j = 0; j < N; j++)
            wm[i][woman[i][j]] = j; //对男孩woman[i][j]的排名是第j位
    }

    bool bSingle = false; //是否所有男孩都有了女友
    while(!bSingle) //每个男孩当前轮选女友
    {
        bSingle = true; //尚未发现单身男孩
        for(i = 0; i < N; i++) //每个男孩i选择尚未被拒绝的最喜欢的女孩
        {
            if(match[i] != -1) //男孩i已经有女友
                continue;
            bSingle = false;
            j = manIndex[i]++;
            w = man[i][j]; //男孩i第j喜欢的女孩w
            m = choose[w]; //女孩w当前的男友m
            if((m == -1) || (wm[w][i] < wm[w][m])) //女孩w更喜欢男孩i
            {
                match[i] = w;
                choose[w] = i;
                if(m != -1)
                    match[m] = -1; //女孩w抛弃前男友m
            }
        }
    }
}
```

改进相互表白的策略？

- 首轮a：所有男孩同时对各自最喜欢的女孩表白：
 - 若女孩a只有1个男孩i表白，接受男孩i；
 - 若女孩a有多位男孩表白，选择她最喜欢的男孩x
 - 若女孩a没有男孩表白，静静等待下一轮。
- 首轮b：所有女孩同时对各自最喜欢的男孩表白：
 - 策略同a
- 第k轮a：所有单身男孩对各自第k喜欢的女孩表白：
 - 无论该女孩是否有男友；
 - 若女孩a收到了男孩i的表白：
 - 若女孩a没有男友，则接受男孩i；
 - 若相对于现任男友j，女孩a更喜欢男孩i，则接受男孩i；
- 第k轮b：所有单身女孩对各自第k喜欢的男孩表白：
 - 策略同a

一个反例

□ 男女孩各3位：

□ 每个男孩的钟情度矩阵为：

□ 每个女孩的钟情度矩阵为：

□ 一个可行解为：2、0、1

	0	1	2
0	0	1	1
1	1	0	2
2	1	2	0

	0	1	2
0	1	2	0
1	2	1	0
2	1	2	0

算法的思考和总结

□ 本算法能够解决这种情况吗？

■ 如果男孩*i*和女孩*a*牵手，但男孩*i*对女孩*b*更喜欢，而女孩*b*的男友*j*却更喜欢女孩*a*。

□ 这即著名的对稳定婚姻策略的Gale-Shapley算法，由于女孩选择对其表白的男孩的最优者，因此也称为延迟认可算法。

■ 可用于学生志愿填报(员工选择职位、毕业生选择单位)等“二部”问题。

■ 如果没有“有向边”，则该算法不适用。

猜数字游戏

- 两个聪明人A和B玩猜数字的游戏。他们在脑门上各贴一个正整数数字，两个数字只相差1，A和B只能看到对方的数组而看不到自己的。
- 以下是两人的对话：
 - A：我不知道
 - B：我也不知道
 - A：我知道了
 - B：我也知道了
- 上述4句对话结束后，聪明的你帮助A、B推算下，他们的数字各是多少呢？

引理

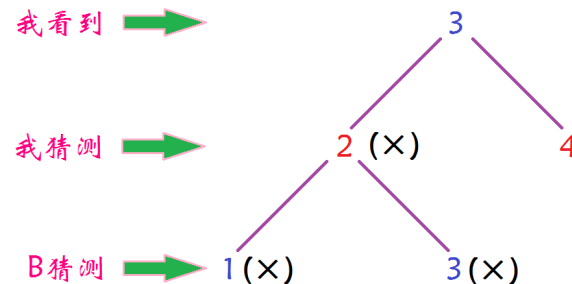
- 引理1：如果A看到B是1，则A马上可以断定：自己是2。
 - 因为条件给定了数字都是正整数。
- 引理2：如果A看到B是2，并且B说“我不知道”，则A马上可以断定：自己是3。
 - 因为A根据B是2可以推断自己是1或者3；
 - 如果自己是1，根据引理1，B会说“我知道”。

考察 “A是3，B是2”这种情况

- 第一次，A看到2，无法判断自己是1或3，只好说不知道；
- 第二次，B看到3，无法判断自己是2或4，只好说不知道；
- 第三次，A得知了“B不知道”，因此，B看到的一定不是1(根据引理1)，所以，A断定了自己是3；
- 第四次，B得知了“A知道”，因此，A如果看到4是无法断定自己是3还是5，因此，A一定是看到了自己是2。

考察 “A是4，B是3”这种情况

- 前两次，A看到3，B看到4，无法判断自己是几，都说不知道；
- 第三次，A的心理活动：
 - 可以断定我是4而不可能是2。理由如下：
 - 整理当前信息：假定我是2，且我说“我不知道”。根据引理2，B一定会断定自己是3。与当前B说“我不知道”矛盾。
- 第四次，B的心理活动：
 - 可以断定自己是3而不可能是5。理由如下：
 - 整理当前信息：假定A看到我是5，A会猜测 he 自己是4或者6
 - 如果A自己是4或者6，A和我都会顺次说“我不知道”；
 - 因此，A无法得出自己是5的结论。
 - 我不是5，那么我只能3。



我看到 →

我猜测 →

B猜测 →

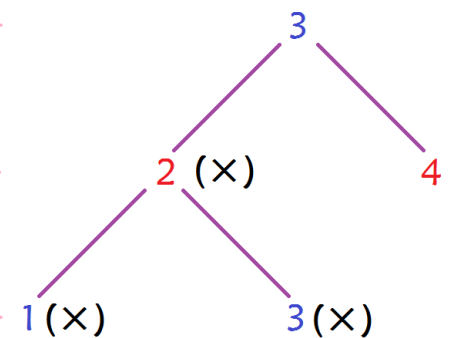
逻辑总结

□ 复杂的逻辑题可以用**二叉树(N叉树)**做辅助推理，原理是：只要某个结点的两个孩子(所有孩子)都不可能，则这个结点不可能。

■ 如A4B3情况下，A的**推理树**如右上图所示。

□ 注意：两人的说话顺序是有决定作用的，是**不对称消息**：

■ A说话，则A是在看到B的内容后**做判断**，B可根据A的内容在自己的推理树上**做剪枝**。



聪明的博弈 $\frac{2}{3} \cdot E$

□ 现有N个人，每人都从1到100按照自己的判断选一个整数，根据这N个数求均值E。规定选择最接近 $\frac{2}{3} \cdot E$ 的那个人是获胜者。聪明的你作为参与者之一，你会选择哪个数呢？

问题分析

- 如果每个人都**随机选择**，则N个数的期望收敛于50。因此，选择 $50 * 2/3 = 33.3$ 的人将获胜。如果所有人都如此思考，则大家都会选择33.3，从而获胜的人应该是选择 $33.3 * 2/3 = 22.2$ ，**继续塌缩**，最终所有人都选择最小数1，从而期望是0.66，大家全获胜。
- 问题是：
 - 不是所有人都**足够聪明**；
 - 总有“**更**”**聪明**的人选择对自己更有利的数；
 - 总有不打算胜利的人**搅局**：随机选数甚至选100等；

将人群分成若干类别

□ 最直接想法的人

- 什么都不想，随便从1-100中选一个，碰运气：假定占比是25%

□ 稍微思考的人

- 稍作思考发现：如果所有人都随机选择，那么平均是应该是50。所以，将50的2/3，即33.33作为我的选择：假定占比25%

□ 继续思考的人

- 如果所有人都选33.33，则平均数为33.33，那么，33.33的2/3，即22.22是胜利者。既然我会这么想，别人也会这么想，那么，如果大家都选22.22，则22.22的2/3，即14.8133是胜利者。继续塌缩下去，大家会选择1。所以，我应该选择最小的数1：假定占比30%
- 注意：由于是从1-100选数，是不能选0的。

将人群分成若干类别

□ 再思考的人

- 不是所有人都会进行理性思考，从而不是所有人都选择1。因此，这个数字应该很小，但由于前面三类人的存在，大体应该是10左右。所以，我选择10；
- 假定这部分人的占比是19%

□ 最聪明的人：

- 建立了“后验”模型，能够预知未来，他们能够直接判断出包括自己在内的N个数的平均值；
- 假定这部分人的占比是1%

□ 不打算胜利的搅局人：

- 随机选数或者直接选100等；
- 假定以上5类人中，以1%的概率随机产生。

六类人的总结

□ 人群分类

- A.最直接想法的人：25% B.稍微思考的人：25%
- C.继续思考的人：30% D.再思考的人：19%
- E.最聪明的人：1% F.搅局的人：低概率随机产生

- 若选中A类的人，则**均匀分布**随机产生1-100的值；
- 若选中B/C/D类，则产生**均值**为**33.3/1/10**的**高斯分布**的值，**方差**影响可以忽略，假定B/C/D的标准差分别为**10/3/3**；
- 由于E类绝顶聪明，他们的值和答案非常接近，则E类的均值和答案相同。所以，选中E类，则样本不累加，**直接丢弃**即可；
- 搅局者可能异常聪明或愚钝，无关智商，因此在上述五类人群中以**2%**的概率随机产生搅局者。
- 1万/10万/100万个样本的均值模拟为：**25.84/25.25/25.18**
- 如果在**大量现实人群**中，**推荐选择25**。

Code

```
213213010203134321200313232200100012120310321210
113022320021132010122212022103110014311213022202
201321211201310212021322121101020200211103122203
203020230231232320230101210302220313211431022223
012221230332300100120213133210233032331112130311
220112233022211201023103223033303202201031011120
200332220031030111323033022110312300300203200102
12222200111021011331012001022112022223301002223
211233013221233312110220312203230200213322202320
122022233310300232023230122220002023122303124310
22111322033113003333210011002222022221031220322
312230121121321022302110203303031231214232003100
230210021302112220211200200232122302113103012111
330312002102023230100131201102123230311023032333
120211111020132311203300320110330022312331100022
200103010010021020120023012223032121312331122211
221102020032323121131132232132123303133122223113
202220220021121202322223231021101213212302031021
312210321133102322010120011023311212211002311200
220222111130033001020202023302110230212111210022
010110330213101223302100121212103211133012021002
341102032003300201012330021121231130311331012033
2233101030003320
2499345
25.24922465
```

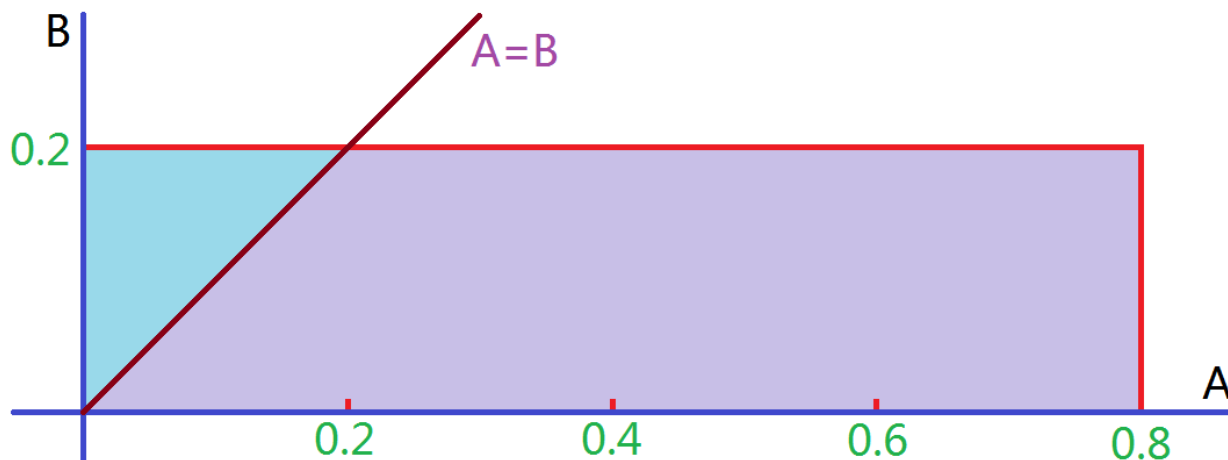
```
int GameEquilibrium()
{
    int category[] = {25, 25, 30, 19, 1};
    int size = sizeof(category) / sizeof(int);
    int* pCumulate = new int[size]; //累积概率, 方便随机选择类别
    CalcCumulate(category, size, pCumulate); //计算累积概率
    int N = 100000; //样本数目
    int M = 0; //除绝顶聪明人以外的有效样本数
    double cur = 0; //当前有效样本的总和
    int nCategory; //选择某个类别的人
    for (int i = 0; i < N; i++)
    {
        nCategory = RandSang(category, size, pCumulate);
        if (Randf(100) < 2) //恶作剧的人
        {
            cur += 100;
            M++;
        }
        else if (nCategory == 0) //随机化的人
        {
            cur += Rand(100);
            M++;
        }
        else if (nCategory == 1) //稍思维的人
        {
            cur += Gauss(33.3, 10, 1, 100);
            M++;
        }
        else if (nCategory == 2) //思维的人
        {
            cur += Gauss(1, 3, 1, 100);
            M++;
        }
        else if (nCategory == 3) //再思维的人
        {
            cur += Gauss(10, 3, 1, 100);
            M++;
        }
    }
    delete[] pCumulate;
    return (int) (cur/M+0.5);
}
```

商品推荐

- ❑ 商品推荐场景中过于聚焦的商品推荐往往会损害用户的购物体验，在有些场景中，系统会通过一定程度的随机性给用户带来发现的惊喜感。
- ❑ 假设在某推荐场景中，经计算A和B两个商品与当前访问用户的匹配度分别为0.8分和0.2分，系统将随机为A生成一个均匀分布于0到0.8的最终得分，为B生成一个均匀分布于0到0.2的最终得分，试计算最终B的分数大于A的分数的概率。

商品推荐

- $A=B$ 的直线上方区域，即为 $B>A$ 的情况。
- $S_{\text{蓝色}}=0.02$ $S_{\text{矩形}}=0.16$
- $p=0.02/0.16=0.125$



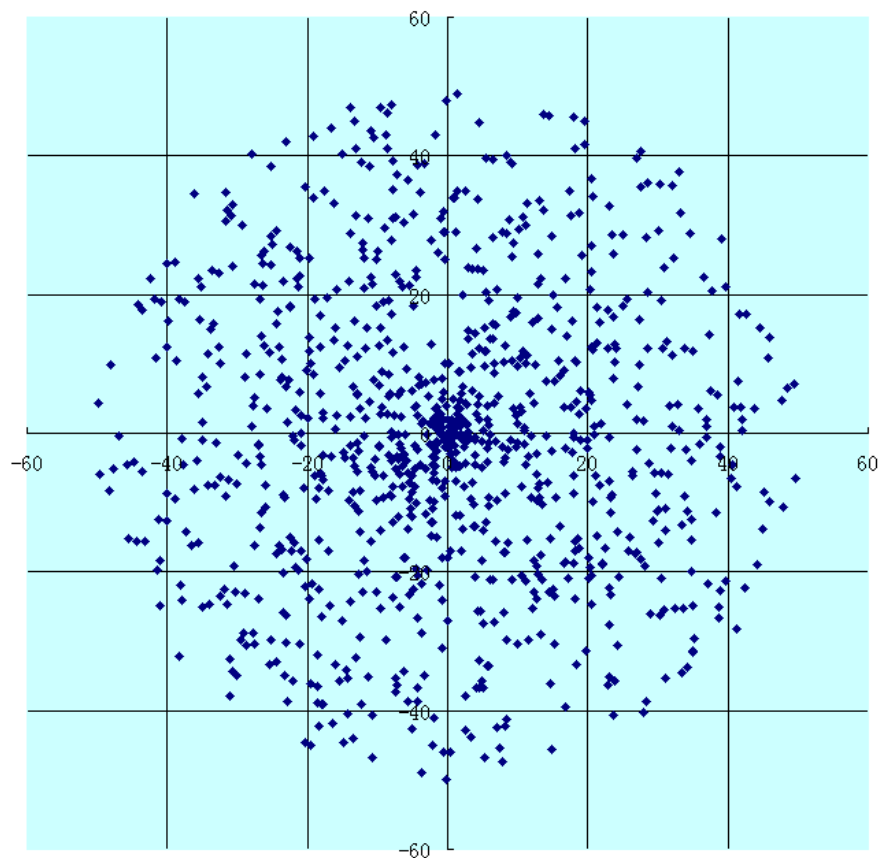
圆内均匀取点

- 给定定点 $O(x_0, y_0)$ 和半径 r ，使得二维随机点 (x, y) 等概率落在圆内。
- 分析
 - 因为均匀分布的数据是具有平移不变性，生成半径为 r ，定点为圆心的随机数 (x_1, y_1) ，然后平移得到 $(x_1 + x_0, y_1 + y_0)$ 即可。
 - 直接使用 $x = r * \cos\theta$ ， $y = r * \sin\theta$ 是否可以呢？
 - 具体试验一下。

圆内均匀取点代码与效果

```
int rand50()
{
    return rand() % 100 - 50;
}

int _tmain(int argc, _TCHAR* argv[])
{
    ofstream oFile;
    oFile.open(_T("D:\\rand.txt"));
    double r, theta;
    double x, y;
    for(int i = 0; i < 1000; i++)
    {
        r = rand50();
        theta = rand();
        x = r*cos(theta);
        y = r*sin(theta);
        oFile << x << '\\t' << y << '\\n';
    }
    oFile.close();
    return 0;
}
```



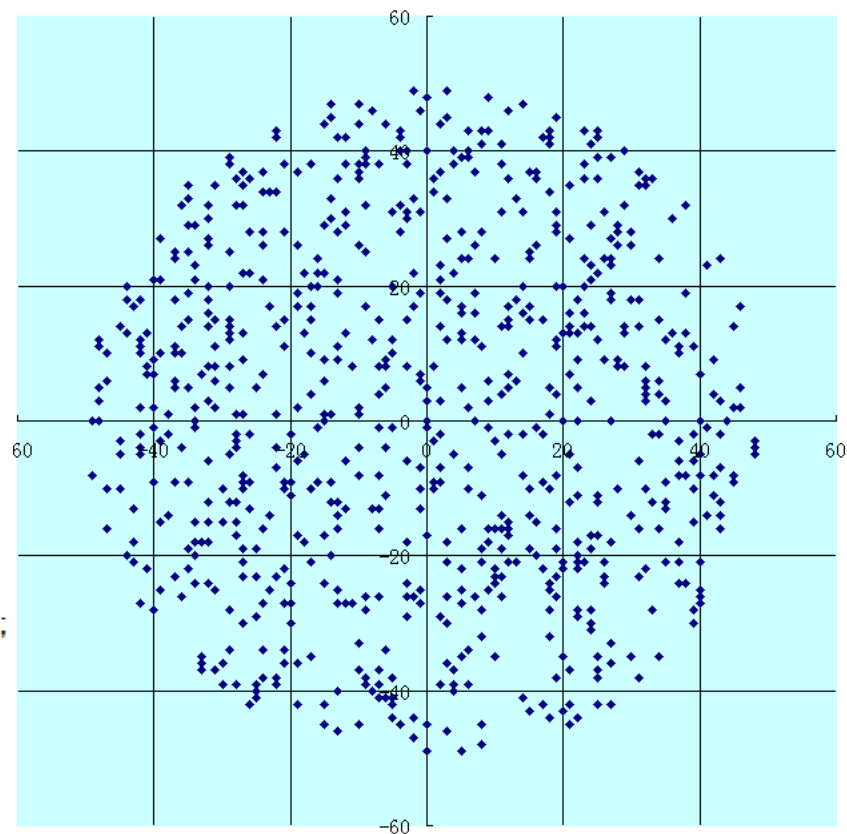
代码与效果

- 显然上述做法是不对的。但可以使用二维随机点的做法，若落在圆外，则重新生成点。结果如下。

代码与效果

```
int rand50()
{
    return rand() % 100 - 50;
}

int _tmain(int argc, _TCHAR* argv[])
{
    ofstream oFile;
    oFile.open(_T("D:\\rand.txt"));
    int x, y;
    for(int i = 0; i < 1000; i++)
    {
        x = rand50();
        y = rand50();
        if(x*x + y*y < 2500)
            oFile << x << '\\t' << y << '\\n';
    }
    oFile.close();
    return 0;
}
```



思想分析

- 不是每次生成随机数都能退出该算法
 - 有一定的接受率。
 - 思考：
 - 以多大的概率1次退出：接受率是多少？
 - 得到随机数需要的平均次数(期望)是多少？
 - 利用该方法计算圆周率？
- 这个做法简洁、有效，值得推荐；
 - 许多相关问题，往往可以如此解决。

一定接受率下的采样

- 给定函数rand7()随机返回自然数1~7，利用rand7()构造随机返回1~10的函数rand10()。
- 解：因为rand7仅能返回1~7的数字，少于rand10的数目。因此，多调用一次，从而得到49种组合。超过10的整数倍部分，直接丢弃。

Code

```
int rand10()
{
    int a1, a2, r;
    do
    {
        a1 = rand7() - 1;
        a2 = rand7() - 1;
        r = a1 * 7 + a2;
    } while (r >= 40);
    return r / 4 + 1;
}
```

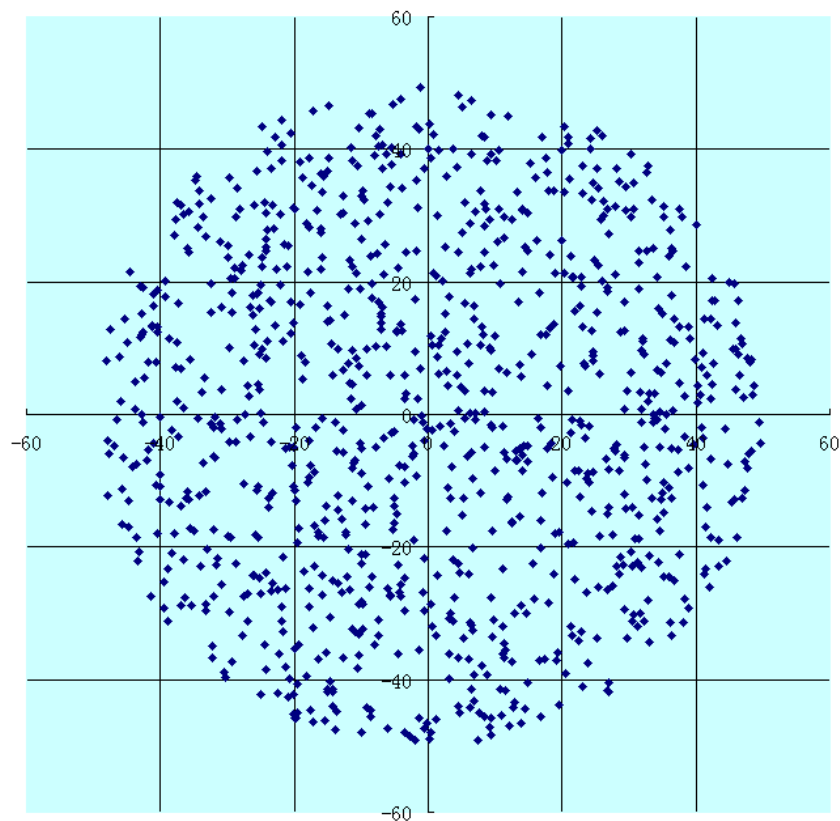
圆内均匀取点的1次成功算法(朴素)

- 问题分析：把随机点看做面积很小的区域，圆内均匀取点意味着随机点P的面积与圆的面积成正比。
- $S_p = kS$
- $S = \pi r^2$ ，与半径的平方成正比
- 从而， $S_p(r) = k\pi r^2$
- 将均匀生成的随机数x取平方根赋值给r；则 $S_p(r)$ 即为均匀分布。
- 同时，是与角度 θ 无关的，即：取均匀分布的随机数 θ 作为旋转角即可。

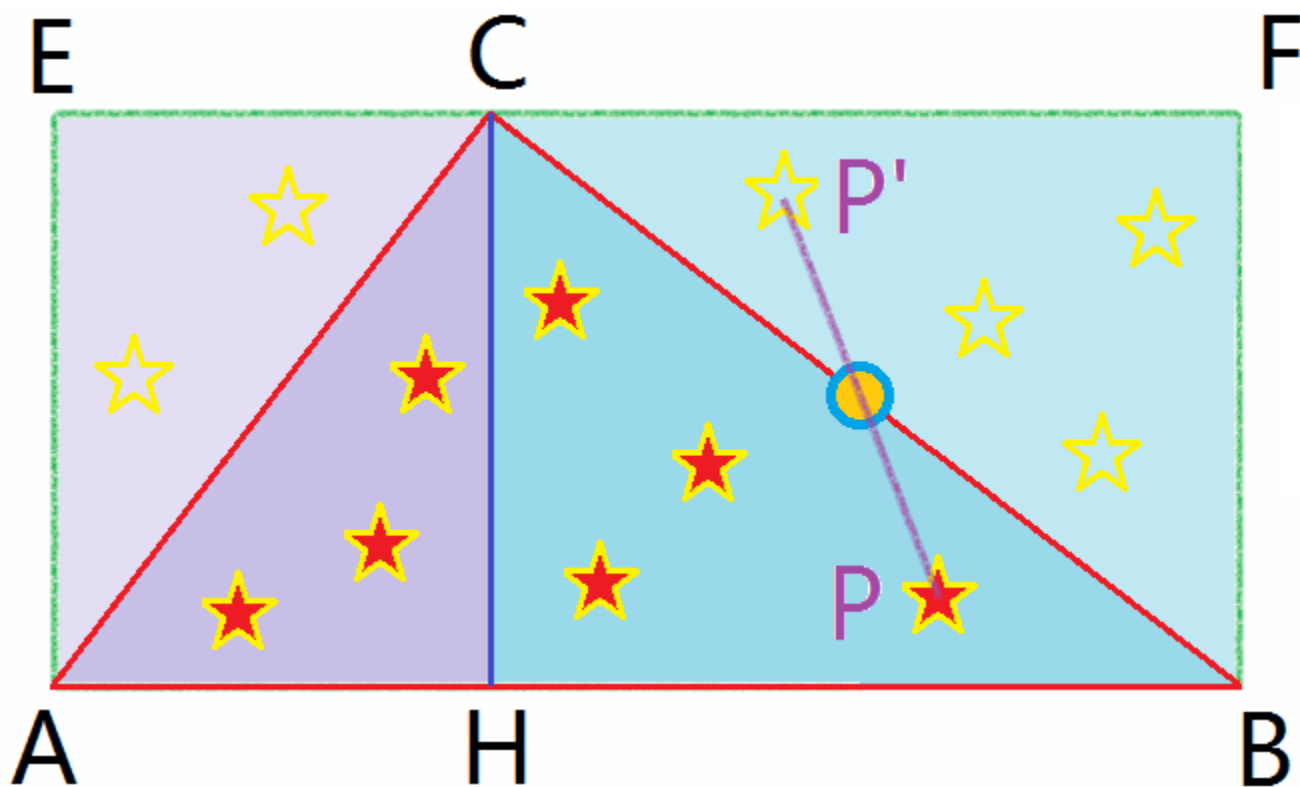
代码与效果

```
double rand2500()
{
    return rand() % 2500;
}

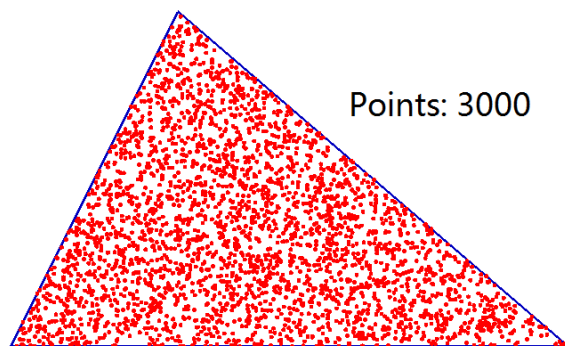
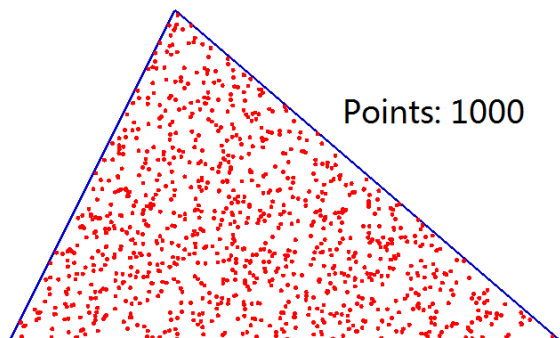
int _tmain(int argc, _TCHAR* argv[])
{
    ofstream oFile;
    oFile.open(_T("D:\\\\rand.txt"));
    double r, theta;
    double x, y;
    for(int i = 0; i < 1000; i++)
    {
        r = sqrt(rand2500());
        theta = rand();
        x = r*cos(theta);
        y = r*sin(theta);
        oFile << x << '\\t' << y << '\\n';
    }
    oFile.close();
    return 0;
}
```



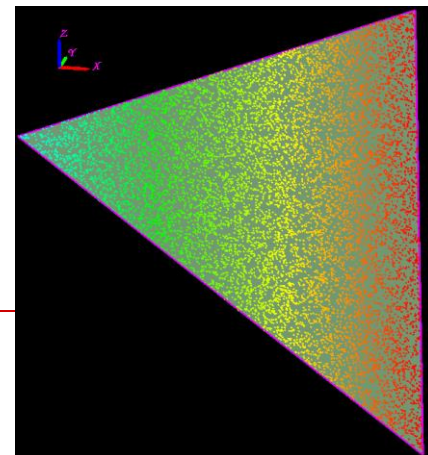
思考：将圆域换成三角形呢？



代码与效果



```
void CRandomTriangle::Random2(int nSize)
{
    CalcRotate();
    m_nSize = nSize;
    if(m_pRandomPoint)
        delete[] m_pRandomPoint;
    m_pRandomPoint = new CDelPoint[nSize];
    CDelPoint pt;
    for(int i = 0; i < nSize; i++)
    {
        pt.RandomInRectangle(m_ptExtend, m_ptHeight);
        if(m_tsBig.IsIn(pt))
        {
            pt += m_ptBase;
            m_pRandomPoint[i] = pt;
        }
        else if(m_tsLeft.IsIn(pt))
        {
            CDelPoint::MirrorPoint(pt, m_ptLeft0);
            pt += m_ptBase;
            m_pRandomPoint[i] = pt;
        }
        else if(m_tsRight.IsIn(pt))
        {
            CDelPoint::MirrorPoint(pt, m_ptRight0);
            pt += m_ptBase;
            m_pRandomPoint[i] = pt;
        }
    }
    CDelPoint::Save(m_pRandomPoint, m_nSize, _T("D:\\random.pt"), 0);
}
```



进一步思考

- 由于三点共面，所以三角形内的所有点必然在某平面上，因此，上述算法能够方便的推广到三维空间。
- 问题：请设计多边形内随机取点算法。
 - 圆内取点的思想：计算多边形的外包围矩形盒，生成外包围盒内的二维点，若点在多边形内，则退出；否则，继续探测。
 - 将多边形剖分成三角形集合，调用三角形内均匀取点算法。
- 算法2思路：
 - 按照面积为权重，选择某个三角形；
 - 生成该三角形内的随机点。
- 拓展
 - 每首歌有不同的分值，设计算法，根据分值随机推荐歌曲。
 - 如何将多边形快速剖分成三角形？注：Delaunay三角剖分

分值推荐

- 假定歌曲库中 N 首歌，每首歌给定一个整数分数。现在要求从 N 首歌中随机选择若干首推荐给用户，要求推荐的这些歌是和其分数作为正比的。
- 给定整数数组 $A[0 \dots N-1]$ ，按照 $A[i]$ 的值作为权值随机取数。

分析

□ 根据权值 $A[0 \dots N-1]$ 计算累积权值 $B[0 \dots N-1]$

■ $B_0 = A_0$

■ $B_i = B_{i-1} + A_i$

□ 区间 $[B_{i-1}, B_i)$ 对应元素 A_i



Code

```
int RandSong(const int* song, int size)
{
    int i;
    int* pCumulate = new int[size]; // i的范围: [i-1, i)
    pCumulate[0] = song[0];
    for(i = 1; i < size; i++)
        pCumulate[i] = pCumulate[i-1] + song[i];

    int nRec = rand() % pCumulate[size-1];
    int low = 0;
    int high = size-1;
    int mid;
    int nSong = -1;
    while(low < high)
    {
        mid = (low + high) / 2;
        if(nRec < pCumulate[mid])
            high = mid;
        else if(nRec > pCumulate[mid])
            low = mid + 1;
        else
        {
            nSong = mid+1;
            break;
        }
    }
    if(nSong == -1)
        nSong = low;
    delete[] pCumulate;
    return nSong;
}
```

另外的思路

- 计算所有歌的权值和sum，每首歌的权值除以sum，认为是各自的概率 $P[0 \dots N-1]$ ；
- 等概率选择 $nSong \in [0, N-1)$
- 生成 $p \in [0, 1]$ ，若 $p < P[nSong]$ ，则选择 $nSong$ ，否则，计算随机生成新的 $nSong$ ，继续探测。
 - 在KMeans++、标签传递算法LPA、主题模型LDA、随机游走等机器学习算法中都有应用。

Code

```
int RandSong2(const int* song, int size)
{
    int nSum = song[0];
    for(int i = 1; i < size; i++)
        nSum += song[i];

    bool bFind = false;
    int nCandidate = 0;
    while(!bFind)
    {
        nCandidate = rand() % size; //候选
        if(rand() % nSum < song[nCandidate])
        {
            bFind = true;
            break;
        }
    }
    return nCandidate;
}
```

试验结果

□ 初始值: $\text{song} = \{43, 63, 43, 89, 322, 2, 5, 32\}$

□ 真实概率:

■ 0.0718 0.105 0.0718 0.149 0.538 0.00334 0.00835 0.0534

□ 算法1

■ 10^3 : 0.073 0.096 0.077 0.152 0.536 0.004 0.013 0.049

■ 10^4 : 0.0758 0.107 0.0723 0.143 0.54 0.0028 0.0087 0.0505

□ 算法2

■ 10^3 : 0.076 0.095 0.084 0.137 0.539 0.003 0.008 0.058

■ 10^4 : 0.0719 0.101 0.0726 0.144 0.542 0.0025 0.0093 0.0567

进一步思考

- 将 $[0, B_n]$ 等分成 m 份， m 取合适的正整数，则正整数 $x \in [1, m]$ 对应一个样本 A_i 。



- 结论：

- 空间复杂度 $O(m)$
- 时间复杂度 $O(1)$

- 词潜入开源包word2vec
即使用该策略完成
Negative Sampling。

```
void InitUnigramTable() {
    int a, i;
    double train_words_pow = 0;
    double d1, power = 0.75;
    table = (int *)malloc(table_size * sizeof(int));
    for (a = 0; a < vocab_size; a++)
        train_words_pow += pow(vocab[a].cn, power);
    i = 0;
    d1 = pow(vocab[i].cn, power) / train_words_pow;
    for (a = 0; a < table_size; a++) {
        table[a] = i;
        if (a / (double)table_size > d1) {
            i++;
            d1 += pow(vocab[i].cn, power) / train_words_pow;
        }
        if (i >= vocab_size) i = vocab_size - 1;
    }
}
```


随机排列

- 给定N个数，设计算法，输出随机排列的一个结果。
 - 要求：输出任何一个排列的概率是相同的。
 - STL `std::random_shuffle`

算法1

```
int Random(int N)    //[0, N]
{
    return rand() % (N+1);
}

void RandomShuffle(int* a, int size)
{
    int j;
    for(int i = 1; i < size; i++)    //待生成倒数第i个数
    {
        j = Random(size-i);    //[0, size-i]
        swap(a[j], a[size-i]);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    const int N = 10;
    int a[N];
    for(int i = 0; i < N; i++)
        a[i] = i+1;
    RandomShuffle(a, N);
    Print(a, N);
    return 0;
}
```

算法2

```
int Random2(int a, int b)
{
    return rand() % (b-a+1) + a;
}

void RandomShuffle2(int* a, int size)
{
    int j;
    for(int i = 0; i < size-1; i++) //i为待生成的第几个数
    {
        j = Random2(i, size-1); //[i, size-1]
        swap(a[i], a[j]);
    }
}
```

金钗赠诗问题

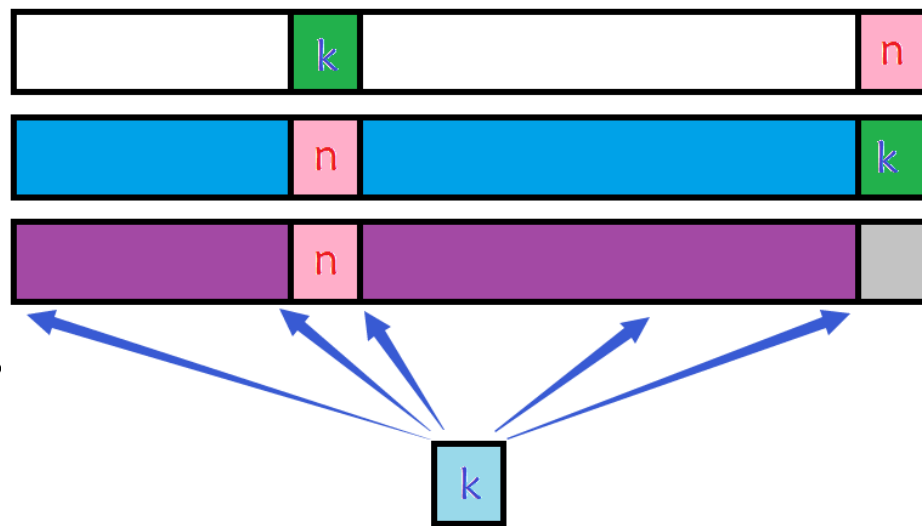
- 赛诗会后，十二金钗待奔前程。分别宴上，12人各写一首诗放入宝囊。大家任取，若取到自己的诗，则再取一首并放回自己的诗。12人都拿到别人的诗作算一种分配。问：共有多少种不同的分配？

问题的由来

- 本质即给定 n 个人写 n 首诗，要求赠给其他人，共有多少种分配方法。
- 一般提法：1到 n 的全排列中，第 i 个数不是 i 的排列共有多少种？
 - 最早是由丹尼尔·伯努利(Danid Bernoulli)提出的“错位排列”问题。

问题分析

- 假定 n 个数的错位排列数目为 $dp[n]$
- 先考察数字 n 的放置方法：显然， n 可以放在从1到 $n-1$ 的某个位置，共 $n-1$ 种方法；假定放在了第 k 位。
- 对于数字 k ：
 - 要么放置在第 n 位，
 - 要么不放置在第 n 位。
 - 下面分别讨论



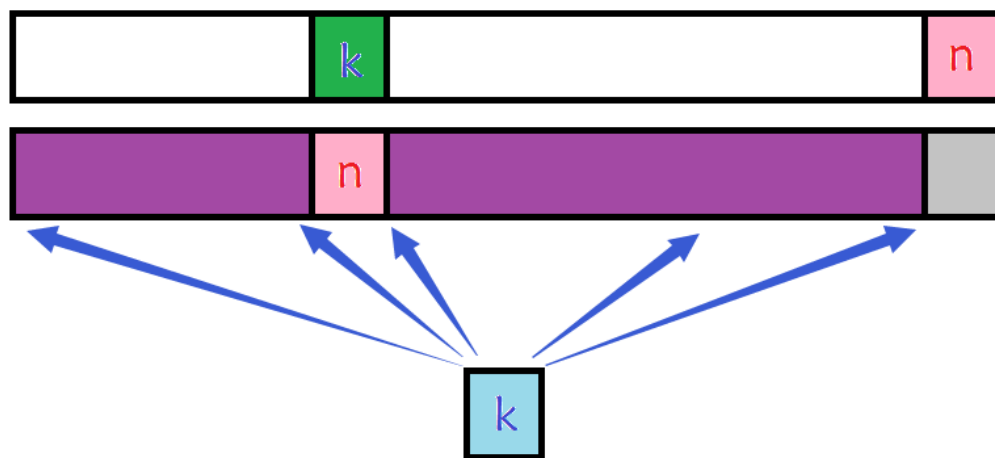
数字k放置在第n位

- 相当于数字k和数字n交互位置后，其他n-2个数字做错位排列，因此有 $dp[n-2]$ 种方法。



数字k不放置在第n位

- 将数字k暂时更名为n(这是可以做到的：因为真正的n已经放在第k位上，真正的n不再考虑之列)，现在需要将1到k-1以及k+1到n这n-1个数放置在相应位置上，要求数字和位置不相同！显然是n-1个数的错位排列，有 $dp[n-1]$ 种方法。



错位排列递推公式

□ 综上， $dp[n]=(n-1)*(dp[n-1]+dp[n-2])$

□ 初值：

■ 只有1个数字，错位排列不存在， $dp[1]=0$ ；

■ 只有2个数字，错位排列即交换排列， $dp[2]=1$ ；

$$a(n)=\begin{cases} (n-1)\cdot[a(n-1)+a(n-2)] & n > 2 \\ 1 & n = 2 \\ 0 & n = 1 \end{cases}$$

错位排列的通项公式

- 使用基本的加法和乘法原理，能够得到错位公式的**通项形式**：

$$S(n) = P_n^n - C_n^1 P_{n-1}^{n-1} + -C_n^2 P_{n-2}^{n-2} + \cdots + (-1)^n - C_n^n P_0^0$$

- 或等价形式：

$$S(n) = n! \left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \frac{1}{4!} + \cdots + (-1)^n \frac{1}{n!} \right)$$

思考题

□ 毕业典礼后，某宿舍三位同学把自己的毕业帽扔了，随后每个人随机地拾起帽子，试计算三个人中没有人选到自己原来带的帽子的概率。

金钗赠诗问题II

- 赛诗会上十二金钗各赋诗一首，12人各自随机挑选一首后，李纨曰：“大家通过两两交换的方式，换回自己的律诗；但要求只能跟我交换”。现已知：

黛玉、宝钗、元春、探春、湘云、妙玉、
迎春、惜春、熙凤、巧姐、李纨、可卿

各自拿到的律诗作者为：

熙凤、黛玉、迎春、惜春、湘云、可卿、
探春、元春、宝钗、巧姐、妙玉、李纨

- 试计算至少需要多少次交换，才能使得所有人交换得到自己的律诗？

问题分析

- 十二金钗分别标号为 $0, 1, 2, \dots, 11$ ，其中，李纨为0号。题目实际上给定了12个整数的一个排列，要求只和0交换，最终形成升序数组，求最少的交换次数。
- 统计每个“环”的长度 L ：
 - 若包含0，则该环的最少交换次数为 $L-1$ ；
 - 若不包含0，则该环的最少交换次数为 $L+1$ ；
- 将所有环的交换次数累积即可。
 - 时间复杂度 $O(N)$ ，空间复杂度 $O(N)$

Code

```
int _tmain(int argc, _TCHAR* argv[])
{
    int N = 12;
    int* a = new int[N];
    for(int i = 0; i < N; i++)
        a[i] = i;
    random_shuffle(a, a+N);
    cout << Exchange0(a, N) << endl;
    delete[] a;
    return 0;
}
```

```
int Exchange0(const int* a, int size)
{
    bool* visit = new bool[size];
    memset(visit, 0, sizeof(bool)*size);
    int j;
    int c = (a[0] == 0) ? 2 : 0;
    for(int i = 0; i < size; i++)
    {
        if(visit[i] || (a[i] == i))
            continue;
        j = a[i];
        while(j != i)
        {
            c++;
            visit[j] = true;
            j = a[j];
        }
        c += 2;
    }
    c -= 2;
    delete[] visit;
    return c;
}
```

我们在这里

□ <http://wenda.ChinaHadoop.cn>

■ 视频/课程/社区

□ 微博

■ @ChinaHadoop

■ @邹博_机器学习

□ 微信公众号

■ 小象学院

■ 大数据分析挖掘

互联网新技术在线教育领航者

小象问答 搜索标题、用户 全站内容搜索 提问 首页 动态 发现 话题 通知

全部 招聘求职 机器学习 大数据平台技术 DCon 大数据行业应用 NoSQL数据库 数据科学 江湖救急

发现 最新 推荐 热门 等待回复

graphviz has no attribute 'write' 贡献
邹博 回复了问题 • 2 人关注 • 1 个回复 • 3 次浏览 • 2017-04-09 15:48

sklearn中如何理解Pipeline机制 贡献
数据分析与数据挖掘 邹博 回复了问题 • 2 人关注 • 1 个回复 • 28 次浏览 • 2017-04-09 15:39

关于9.Logistic回归的ppt中第9页的对数线性函数 贡献
机器学习 邹博 回复了问题 • 3 人关注 • 3 个回复 • 39 次浏览 • 2017-04-09 15:35

关于“贝叶斯估计中，最大后验概率估计就是结构化风险最小化的例子：当模型是条件概率分布，损失函数为对数损失函数，模型的复杂度由模型的先验概率表示，结构化风险最小化就等价于最大后验概率估计” 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 26 次浏览 • 2017-04-09 15:27

关于连续值的预测 贡献
咨询 邹博 回复了问题 • 2 人关注 • 1 个回复 • 31 次浏览 • 2017-04-09 15:24

拉格朗日对偶函数为什么一定是凸函数 贡献
数据科学 邹博 回复了问题 • 2 人关注 • 2 个回复 • 26 次浏览 • 2017-04-09 15:20

梯度下降公式中的斯堪J 是 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 29 次浏览 • 2017-04-09 15:17

深度学习适合做预测吗？ 贡献
深度学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 27 次浏览 • 2017-04-09 15:15

关于6.4PCA_FeatureSelection.py中plt.legend的参数疑问 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 28 次浏览 • 2017-04-09 15:04

@邹博 有哪些可以下载数据源的网站？ 贡献
数据分析与数据挖掘 邹博 回复了问题 • 4 人关注 • 1 个回复 • 31 次浏览 • 2017-04-09 14:53

LDA主题模型 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 29 次浏览 • 2017-04-09 14:45

代码10.6bagging_ridged老师提到了采样率设为0.2能够使峰值部分的数据被体现出来。这是为什么呢？ 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 22 次浏览 • 2017-04-09 14:26

GraphViz's executables not found 贡献
机器学习 邹博 回复了问题 • 3 人关注 • 2 个回复 • 23 次浏览 • 2017-04-09 13:47

决策树中关于feature_importances代码的问题 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 6 次浏览 • 2017-04-09 13:11

专题
招聘求职
大数据行业应用
数据科学
系统与编程
云计算技术

热门话题 更多 >
机器学习 907 个问题, 230 人关注
spark 387 个问题, 172 人关注
hadoop 1059 个问题, 155 人关注
python数据分析 171 个问题, 28 人关注
数据分析与数据挖掘 54 个问题, 111 人关注

热门用户 更多 >
小心巴 14 个问题, 0 次赞同
叉叉V 45 个问题, 22 次赞同
铁甲无声 10 个问题, 0 次赞同
带刀锦衣卫 13 个问题, 0 次赞同

感谢大家！

恳请大家批评指正！