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

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
- Buffon’s Needle
- Gale-Shapley算法
- 猜数字游戏
- 概率化商品推荐
- 圆内均匀取点
- 带拒绝的采样
- 带权推荐
- 金钗赠诗问题

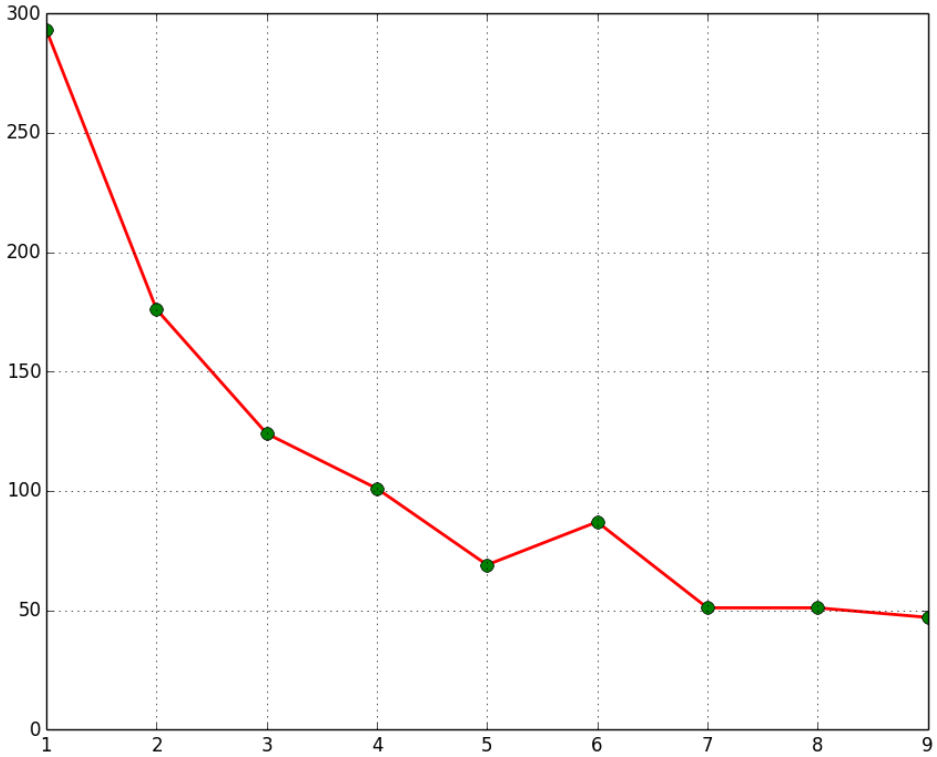
统计数字的概率

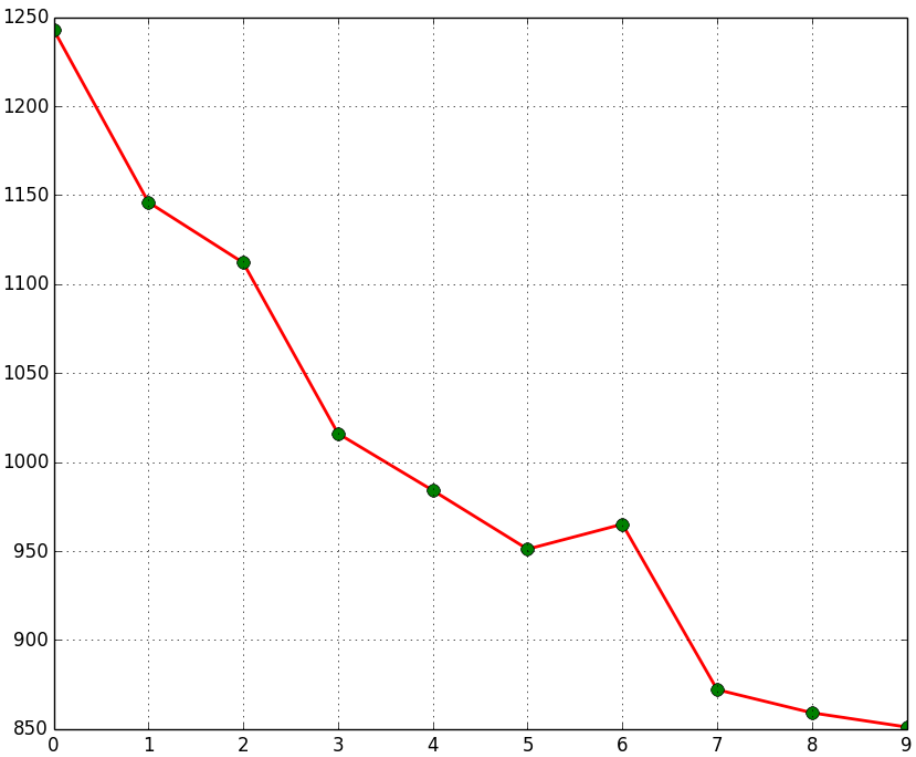
- 给定某正整数N，统计从1到N!的所有数中，首位数字出现1的概率。
- 进而，可以计算首位数字是2的概率，是3的概率，从而得到一条“九点分布”。

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[1243/1146/1112/1016/984/951/965/872/859/851]



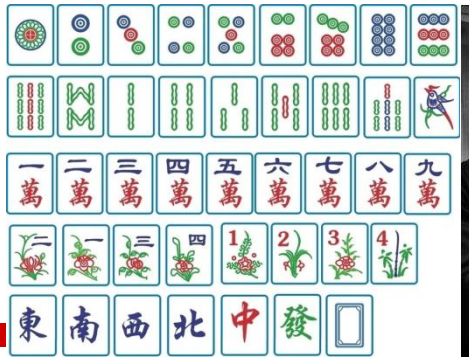


本福特定律

- 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4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：

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

计算概率

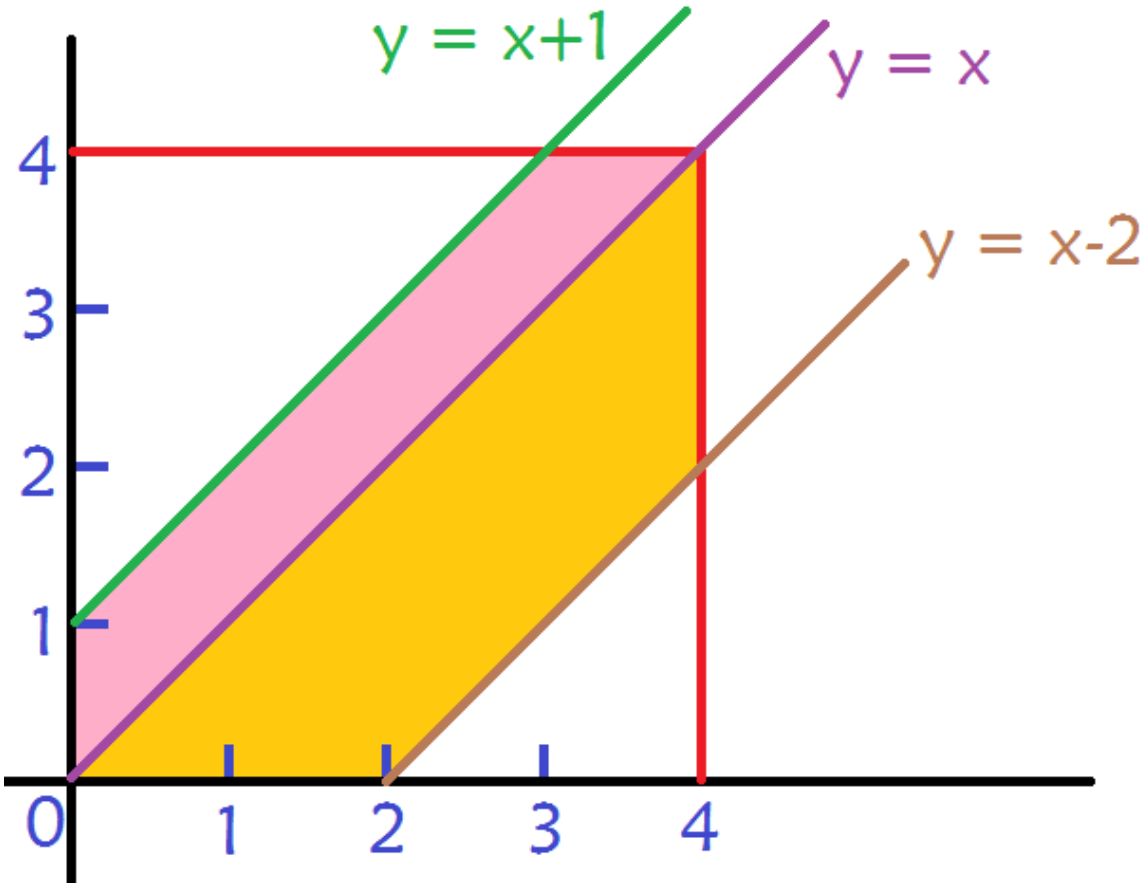
- A、B两国元首相约在首都机场晚20点至24点交换一份重要文件。如果A国的飞机先到，A会等待1个小时；如果B国的飞机先到了，B会等待2个小时。假设两架飞机在20点至24点降落机场的概率是均匀分布，试计算能够在20点至24点完成交换的概率。
  - 假设交换文件本身不需要时间。

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。

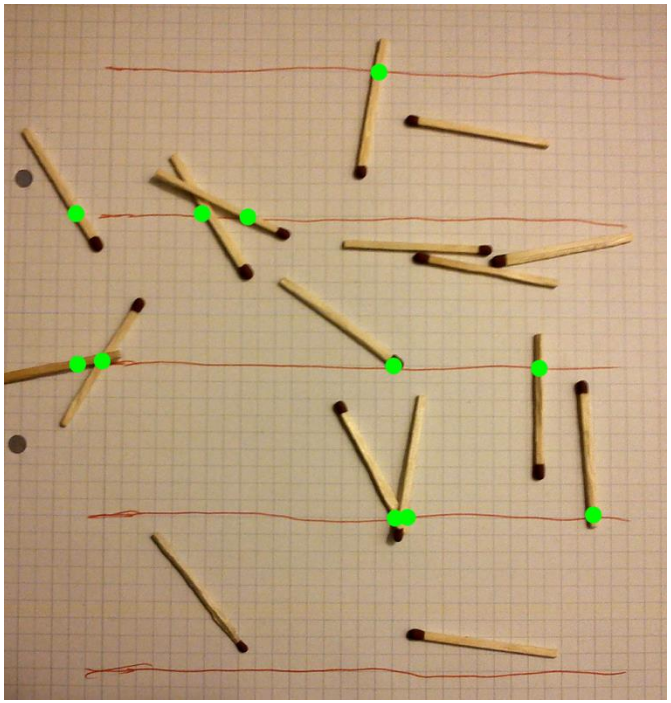
计算面积

- 三角形面积
  - 9/2、2
- 矩形面积
  - 16
- 概率
  - 19/32



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∈[0,a/2]，投针是随机的，y为均匀分布：

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

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

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

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

蒙特卡洛模拟

- 有交点的概率：

$$P(X)=\int_0^{\pi}\int_0^{\frac{L}{2}\sin\theta} f(y,\theta)dyd\theta=\int_0^{\pi}\int_0^{\frac{L}{2}\sin\theta} f(y)f(\theta)dyd\theta$$

$$=\int_0^{\pi}\int_0^{\frac{L}{2}\sin\theta}\frac{2}{a}\cdot\frac{1}{\pi}dyd\theta=\frac{2L}{a\pi}$$

- 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);
}
```

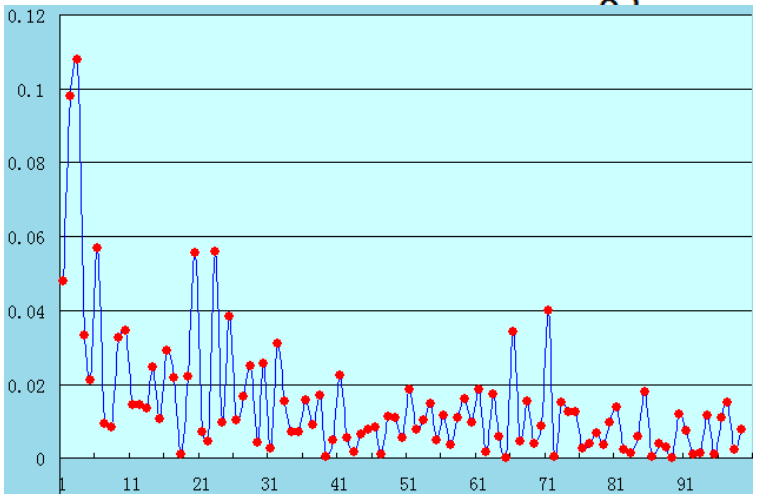
Code2

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



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×N的矩阵A；同时，N个女孩的钟情度形成N×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：

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

- 每个女孩的钟情度矩阵为：

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

- 一个可行解为：2、0、1

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

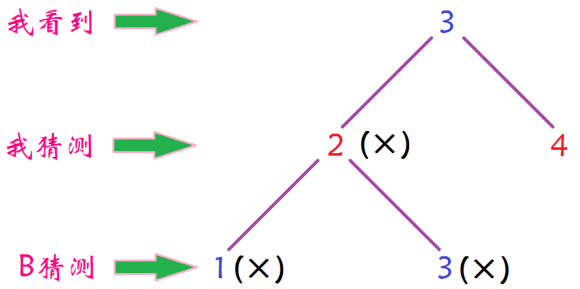
- 引理1：如果A看到B是1，则A马上可以断定：自己是2。
  - 因为条件给定了数字都是正整数。
- 引理2：如果A看到B是2，并且B说“我不知道”，则A马上可以断定：自己是3。
  - 因为A根据B是2可以推断自己是1或者3；
  - 如果自己是1，根据引理1，B会说“我知道”。

考察“A是3，B是2”这种情况

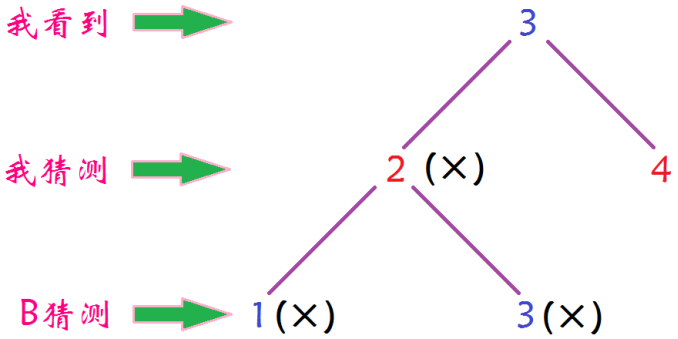
- 第一次，A看到2，无法判断自己是1或3，只好说不知道；
- 第二次，B看到3，无法判断自己是2或4，只好说不知道；
- 第三次，A得知了“B不知道”，因此，B看到的一定不是1(根据引理1)，所以，A断定了自己是3；
- 第四次，B得知了“A知道”，因此，A如果看到4是无法断定自己是3还是5，因此，A一定是看到了自己是2。

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会猜测他自己是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，那么我只能是3。



逻辑总结



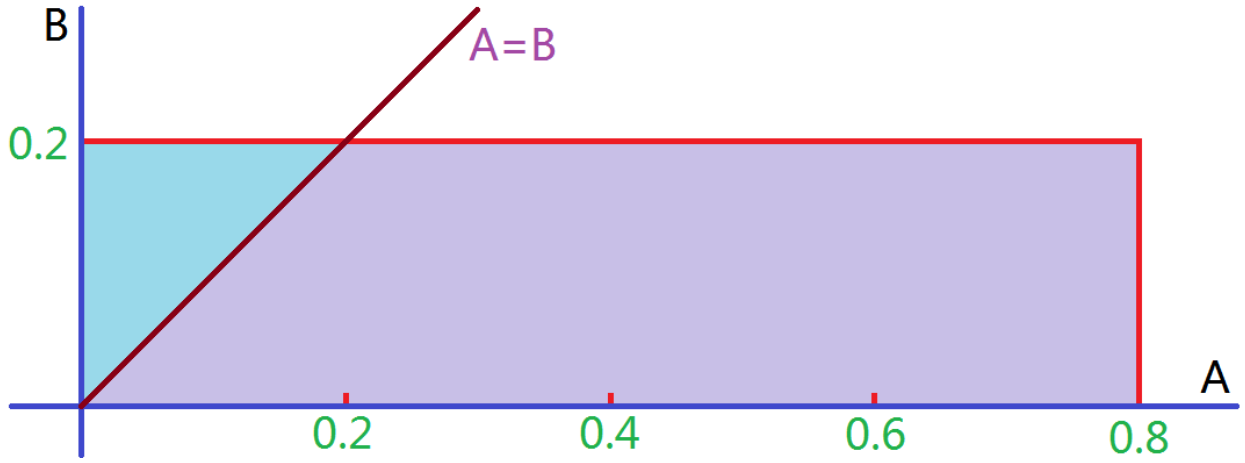
- 复杂的逻辑题可以用二叉树(N叉树)做辅助推理，原理是：只要某个结点的两个孩子(所有孩子)都不可能，则这个结点不可能。
  - 如A4B3情况下，A的推理树如右上图所示。
- 注意：两人的说话顺序是有决定作用的，是不对称消息：
  - A说话，则A是在看到B的内容后做判断，B可根据A的内容在自己的推理树上做剪枝。

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。
- $S_{蓝色}=0.02$　　$S_{矩形}=0.16$
- p=0.02/0.16=0.125

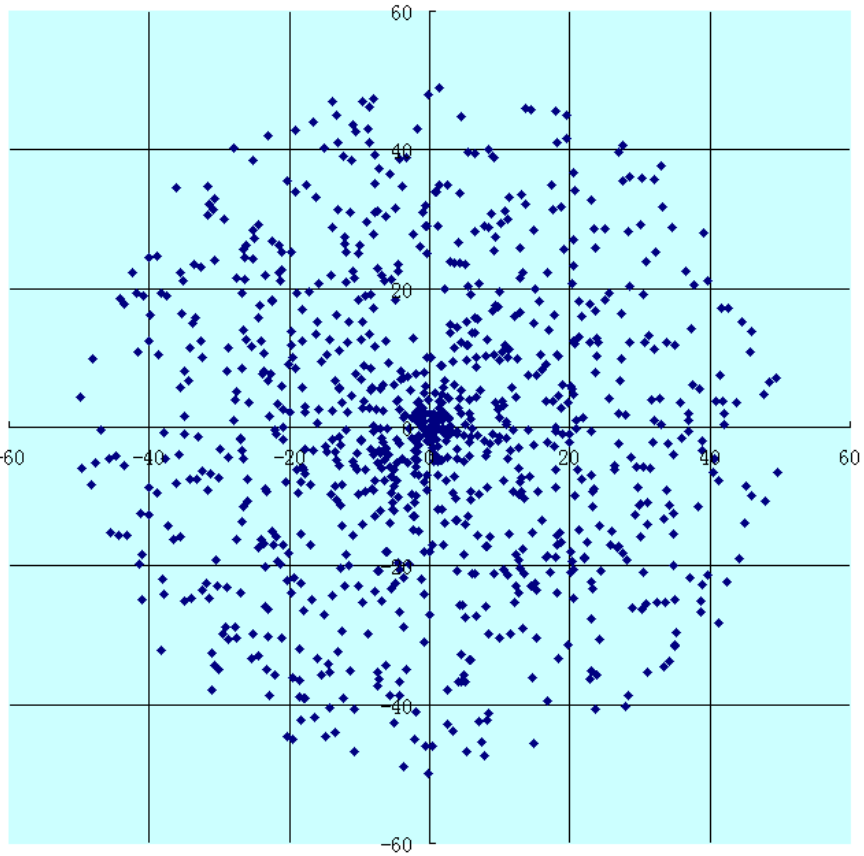


圆内均匀取点

- 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;
}
```

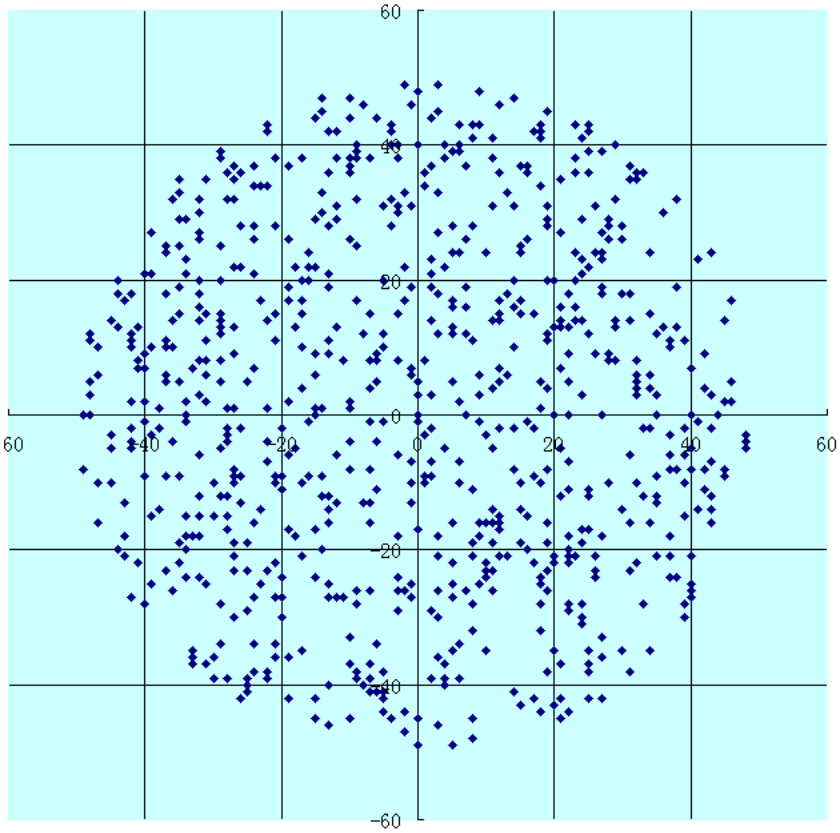


代码与效果

- 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？
    - 得到随机数的需要的平均次数(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

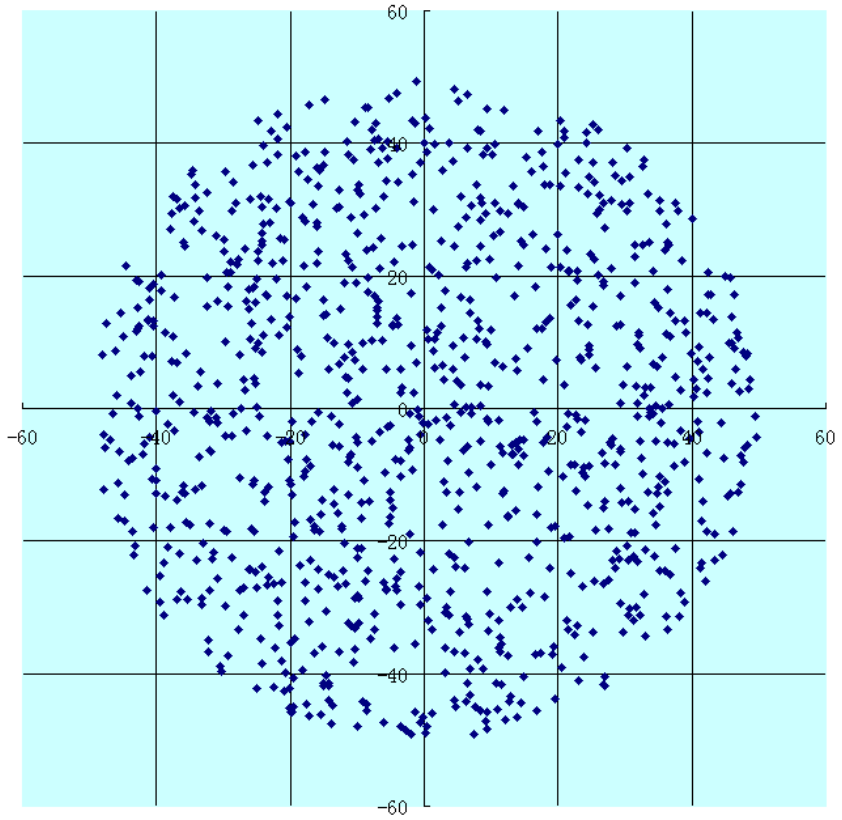
```
int rand10()
{
    int a1, a2, r;
    do
    {
        a1 = rand7() - 1;
        a2 = rand7() - 1;
        r = a1 * 7 + a2;
    } while (r >= 40);
    return r / 4 + 1;
}
```

圆内均匀取点的1次成功算法(朴素)

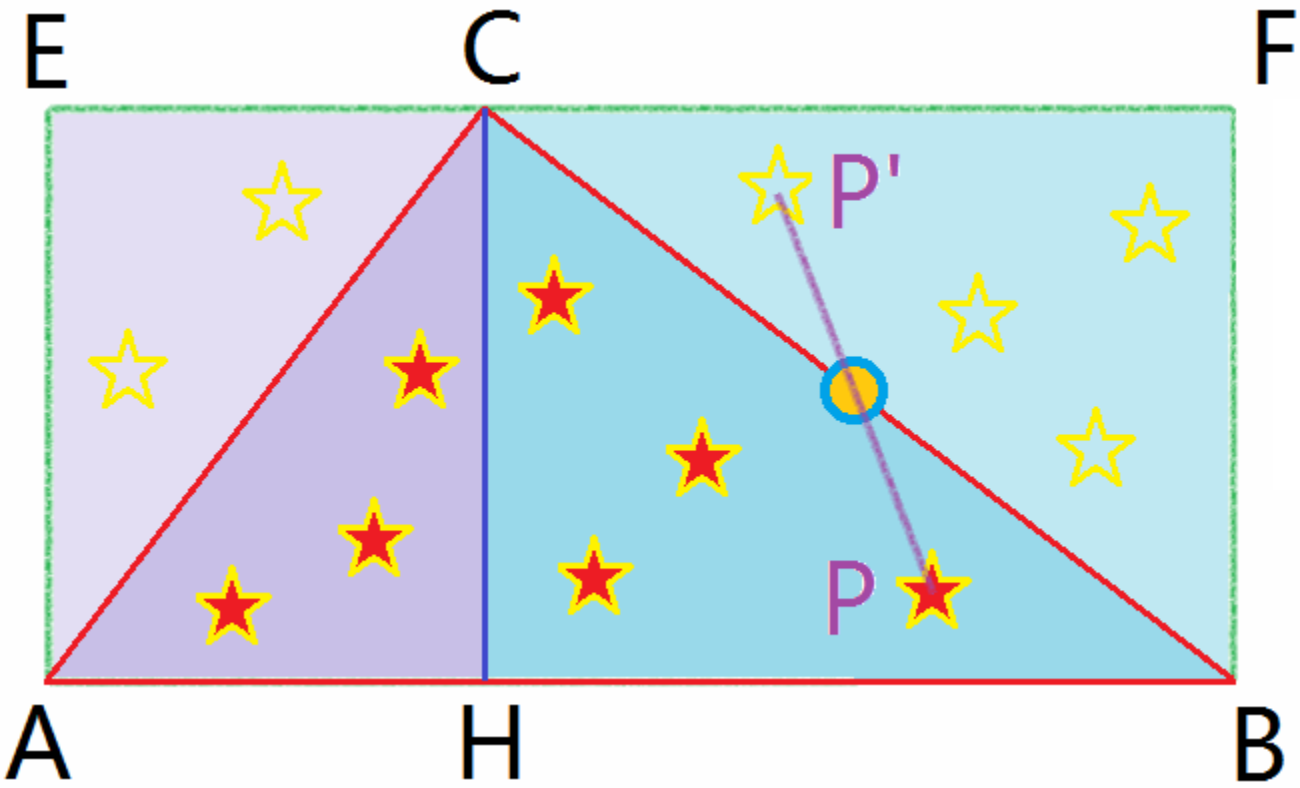
- 问题分析：把随机点看做面积很小的区域，圆内均匀取点意味着随机点P的面积与圆的面积成正比。
- $S_p=kS$
- $S=\pi r^2$，与半径的平方成正比
- 从而，$S_p(r)=k\pi r^2$
- 将均匀生成的随机数x取平方根赋值给r；则$S_p(r)$即为均匀分布。
- 同时，是与角度θ无关的，即：取均匀分布的随机数θ作为旋转角即可。

代码与效果

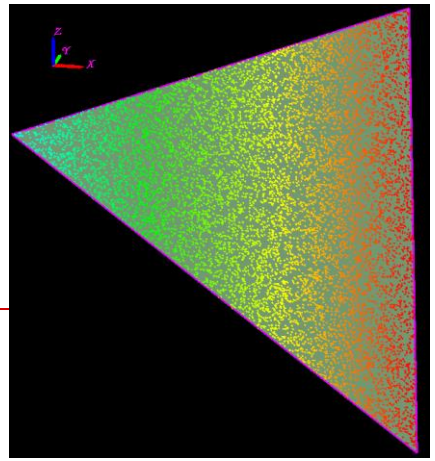
```
double rand2500()
{
    return rand() % 2500;
}

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;
}
```

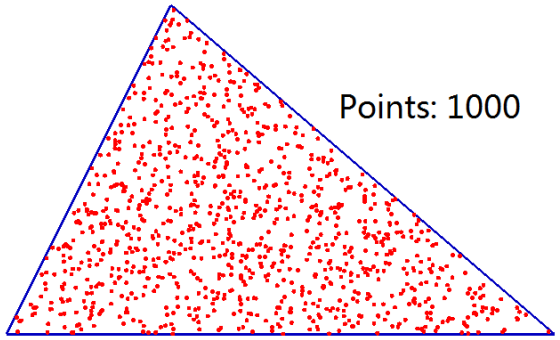


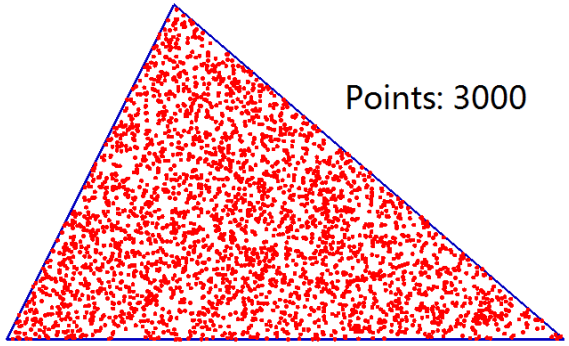
思考：将圆域换成三角形呢？



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…N-1]，按照A[i]的值作为权值随机取数。

分析

- 根据权值A[0...N-1]计算累积权值B[0...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…N-1]；
- 等概率选择nSong∈[0,N-1)
- 生成p∈[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。

  - 该问题在标签传递算法LPA、主体模型LDA、随机游走RW等机器学习问题中都有应用。

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： 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

思考

- 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

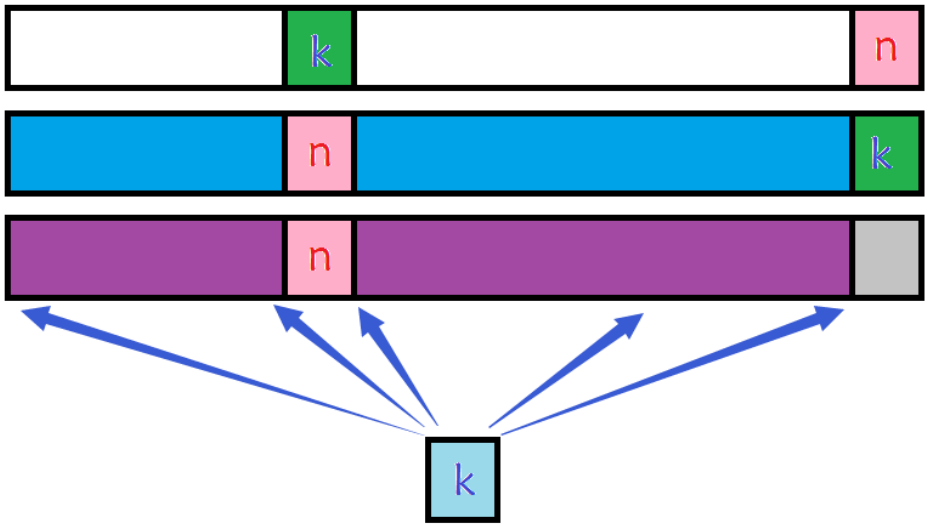
- 赛诗会后，十二金钗待奔前程。分别宴上，12人各写一首诗放入宝囊。大家任取，若取到自己的诗，则再取一首并放回自己的诗。12人都拿到别人的诗作算一种分配。问：共有多少种不同的分配？

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

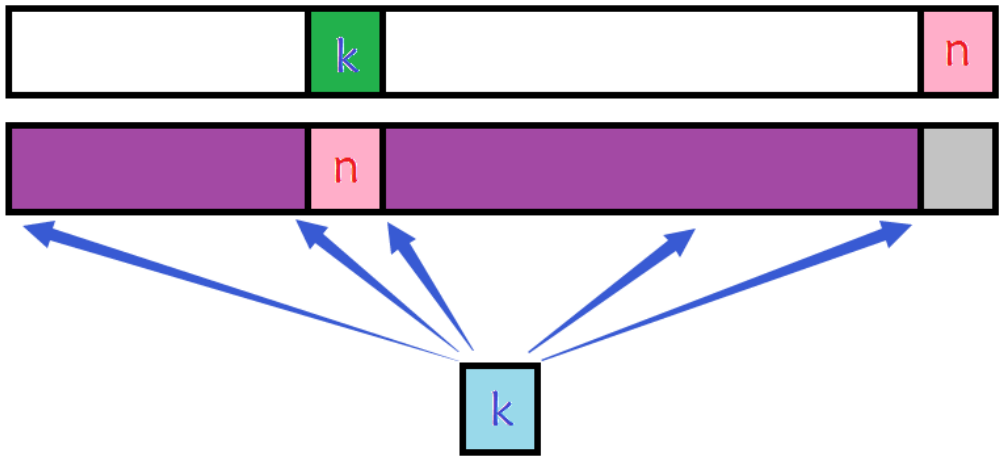


数字k放置在第n位

- 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：

$$S(n)=P_n^n-C_n^1P_{n-1}^{n-1}+-C_n^2P_{n-2}^{n-2}+\cdots+(-1)^n-C_n^nP_0^0$$

- 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,…,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
  - 大数据分析挖掘

感谢大家！

恳请大家批评指正！