装备套装问题

问题：

某套装A由n件装备组成：

$$A=\{a_1,a_2\ldots a_n\}$$

每件装备a_i都由某BOSS掉落，且每次只掉落一件（掉落事件互斥），每件装备的掉落概率由概率向量p给出：

$$P=(p_1,p_2\ldots p_n)，其中\sum_{k=1}^{n}p_k\leq 1$$

为收集齐此套装的n件装备，平均需要杀多少次BOSS？为具体起见，设

$$P=(0.05,0.1,0.15,0.2,0.25)$$

解答：

方法一、循环模拟

模拟收集次数为 1000000 次。

```
clc;clear all;format long;
n=input('输入套装数目： ');
P=zeros(1,n);
for i=1:n
    fprintf('输入第%g件装备掉落的概率： ',i);
    P(i)=input('');
end
P_sum=zeros(1,n+1);
for i=1:n
    P_sum(i+1)=sum(P(1:i));
end
Time=1000000;
Total_pick=0;
for i=1:Time
    select=zeros(1,n);
    N=0;
    while sum(select)<n
       x=rand();
      N=N+1;
      for i=1:n
         if x>P_sum(i) && x<=P_sum(i+1)
              select(i)=1;
         end
      end
    end
    Total_pick=Total_pick+N;
end
fprintf('循环模拟方法结果： %f次。\n',Total_pick/Time);
```

运行结果：

输入套装数目： 5
输入第1件装备掉落的概率： 0.05
输入第2件装备掉落的概率： 0.10
输入第3件装备掉落的概率： 0.15
输入第4件装备掉落的概率： 0.20
输入第5件装备掉落的概率： 0.25
循环模拟方法结果： 24.903420次。

方法二、递归方法

设要求的次数期望为$F(A)$。根据第一次的结果分成以下 2 种可能：

1、 第一次收集到装备a_i，那么还需收集剩下的$A-\{a_i\}$，之后进行的收集次数为

$$p_i \cdot F(A-\{a_i\})$$

2、第一次没有收集到任何装备（若$\sum_{k=1}^{n} p_k < 1$），那么之后还需进行的收集次数为

$$\left(1-\sum_{k=1}^{n} p_k\right) \cdot F(A)$$

根据数学期望的线性特性，有：

$$F(A) = 1 + \sum_{i=1}^{n} [p_i \cdot F(A-\{a_i\})] + \left(1-\sum_{k=1}^{n} p_k\right) \cdot F(A)$$

从而：

$$F(A) = \frac{1+\sum_{i=1}^{n} [p_i \cdot F(A-\{a_i\})]}{\sum_{k=1}^{n} p_k}$$

$$F(\{a_i\}) = \frac{1}{p_i}$$

上面的分析过程有很明显的递归性质，因此可以编写递归函数进行求解。在编写递归程序之前首先要实现$A-\{a_i\}$，即从向量P中删除第i个元素，剩下的元素组成新向量，为此编写一个函数如下：

```
function vector=Left(P,n)
% 从向量P中删除第i个元素，剩下的元素组成新向量赋值给vector
P_size=size(P);
vector=[];
j=1;
for i=1:P_size(2)
    if i==n
        i=i+1;
    else
        vector(j)=P(i);
        j=j+1;
    end
end
```

然后编写递归函数如下：

```
function select_time=Suit(P)
size_P=size(P);
if size_P(2)==1
    select_time=1/ P;
else
    select_time=1/(sum(P));
    for i=1:size_P(2)
        select_time=select_time+ P(i)*Suit(Left(P,i))/(sum(P));
    end
end
```

运行时在命令窗口直接调用此函数：

```
fprintf('递归方法结果： %f次。\n',Suit(P))
```

结果：

递归方法结果： 24.898046398046397次。

如果套装件数不是很多，完全可以人工计算。

方法三、容斥方法

当$n=2$的时候，

$$F(\{a_1,a_2\})=\frac{1+p_1\cdot F(\{a_2\})+p_2\cdot F(\{a_1\})}{p_1+p_2}=\frac{1+p_1\cdot\frac{1}{p_2}+p_2\cdot\frac{1}{p_1}}{p_1+p_2}=\frac{1}{p_1}+\frac{1}{p_2}-\frac{1}{p_1+p_2}$$

当$n=3$的时候，

$$F(\{a_1,a_2,a_3\})=\frac{1+p_1\cdot F(\{a_2,a_3\})+p_2\cdot F(\{a_1,a_3\})+p_3\cdot F(\{a_1,a_2\})}{p_1+p_2+p_3}$$

$$=\frac{1+p_1\cdot\left(\frac{1}{p_2}+\frac{1}{p_3}-\frac{1}{p_2+p_3}\right)+p_2\cdot\left(\frac{1}{p_1}+\frac{1}{p_3}-\frac{1}{p_1+p_3}\right)+p_3\cdot\left(\frac{1}{p_1}+\frac{1}{p_2}-\frac{1}{p_1+p_2}\right)}{p_1+p_2+p_3}$$

$$=\left(\frac{1}{p_1}+\frac{1}{p_2}+\frac{1}{p_3}\right)-\left(\frac{1}{p_1+p_2}+\frac{1}{p_1+p_3}+\frac{1}{p_2+p_3}\right)+\frac{1}{p_1+p_2+p_3}$$

…………

由数学归纳法可以证明（证明略），当

$$A=\{a_1,a_2\dots a_n\}$$

时，

$$F(A)=\sum_i\frac{1}{p_i}-\sum_{i\neq j}\frac{1}{p_i+p_j}+\sum_{i\neq j\neq k}\frac{1}{p_i+p_j+p_k}+\cdots+\frac{(-1)^{n+1}}{\sum p_i}$$

记$S(P,k)$为集合P的k−子集（元素数为k）的元素之和，那么上面的结果可以写成如下的形式：

$$F(A)=\sum_{k=1}^{n}(-1)^{k+1}\sum\frac{1}{S(P,k)}$$

根据这个结果可以编写新函数：

```
function result=Taozhuang(P)
size_P=size(P);
```

```
result=0;
for i=1:size_P(2)
    result=result+(-1)^(i+1)*sum(1./sum(combntns(P,i)',1));
end
```

其中函数combntns(P,i)的结果是P的所有元素数为i的子集。

运行时在命令窗口直接调用此函数：

```
fprintf('容斥方法结果： %f次。\n', Taozhuang(P))
```

结果：

```
容斥方法结果： 24.898046398046397次。
```

虽然此方法是由第二种方法归纳出的，但是从结果中可以看出明显的容斥原理的影子，这也是把此方法叫作“容斥方法”的原因。

小结：

模拟方法耗时长，资源占用大，结果精度低；递归方法耗时短，资源占用大，结果精度高；容斥方法耗时短，资源占用小，结果精度高。优劣立现。

QQ：707509279