

## 摘 要

2019 春节前夕，支付宝如往年一样举办“集五福”活动，在参与的过程中灵感随之产生，我希望能计算集齐全部 5 种卡片所需次数的期望，和限制抽取次数能够集齐的概率。

即给定  $N$  种卡片，每种卡片都有一定的概率  $p_i$  能被抽到且能被多次抽到。

1. 求恰好集齐时抽取次数的期望。
2. 限制最多抽  $M$  次，能够集齐的概率。

该问题即计算多重集合卡片抽取期望与限制次数下的概率。本文从该问题出发，研究多种解法，包括蒙特卡洛解法、斯特林数解法，动态规划解法与转化为生成函数后的快速傅里叶变换解法，并比较其复杂度与精确度。

**关键词：** 多重集合抽卡概率，蒙特·卡罗方法，状态压缩，概率，动态规划，容斥原理，卡特兰数，快速傅里叶变换



## 目 录

第一章 绪 论 .....	1
1.1 研究问题的背景 .....	1
1.2 研究问题的意义 .....	1
1.3 本文应用的主要算法 .....	1
第二章 问题建模 .....	2
2.1 问题建模 .....	2
2.2 符号表示 .....	2
第三章 蒙特·卡罗解法 .....	3
3.1 蒙特·卡洛方法介绍 .....	3
3.1.1 算法分析 .....	3
3.1.2 实现过程 .....	3
3.2 伪代码 .....	4
第四章 问题一：期望的计算 .....	6
4.1 问题分析 .....	6
4.2 从特殊到一般：卡片概率相同的特殊情况 .....	6
4.3 动态规划 .....	7
4.3.1 状态表示 .....	7
4.3.2 转移方程 .....	7
4.3.3 算法分析 .....	8
4.3.4 简单代码 .....	8
第五章 问题二：限制抽取次数求集齐概率 .....	10
5.1 问题分析 .....	10
5.2 从特殊到一般：卡片概率相同的特殊情况 .....	10
5.3 动态规划解法 .....	11
5.3.1 状态定义 .....	11
5.3.2 状态转移 .....	11
5.3.3 算法分析 .....	11
5.4 生成函数与快速傅里叶变换 .....	11
5.4.1 问题分析 .....	11
5.5 多项式乘法 .....	12

5.6 多项式除法与多项式求逆 .....	12
第六章 总结.....	13
6.1 解法对比.....	13
6.2 总结.....	13



## 第一章 绪论

### 1.1 研究问题的背景

本问题来源于生活。2019 春节前夕，支付宝如往年一样举办“集五福”活动<sup>①</sup>，在参与的过程中，运气极差的我依然没有在规定的次数内集齐五张卡片，于是我萌生了计算抽取成功概率的想法。

问题转化为，以五张卡片的概率为输入变量，求解集齐卡片的抽取次数期望。在对问题的建模与思考后，我想出状态压缩动态规划的解法后，将其作为 2019 年 3 月的南开大学 ACM 校队选拔赛网络赛的题目<sup>②</sup>之一。

但本篇论文不仅仅是题目原题，我在原有问题的基础上进行了修改与条件限制，提出其它解法，并将各算法进行对比。

### 1.2 研究问题的意义

抽卡问题在生活中出现广泛，除了支付宝“集五福”活动，诸多游戏的抽卡、宝箱机制均有类似的场景。

### 1.3 本文应用的主要算法

本文应用的主要算法包括单纯暴力解法、卡特兰数的计算、状态压缩下的动态规划、快速傅里叶变换。

---

① 描述：目标是集齐 5 张福卡：“爱国福”、“富强福”、“友善福”、“和谐福”、“敬业福”。有若干次的抽取机会，每次抽取你都有一定概率抽到五张福卡中的一张或抽空，需要在规定次数内集齐五种卡片。

② 题目链接：<http://acm.nankai.edu.cn/problem/1027>

## 第二章 问题建模

### 2.1 问题建模

给定  $n$  个互不相同的卡片，与数组  $P = \{p_1, p_2, p_3, \dots, p_n\}$ ，进行若干次抽取实验，对于每次实验，卡片  $i$  被抽中的概率均为  $p_i$ 。 $(0 < \sum_{i=1}^{i=n} p_i \leq 1)$

求解：

1. 若每个卡片可被无限次抽取，不限制抽取次数，求能够恰好抽取全部卡片的抽取次数期望。记为问题一。
2. 若每个卡片可被无限次抽取，且限制抽取  $m$  次，求全部卡片均有被命中的概率。记为问题二。

### 2.2 符号表示

符号	描述
$n$	卡片的数量
$p_i$	第 $i$ 个卡片被命中的概率
$m$	限制抽取的次数

符号表示

## 第三章 蒙特·卡罗解法

### 3.1 蒙特·卡洛方法介绍

蒙特卡罗方法也称统计模拟方法，是 1940 年代中期由于科学技术的发展和电子计算机的发明，而提出的一种以概率统计理论为指导的数值计算方法。是指使用随机数（或更常见的伪随机数）来解决很多计算问题的方法。

20 世纪 40 年代，在冯·诺伊曼，斯塔尼斯拉夫·乌拉姆和尼古拉斯·梅特罗波利斯在洛斯阿拉莫斯国家实验室为核武器计划工作时，发明了蒙特卡罗方法。因为乌拉姆的叔叔经常在摩纳哥的蒙特卡洛赌场输钱得名，而蒙特卡罗方法正是以概率为基础的方法。

#### 3.1.1 算法分析

蒙特·卡洛算法利用计算机生成一个随机数的特性，模拟实际抽取卡片的随机过程。对于每次实验，计算机都需要模拟出本次命中的是哪个卡片，直到满足结束条件。

我们利用蒙特卡洛算法作为研究问题的暴力解法，同时也作为答案的一种可信解法。

#### 3.1.2 实现过程

我们记总实验次数为  $T$ 。

1. 按照概率随机选择一个卡片，该卡片数加一
2. 重复过程一，直到满足结束条件。
3. 进行下一次实验

按照概率的选择卡片的方法可优化至  $O(\log N)$ ：

将卡片对应的概率分布映射到  $[0,1]$  的一维区间上，通过生成  $[0,1]$  的随机数，从而确定对应的卡片。记第  $i$  张卡片区间为  $[\sum_{k=1}^{i-1} p_k, \sum_{k=1}^i p_k]$

生成一个随机数  $x$  后，利用二分查找可以得到随机数  $x$  对应的区间，复杂度  $O(\log N)$ 。

复杂度为  $O(TE \log N)$ ，其中  $E$  为抽取次数期望。

对这个问题来说，蒙特卡洛算法实际上是一种暴力解法，不妨以此解法的答案作为下列解法正确性的标准。



### 3.2 伪代码

对于问题一，计算期望的蒙特卡洛伪代码如下：

```
initialization;
while iteration < T do
    count=0;
    while Not all the cards were collected do
        count++;
        rand=random[0.0,1.0];
        i=lower_bound(prefix_p,rand) ;
        set card i collected;
    end
    ans+=count/T;
end
```

算法 3-1 问题一：蒙特卡洛算法流程

预处理时，用 *prefix\_p* 数组存储卡片概率的前缀和，以便在生成随机数进行二分。在判断卡片是否集齐时，可采用将卡片状态转换为一个  $N$  位二进制数，若第  $i$  卡片抽到了，则将二进制的第  $i$  位置为 1，如此可  $O(1)$  判断卡片是否集齐。

对于问题二，限制抽取次数小于等于  $M$  时集齐的概率，伪代码如下：

```

initialization;
while  $iteration < T$  do
    count=0;
    while  $count \leq M$  and Not all the cards were collected do
        count++;
        rand=random[0.0,1.0];
        i=lower_bound(prefix_p,rand) ;
        set card i collected;
    end
    if  $count \leq M$  then
        Got++;
    else
        Nothing;
    end
end
ans=Got/T;

```

算法 3-2 问题二：蒙特卡洛算法流程

## 第四章 问题一：期望的计算

### 4.1 问题分析

定义刚好抽到  $N$  张卡片的次数是  $X$ ，设其为事件  $A$ ，则  $A$  满足前  $X-1$  次抽取到的卡片属于某  $N-1$  种卡片的集合，而第  $X$  次刚好抽到剩下的第  $N$  种卡片。

设  $P_X$  为  $X$  次恰好集齐的概率，则期望为：

$$E(X) = \sum_{X=1}^{\infty} X * P_X$$

根据上述期望公式，若不进行蒙特卡洛随机模拟，似乎必须将  $X$  的所有情况全部计算一遍，才能得到一个准确的答案。但这里可以凭借期望可累加的性质和状态转移的思想，利用动态规划求解。

### 4.2 从特殊到一般：卡片概率相同的特殊情况

先考虑一种比较特殊的情况，即卡片概率均相同的特殊情况，即每张卡片的抽取概率都是  $\frac{1}{n}$ 。

接下来存储每个状态，记状态  $s(x)$  为当前已经抽取了  $x$  种卡片，记  $dp[x]$  为当前处于状态  $s(x)$  时，抽完剩下  $n-x$  种卡片所需要次数的期望。

在当前已抽取了  $x$  张卡片的情况下，下一次抽取到的是当前已抽取的卡片种类的事件记为  $A$ ，下一次抽取为剩下的  $n-x$  种张卡片的事件记为  $B$ ，则

$$\begin{aligned} P(A) &= \frac{x}{n} \\ P(B) &= \frac{n-x}{n} \end{aligned}$$

则本次实验服从几何分布<sup>①</sup>，根据几何分布的期望计算公式，可知  $s(x)$  转移至  $s(x+1)$  状态的期望次数为  $\frac{n}{n-x}$ ，所以状态转移公式为：

$$dp[x] = dp[x+1] + \frac{n}{n-x}$$

那么  $dp[0]$  即为所求，即当前是没有任何卡片的状态时，集齐剩下  $n$  种卡片的

① 几何分布：在伯努利试验中，记每次试验中事件  $A$  发生的概率为  $p$ ，试验进行到事件  $A$  出现时停止，此时所进行的试验次数为  $X$ ，期望  $E(X) = \frac{1}{p}$

期望次数。

我们将该状态转移公式逆推：

$$\begin{aligned} ans &= \frac{n}{n} + \frac{n}{n-1} + \frac{n}{n-2} + \dots + n \\ &= n(1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}) \end{aligned}$$

所以该特例情况，答案可用  $O(N)$  复杂度计算出来。同时，计算级数

$$\lim_{k \rightarrow +\infty} \sum_{k=1}^N \frac{1}{k} = \ln N + C$$

$C$  为欧拉常数，于是我们可知问题的答案近似于  $n \ln n$ 。

本问题的题目与数据已上传至 OJ<sup>①</sup>，原创题目。

## 4.3 动态规划

将上述的特殊情况推广到一般情况，即每种卡片的概率不一定相同。这个时候，就不能简单地将状态划分为  $n$  种，事实上，由于每种概率的不同，我们需要划分为  $2^n$  个状态。

### 4.3.1 状态表示

状态共有  $2^n$  种状态，为了方便记录，不妨将每个状态映射为一个  $n$  位二进制。

状态  $S(x)$ ：其中  $x$  为一个  $n$  位二进制，若  $x$  的第  $i$  位为 1，则该状态下第  $i$  种卡片已被抽到，否则还未被收集。

这样记录的空间复杂度为  $O(2^N)$ 。

### 4.3.2 转移方程

我们对  $dp[x]$  做相同的定义，即状态为  $S(x)$  时，将剩下卡片集齐的期望次数。

转移很类似上述的特殊情况，只不过状态  $S(x)$  被转移的对象不仅来源于一个状态，而是多个状态。

假设当前有 3 张卡片， $x = 001$ ，即当前状态下第 1 张卡片已经收集（假设从 1 开始标号）。那么  $S(001)$  将从  $S(101)$  和  $S(011)$  两个状态转移过来，记  $P_x$  为  $S(x)$  所代表的状态下没集齐的卡片概率和，比如  $P(001)$  为第 2、3 种卡片的概率和。根据特殊问题分析的几何分布情况，能抽到新卡的期望次数为  $\frac{p_1 + p_2 + p_3}{p_2 + p_3}$ ，那么在状态

① 题目链接：<http://acm.nankai.edu.cn/problem/1069>

001 能抽到新卡的情况下，能变为状态 101 的概率为  $\frac{p_3}{p_2+p_3}$ ，能变为状态 011 的概率为  $\frac{p_2}{p_2+p_3}$ 。

于是可知状态转移公式为：

$$dp[x] = \sum_{i=1}^n \frac{p[i]}{P_x} dp[1 \ll i | x] + \frac{1}{P_x} \text{ if } 1 \ll i | x \neq x$$

$1 \ll i | x \neq x$  的意思是，状态  $x$  能被转移的状态所代表的二进制一定比  $x$  多一个 1。

### 4.3.3 算法分析

动态规划与蒙特卡洛解法的不同在于，动态规划解法可以求出问题的一个准确值，不会因为模拟次数影响结果精度。

其时间复杂度为  $O(N * 2^N)$ ，空间复杂度  $O(2^N)$ ，虽然时间复杂度为指数型，但实际很难有一个多项式的解法求得该问题的准确答案，状态的复杂性很难避免的。

本问题的题目与数据已上传至 OJ<sup>①</sup>，原创题目。

### 4.3.4 简单代码

```

1      #include <bits/stdc++.h>
2
3      using namespace std;
4
5      const int N = 25 + 2;
6      double dp[1 << N], p[N];
7
8      int main() {
9          int n;
10         cin >> n;
11         for (int i = 0; i < n; ++i) cin >> p[i];
12         for (int s = (1 << n) - 2; s >= 0; --s) {
13             double P = 0;
14             for (int i = 0; i < n; ++i) {
15                 if ((s | (1 << i)) != s) {
16                     P += p[i];
17                     dp[s] += p[i] * dp[s | (1 << i)];
18                 }
19             }
20         }
21     }

```

① 题目链接：<http://acm.nankai.edu.cn/problem/1070>

```
17         dp[s] = dp[s] / P + 1.0 / P;  
19     }  
20     cout<<dp[0]<<endl;  
21     return 0;  
22 }
```

## 第五章 问题二：限制抽取次数求集齐概率

### 5.1 问题分析

在问题一的基础上，我们将抽取次数从无限次限制为  $M$  次，并求在此情况下的能集齐全部种类卡片的概率。

在限制抽取次数时，问题从无限域转化到有限域，下面讨论该问题的不同解法，和如何通过快速傅里叶变化优化复杂度。

### 5.2 从特殊到一般：卡片概率相同的特殊情况

我们还是先从每张卡片的概率均相同的情况讨论起，即每张卡片的抽取概率都是  $\frac{1}{n}$ 。

我们不妨枚举恰好集齐所需的次数  $K$ ，则  $K$  一定满足  $n \leq K \leq m$ 。

一定存在一种卡片，它只被抽取过 1 次，它就是第  $K$  次所抽到的卡片种类。那么前  $K-1$  次抽取一定在  $n-1$  种卡片中，且  $n-1$  种卡片均有覆盖。

不难发现，问题转换为一个放球模型，即将  $K-1$  个不同的球放入  $n-1$  不同的盒子中。根据第二类斯特数<sup>①</sup>计算公式：

$$S(i, j) = S(i-1, j-1) + j * S(i-1, j)$$

由于这里计算的是放在不同盒子的情况，需要将计算得的斯特林数乘上盒子数的阶乘，即：

$$C(i, j) = S(i, j) * j!$$

而最后一次抽到卡片一共存在  $n$  种情况，所以当集齐次数为  $K$  时，该情况的概率为  $A(k) = n * C(k-1, n-1) * \frac{1}{n^k}$

所以在每张卡片概率都相同的情况下，答案为

$$ans = \sum_{k=n}^m A(k)$$

① 第二类斯特林数：把  $n$  个不同的小球放在  $m$  个相同的盒子里方案数。

时间复杂度和空间复杂度均为  $O(NM)$

本问题的题目与数据已上传至 OJ<sup>①</sup>，原创题目。

## 5.3 动态规划解法

### 5.3.1 状态定义

卡片已抽取的状态采用与问题一相同的方式，用  $n$  位二进制表示。但问题二因为限制了抽取次数，所以此时我们需要增加一个维度，表示当前状态已经抽取的次数。

定义  $dp[x][i]$  为，当前的抽取状态为  $x$ ，并进行了  $i$  次抽取时达到此状态的概率， $x$  是二进制表示哪些卡片被抽到。

### 5.3.2 状态转移

令  $Q_x$  为卡片的状态为  $x$  时，没有在卡片集合的概率和（包括抽空的概率）。

比如  $n = 3, m = 5, p = [0.1, 0.2, 0.3]$ ，那么  $Q_{101}$  为  $p_2$  和抽空的总和，即  $0.2 + 0.4 = 0.6$ 。

假设若计算  $dp[101][3]$ ，那么它一定可以从  $dp[101][2], dp[001][2], dp[100][2]$  三种状态转移过来，三种状态转移的参数分别为  $Q_x, p_3, p_1$ 。

初始状态  $dp[0][0] = 1$

状态转移公式为：

$$dp[x][i] = dp[x][i-1] * Q_x + \left( \sum_{j=1}^n dp[x \setminus j][i-1] * p[j] \text{ if } j \in s \right)$$

其中， $x \setminus j$  代表将二进制  $x$  去掉第  $j$  位的 1。

### 5.3.3 算法分析

该算法的空间复杂度为  $O(M * 2^N)$ ，时间复杂度为  $O(NM * 2^N)$ 。

本问题的题目与数据已上传至 OJ<sup>②</sup>，原创题目。

## 5.4 生成函数与快速傅里叶变换

### 5.4.1 问题分析

根据离散数学的知识，我们可以将每个卡片作为母函数展开，假设第  $i$  张卡片所对应的生成函数为  $F_i(x)$ ，则

① 题目链接：<http://acm.nankai.edu.cn/problem/1069>

② 题目链接：<http://acm.nankai.edu.cn/problem/1070>



$$\begin{aligned}
 F_1(x) &= p_1x + \frac{p_1^2}{2!}x^2 + \frac{p_1^3}{3!}x^3 + \frac{p_1^4}{4!}x^4 \dots + \frac{p_1^m}{m!}x^m \\
 F_2(x) &= p_2x + \frac{p_2^2}{2!}x^2 + \frac{p_2^3}{3!}x^3 + \frac{p_2^4}{4!}x^4 \dots + \frac{p_2^m}{m!}x^m \\
 F_3(x) &= p_3x + \frac{p_3^2}{2!}x^2 + \frac{p_3^3}{3!}x^3 + \frac{p_3^4}{4!}x^4 \dots + \frac{p_3^m}{m!}x^m \\
 &\dots \\
 F_n(x) &= p_nx + \frac{p_n^2}{2!}x^2 + \frac{p_n^3}{3!}x^3 + \frac{p_n^4}{4!}x^4 \dots + \frac{p_n^m}{m!}x^m
 \end{aligned}$$

将上面的  $n$  个生成函数相乘并展开、合并同类项，得到的  $x^k$  项所对应的系数就是当抽取次数为  $k$  时，卡片所有组合情况的概率总和。这里就体现了每项系数均除以一个阶乘的用处，即去重。

## 5.5 多项式乘法

考虑到题目的特殊性，即最后一张卡片随对应的种类仅可能被抽一次。所以我们不妨枚举最后抽到的卡片种类，将剩余多项式相乘，然后计算  $x^n - x^m$  每个项系数和，再将每次枚举的系数和再求和。

所得的总和即为答案。

通过快速傅里叶变换，每两个多项式相乘的复杂度为  $(M \log M)$ ，在两个多项式相乘后，可以将  $x^m$  以上的项舍弃掉，使得复杂度不会增加。而对  $n - 1$  个多项式连乘的复杂度为  $O(NM \log M)$ ，最外层枚举  $n$  种最后抽到的卡片，所以总的时空复杂度为  $O(N^2 M \log M)$ 。

空间复杂度为  $O(M)$ 。

## 5.6 多项式除法与多项式求逆

我们发现，枚举最后一次抽到的卡片种类会使得复杂度多乘一个  $N$ 。但实际上，我们发现，每次枚举后都会得到  $n - 1$  个多项式相乘的结果，所不同的不过是少乘的该多项式。

所以，我们不妨将  $n$  个多项式全部乘起来，结果为  $G(x)$ 。枚举最后一个抽到的卡片种类  $i$ 。此时，需要将  $G(x)$  除以  $F_i(x)$ ，由于  $F_i(x)$  是  $G(x)$  的因子，所以一定能被整除。可以通过乘  $F_i(x)$  的逆多项式，从而算得多项式的商。

时间复杂度优化为  $O(NM \log N)$

## 第六章 总结

经过在 OJ 上传题目后的验证，解法均可通过。

### 6.1 解法对比

对于问题一，复杂度比较如下

表 6-1 问题一复杂度比较

算法	时间复杂度	空间复杂度	精确性
蒙特卡洛	$O(TE\log N)$	$O(N)$	受迭代次数 $T$ 的影响
动态规划（概率相同的情况）	$O(N)$	$O(N)$	精确结果
动态规划（一般情况）	$O(N2^N)$	$O(2^N)$	精确结果

对于问题二，复杂度比较如下

表 6-2 问题二复杂度比较

算法	时间复杂度	空间复杂度	精确性
蒙特卡洛	$O(TM\log N)$	$O(N)$	受迭代次数 $T$ 的影响
第二类斯特林数（概率相同的情况）	$O(NM)$	$O(NM)$	精确结果
动态规划	$O(NM2^N)$	$O(M2^N)$	精确结果
生成函数多项式乘法	$O(N^2 M \log M)$	$O(M)$	精确结果
生成函数多项式求逆	$O(NM \log M)$	$O(M)$	精确结果

### 6.2 总结

本文从生活实际出发，从现实中对一个问题的思考，引申出解决问题的不同方法，逐步优化多重集合抽卡问题的算法复杂度。

该问题具有拓展性，如每种卡片限制个数，也可用类似的解法求取；同时具有实用性，对游戏主办方卡片概率制定的选择具有参考意义，对游戏玩家方也具有指导意义。