The Research on the Derangements Problem

Team Member

Qian Xinqi Yu Haojia

Teacher

Ji Jianfeng

School

Suzhou NO. 10 High School of Jiangsu Province

Abstract：

In a math class, we studied a problem of receiving postcards from others: The simple case starts with 4 people in one dorm. Everyone writes a postcard and each person chooses one card from other people. The question is how many ways of arrangements are there? We find the answer by using two different methods. We extend the problem to the case of n people, and find our own solution: recurrence. As we do more research on it, we find that this problem is one of the most famous 100 math problems: the problem of derangements. After examing all the solutions provided to the problem, we find that none of them gives a simplified answer. The goal of our research is to simplify the final solution, and we find that we can use the bracket function to solve the problem.

Keywords: **Permutations and Combinations**, **Maclaurin's formula**, **Rounding Function**, **Triangle**.

1.Introduction：

Here is a problem：Four people in the same dorm wrote 4 different New Year cards. Collected them first and then let each person pick one written by others. Find all the number of possible ways.

This is the famous “Derangements Problem ” also called “The Bernoulli-Euler Problem of the Misaddressed Letters”. The problem is equivalent to this: There are four numbers 1,2,3,4. All the numbers are not in the original location after rearranging.

There are only 4 numbers so we think that enumeration should be a good idea. we can work out the problem through the following table：

1	2	3	4

2	1	4	3
2	3	4	1
2	4	1	3
3	1	4	2
3	4	1	2
3	4	2	1
4	1	2	3
4	3	1	2
4	3	2	1

We can get the answer “9 ways” easily by enumeration.

This problem is not too difficult, but what about n numbers: How many possible ways that all the n elements of a sequence are not in their original locations after rearranging.

When facing this kind of problems, we come up with the idea of including-excluding principle first. So we try to work out the problem by using this principle.

2.Work out the problem according to including-excluding principle：

The amount of the ways to rearrange the sequence is: $N = n! = A_n^n$

Get rid of the situations that one element is still in its original location: $m_1 = C_n^1(n-1)! = A_n^{n-1}$

Add the situations that two elements are still in their original locations: $m_2 = C_n^2(n-2)! = A_n^{n-2}$

Get rid of the situations that three elements are still in their original locations: $m_3 = C_n^3(n-3)! = A_n^{n-3}$

……

So the last situation is that all elements are in their original locations: $m_n = C_n^n(n-n)!= A_n^{n-n} = A_n^0$

According to including-excluding principle, for a sequence of n elements, the amount of ways to rearrange a sequence and make each element in a new location is $N - m_1 + m_2 - m_3 + \ldots + (-1)^n m_n$

That is $A_n^n - A_n^{n-1} + A_n^{n-2} - A_n^{n-3} + \ldots + (-1)^n A_n^0, (n \geq 2)$

That is $A_n^{n-2} - A_n^{n-3} + A_n^{n-4} - A_n^{n-5} + \ldots + (-1)^n A_n^0, (n \geq 2)$

That is $n!\left(\frac{(-1)^2}{2!} + \frac{(-1)^3}{3!} + \frac{(-1)^4}{4!} + \ldots + \frac{(-1)^{n-1}}{(n-1)!} + \frac{(-1)^n}{n!}\right), (n \geq 2)$

Use the Principle of Fixed Point to explain the solution of the problem:

Suppose that A1 means the set of all the sequences which element No. 1 is in the original location and A2 means the set of all the sequences which element No. 2 is in the original location. The rest may be deduced by analogy. So the answer to the original problem is equal to the number of objects of the complementary set of the intersection of the sets from A1 to An.

During this process, we found progression could be used to solve this problem. The following is how we get the recurrence formula of the progression.

3. Find the recurrence formula of the sequence:

First, suppose a_n means the amount of ways to deranging n elements

Now we discuss element No. 1. The following is two cases.

Case 1: Suppose element No. 1 and No. r exchange after rearrangement. Since element No. r is chosen randomly, there are (n-1) ways.

The rest (n-2) numbers derange, which has nothing to do with element No. 1 and No. r, so there are a_{n-2} ways

$\therefore$ In this case, we have $m_1 = (n-1)a_{n-2}$ ways

Case 2: Suppose element No. 1 does not exchange with any other elements. To deal with this situation, we have two solutions:

Solution 1:

We put element No. 1 in a new location naming location No. (n+1) and choose one element from element No. 2 to No. n to fill in location No. 1. There are (n-1) ways. Suppose the element we choose is No. k ($k \neq 1$)

We put (n-1) elements ranging from No. 2 to No. (n+1) in location ranging from No. 2 to No. n. Notice that element No. 1 could not be put in location No. k, which equals to the derangement of (n-1) elements. The

number of ways is a_{n-1}

$\therefore$ In this case, we have $m_2 = (n-1)a_{n-1}$ ways

Solution 2:

First, we derange n-1 elements ranging from No.2 to No. n. There are a_{n-1} ways.

Secondly, we pick up one element form the elements above. There are (n-1) ways.

Suppose the element we chose is No. s locating in No. k after derangements, and $k \neq s \neq 1$. Then we exchange the location of element No.1 and element No. s, which means element No. s locates in No.1 and element No.1 locates in No. k rather than No. s ($k \neq s \neq 1$). As a result, No.1 does not exchange the location with any other elements and all possible situations have been considered.

$\therefore$ In this cases, we have $m_2 = (n-1)a_{n-1}$ ways

Based on the above two cases, we get the recurrence formula of the progression: $a_n = (n-1)(a_{n-1} + a_{n-2})$

Then we start to explore the formula for general term.

4. Explore the formula for general term from the recurrence formula:

In the sequence, we get $a_1 = 0$, $a_2 = 1$, $a_3 = 2$, $a_4 = 9$

So the question is :It is known that $a_n = (n-1)(a_{n-1} + a_{n-2})$, $(n \geq 3)$ and $a_1 = 0$, $a_2 = 1$, explore the formula for general term $\therefore a_n - na_{n-1} = -[a_{n-1} - (n-1)a_{n-2}]$

Suppose $b_n = a_n - na_{n-1}$

$\therefore b_n = -b_{n-1}$, $b_2 = 1 \neq 0$

$\therefore b_n = (-1)^n, (n \geq 2)$

$\therefore a_n - na_{n-1} = (-1)^n, (n \geq 2)$

(1) Method 1:

$$\therefore \frac{a_n}{(-1)^n} + n\frac{a_{n-1}}{(-1)^{n-1}} = 1, (n \geq 2)$$

Suppose $c_n = \frac{a_n}{(-1)^n}, (n \geq 1)$

Then $c_n = 1 - nc_{n-1}, (n \geq 2), c_1 = 0, c_2 = 1, c_3 = -2$

$$\therefore c_n = 1 - nc_{n-1}$$

$$= 1 - n\left[1 - (n-1)c_{n-2}\right]$$
$$= 1 - n + n(n-1)\left[1 - (n-2)c_{n-3}\right]$$
$$= 1 - n + n(n-1) - n(n-1)(n-2)\left[1 - (n-3)c_{n-4}\right]$$

.....

$$= \begin{array}{l} 1 - n + n(n-1) - n(n-1)(n-2) + \cdots \\ + (-1)^{n-3} n(n-1)(n-2)\cdots\left[n - (n-4)\right]\left[1 - (n - (n-3))c_{n-(n-2)}\right] \end{array}$$

$$= (-1)^0 1 + (-1)^1 n + (-1)^2 n(n-1) + \cdots + (-1)^{n-2} n(n-1)(n-2)\cdots * 4 * 3 * c_2$$

That is $c_n = (-1)^0 \frac{n!}{n!} + (-1)^1 \frac{n!}{(n-1)!} + (-1)^2 \frac{n!}{(n-2)!} + \cdots + (-1)^{n-2} \frac{n!}{2!}, (n \geq 2)$

$$\therefore a_n = (-1)^n \left[(-1)^0 \frac{n!}{n!} + (-1)^1 \frac{n!}{(n-1)!} + (-1)^2 \frac{n!}{(n-2)!} + \cdots + (-1)^{n-2} \frac{n!}{2!} \right], (n \geq 2)$$

That is $a_n = (-1)^n \left(A_n^0 - A_n^1 + A_n^2 - \ldots\ldots + (-1)^{n-2} A_n^{n-2} \right), (n \geq 2)$

That is $a_n = A_n^{n-2} - A_n^{n-3} + A_n^{n-4} - A_n^{n-5} + \ldots + (-1)^n A_n^0, (n \geq 2)$

That is $a_n = n! \left(\frac{(-1)^2}{2!} + \frac{(-1)^3}{3!} + \frac{(-1)^4}{4!} + \ldots + \frac{(-1)^{n-1}}{(n-1)!} + \frac{(-1)^n}{n!} \right), (n \geq 2)$

(2) Method 2:

$$\because a_n - na_{n-1} = (-1)^n, (n \geq 2)$$

$$\therefore \frac{a_n}{n!} - \frac{a_{n-1}}{(n-1)!} = \frac{(-1)^n}{n!}, (n \geq 2)$$

$$\frac{a_{n-1}}{(n-1)!} - \frac{a_{n-2}}{(n-2)!} = \frac{(-1)^{n-1}}{(n-1)!}$$

.....

$$\frac{a_4}{4!} - \frac{a_3}{3!} = \frac{(-1)^4}{4!}$$

$$\frac{a_3}{3!} - \frac{a_2}{2!} = \frac{(-1)^3}{3!}$$

$$\therefore \frac{a_n}{n!}-\frac{a_2}{2!}=\frac{(-1)^n}{n!}+\frac{(-1)^{n-1}}{(n-1)!}+\frac{(-1)^4}{4!}+\frac{(-1)^3}{3!},(n\geq 2)$$

$$\therefore a_n=n!\left(\frac{(-1)^2}{2!}+\frac{(-1)^3}{3!}+\frac{(-1)^4}{4!}+\ldots+\frac{(-1)^{n-1}}{(n-1)!}+\frac{(-1)^n}{n!}\right),(n\geq 2)$$

In order to make sure the exactitude of our conclusion, we use **mathematical induction** to prove it.

(3) Using mathematical induction to prove the formula for general term.

Since $a_n=(n-1)(a_{n-1}+a_{n-2}),(n\geq 2)$, and $a_2=1,a_3=2$.

To prove: $a_n=n!\left(\frac{(-1)^2}{2!}+\frac{(-1)^3}{3!}+\frac{(-1)^4}{4!}+\ldots+\frac{(-1)^{n-1}}{(n-1)!}+\frac{(-1)^n}{n!}\right)(n\geq 2)$

Proof：

ⅰ. When $n=2$, $a_2=2!\left(\frac{(-1)^2}{2!}\right)=1$ is correct,

When $n=3$, $a_3=3!\left(\frac{(-1)^2}{2!}+\frac{(-1)^3}{3!}\right)=2$ is correct;

ⅱ. If when $n\leq k\ (k\geq 3)$, $a_n=n!\left(\frac{(-1)^2}{2!}+\frac{(-1)^3}{3!}+\frac{(-1)^4}{4!}+\ldots+\frac{(-1)^{n-1}}{(n-1)!}+\frac{(-1)^n}{n!}\right)$ is correct,

that is if $a_k=k!\left(\frac{(-1)^2}{2!}+\frac{(-1)^3}{3!}+\ldots+\frac{(-1)^{k-1}}{(k-1)!}+\frac{(-1)^k}{k!}\right)$ and

$a_{k-1}=(k-1)!\left(\frac{(-1)^2}{2!}+\frac{(-1)^3}{3!}+\ldots+\frac{(-1)^{k-1}}{(k-1)!}\right)$ is correct at the same time.

Then when $n=k+1$, according to $a_n=(n-1)(a_{n-1}+a_{n-2}),(n\geq 2)$

$$a_{k+1}=k(a_k+a_{k-1})$$

considering $\frac{a_{k+1}}{(k+1)!}=\frac{ka_k}{(k+1)!}+\frac{ka_{k-1}}{(k+1)!}$

that is $\frac{a_{k+1}}{(k+1)!}=\frac{k\left(\frac{(-1)^2}{2!}+\ldots+\frac{(-1)^{k-1}}{(k-1)!}+\frac{(-1)^k}{k!}\right)}{(k+1)}+\frac{\left(\frac{(-1)^2}{2!}+\ldots+\frac{(-1)^{k-1}}{(k-1)!}\right)}{(k+1)}$

that is $\frac{a_{k+1}}{(k+1)!}=\left(\frac{(-1)^2}{2!}+\ldots+\frac{(-1)^{k-1}}{(k-1)!}\right)+\frac{k(-1)^k}{(k+1)k!}$

that is $\frac{a_{k+1}}{(k+1)!}=\frac{(-1)^2}{2!}+\ldots+\frac{(-1)^{k-1}}{(k-1)!}+\frac{(-1)^k}{k!}\left(1-\frac{1}{k+1}\right)$

that is $\frac{a_{k+1}}{(k+1)!}=\frac{(-1)^2}{2!}+\ldots+\frac{(-1)^{k-1}}{(k-1)!}+\frac{(-1)^k}{k!}+\frac{(-1)^{k+1}}{(k+1)!}$

$\therefore a_{k+1}=(k+1)!\left(\frac{(-1)^2}{2!}+\frac{(-1)^3}{3!}+\ldots+\frac{(-1)^k}{k!}+\frac{(-1)^{k+1}}{(k+1)!}\right)$

That is if when $n\le k\ (k\ge 3)$, $a_n=n!\left(\frac{(-1)^2}{2!}+\frac{(-1)^3}{3!}+\frac{(-1)^4}{4!}+\ldots+\frac{(-1)^{n-1}}{(n-1)!}+\frac{(-1)^n}{n!}\right)$ is correct,

then when $n=k+1$, the proposition is still correct

Given ⅰ ⅱ, according to the principle of **mathematical induction,**

We get: $\forall n\ge 2$, $a_n=n!\left(\frac{(-1)^2}{2!}+\frac{(-1)^3}{3!}+\frac{(-1)^4}{4!}+\ldots+\frac{(-1)^{n-1}}{(n-1)!}+\frac{(-1)^n}{n!}\right)$

After we getting the formula for general term, we tried to simplify it.

5. Simplify the formula for general term by Maclaurin's formula:

Thinking about Maclaurin's formula $f(x)=e^x$, let $x=-1$:

$\therefore e^{-1}=\frac{(-1)^2}{2!}+\frac{(-1)^3}{3!}+\frac{(-1)^4}{4!}+\ldots+\frac{(-1)^{n-1}}{(n-1)!}+\frac{(-1)^n}{n!}+\frac{(-1)^{n+1}}{(n+1)!}e^{-\theta},\theta\in(0,1)$

Compared with $a_n=n!\left(\frac{(-1)^2}{2!}+\frac{(-1)^3}{3!}+\frac{(-1)^4}{4!}+\ldots+\frac{(-1)^{n-1}}{(n-1)!}+\frac{(-1)^n}{n!}\right)$

$\therefore a_n=n!\left(e^{-1}-\frac{(-1)^{n+1}}{(n+1)!}e^{-\theta}\right),\theta\in(0,1)$

Then $\frac{n!}{e}=a_n+\frac{(-1)^{n+1}}{(n+1)e^{\theta}},\theta\in(0,1)$

Considering $\left[\frac{n!}{e}+\frac{1}{e}-a_n\right]=\left[\frac{1}{e}+\frac{(-1)^{n+1}}{(n+1)e^{\theta}}\right]$

$\because \theta\in(0,1)$, when $(n\ge 2)$

$$\therefore \frac{1}{e^{\theta}} \in \left(\frac{1}{e},1\right), \frac{1}{n+1} \in \left(0, \frac{1}{3}\right]$$

$$\therefore \frac{1}{(n+1)e^{\theta}} \in \left(0,\frac{1}{3}\right)$$

$$\therefore \frac{(-1)^{n+1}}{(n+1)e^{\theta}} \in \left(-\frac{1}{3},0\right) \cup \left(0,\frac{1}{3}\right)$$

$$\frac{(-1)^{n+1}}{(n+1)e^{\theta}} + \frac{1}{e} \in \left(\frac{1}{e}-\frac{1}{3},\frac{1}{e}\right) \cup \left(\frac{1}{e},\frac{1}{3}+\frac{1}{e}\right) \in (0,1)$$

So $\left[\frac{n!+1}{e} - a_n\right] = \left[\frac{1}{e} + \frac{(-1)^{n+1}}{(n+1)e^{\theta}}\right] = 0$

$\because a_n \in N^*$

$$\therefore a_n = \left[\frac{n!+1}{e}\right], (n \geq 2)$$

And when $n = 1$, the equation is still right.

So $a_n = \left[\frac{n!+1}{e}\right], (\forall n \in N^*)$

6. Euler's study of this problem:

Assume the number of the ways to derange a sequence of n elements as a_n.

If we check the elements of rearranged sequence one by one, the unsuccessful derangements have n cases.

1. If element No. 1 is in location No. 1, the derangements are failed. $b_{(n,1)}$ is referred to the number of this case.

2. If element No. 1 is not in location No. 1, but element No. 2 is in location No. 2, the derangements are failed. $b_{(n,2)}$ is referred to the number of this case.

……

m. If the previous (m-1) elements do not locate in their original location, but element No. m is in location No. m, the derangements are failed. $b_{(n,m)}$ is referred to the number of this case.

……

n. If the previous (n-1) elements do not locate in their original location, but element No. n is in location No.

n, the derangements are failed. $b_{(n,n)}$ is referred to the number of this case.

Besides, the number of the ways to rearrange the sequence with n elements is: $n!$

So the number of derangements is $a_n = n! - \left(b_{(n,1)} + b_{(n,2)} + \cdots + b_{(n,n)}\right)$

Obviously, $b_{(n,1)} = (n-1)!$

Now we discuss the sequence of (n+1) numbers.

Obviously, the number of the ways to rearrange the sequence with (n+1) elements is: $(n+1)!$

$$b_{(n+1,1)} = n!$$

As for $b_{(n+1,2)}$, it equals to the amount of ways to put element No. 2 in the original location without other limitation, which is $n!$, minuses the number of ways to put element No. 2 and another element existing before element No. 2 in their original location.

The number of ways to put element No. 2 and another element No. 1 existing before No. 2 in their original location is $(n-1)!$. In other words, this case can be defined as arranging n numbers randomly while putting element No. 1 in location No. 1.

That is $b_{(n+1,2)} = n! - b_{(n,1)}$

In the same way, $b_{(n+1,3)}$ can be defined as the rearrangement of n elements without considering element No. 3. The answer is equal to the number of ways to arrange without any limitation, which is $n!$, minus the number of ways to putting element No. 1 and No. 2 in their original location, which is the sum of the number of situations where element No. 1 is in location No. 1 and the number of situations where element No. 1 is not in location No. 1, while element No. 2 is in location No. 2, $b_{(n,1)} + b_{(n,2)}$

$$\therefore b_{(n+1,3)} = n! - \left(b_{(n,1)} + b_{(n,2)}\right)$$

That is $b_{(n+1,3)} = b_{(n+1,2)} - b_{(n,2)}$

So $b_{(n+1,m)}$ can be defined as the rearrangement of n elements without considering element No. m. The answer is the number of ways to arrange without any limitation, which is $n!$, minus the number of ways to putting the first (m-1) element in their original location, which is the sum of the number of situations which element No. 1 is in location No. 1 and the number of situations where element No. 1 is not in location No. 1while element No. 2 is in location No. 2,.Adding up until we find element No. (m-1) is in location No. (m-1).

That is: $b_{(n,1)} + b_{(n,2)} + \ldots + b_{(n,m-1)}$

$$\therefore b_{(n+1,m)} = n! - \left(b_{(n,1)} + b_{(n,2)} + \ldots + b_{(n,m-1)}\right)$$

That is $b_{(n+1,m)} = b_{(n+1,m-1)} - b_{(n,m-1)}$

So we can get the following table：

$$b_{(n+1,1)} = n!$$

$$b_{(n+1,2)} = b_{(n+1,1)} - b_{(n,1)}$$

$$b_{(n+1,3)} = b_{(n+1,2)} - b_{(n,2)}$$

……

$$b_{(n+1,m)} = b_{(n+1,m-1)} - b_{(n,m-1)}$$

……

$$b_{(n+1,n+1)} = b_{(n+1,n)} - b_{(n,n)}$$

So $\frac{b_{(n,1)}}{n!} = \frac{(n-1)!}{n!}$

$$\frac{b_{(n,2)}}{n!} = \frac{b_{(n,1)}}{n!} - \frac{b_{(n-1,1)}}{n!} = \frac{1}{n} - \frac{1}{n(n-1)}$$

$$\frac{b_{(n,3)}}{n!} = \frac{b_{(n,2)}}{n!} - \frac{b_{(n-1,2)}}{n!} = \frac{1}{n} - \frac{1}{n(n-1)} - \left(\frac{1}{n(n-1)} - \frac{1}{(n-1)(n-2)} \right)$$

$$= \frac{1}{n} - \frac{2}{n(n-1)} + \frac{1}{(n-1)(n-2)}$$

In the same way, $\frac{b_{(n,4)}}{n!} = \frac{b_{(n,3)}}{n!} - \frac{b_{(n-1,3)}}{n!} = \frac{1}{n} - \frac{3}{n(n-1)} + \frac{3}{(n-1)(n-2)} - \frac{1}{(n-2)(n-3)}$

We find the numbers are in **Pascal's triangle** in the numerator

……

That is $b_{(n,1)} = C_0^0 (-1)^0 (n-1)!$

$$b_{(n,2)} = C_1^0 (-1)^0 (n-1)! + C_1^1 (-1)^1 (n-2)!$$

$$b_{(n,3)} = C_2^0 (-1)^0 (n-1)! + C_2^1 (-1)^1 (n-2)! + C_2^2 (-1)^2 (n-3)!$$

……

$$\therefore\ b_{(n,m)} = \sum_{i=0}^{m-1} C_{m-1}^i (-1)^i (n-i-1)!$$

……

$$b_{(n,n)} = \sum_{i=0}^{n-1} C_{n-1}^i (-1)^i (n-i-1)!$$

Summation：

$$\sum_{i=1}^{n} b_{(n,i)} = \sum_{i=0}^{n-1}\left((-1)^i (n-i-1)! \sum_{k=i}^{n-1} C_k^i\right)$$

$$= \sum_{i=0}^{n-1}\left((-1)^i (n-i-1)! C_n^{i+1}\right)$$

$$= \sum_{i=0}^{n-1}\left((-1)^i (n-i-1)! \frac{n!}{(n-i-1)!(i+1)!}\right)$$

$$= \sum_{i=0}^{n-1}\left((-1)^i \frac{n!}{(i+1)!}\right)$$

$$= n! \sum_{i=0}^{n-1}\left(\frac{(-1)^i}{(i+1)!}\right)$$

and $a_n = n! - \sum_{i=1}^{n} b_{(n,i)}$

$$\therefore\ a_n = n!\left(1 - \sum_{i=0}^{n-1}\frac{(-1)^i}{(i+1)!}\right)$$

$$a_n = n!\left(1 + \sum_{i=1}^{n}\frac{(-1)^i}{i!}\right)$$

That is $a_n = n! \sum_{i=2}^{n}\frac{(-1)^i}{i!}$

7. Generalizations

Generalization 1： Find the amount of methods that only r elements selected from n elements are deranged

It is obviously that this problem can be solved step-by-step：

Step1：Considering which r numbers are deranging: $m_1 = C_n^r$

Step2：The amount of methods that these r elements are deranged but other n-r numbers are still at their original locations: $m_2 = a_r = \left[\frac{r!+1}{e}\right]$

On the basis of the addition principle, the number of ways of this generalization is: $C_n^r\left[\frac{r!+1}{e}\right]$

Generalization 2: Use s elements out of the set to replace the s original elements, and (n-s) elements in the set to do the derangement.

As the s elements are out of the set, when deranging, none of the s elements has the corresponding location, which means that any derangement of these s elements are legal on the condition of derangement. So we just need to guarantee that none of the (n-s) elements, which are in the set originally, is in their original location.

STEP1: To consider which s elements in the set are selected to be replaced.

The number of ways in this step is: $m_1 = C_n^s$

STEP2: To arrange the other (n-s) elements to the location No. 1 to No. n, on the condition of derangement.

To consider this step as an independent problem: To suppose the number of ways of deranging m elements to n locations S_n^m

In order to simplify, let location No. 1 region Ⅰ; location No. 2 to m region Ⅱ; and location No.(m+1) to n region Ⅲ.

Discuss element "1", there are 2 situations:

"1" in region Ⅲ, the number of ways in this step is (n-m)

In this situation, for the left (m-1) elements, region Ⅰ is the same as region Ⅲ when considering the condition of derangement. So, in this step, the number of ways in this step is S_{n-1}^{m-1}

So, the number of ways of situation (1) is $(n-m)S_{n-1}^{m-1}$

"1" in region Ⅱ, The number of ways in this step is (m-1)

Without affecting the point of the problem, we can suppose "1" is in the location No. k

To simplify the problem, we suppose the number of ways of derangement in this step is B_n^m, and the analysis of B_n^m can be divided into 3 situations:

①"k" is in the location No. 1. This situation is the same as the situation that "1" and "k" replace each other, in which case, other elements are not affected. So the number of ways of situation① is S_{n-2}^{m-2}

②"k" is in region Ⅲ, the number of ways in this step is (n-m).

As "k" is fixed in region Ⅲ, the location it takes is the same as location No. 1 for the left (m-2) elements, since these locations are all legal on the condition of derangement. So actually, to some degree, this case is the same as situation①, the number of ways in this step is S_{n-2}^{m-2}. So the total number of ways of situation② is $(n-m)S_{n-2}^{m-2}$

③"k" is in region Ⅱ, the number of ways in this step is (m-2).

As "k" is fixed in region Ⅱ, the location it takes(suppose it takes location No. g) is, to some degree, is be taken by the element "1",because the case is the same as the one that there's no element "k", location No. k, and element "1" takes the location No. g, since "k" has no effect on other (m-1) elements. Because g is arbitrary, the number of ways in this step is B_{n-1}^{m-1}. So the total number of ways of situation③ is $(m-2)B_{n-1}^{m-1}$

$\therefore\ B_n^m = (n-m+1)S_{n-2}^{m-2} + (m-2)B_{n-1}^{m-1}$

To sum up, $S_n^m = (n-m)S_{n-1}^{m-1} + (m-1)B_n^m$

$$S_n^m = (n-m)S_{n-1}^{m-1} + (m-1)(n-m+1)S_{n-2}^{m-2} + (m-1)(m-2)B_{n-1}^{m-1}$$

$$S_n^m = (n-m)S_{n-1}^{m-1} + (m-1)(n-m+1)S_{n-2}^{m-2} + (m-1)(m-2)(n-m+1)S_{n-3}^{m-3} + (m-1)(m-2)(m-3)B_{n-2}^{m-2}$$

$$S_n^m = (n-m+1)\left(A_{m-1}^0 S_{n-1}^{m-1} + A_{m-1}^1 S_{n-2}^{m-2} + A_{m-1}^2 S_{n-3}^{m-3}\right) - A_{m-1}^0 S_{n-1}^{m-1} + A_{m-1}^3 B_{n-2}^{m-2}$$

$$S_n^m = (n-m+1)\left(A_{m-1}^0 S_{n-1}^{m-1} + A_{m-1}^1 S_{n-2}^{m-2} + A_{m-1}^2 S_{n-3}^{m-3} + A_{m-1}^3 S_{n-4}^{m-4}\right) - A_{m-1}^0 S_{n-1}^{m-1} + A_{m-1}^4 B_{n-3}^{m-3}$$

……

By iterating, we can get the following equation:

$$S_n^m = (n-m+1)\sum_{i=1}^{m-1} A_{m-1}^{i-1} S_{n-i}^{m-i} - A_{m-1}^0 S_{n-1}^{m-1} + A_{m-1}^{m-1} B_{n-m+2}^2$$

By the definition, we know: $B_{n-m+2}^2 = n-m+1$, $S_{n-m+1}^1 = n-m+1$

So, $S_{n-1}^{m-1} = (n-m+1)\sum_{i=1}^{m-2} A_{m-2}^{i-1} S_{n-1-i}^{m-1-i} - A_{m-2}^0 S_{n-2}^{m-2} + A_{m-2}^{m-2} B_{n-m+2}^2$

STEP3：Arrange the s elements out of the set to the left s locations, the amount of this step is: $m_3 = s!$

In conclusion, the amount of the generalization 2 is $C_n^s S_n^{n-s} s!$

In this answer, the divisor S_n^{n-s} can be calculated by the recursive algorithm. We can calculate it with the help of compute, when necessary.

8. Application

It is said that one of the lotteries in Jiangsu Province is unfair. That kind of lottery is called “Choose 5 from 11”. So we decided to check that lottery is unfair or not by our research results.

First I want to introduce the rule of that lottery：Buyers choose 5 numbers from 1 to 11 and arrange them in any orders they want. Then obtain corresponding awards according to the corresponding level of the numbers chosen by buyers with the winning sequence.

Different winning sequences can be considered as derangements. In this problem, **Derangement L0** is referred as the situation that all numbers are different in two winning sequences. **Derangement L1** is referred as the situation that one number is same in two winning sequences. The rest may be deduced by analogy. So **Derangement L5** is referred as the situation that five numbers are same in two winning sequences. In order to justify “Choose 5 from 11”, we need compare two results. The first result is the theoretical probability of the above situations. The second is the practical probability coming from different winning sequences by using our

own computer program.

As for the practical probability, we make up the standard sequence: 1,2,3,4,5. Comparing different winning sequences and the standard sequence, we can get the practical probability.

As for the theoretical probability, we get one more application of the derangement problem.

Generalization 3: Suppose E is the universal set with m elements and c_n is a finite sequence with n terms, and $c_n \in$ E, n $<$ m. Now we pick up n elements from E and arrange them randomly. Comparing with the original sequence, the number of new sequences which has Derangement L0 is S_0, the number of new sequences which has Derangement L1 is S_1, …… the number of new sequences which has Derangement Ln is S_n. Please figure out the formula for general term.

Solution：

Supposed m-n=t, and the t elements belong to the set T.

Considering Derangement Ln, there are the following n+1 cases.

1. Pick up no number from T to make a sequence with other numbers.
2. Pick up one number from T to make a sequence with other numbers.
3. Pick up two numbers from T to make a sequence with other numbers.

……

n+1. Pick up n numbers from T to make a sequence with other numbers.

Then discuss how to choose the numbers:

Step1. Choose x locations that deranged from n locations. There are C_n^x ways.

Step2. Choose i elements from set T. There are C_t^i ways.

Step3. Insert the above i elements into the x places. There are A_x^i ways.

Step4. Insert the X elements into the remaining (x-i) places and make sure no element is in its original location. There are H_x^{x-i}① ways.

So, $S_x = \sum_{i=0}^{x} C_n^x C_t^i A_x^i H_x^{x-i}$

① H_x^{x-i} is a new definition we defined in this problem.

Definition： We define H_n^r as the permutation of n elements to r positions of the original sets such that none of the n elements appear in their original position

In the research of H_n^r we found a triangle which looks like **Pascal's triangle**:

1

1 0

1 1 1

1 2 3 2

1 3 7 11 9

1 4 13_{15} 32 53 44

1 5 21 71 181 309 265

Of which $H_n^0 = 1; H_n^r = rH_{n-1}^{r-1} + H_{n-1}^r; H_n^n = \left[\frac{n!+1}{e}\right]$

We will further research the general expressions of H_n^r and the relationship between H_n^r and this triangle.

Despite this we can solve the lottery problem now.

Theoretical value：	S_0=1	P_0 =0.000018037518	Actual value：	P_0=0.000113623452
	S_1=30	P_1=0.000541125541		P_1=0.000568117259
	S_2=430	P_2=0.007756132756		P_2=0.007158277468
	S_3=3560	P_3=0.064213564214		P_3=0.062379275082
	S_4=16665	P_4=0.300595238095		P_4=0.300874900579
	S_5=34754	P_5=0.626875901876		P_5=0.628905806158

When computing the practical probability, we used 10,000 groups of data. The first two group is not very accurate due to the low probability and small sample size. But the error of the next four groups is less than 0.2%. As a conclusion, “Choose 5 from 11” is fair.

Reference

[1]C. Edward Sandifer ,How Euler Did It ,The Mathematical Association of America ,（2007-7-3）
[2] 卢开澄 卢华明 ,组合数学（第 3 版） ,清华大学出版社 ,（2002）
[3] 孙淑玲 许胤龙 ,组合数学引论 ,中国科技大学出版社 ,（1999）

Appendix

1. Use Java to do the verification of calculation derangements using Maclaurin’s formula:

```
import java.io.*;
import java.math.*;
import java.util.*;

public class the_verification_of_calculation_derangements_using_Maclaurin’s_formula
{
    /**
     * @param args
```

```
 */

static final BigDecimal c_1 = BigDecimal.valueOf(-1);//c_1=-1
static final BigDecimal c0 = BigDecimal.valueOf(0); //c0=0
static final BigDecimal c1 = BigDecimal.valueOf(1); //c1=1

public static BigDecimal factorial(BigDecimal n)
{
    BigDecimal x = BigDecimal.valueOf(1);// define x=1 (BigDecimal) to store the answer
    BigDecimal i;
    for(i = BigDecimal.valueOf(1);
            ((i.compareTo(n) == -1)||(i.compareTo(n) == 0));
             i = i.add(c1))
    {
        x = x.multiply(i);// to realize the factorial
    }
    return x;
}

public static BigDecimal finde(BigDecimal precision,int partpre)
{
    BigDecimal x = BigDecimal.valueOf(1);//define x=1 (BigDecimal) to store the answer
    BigDecimal part;   part = BigDecimal.valueOf(1);//define part=1 (BigDecimal)
    BigDecimal i;
    for (i = BigDecimal.valueOf(1);
            ((i.compareTo(precision) == -1)||(i.compareTo(precision) == 0));
             i = i.add(c1))
    {
        part =  part.divide(i,partpre,BigDecimal.ROUND_DOWN);
        x = x.add(part);
    }
    return x;
}

public static BigDecimal  progression(BigDecimal n)
{
    BigDecimal x = BigDecimal.valueOf(0); //define x=0 (BigDecimal) to store the answer
    BigDecimal i;
    BigDecimal part = factorial(n);//part = n!
    for (i = BigDecimal.valueOf(2);
            ((i.compareTo(n) == -1)||(i.compareTo(n) == 0));
             i = i.add(c1))
    {
        part =  part.divide(i); //part = part / i;
        x = x.add(part); //xx += part;
```

```
            part =  part.multiply(c_1); //part = part * (-1);
        }
        return x;
    }

    public static BigDecimal Maclaurin(BigDecimal n,BigDecimal ce)
    {
        BigDecimal a; // define a=1 (BigDecimal) to store the answer
        a = factorial(n); //a = n!;
        a = a.add(c1); //a++;
        a = a.divide(ce,100,BigDecimal.ROUND_DOWN); // a = a/e with the precision of 100
        return a;
    }

    public static void main(String[] args)//program starts from here
    {
        // TODO Auto-generated method stub
        Scanner cin = new Scanner (new BufferedInputStream(System.in));
        BigDecimal a,b;    BigDecimal asubb;
        BigDecimal ce ;
        BigDecimal n;
        int nn = 1;    int nnmax;
        System.out.println("Please enter the upper limit you want to ensure this method to.");
        nnmax = cin.nextInt();
        boolean CF = true;
        int nnp;
        do
        {
            n = BigDecimal.valueOf(nn);
            nnp = nn; //define the precision(int) of a
            BigDecimal np;  np = n; //define the precision(BigDecimal) of a
            nnp *= (nn/500+1);
            if (nn<=300) nnp *= 10;
                    //when nn<=300, let precision be 10*nn*(nn/500+1)
            if ((nn>300)&&(nn<=500)) nnp *= 5;
                    //when nn>300&<=500, let precision be 5*nn*(nn/500+1)
            if ((nn>500)&&(nn<=1000)) nnp *= 2;
                    // when nn>500&<=1000, let precision be 2*nn*(nn/500+1)
            if(nn>=2000) nnp *= 0.8;
                    //when nn>2000, let precision be 0.8*nn*(nn/500+1)
            /*These steps are made to ensure precision nnp is a sutiable function of nn*/

            np = BigDecimal.valueOf(nnp); //np(BigDecimal)=nnp(Int)
            ce = finde(np,nnp); //find e under different precision
```

```
            a = Maclaurin(n,ce); /*Find a[n] by using Maclaurin's formula(here the answer
hasn't been truncated)*/
            b = progression(n); /*Find a[n] by using the general term of the progression*/
            asubb = a.subtract(b);
            /*Find the difference of a and b, if the difference asubb >=0 &<1 then the method
is correct*/

            if ((asubb.compareTo(c1) >= 0)||(asubb.compareTo(c0) < 0))
            {
                CF = false;
                break;
            }/*When the difference is not a decimal, which means a is not equal to b after
being truncated, take the wrong signal.*/
            else
            {
                System.out.print("The answer according to the method of using Maclaurin's
formula is correct until  ");
                System.out.println(nn);
            }
            nn += 1; //check every 1 number
        }while(nn <= nnmax);

        if (CF)
        {
            System.out.print
                ("CONGRATULATIONS!The method of using Maclaurin's formula is right.");
        }
        else
        {
            System.out.println("The method of using Maclaurin's formula is WRONG");
        }
        return ;
    }
}
```

2. Use Java to calculate the number of ways of Generalization 2 with the recursive algorithm:

```
import java.io.*;
import java.math.*;
import java.util.*;
public class Generalization_2
{
    /**
```

```
 * @param args
 */

static final BigInteger c0 = BigInteger.valueOf(0); //c0=0
static final BigInteger c1 = BigInteger.valueOf(1); //c1=1
static final BigInteger c2 = BigInteger.valueOf(2); //c2=2
static final BigInteger c3 = BigInteger.valueOf(3); //c3=3
static final BigInteger c_1 = BigInteger.valueOf(-1); //c_1=-1

public static BigInteger factorial(BigInteger n)
{
    if (n.compareTo(c0) == 0) return c1;
    BigInteger x = BigInteger.valueOf(1);
    BigInteger i;
    for(i = BigInteger.valueOf(1);
            ((i.compareTo(n) == -1)||(i.compareTo(n) == 0));
             i = i.add(c1))
    {
        x = x.multiply(i);
    }
    return x;
}

public static BigInteger C_rn(BigInteger r,BigInteger n)
{
    BigInteger Crn = c1;
    Crn = (factorial(n).divide(factorial(r))).divide(factorial(n.subtract(r)));
    return Crn;
}

public static BigInteger  progression(BigInteger n)
{
    if (n.compareTo(c0) == 0) return c0;
    BigInteger x = BigInteger.valueOf(0);
    BigInteger i;
    BigInteger part = factorial(n);//part = n!

    for (i = BigInteger.valueOf(2);
            ((i.compareTo(n) == -1)||(i.compareTo(n) == 0));
             i = i.add(c1))
    {
        part =  part.divide(i); //part = part / i;
        x = x.add(part); //xx += part;
        part =  part.multiply(c_1); //part = part * (-1);
    }
```

```
        return x;
    }

    public static BigInteger S_mn(BigInteger m,BigInteger n)
    {
        BigInteger Smn = BigInteger.valueOf(1);
        if (m.compareTo(n) == 0) {Smn = progression(n);return Smn;}
        if (m.compareTo(c0) == 0) {Smn = n.subtract(c0);return Smn;}
        if (m.compareTo(c1) == 0) {Smn = n.subtract(c1);return Smn;}
        if (m.compareTo(c2) == 0)
        {
            Smn = (n.subtract(c2)).multiply(n.subtract(c2));
            Smn = Smn.add( n.subtract(c1));
            return Smn;
        }

        BigInteger temps,tempb;
        temps = S_mn((m.subtract(c1)),(n.subtract(c1)));
        temps = temps.multiply(n.subtract(m));
        tempb = B_mn(m,n);
        tempb = tempb.multiply(m.subtract(c1));
        Smn = temps.add(tempb);
        return Smn;
    }
    public static BigInteger B_mn(BigInteger m,BigInteger n)
    {
        BigInteger Bmn = BigInteger.valueOf(1);
        if (m.compareTo(c0) == 0) {Bmn = c0;return Bmn;}
        if (m.compareTo(c1) == 0) {Bmn = c0;return Bmn;}
        if (m.compareTo(c2) == 0) {Bmn = n.subtract(c1);return Bmn;}
        if (m.compareTo(c3) == 0)
        {
            Bmn = (n.subtract(c2)).multiply(n.subtract(c2));
            return Bmn;
        }
        if (m.compareTo(n) == 0)
        {
            Bmn = progression(n.subtract(c1));
            Bmn = Bmn.add(progression(n.subtract(c2)));
            return Bmn;
        }
        BigInteger temps,tempb;
        temps = S_mn((m.subtract(c2)),(n.subtract(c2)));
        temps = temps.multiply(n.subtract(m).add(c1));
        tempb = B_mn((m.subtract(c1)),(n.subtract(c1)));
```

```
        tempb = tempb.multiply(m.subtract(c2));
        Bmn = temps.add(tempb);
        return Bmn;
    }

    public static void main(String[] args)//program starts from here
    {
        // TODO Auto-generated method stub
        Scanner cin = new Scanner (new BufferedInputStream(System.in));
        BigInteger Smn;
        BigInteger Ans;
        System.out.println
            ("Please enter the 's' and 'n' of the problem.Enter '-1''-1' to finish.");
        BigInteger s; s = cin.nextBigInteger();
        BigInteger n; n = cin.nextBigInteger();
        BigInteger m; m = n.subtract(s);
        while (!((n.compareTo(c_1) == 0) && (m.compareTo(c_1) == 0)))
        {
            if (!((n.compareTo(c0) < 0 )
                    ||(n.compareTo(m) < 0)||(m.compareTo(c0) < 0)) )
            {
                Smn = S_mn(m,n);
                Ans = Smn.multiply(C_rn(s,n));
                Ans = Ans.multiply(factorial(s));
                System.out.println(Ans);
            }
            else
            {
                System.out.println("Illegal Input.");
            }
            System.out.println
                ("Please enter the 's' and 'n' of the problem.Enter '-1''-1' to finish.");
            s = cin.nextBigInteger();
            n = cin.nextBigInteger();
            m = n.subtract(s);
        }
        return ;
    }
}
```

3. Use Java to calculate H_n^r triangle in Generalization 3:

```
import java.io.*;
import java.math.*;
import java.util.*;

public class Hrn_triangle {
    /**
     * @param args
     */

    static final BigInteger c0 = BigInteger.valueOf(0); //c0=0
    static final BigInteger c1 = BigInteger.valueOf(1); //c1=1
    static final BigInteger c_1 = BigInteger.valueOf(-1); //c_1=-1

    public static BigInteger factorial(BigInteger n)
    {
        if (n.compareTo(c0) == 0) return c1;
        BigInteger x = BigInteger.valueOf(1);
        BigInteger i;
        for(i = BigInteger.valueOf(1);
                ((i.compareTo(n) == -1)||(i.compareTo(n) == 0));
                 i = i.add(c1))
        {
            x = x.multiply(i);
        }
        return x;
    }

    public static BigInteger  progression(BigInteger n)
    {
        BigInteger x = BigInteger.valueOf(0);
        BigInteger i;
        BigInteger part = factorial(n);//part = n!

        for (i = BigInteger.valueOf(2);
                ((i.compareTo(n) == -1)||(i.compareTo(n) == 0));
                 i = i.add(c1))
        {
            part =  part.divide(i); //part = part / i;
            x = x.add(part); //xx += part;
            part =  part.multiply(c_1); //part = part * (-1);
        }
        return x;
    }
```

```
    public static BigInteger H_rn_tri(BigInteger r,BigInteger n)
    {
        BigInteger Hrn = BigInteger.valueOf(1);
        if (r.compareTo(c0) == 0) {Hrn = c1; return Hrn;}
        if (r.compareTo(n) == 0) {Hrn = progression(n); return Hrn;}
        BigInteger temp;
        temp = H_rn_tri((r.subtract(c1)),(n.subtract(c1)));
        temp = temp.multiply(r);
        Hrn = H_rn_tri(r,(n.subtract(c1))).add(temp);
        return Hrn;
    }

    public static void main(String[] args)//program starts from here
    {
        // TODO Auto-generated method stub
        Scanner cin = new Scanner (new BufferedInputStream(System.in));
        BigInteger Hrn;
        System.out.println
            ("Please enter the 'r' and 'n' of the H(r,n).Enter '-1''-1' to finish.");
        BigInteger r; r = cin.nextBigInteger();
        BigInteger n; n = cin.nextBigInteger();
        while (!((n.compareTo(c_1) == 0) && (r.compareTo(c_1) == 0)))
        {
            if (!((n.compareTo(c0) < 0 )
                    ||(n.compareTo(r) < 0)||(r.compareTo(c0) < 0)) )
            {
                Hrn = H_rn_tri(r,n);
                System.out.println(Hrn);
            }
            else
            {
                System.out.println("Illegal Input.");
            }
            System.out.println
                ("Please enter the 'r' and 'n' of the H(r,n).Enter '-1''-1' to finish.");
            r = cin.nextBigInteger();
            n = cin.nextBigInteger();
        }
        return ;
    }
}
```

4. Use C++ to do the verification of the fairness of the lottery "Choose 5 from 11" :

```
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;

int main()
{
    ifstream in_file("E:\\ DataInput.txt",ios::in);
    if (!in_file)
    {
        cout<<"Failure in opening the document!" <<endl;
        system("pause");
        return -1;
    }
    double derangeP[6];
    int countnum=0;
    int derangenum[6];
    int x[6];
    int example[6];
    int i=0,j=0,k=0;

    for (k=0;k<=5;k++)     derangenum[k]=0;
    for (i=1; i<=5; i++)     example[i]=i    ;

    while(!in_file.eof())
    {
        for(i=1;i<=5;i++)
        {
            in_file>> x[i] ;
            if (x[i] != example [i]) countnum++;
        }
        derangenum[countnum]++;
        countnum = 0;
        n++;
    }

    for(i=1;i<=5;i++)
        {
            if (x[i] != example [i])       countnum++;
        }
    derangenum[countnum]--;
    countnum = 0;
    n--;
```

```
    cout<<setiosflags(ios::fixed);
    cout<<setprecision(20);
    in_file.close();
    cout<<"There are together"<<n<<"lotteries"<<endl;
    for (k=0 ;k<=5; k++) derangeP[k] = (double)derangenum[k] / n;
    for (k=0 ;k<=5; k++) cout<<"The numbers of "<<k
                             <<"elements are deranged are"<<derangenum[k]<<endl;
    for (k=0 ;k<=5; k++) cout<<"The probability of"<<k
                             <<"elements are deranged are"<<derangep[k]<<endl;
    cout<<"The theoretical value of Derangement L0 is"<< (double)1/55440<<endl;
    cout<<"The theoretical value of Derangement L1 is "<< (double)30/55440 <<endl;
    cout<<"The theoretical value of Derangement L2 is "<< (double)430/55440<<endl;
    cout<<"The theoretical value of Derangement L3 is "<< (double)3560/55440<<endl;
    cout<<"The theoretical value of Derangement L4 is "<< (double)16665/55440<<endl;
    cout<<"The theoretical value of Derangement L5 is "<< (double)34754/55440<<endl;
    system ("pause");
    return 0;
}
```