康托展开

康托展开的公式是 X=an*(n-1)!+an-1*(n-2)!+...+ai*(i-1)!+...+a2*1!+a1*0!

其中，ai为当前未出现的元素中是排在第几个（从0开始）。

这个公式可能看着让人头大，最好举个例子来说明一下。例如，有一个数组 s = ["A", "B", "C", "D"]，它的一个排列 s1 = ["D", "B", "A", "C"]，现在要把 s1 映射成 X。n 指的是数组的长度，也就是4，所以

X(s1) = a4*3! + a3*2! + a2*1! + a1*0!

关键问题是 a4、a3、a2 和 a1 等于啥？

a4 = "D" 这个元素在子数组 ["D", "B", "A", "C"]

中是第几大的元素。"A"是第0大的元素，"B"是第1大的元素，"C"

是第2大的元素，"D"是第3大的元素，所以 a4 = 3。

a3 = "B" 这个元素在子数组 ["B", "A", "C"]

中是第几大的元素。"A"是第0大的元素，"B"是第1大的元素，"C" 是第2大的元素，所以 a3 = 1。

a2 = "A" 这个元素在子数组 ["A", "C"]

中是第几大的元素。"A"是第0大的元素，"C"是第1大的元素，所以 a2 = 0。

a1 = "C" 这个元素在子数组 ["C"] 中是第几大的元素。"C" 是第0大的元素，所以 a1 = 0。（因为子数组只有1个元素，所以a1总是为0）

所以，X(s1) = 3*3! + 1*2! + 0*1! + 0*0! = 20

A B C | 0

A C B | 1

B A C | 2

B C A | 3

C A B | 4

C B A | 5

通过康托逆展开生成全排列

如果已知 s = ["A", "B", "C", "D"]，X(s1) = 20，能否推出 s1 = ["D", "B", "A", "C"] 呢？

因为已知 X(s1) = a4*3! + a3*2! + a2*1! + a1*0! = 20，所以问题变成由 20

能否唯一地映射出一组 a4、a3、a2、a1？如果不考虑 ai 的取值范围，有

3*3! + 1*2! + 0*1! + 0*0! = 20

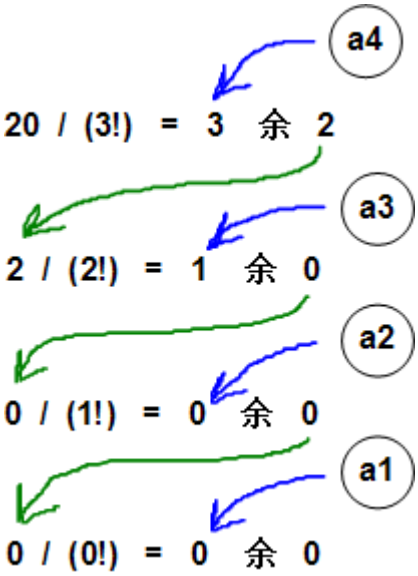
2*3! + 4*2! + 0*1! + 0*0! = 20

1*3! + 7*2! + 0*1! + 0*0! = 20

0*3! + 10*2! + 0*1! + 0*0! = 20

0*3! + 0*2! + 20*1! + 0*0! = 20

等等。但是满足 0 <= ai <= n-1 的只有第一组。可以使用辗转相除的方法得到ai，如下图所示：



知道了a4、a3、a2、a1的值，就可以知道s1[0] 是子数组["A", "B", "C", "D"]中第3大的元素"D"，s1[1] 是子数组 ["A", "B", "C"] 中第1大的元素"B"，s1[2] 是子数组 ["A", "C"] 中第0大的元素"A"，s[3] 是子数组 ["C"] 中第0大的元素"C"，所以s1 = ["D", "B", "A", "C"]。

这样我们就能写出一个函数 Permutation3()，它可以返回 s 的第 m 个排列。

前面的内容从http://archive.cnblogs.com/a/2026276/转载

```
1.#include<iostream>
2.#include<algorithm>
3.#include<vector>
4.#include<cstdlib>
5.using namespace std;
6.class cantor{
7.public:
8.    int n;//字符串的长度
9.    string s;
10.    int pos;//字符串在全排列中的字典位置，从0开始
11.    vector<int>num;//所有的字符
12.    cantor(string s):s(s){n=s.size();}
```

```
13.     cantor(int n,int pos):n(n),pos(pos){
14.         int i;
15.         for(i=0;i<n;i++)
16.             num.push_back(i);
17.     }
18.     int fac(int);
19.     void encode();
20.     void decode();
21.
22. };
23. int cantor::fac(int num){
24.     if(num==0) return 1;
25.         else return num*fac(num-1);
26. }
27. void cantor::encode(){
28.     int i,j,count;
29.     vector<int>vec(n);
30.     for(i=0;i<n;i++){
31.         count=0;
32.         for(j=i;j<n;j++)
33.             if(s[i]>s[j]) count++;
34.         vec[n-i-1]=count;
35.                 }
36.     pos=0;
37.     for(i=0;i<s.size();i++)
38.         pos+=vec[i]*fac(i);
39. }
40. void cantor::decode(){
41.     int i;
42.     div_t divresult;
43.     for(i=n-1;i>=0;i--){
44.         divresult=div(pos,fac(i));求余数与除数
45.         s.push_back(num[divresult.quot]+'0');
46.         num.erase(num.begin()+divresult.quot);
47.         pos=divresult.rem;
```

```
48.             }
49. }
50. int main(){
51.     cantor test(4,2);
52.     test.decode();
53.     cout<<test.s<<endl;
54. }
```