二项式反演

吴一岐

2020 年 5 月 20 日

1 二项式反演

1.1 描述

假设有两个函数 g 和 f 满足

$$g(i)=\sum_{j=1}^{i}\binom{i}{j}f(j)$$

那么一定有

$$f(i)=\sum_{j=1}^{i}(-1)^{i-j}\binom{i}{j}g(j)$$

1.2 证明

我们将上面的式子代入到下面可以得到

$$f(i)=\sum_{j=1}^{i}(-1)^{i-j}\binom{i}{j}\sum_{k=1}^{j}\binom{j}{k}f(k)$$

接下去我们只需要证明 $f(k)$ 的系数等价于 $[i==k]$ 即可
这样子左右两边就相等了
$f(k)$ 的系数为

$$\sum_{j=k}^{i}(-1)^{i-j}\binom{i}{j}\binom{j}{k}$$

用阶乘形式替代组合数

$$\sum_{j=k}^{i}(-1)^{i-j}\frac{i!}{j!(i-j)!}*\frac{j!}{k!(j-k)!}$$

化简之和得到

$$\left(\sum_{j=k}^{i}(-1)^{i-j}\frac{(i-k)!}{(i-j)!(j-k)!}*\right)\frac{i!}{k!(i-k)!}$$

$$\left(\sum_{j=k}^{i}(-1)^{i-j}\binom{i-k}{j-k}\right)\binom{i}{k}$$

我们发现左边的式子只有在 $i=k$ 的时候才为 1，其他情况为 0，因此得证

另外，下标无论是 0 开始还是 1 开始都满足这个二项式反演

1.3 其他形式

二项式反演还有其他几种情况，证明方法类似

$$g(i)=\sum_{j=i}^{n}\binom{j}{i}f(j) <=> f(i)=\sum_{j=i}^{n}(-1)^{j-i}\binom{j}{i}g(j)$$

$$g(i)=\sum_{j=1}^{i}(-1)^{j}\binom{i}{j}f(j) <=> f(i)=\sum_{j=1}^{i}(-1)^{j}\binom{i}{j}g(j)$$

$$g(i)=\sum_{j=i}^{n}(-1)^{j}\binom{j}{i}f(j) <=> f(i)=\sum_{j=i}^{n}(-1)^{j}\binom{j}{i}g(j)$$

2 例题一

错排问题

从容斥角度看错排问题

2.0.1 问题求解

让我们从二项式反演的角度看错排问题

设 $g(i)$ 表示 i 个人随便站的方案数，$f(i)$ 表示 i 个人都站错的方案数，那么 g 和 f 满足

$$g(i)=\sum_{j=0}^{i}\binom{i}{j}f(j)$$

$$g(i)=i!$$

备注：这个题需要 j 从 0 开始，但还是可以使用二项式反演

我们反演出 $f(i)=\sum\limits_{j=0}^{i}(-1)^{i-j}\binom{i}{j}g(j)$

因此我们可以得出恰好 i 个人站错的方案，答案就是 $f(n)$

我们来对比一下普通容斥的做法，设 $h(i)$ 表示 i 个人站对，其他人随便站的方案数

那么 $ans = h(0) - h(1) + h(2) - ... + (-1)^n h(n)$

$h(i) = \binom{n}{i} i!$

这样子是对的原因是：恰好 i 个人站对的容斥系数都恰好是 0，因为恰好 i 个人站对的方案数会在 $h(j)(j \le i)$ 里面出现 $\binom{i}{j}$ 次，组合数交错相加减最终等于 0

我们发现其实二项式反演推出的式子就是容斥原理

2.0.2 代码实现

错排问题

```
#include <bits/stdc++.h>
using namespace std;


long long c[22][22];
long long fac[22];

void init() {
  c[0][0] = 1;
  for (int i = 1; i <= 20; i++) {
    c[i][0] = c[i][i] = 1;
    for (int j = 1; j < i; j++) {
      c[i][j] = c[i - 1][j] + c[i - 1][j - 1];
    }
  }
}

int main() {
  fac[0] = 1;
  for (int i = 1; i <= 20; i++) {
    fac[i] = fac[i - 1] * i;
  }
  int n;
  init();
```

```
  while(scanf("%d", &n)==1) {
    long long ret = 0;
    for (int i = 0; i <= n; i++) {
      if (i % 2 == 0) {
        ret += c[n][i] * fac[n - i];
      } else {
        ret -= c[n][i] * fac[n - i];
      }
    }
    printf("%lld\n", ret);
  }
  return 0;

}
```