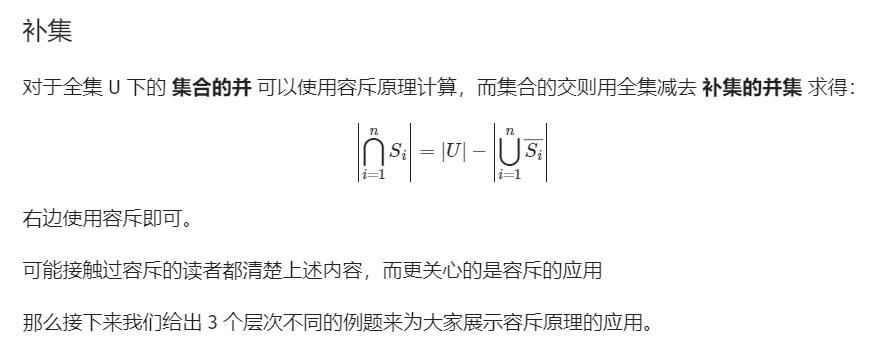
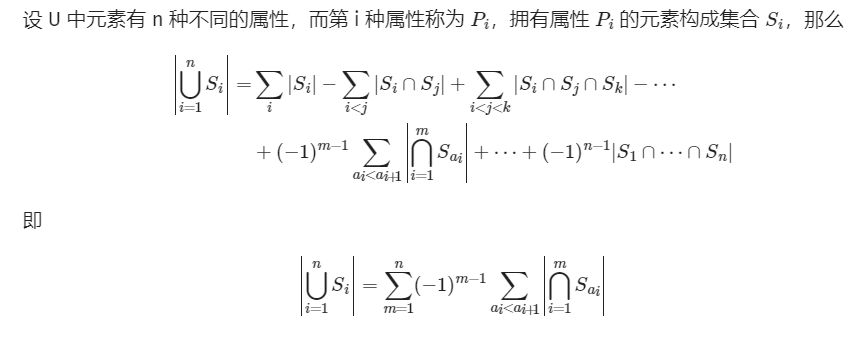


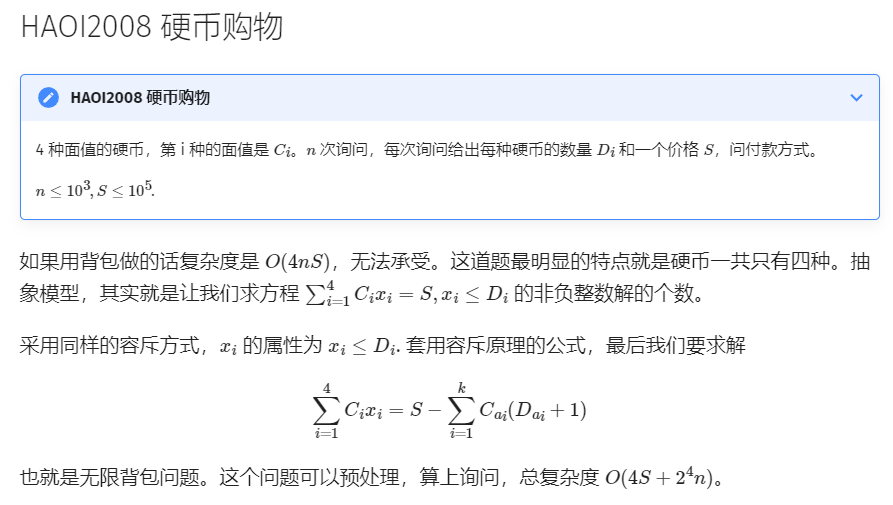
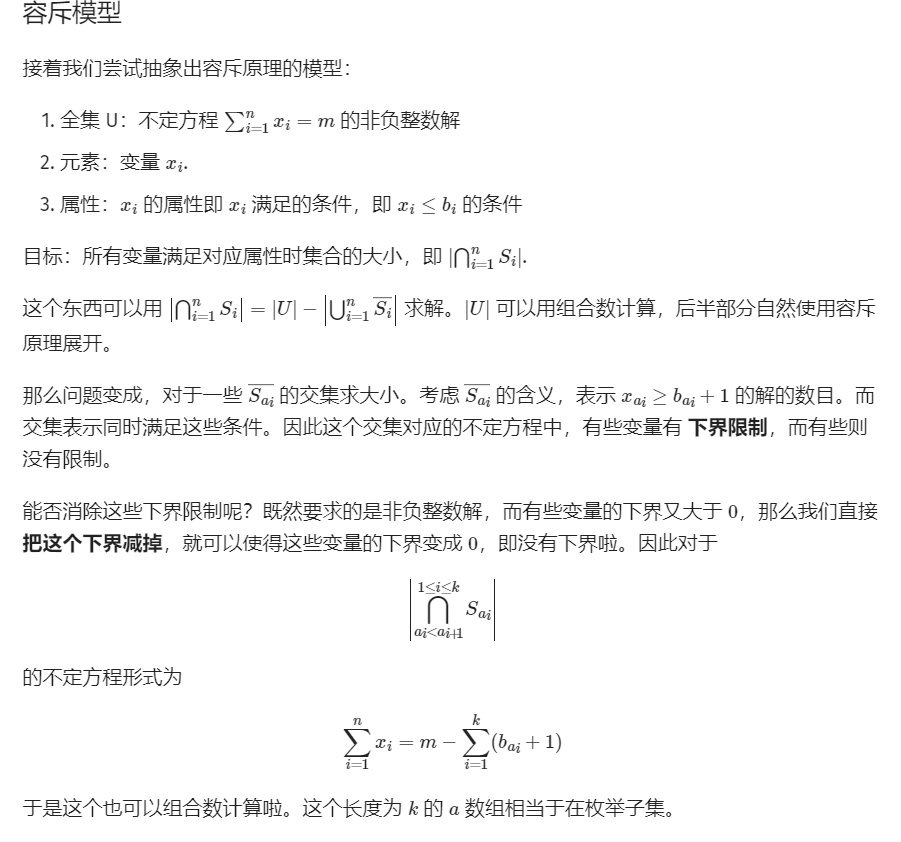
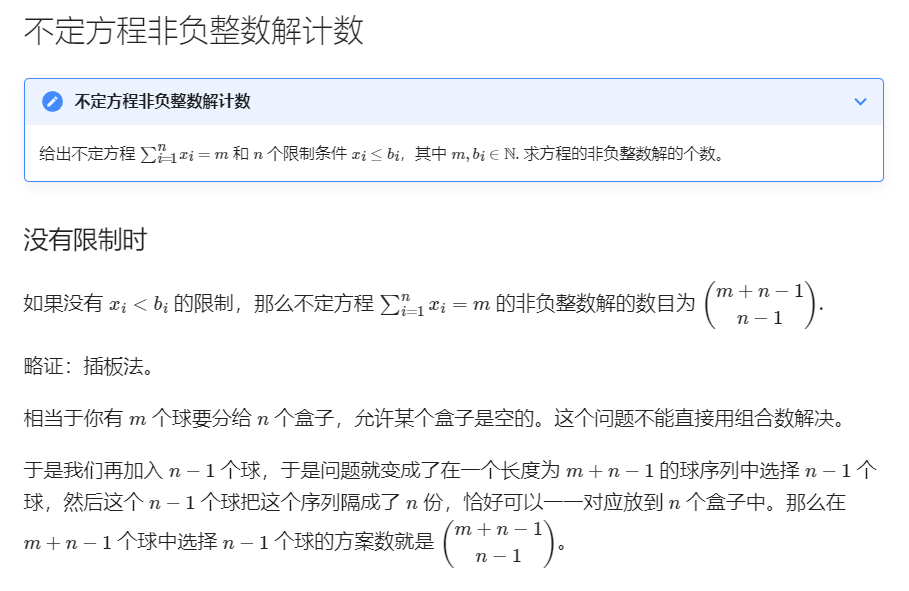
上面的式子比较形象,抽象概括的可以为下面两个式子:



从这里可以看出来什么情况可以用容斥定理:

1.求要满足n个约束条件的结果计数(**集合的并** 可以使用容斥原理计算，而集合的交则用全集减去 **补集的并集**, 补集的并集用容斥定理求)

2.求至少要满足一个约束条件的结果计数(也就是求**集合的并**A1 U A2 U A3)



容斥原理的板子

while(cin>>n)

{

ll ans=0;

//遍历所有可选择的二进制数,值得注意的是,这里涉及到二进制数,有可能产生借用前面的二进制子集来进行状态转移上的优化

for(int i=1;i<(1ll<<n);i++)

{

ll mul=1;

int flag=0;

for(int j=0;j<n;j++)

{

if(i&(1<<j))

{

mul\*=p[j];//不必要的变量

flag++;//flag为表示当前选择了哪些的二进制数里面1的个数

}

}

ll now=(ll)(pow((double)n,(1.0/mul))+eps);//now为对答案的贡献

if(flag%2)//奇数+,偶数-

{

ans+=now;

}

else

{

ans-=now;

}

}

cout<<ans<<endl;

}