暴力枚举

**一、循环枚举:**使用多重循环枚举所有的情况。

枚举剪枝

1. **子集(组合)枚举**
2. **选数。从n(n<=20)个整数中任选k个整数相加，求有多少种选择情况可以使和为质数。**

**分析：**本题可以认为是从一个有n个数字的集合中挑选出一些数字(也就是子集)，然后判断该子集是否满足某个性质(其和是质数)。集合枚举的意思是从一个集合中找出它的所有子集。

集合中每个元素都可以被选或不选，含有n个元素的集合总共有2^n个子集(包括全集和空集)。

考虑集合A={1,2,3,4,5}和它的4个子集A1={1,3,4,5}、A2={1,4,5}、A3={3}、A4={2,3}.按照某个顺序，把全集A中的每个元素在每个子集中的出现状况用0(没出现)和1(出现了)表示出来，如下表所示：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A中元素 | 1 | 2 | 3 | 4 | 5 | 二进制 | 对应十进制 |
| 在A1中的出现情况 | 1 | 0 | 1 | 1 | 1 | 11101 | a1=29 |
| 在A2中的出现情况 | 1 | 0 | 0 | 1 | 1 | 11001 | a2=25 |
| 在A3中的出现情况 | 0 | 0 | 1 | 0 | 0 | 00100 | a3=4 |
| 在A4中的出现情况 | 0 | 1 | 1 | 0 | 0 | 00110 | a4=6 |

那么，可以发现A的子集A1可以表示为一个“二进制数”11101，对应十进制变量a1=29(低位在右，表示元素1的二进制位在最右边，因此二进制数字需要左右反过来)；反之，这个数字也可以表示子集A1.注意，这边的集合是大写字母，集合对应的数字是小写字母。同理，A2可以表示为二进制数11001，A3可以表示成00100，A4可以表示成00110.此时找到了一种让子集对应于二进制数的很直观的方法。

本例一共有5个元素，表示仅包含第i(1<=i<=5)个元素的集合的数字可以使用**位移运算**构造，写成1<<(i-1)；而包含所有元素的全集可以表示成a=(1<<n)-1，空集表示为0.

**集合的常用关系有下面几种：**

1. 并集：当需要表示两个子集的并集时，可以把表示两个子集的二进制数进行“或”运算，即a1=a2|a3.
2. 交集：当需要表示两个子集的交集时，可以把表示两个子集的二进制数进行“与”运算，即a3=a1&a4.
3. 包含：集合A2的所有元素都在A1中出现，说明A1包含A2.易知A1并A2是A1，同时A1交A2是A2，也就是判断A1是否包含A2可以写成(a1|a2==a1)&&(a1&a2==a2).
4. 属于：是指某个元素在集合中，是包含的一种特殊情况——只需要检查单独某项元素构成的集合是否是另一个集合的子集。一般地，可以先用左位移运算构造出那个仅含一项的集合，然后再和原集合取交，若不为空集，则命题为真。如果要判断第3个元素是否属于A1，可以写成1<<(3-1)&a1.
5. 补集：例如A2对于全集的补集就是A3，A2的补集可以表示为a^a2.

#include<bits/stdc++.h>  
using namespace std;  
int a[30];  
bool check(int x){//判断质数  
 for(int i=2;i\*i<=x;i++)  
 if(x%i==0)return 0;  
 return 1;  
}  
int main(){  
 int n,k,ans=0;  
 cin>>n>>k;  
 for(int i=0;i<n;i++)scanf("%d",&a[i]);  
 int U=1<<n;//U-1即为全集  
 for(int S=0;S<U;S++)//枚举所有子集[0,U)  
 if(\_\_builtin\_popcount(S)==k){//找到k元子集  
 int sum=0;  
 for(int i=0;i<n;i++)  
 if(S&(1<<i))sum+=a[i];//如果第i个元素在S中  
 if(check(sum))ans++;  
 }  
 cout<<ans;  
 return 0;  
}

1. **组合的输出。从自然数1，2，…，n中任选r个数(1<=r<=n<=21)作为一个组合，并输出所有的组合情况，每个组合中的数字从小到大输出。**

#include<bits/stdc++.h>  
using namespace std;  
int a[30];  
int main(){  
 int n,r;  
 cin>>n>>r;  
 for(int S=(1<<n)-1;S>=0;S--){//从全集枚举到0  
 int cnt=0;  
 for(int i=0;i<n;i++)  
 if(S&(1<<i))  
 a[cnt++]=i;//分离记录每一位  
 if(cnt==r){  
 for(int i=r-1;i>=0;i--)  
 printf("%3d",n-a[i]);//因为用高位表示1，所以需要反过来输出  
 puts("");  
 }  
 }  
 return 0;  
}

**三、排列枚举:**

绝大多数题目可以使用next\_permutation函数，生成各个元素的不同排列，然后再进行判断或者统计即可轻松解决。

1. **将1，2，…，9共9个数分成3组，分别组成3个3位数，且使这3个3位数的比例是A:B:C(A<B<C<10^9)，试求出所有满足条件的3个3位数，若无解，输出“NO”。**

**思路：**

直接枚举3个3位数再检验，算法的复杂度为O(9!),其实还是可行的。

这里介绍一个用STL的简便写法。next\_permutation(start,end)是algorithm标准库中的一个标准函数，它可以在表示[start,end)内存的数组中产生严格的下一个字典序排列。具体来说就是[2,3,1]可以变成[3,1,2],再下一步就是[3,2,1]。

那么这道题只需要利用9的全排列，把全排列里的9个数位分别“划给”3个3位数就可以产生所有情况了。

#include<bits/stdc++.h>  
using namespace std;  
typedef long long LL;  
int a[10];  
int main(){  
 LL A,B,C,x,y,z,cnt=0;  
 cin>>A>>B>>C;  
 for(int i=1;i<=9;i++)  
 a[i]=i;  
 do{  
 x=a[1]\*100+a[2]\*10+a[3];  
 y=a[4]\*100+a[5]\*10+a[6];  
 x=a[7]\*100+a[8]\*10+a[9];  
 if(x\*B==y\*A&&y\*C==z\*B)/\*避免浮点误差和爆long long的小技巧\*/  
 printf("%lld %lld %lld\n",x,y,z),cnt++;  
 }while(next\_permutation(a+1,a+10));  
 if(!cnt)puts("NO!");  
 return 0;  
}

1. **全排列问题。输出自然数1到n所有不重复的排列，即n的全排列。**

**思路：**

从最小的排列开始(所有元素从小到大)，每次更换成下一个排列，然后输出。next\_permutation是有返回值的，没有下一个排列的时候就会返回0.

#include<bits/stdc++.h>  
using namespace std;  
int a[10],n;  
int main(){  
 cin>>n;  
 for(int i=1;i<=n;i++)  
 a[i]=i;  
 do{  
 for(int i=1;i<=n;i++)  
 printf("%5d",a[i]);  
 puts("");  
 }while(next\_permutation(a+1,a+n+1));  
 return 0;  
}

1. **对于n的全部排列，找到一个给定排列向后的第m个排列。**

#include<bits/stdc++.h>  
using namespace std;  
int a[10010],n,m;  
int main(){  
 cin>>n>>m;  
 for(int i=1;i<=n;i++)  
 cin>>a[i];  
 for(;m--;)  
 next\_permutation(a+1,a+n+1);  
 for(int i=1;i<=n;i++)  
 cout<<a[i]<<' ';  
 return 0;  
}