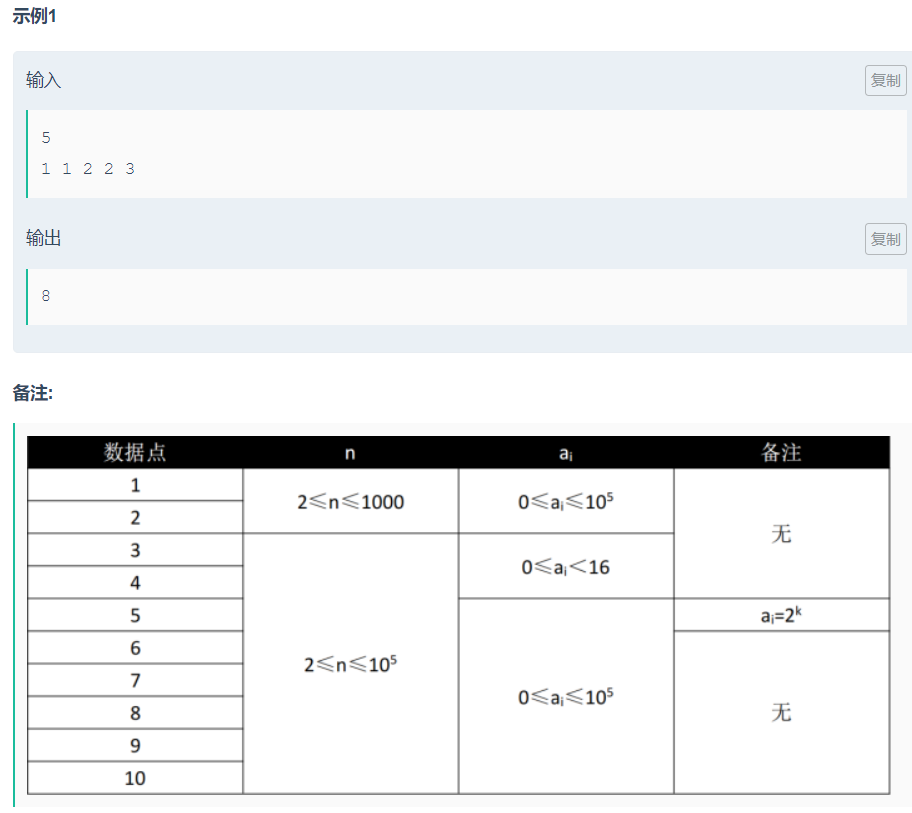
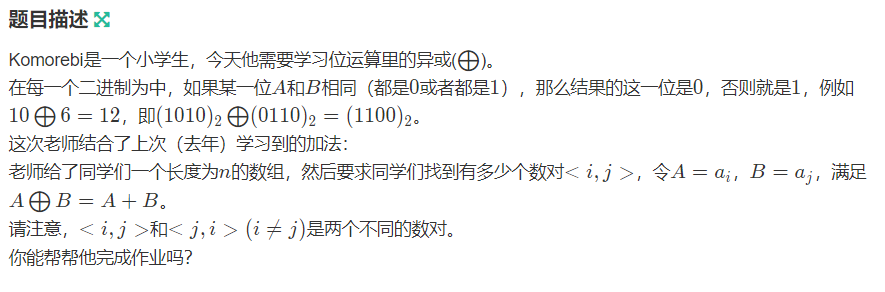
<https://ac.nowcoder.com/acm/contest/52244/G>



题意大概就是A位上的值是1的,B位上面只能是0,A上面为0,B上面可为1也可为0,首先我们大概能猜出来要用一种hash的算法来减少复杂度,那么我们取A位运算的反数(也就是2^20-A),那么B一定为反数的子集,所以我们可以采取一种特殊的枚举子集的方式:把所有A数的反数都放在f[2^20]数组里面,然后从最大数2^20开始往下枚举,枚举i的每一位,如果为1就把这个f数组往f[i^(1<<j)]上叠加,那么dp到B数的时候,就可以得到一个关于B的所有父集的f数组的一个前缀和,那么就可以得出所有能与B匹配的A.

总之,这是一种特殊的算法,可以用于解决所有类似这种的二进制子集/母集问题,包括前缀最大值,前缀和等问题,简称sosdp(但是我不是很熟),必须要记住相关格式

#define mn 2000005

using namespace std;

ll ans=0,f[mn],a[mn],n,m;

int main()

{

ll x,y,z,i,j,k;

char ch;

cin>>n;

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

{

scanf("%lld",&a[i]);

f[(1<<20)-1-a[i]]++;

}

for(j=0;j<=19;j++)

for(i=(1<<20)-1;i>=0;i--)

if((i&(1<<j)))f[i^(1<<j)]+=f[i];

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

ans+=f[a[i]];

cout<<ans;

return 0;

}