位运算

位运算是编程中非常有趣的操作，它主要利用计算机中以二进制的形式存储和操作数值的特点来优化一些数值处理。所以，我们要理解位运算，先理解二进制。

二进制

二进制的特点就是逢二进一，这个我们在上一讲进制转换中已经了解了。二进制的运算法则也非常简单，如下：

二进制加法：
0+0=0
0+1=1
1+0=1
1+1=10

二进制乘法：
0*0=0
1*0=0
0*1=0
1*1=1

编码

在计算机中，数值分正数和负数，两种数值在计算机中的存储规则是不一样的，理解它们的存储规则，需要了解编码的知识。

原码：计算机存储的是二进制数，第 1 位 0 表示正数，第 1 位为 1 表示负数。如 $(0010)_2=2$，$(1010)_2=-2$
反码：正数的反码与原码相同，负数的反码对原码按位取反
补码：正数的补码与原码相同，负数的补码为反码加 1

	原码	反码	补码
2	0010	0010	0010
-2	1010	1101	1110

负数在计算机中是以补码的形式存储的，或者说，一个数的负数即为该数的负数补码。

为什要用补码呢？

使用补码可以让计算用加法的方式做减法，提高计算机的计算性能。怎么用加法做减法？我们以十进制数为例：假如我们的整数只能存储两位十进制数，那么

99-43=99-(100-57)=99+57-100=156

补码

因为只能存储两位整数，舍弃百位后得 56，舍弃百位，我们也叫自然溢出。

同理，在计算机二进制中，减去某数，就相当于加上该数的补码。

此外需要特别注意的是，计算机中二进制数的存储方式如下图（以 4 位二进制为例），构成一个环。

二进制数	原数	备注
0000	0	
0001	1	
0010	2	
0011	3	
0100	4	
0101	5	
0110	6	
0111	7	
1000	-8	负数从小到大
1001	-7	
1010	-6	
1011	-5	
1100	-4	
1101	-3	
1110	-2	
1111	-1	再加 1 变成了 0

练习：对照上表，模拟以下表达式在二进制下的运算：

2+3

3+（-4）

（-5）+（-2）

3-5

5-（-3）

（-3）-（-5）

位运算操作符

c++提供了丰富的位运算操作符，如下，使得我们很便捷地在二进制下操作。

运算符	含义
&	按位与
|	按位或
^	按位异或
~	取反
<<	左移
>>	右移

1、 按位与（&）

按位与运算是指一位的二进制进行与运算，其运算规则如下：

- 0&0=0
- 0&1=0
- 1&0=0
- 1&1=1

例：两个整数的与运算

short a=10,b=12;//类型定义为 short，该类型为存储空间为 16 位

a&b=（0000000000001010 ）& （0000000000001100 ）=（ 0000000000001000 ）=8

两个整数进行与运算时，是逐位相与的。以上格式也可以这么看：

```
  1010
& 1100
  1000
```

练习：判断一个数 a 是否为偶数的表达式（位运算写法）

答案：a&1==0

解释：如果 a 的二进制末位是 1，则是奇数，如果是 0，则是偶数，将 a 与 1 相与，如末位是 1，与后结果是 1，如末位是 0，与后结果是 0。结果 1 表示奇数，结果 0 表示偶数。

2、 按位或（|）

按位或运算是指一位的二进制进行或运算，其运算规则如下：

- 0|0=0
- 0|1=1
- 1|0=1
- 1|1=1

例：两个整数的或运算

short a=10,b=12;

A|b=（0000000000001010 ）|（0000000000001100 ）=（ 0000000000001110 ）=14

练习：把 x 的二进制最后一位变为 1
答案：x=x|1
解释：x 的最后一位与 1 或运算，则最后一位变为 1，其它位不变。

3、 按位异或（^）

异或运算符在位运算中较常用到。其运算规则如下：

- 0^0=0
- 0^1=1
- 1^0=1
- 1^1=0

例：两个整数的异或运算：
short a=10,b=12;
a^b=（0000000000001010 ）^（0000000000001100 ）=（ 0000000000000110 ）=6

异或运算符有三个很重要的特性
（1） 一个数异或 0 还是它本身：a^0=a
（2） 一个数异或自己结果是 0：a^a=0
（3） 异或运算的逆运算是它本身:a^b^b=a

练习：用异或交换两个变量
答案：
a=a^b;
b=a^b; //等价于 b=a^b^b,等价于 b=a
a=a^b; //等价于 a=a^b^a，等价于 a=b

例：有一组连续的数字，从 1 到 n 排列，中间丢失了一个数字，顺序也被打乱了，请找出丢失的数字。（n<=10^8）
分析：设丢失的数字为 k，因为 n^n=0;k^0=k;所以（a[1]^a[2]^……^a[n-1]）^1^2^……^n=k^0=k
代码：

```
#include<iostream>
using namespace std;
int main(){
    int x,n,k=0;
    cin>>n;
    for(int i=1;i<=n-1;i++){
        cin>>x;
        k=k^x^i;
    }
    k^=n;
    cout<<k;
```

```
12. }
```

4、 取反(~)

取反运算符类似逻辑非，将 0 变成 1，将 1 变成 0。其运算规则如下：

- ▶ ~0=1
- ▶ ~1=0

注意：
使用取反运算时要格外小心，需要注意整数类型有没有符号。
有符号类型取反后最高位的变化导致了正负颠倒，又因为负数存储使用了补码，效果相当于-a-1。
无符号类型取反后的效果就是把这个数在数轴上的位置“对称翻折到另一边”去，即最大值-a。

5、 左移(<<)

a<<b：表示 a 的所有二进制位整体向左移动 b 位，后面 b 位用 0 补充。
a<<b 实际上 a<<b 就是 a 乘以 2 的 b 次方，可以用 a<<1 代替 a*2 的操作

例：
$1<<2=(100)_2=4$
$23<<2=(10111)_2<<2=(1011100)_2=92$

练习：求整型 x 的二进制第 i 位的值
答案：x&(1<<(i-1))
解释：比如求 9 的第 4 位的值。9 的二进制是（1001）,1<<(4-1)即 1 左移 3 位为（1000），将
（1001）与（1000）进行与运算

```
  1001
& 1000
  1000
```

如果 x 的第 i 位为 0，则结果为 0，如果结果不为 0，则第 i 位的值为 1

6、 右移(>>)

a>>b,表示 a 的所有二进制位整体向右移动 b 位，去掉末 b 位。
右移时需注意符号位问题，务必确保对非负整数进行运算，否则会出错。
a>>b，相当于 a 除以 2 的 b 次方（取整），可以用 a>>1 代替 a/2。

例：
$28>>2=(11100)_2>>2=(111)_2=7$

练习：分离出整数 x 的二进制位（对 x 进行二进制数位分离）

分析：x&1 可以得到 x 的末位的值，x>>1 可以将 x 的末位去除，反复进行这两个操作，直到 x 的值为 0。
代码：

```
#include<iostream>
using namespace std;
int main(){
    int x;
    cin>>x;
    while(x!=0){
        cout<<(x&1)<<" ";
        x=x>>1;
    }
}
```

==========网站练习==========

高低位交换

【题目描述】
给出一个小于 2^32 的正整数。这个数可以用一个 32 位的二进制数表示（不足 32 位用 0 补足）。我们称这个二进制数的前 16 位为“高位”，后 16 位为“低位”。将它的高低位交换，我们可以得到一个新的数。试问这个新的数是多少（用十进制表示）。
例如，数 1314520 用二进制表示为 0000 0000 0001 0100 0000 1110 1101 1000（添加了 11 个前导 0 补足为 32 位），其中前 16 位为高位，即 0000 0000 0001 0100；后 16 位为低位，即 0000 1110 1101 1000。将它的高低位进行交换，我们得到了一个新的二进制数 0000 1110 1101 1000 0000 0000 0001 0100。它即是十进制的 249036820。
【输入】
一个小于 2^32 的正整数
【输出】
将新的数输出
【样例输入】
1314520
【样例输出】
249036820
【分析 1】
因为只有 32 位，可以用位运算完成。对于整数 n，其前 16 位为 `n&((1<<16)-1)`，其后 16 位为 `n>>16`，将这两部分重新组成新数即可。注意，如果用 `int` 类型存储 32 位数，必须声明为 `unsigned` 无符号类型。
【代码 1】

```
#include<iostream>
using namespace std;
```

```
unsigned int n,x,y;  //定义无符号 int 类型
int main(){
    cin>>n;
    x=n&((1<<16)-1);
    y=n>>16;
    cout<<(x<<16)+y;
}
```

【分析 2】
针对此题，因为是移动 16 位，可以利用数值自然溢出的特点，简易处理。
【代码 2】

```
#include<iostream>
#include<cstring>
using namespace std;
unsigned int a,b,n;
int main(){
    cin>>n;
    a=n<<16;  //左移 16 位，高位部分自然溢出
    b=n>>16;  //右移 16 位，低位部分自然去掉
    cout<<a+b;
}
```

密室寻宝

【题目描述】
哈利波特不经意间进入了一座古墓，古墓入口有一道大门，内部有六个密室，每个密室中藏有一件兵器。已知需要两个密码才能从里面打开密室和大门，取出密室内的兵器后从大门撤出。
两个密码均是不大于 63 的整数，将其转化为八位二进制数后对应位进行"与"运算（运算的规则是：当两个位均为"1"时，结果为"1"，否则结果为"0"）。将"与"运算的结果从右往左数，当第 n 位为 1 时，表示可以打开第 n 个密室，取出其中的兵器；只有当取到至少两件兵器时，方可打开大门撤出。
现在哈利波特任意给你两个密码，请你帮他设计一个程序，算算可以从哪些密室取出兵器，并可否从大门撤出。
【输入】
第一行输入第一个密码 P ，
第二行输入第二个密码 Q 。
【输出】
第一行：按从小到大的顺序输出可以打开密室的编号。若没有可以打开的密室，则输出"0"。
第二行：若可打开大门，则输出为"Yes"，否则输出"NO"。
【样例输入】

输入样例一：
2
5
输入样例二：
7
13
【样例输出】
输出样例一：
0
NO
输出样例二：
1 3
Yes
【分析】
直接用位运算模拟。
【代码】

```
#include<iostream>
using namespace std;
unsigned int p,q,x;
int cnt;
int main(){
    cin>>p>>q;
    x=p&q;  //两个整数与运算
    for(int i=1;i<=8;i++){ //分离每一个二进制位
        if(x&1){  //如果该位为 1
            cout<<i<<" ";  //取得宝藏
            cnt++;
        }
        x>>=1;  //左移 1 位，去掉该位
    }
    if(cnt==0)cout<<0;
    cout<<endl;
    if(cnt>=2)cout<<"Yes"<<endl;
    else cout<<"NO"<<endl;
}
```

开灯

【题目描述】
在一条无限长的路上，有一排无限长的路灯，编号为 1，2，3，4，……。
每一盏灯只有两种可能的状态，开或者关。如果按一下某一盏灯的开关，那么这盏灯的状态将发生改变。如果原来是开，将变成关。如果原来是关，将变成开。
在刚开始的时候，所有的灯都是关的。

小明每次可以进行如下的操作：
指定两个数，a，t（a 为实数，t 为正整数）。将编号为[a]，[2*a]，[3*a]，……，[t*a]的灯的开关各按一次。其中[k]表示实数 k 的整数部分。
在小明进行了 n 次操作后，小明突然发现，这个时候只有一盏灯是开的，小明很想知道这盏灯的编号，可是这盏灯离小明太远了，小明看不清编号是多少。
幸好，小明还记得之前的 n 次操作。于是小明找到了你，你能帮他计算出这盏开着的灯的编号吗？
【输入】
第一行一个正整数 n，表示 n 次操作。
接下来有 n 行，每行两个数，ai，ti。其中 ai 是实数，小数点后一定有 6 位，ti 是正整数。
【输出】
仅一个正整数，那盏开着的灯的编号。
【样例输入】
3
1.618034 13
2.618034 7
1.000000 21
【样例输出】
20
【提示】
记 T=t1+t2+t3+……+tn。
对于 30%的数据，满足 T<=1000
对于 80%的数据，满足 T<=200000
对于 100%的数据，满足 T<=2000000
对于 100%的数据，满足 n<=5000，1<=ai<1000，1<=ti<=T
数据保证，在经过 n 次操作后，有且只有一盏灯是开的，不必判错。
【分析】
本题可以用异或找丢失的数的思路来做。设变量 ans=0，如果有一盏灯编号为 k，如果它亮了，则 ans=ans^k，如果再对这张编号为 k 的灯进行操作，则灯灭了，ans^k^k=ans，k 异或两次后对结果 ans 没有影响。所以，本题的思路是将所有操作的灯的编号与 ans 异或，如果有灯操作偶数次，则该等的异或值为 0。剩下的只有一盏灯，即最后的异或值。
【代码】

```
#include<iostream>
using namespace std;
int n,t,ans=0,k;
double a;
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>a>>t;
        for(int j=1;j<=t;j++){
            int k=a*j;
            ans=ans^k;
        }
```

```
    }
    cout<<ans;
}
```

倒水

【题目描述】
一天，CC 买了 N 个容量可以认为是无限大的瓶子，开始时每个瓶子里有 1 升水。接着~~CC 发现瓶子实在太多了，于是他决定保留不超过 K 个瓶子。每次他选择两个当前含水量相同的瓶子，把一个瓶子的水全部倒进另一个里，然后把空瓶丢弃。(不能丢弃有水的瓶子)
显然在某些情况下 CC 无法达到目标，比如 N=3,K=1。此时 CC 会重新买一些新的瓶子(新瓶子容量无限，开始时有 1 升水)，以到达目标。
现在 CC 想知道，最少需要买多少新瓶子才能达到目标呢？
【输入】
一行两个正整数， N,K(1<=N<=10^9，K<=1000)。
【输出】
一个非负整数，表示最少需要买多少新瓶子。
【样例输入】
water1.in
3 1
water2.in
13 2
water3.in
1000000 5
【样例输出】
water1.out
1
water2.out
3
water3.out
15808
【分析】
这题和二进制有关。先看以下分析：

合并前	二进制	合并后
1 个瓶子	1	1 个瓶子
2 个瓶子	10	1 个瓶子
3 个瓶子	11	2 个瓶子
4 个瓶子	100	1 个瓶子
5 个瓶子	101	2 个瓶子
6 个瓶子	110	2 个瓶子
7 个瓶子	111	3 个瓶子
8 个瓶子	1000	1 个瓶子
…	…	…

根据上列式子，我们知道 n 个有水的瓶子，最后合并成的瓶子个数，就是这个数转成二进制

1 的个数。

现在要将该数上的二进制数 1 的个数减少到 k 个，怎么办呢？我们只需要将 1 的位置上再加 1，即可以将该位取消，如：3 个瓶子，保留 1 个瓶子，可以

```
  11
+  1
 100
```

如：14 个瓶子，保留 2 个瓶子，可以

```
   1110
+  0010
  10000
```

所以，解决的方法是在新数的二进制位上的最小的 1 的位置加上 1，直到该数的 1 的个数小于等于 k 为止。

那么，怎么取得二进制下最小的 1 呢？我们用一个叫 lowbit 的函数实现，其实就是一个很简单的表达式：x&-x;

这个式子返回的值就是从后往前数，到第一个 1 出现为止的数(二进制下)。

来列个表：

I	i(2)	i&-I	i&-i(2)
1	1	1	1
2	10	2	10
3	11	1	1
4	100	4	100
5	101	1	1
6	110	2	10
7	111	1	1
8	1000	8	1000

【代码】

```
#include<iostream>
using namespace std;
long long n,k,ans;
int get(long long n){  //获得 n 在二进制下 1 的个数
    int sum=0;
    while(n!=0){
        sum+=n&1;
        n>>=1;
    }
    return sum;
}
int lowbit(long long n){  //找 n 在二进制下最小的 1
    return n&(-n);
}
int main(){
    cin>>n>>k;
    while(get(n)>k){
        ans+=lowbit(n);
```

```
        n+=lowbit(n);
    }
    cout<<ans;
}
```

比萨

【题目描述】
NH 的最大比萨店为即将来临的节日准备了 T 种不同加味的原料，但考虑到 NH 人的口味和其它一些因素，原料的使用有 N 种限制。
T 种不同原料的编号为 1..T。一个限制如"5 3"即表示 5 号和 3 号加味原料不能同时使用。如此，此时使用三种原料 3,5,6 的比萨是不允许的。
现在请你帮忙计算在上面条件下，最多可以制作多少不同的比萨（包括不添加任何加味原料的）。
【数据范围】
1≦T≦20
1≦N≦52
【输入】
第一行：两个整数：T 和 N。
下面有 N 行：每行表示一种限制。每行的第一个整数 Z(1≦Z≦ T)表示这行后面有 Z 个表示原料编号的整数，这些原料不能同时出现在一个比萨中。
【输出】
只一行，一个整数表示在上面的限制下最多可以制成多少种不同比萨。
【样例输入】
6 5
1 1
2 4 2
3 3 2 6
1 5
3 3 4 6
【样例输出】
10
【答案说明】
无加料；2；2 和 3；2 和 6；3；3 和 4；3 和 6；4；4 和 6；6。
【分析】
因为 t 小于等于 20，所以，我们可以暴力枚举每一个方案。以 i 的二进制形式表示方案数，如 13，二进制是 1101,表示用 1、3、4 中调料。那么这样的方案数从 0 到 1<<t-1,对于方案 i，遍历 i 的二进制下 1 的位置，判断有无冲突即可。
【代码 1】

```
#include<iostream>
using namespace std;
int t,n,ans,a[53][21];

```

```
bool checkrow(int x,int k){  //对于第 k 个约束条件，方案 x 是否合法
    bool f=true;
    for(int i=1;i<=a[k][0];i++){
        bool t=x&(1<<(a[k][i]-1));  //获得 x 的第 a[k][i]位的值
        f=f&&t;
    }
    return f;  //值为 true 表示不合法
}

bool check(int x){  //判断 x 的方案是否有冲突
    for(int i=1;i<=n;i++){  //枚举每一种冲突约束
        if(checkrow(x,i))return false;  //如果冲突了返回 false
    }
    return true;
}
int main(){
    cin>>t>>n;
    for(int i=1;i<=n;i++){
        cin>>a[i][0];
        for(int j=1;j<=a[i][0];j++){
            cin>>a[i][j];
        }
    }
    int num=1<<t;
    for(int i=0;i<num;i++){  //枚举方案
        if(check(i))ans++;
    }
    cout<<ans;
}
```

【代码 2】

```
#include<iostream>
using namespace std;
int t,n,ans,a[53];
bool check(int x){
    for(int i=1;i<=n;i++){
        if((x&a[i])==a[i])return false;//将方案 x 与限制条件 a[i]相与，判断冲突
    }
    return true;
}
int main(){
    cin>>t>>n;
    for(int i=1;i<=n;i++){
        int k,x;
```

```
        cin>>k;
        for(int j=1;j<=k;j++){
            cin>>x;
            a[i]=a[i]|(1<<(x-1));  //将限制条件合成一个二进制数
        }
    }
    int num=1<<t;
    for(int i=0;i<num;i++){
        if(check(i))ans++;
    }
    cout<<ans;
}
```

桐桐的运输方案

【题目描述】
桐桐有 N 件货物需要运送到目的地，它们的重量和价值分别记为：
重量：W1，W2，…，Wn；
价值：V1，V2，…，Vn；
已知某辆货车的最大载货量为 X，并且当天只能运送一趟货物。这辆货车应该运送哪些货物，才能在不超载的前提下使运送的价值最大？
【输入】
第一行是一个实数，表示货车的最大载货量 x(1＜x≤100)。
第二行是一个正整数，表示待运送的货物数 n(1＜n≤20)。
后面 n 行每行两个用空格隔开的实数，分别表示第 1 至第 n 件货物的重量 W 和价值 V。
【输出】
第一行为被运送货物的总价值（只输出整数部分）；
第二行为按编号大小顺序输出所有被运送货物的编号（当一件都不能运送时，不输出）。
【样例输入】
输入样例 1:
16
4
3 4
4 5
5 6
6 7
输入样例 2:
20
4
3.5 4
4 5
5 6.8
6.9 7

输入样例 3:
12
3
24 15
18 20
15 30
【样例输出】
输出样例 1:
18
2 3 4
输出样例 2:
22
1 2 3 4
输出样例 3:
0
【分析】
本题与第五题类似。可以利用二进制枚举每一种方案，再判断方案是否合法进行比较。
【代码】

```
#include<iostream>
using namespace std;
double x,w[21],v[21],maxv,maxw;
int n,r;

int main(){
    cin>>x>>n;
    for(int i=1;i<=n;i++){
        cin>>w[i]>>v[i];
    }
    int num=1<<n;
    for(int i=0;i<num;i++){   //枚举每一种方案
        int t=i;
        double tv=0,tw=0;
        for(int j=1;j<=n;j++){  //分解二进制每一位
            if(t&1){  //该位为 1，即表示选择这件货物
                tv+=v[j];
                tw+=w[j];
            }
            t>>=1;
        }
        if(tw>x)continue;  //总重量超过 x，不合法，跳过
        if(tv>maxv){  //比较全局最优价值和
            r=i;
            maxv=tv;
        }
```

```
27.     }
28.     if(maxv==0){
29.         cout<<0<<endl;
30.     }else{
31.         cout<<(int)maxv<<endl;
32.         for(int i=1;i<=n;i++){
33.             if(r&1)cout<<i<<" ";
34.             r>>=1;
35.         }
36.     }
37. }
```