位运算

信智学院　陈琰宏

主要内容

01 位运算

02 举例

03

04

05

位操作

二进制操作又叫做位运算，二进制有很多用处，常用来提高程序的效率。

在算法竞赛中，常用二进制进行数据压缩，就是用二进制来表示复杂的状态，然后通过位运算来完成问题的求解，这样能大大提高算法的效率，这样的状态表示经常用来优化DP或者是搜索。。。

位运算(*bitwise operation*)。其优先级:

~ > &> ^ >|

名称	C/C++样式	简记法则
按位与	&	全一则一，否则为零
按位或	|	有一则一，否则为零
按位取反	~ (!)	是零则一，是一则零
按位异或	^	不同则一，相同则零
左移位	<<	a<<k等价于$a*2^k$
右移位	>>	a>>k等价于$a/2^k$

位运算-左移 <<

假设有个变量x=1；

那么x=x<<1表示的意思如下：

x:0000000000000001（二进制表示）

x<<1:0000000000000010（也就是把x的每一位向左移动1位的结果，右边不够的用0添上）

此时x=x<<1 =2，其实可以看做乘2

x<<i 表示的就是左移i位,可以看做乘 i 次 2

位运算-右移 >>

假设有个变量x=2;

那么x=x>>1表示的意思如下：

x:　　0000000000000010（二进制表示）

x>>1:0000000000000001（也就是把x的每一位向右移动1位的结果，左边不够的用0添上）

此时x=x>>1 ==1，其实可以看做除2

x>>i 表示的就是右移i位,可以看做除 i 次2

位运算-或运算|

设x=101， y=010(都是二进制表示)

那么x|y表示如下

x: 101

y: 010

x|y:111

其实也就是将x和y表示成二进制，然后按位做或运算

此时x|y==7(10进制表示)

位运算-与运算 &

设x=101 y=010(都是二进制表示)

那么x&y表示如下

x: 101

y: 010

x&y: 000

将x和y表示成二进制，然后按位做与运算

此时x&y==0(10进制表示)

位运算-非运算 ~

设x=101(都是二进制表示)

那么~x表示如下

x:　　101

~x:　　010

其实也就是将x表示成二进制，然后按位做取反运算

此时~x==2(10进制表示)

位运算-与运算^

设x=1010 y=1100（都是二进制表示）

那么x^y表示如下

x: 1010

y: 1100

x^y: 0110

其实也就是将x和y表示成二进制，然后按位做异或运算，也就是相同为0，不同为1

此时x^y==6(10进制表示)

位运算-获取一个或多个固定位的值

假设x=1010(10进制的10)

我们要获取从左边数第2为的值，那么我们可以这样来取

x&(1<<1)也就是

x:　　1010

1<<1:　　0010

x&(1<<1)　0010

这样我们就可以通过判断x&(1<<2)是否等于0来知道这一位是0还是1了

当然我们可以用x&(3<<2)来取得第3位和第4位

位运算-把一个或多个固定位的值置零

假设x=1010(10进制的10)

我们要把从左边数第2为的值置零，那么我们可以这样来做

x&(~(1<<1))也就是

x:　1010

~(1<<1):　1101

x&(~(1<<1))：　1000

当然我们可以用x&(~(3<<2))来把第3位和第4位置零

位运算-一个或多个固定位的值取反

假设x=1010(10进制的10)

要把从左边数第2为的值取反，可以这样做

x^(~(1<<1))

也就是如果x=1010

x:	1010
1<<1:	0010
~(1<<1):	1101
	1010
x^(1<<1) :	0111

当然我们可以用x^(3<<2)来把第3位和第4位取反

位运算-lowbit操作

lowbit(n)定义为非负整数n在二进制表示下“最低位的1及其后边所有的0”构成的数值。

例如　n = 10的二进制表示为$(1010)_2$,则

lowbit(n) = 2 = $(10)_2$

下 面我们来推导lowbit(n)的公式

位运算

负数的二进制运算为补码运算

$-x = \sim x + 1$

$x \& -x = x \& (\sim x + 1)$

假设 x=10101...100000

~x=01010...011111

+1

100000

位运算

x&-x=

假设 x=10101...100000

~x=01010...100000

00000...100000

[1213]二进制中1的个数

给定一个长度为 n 的数列，请你求出数列中每个数的二进制表示中 1 的个数。

输入:第一行包含整数 n。

第二行包含 n 个整数，表示整个数列。

$1 \leqslant n \leqslant 100000$　　$0 \leqslant$ 数列中元素的值 $\leqslant 10^9$

输出共一行，包含 n 个整数，其中的第 i 个数表示数列中的第 i 个数的二进制表示中 1 的个数。

样例输入

5

1 2 3 4 5

样例输出

1 1 2 1 2

算法1 (lowbit)

O(nlogn)

使用lowbit操作，每次lowbit操作截取一个数字最后一个1后面的所有位，每次减去lowbit得到的数字，直到数字减到0，就得到了最终1的个数.

[1213]二进制中1的个数

```cpp
#include <iostream>
using namespace std;
int n;
int lowbit(int x) {
    return x & -x;
}
int main() {
    cin >> n;
    while (n -- ) {
        int x, ans = 0;
        scanf("%d",&x);
        while (x) x -= lowbit(x), ans ++ ;
      // x 每次减去二进制中最后一个 1
      //故循环结束后减去的次数就为 x 的二进制中 1 的个数
        cout << ans << ' ';
    }
    cout << endl;
    return 0;
}
```

算法2 枚举

O(nlogn)

对于每个数字a，a&1得到了该数字的最后一位，之后将a右移一位，直到位0，就得到了1的个数

算法2 枚举

```
1  #include<iostream>
2  using namespace std;
3  int n;
4  int a,k;
5  int main(){
6      scanf("%d",&n);
7      for(int i=0;i<n;i++){
8          scanf("%d",&a);
9          k=0;
10         while(a){
11             k+=a&1;
12             a=a>>1;
13         }
14         printf("%d ",k);
15     }
16     return 0;
17 }
```

[6687] 数组中只出现一次的两个数字

一个整型数组里除了两个数字之外，其他的数字都出现了两次。
请写程序找出这两个只出现一次的数字。
你可以假设这两个数字一定存在。
数组长度 1<=n<=1e5
1<=a[i]<=1e9

输入样例 5
1 2 2 3 2
输出样例
1 3

分析1

哈希表
时间复杂度O(n)，
空间复杂度O(n^2)

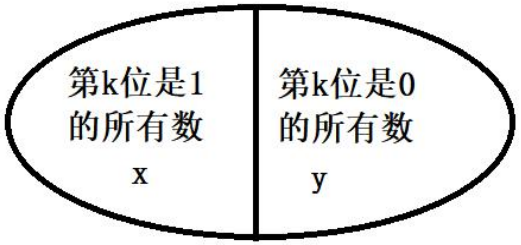
```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n,x;
    vector<int> ans;
    map<int, int> hash;
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>x;
        hash[x]++;
    }
    for (auto & it : hash)
        if (it.second == 1)
            ans.push_back(it.first);
    for(int i=0;i<ans.size();i++)
        cout<<ans[i]<<" ";
    return 0;
}
```

分析2

异或满足交换律,如x1^x2^x3=x1^x3^x2

1. 第一步异或，相同的数其实都抵消了，剩下两个不同的数。这两个数异或结果肯定有某一位为1，不然都是0的话就是相同数。

2. 假设x,y出现1次，其它出现两次。所有元素异或后的结果为 s=x^y^0,因为x、y不相等,则x、y异或后的结果必然不等于0，想x,y至少有一位不同。假设x的第k位为1，y的第k为是0.按此位将所有数分成两组，分开后各自异或，相同的两个数异或肯定为0（而且分开的时候，两个数必为一组）。剩下的每组里就是要找的数。



[6687] 位操作练习

给出两个不大于 65535 的非负整数，判断其中一个的 16 位二进制表示形式，是否能由另一个的 16 位二进制表示形式经过循环左移若干位而得到。
循环左移和普通左移的区别在于：最左边的那一位经过循环左移一位后就会被移到最右边去。
比如：
1011 0000 0000 0001 经过循环左移一位后，变成
0110 0000 0000 0011，
若是循环左移2位，则变成
1100 0000 0000 0110。

输入包含多组测试数据。 每组数据包含两个整数，占一行。
每组数据输出一个结果，如果两个数之间能够进行移位转化则输出 YES，否则输出 NO。

[6687] 位操作练习

2 4	YES
9 18	YES
45057 49158	YES
7 12	NO

分析

最高位是1的情况：

x:1011 最高位是1 ----> 0111
x:0011 减去最高位
x:0110 乘以2
x:0111 //加上循环移过来的1

```
int main(){
    int n,m,f;
    while(cin>>n>>m){
        for(int i=0;i<16;i++){
            f=true;
            if(1 & n>>15){//确定最高位是否是1
                n-=32768;//减去最高位的值
                n*=2;//左移一位
                n++;//加上循环移过来的1
            }
            else{
                n=n*2;
            }
            if(n==m){
                f=false;
                cout<<"YES\n";
                break;
            }
        }
        if(f)
            cout<<"NO\n";
    }
}
```

方法2

（1）分析可知，若两个数能够通过移位转换，则二者都能转换成对方。因此，可以将问题简化成只考虑一方。

（2）若 A 能够通过移位转换成 B，则说明将 A 前面某个连续部分拼接到 A 后面就转化成了 B。

（3）因此，可以将 A 扩充成两倍长度，此时若二者能够移位转换，则 A 中必然包含有等于 B 的子串。

```
4   int main()
5   {
6       int a, b;
7       while(cin >> a >> b)
8       {
9           string x, y;
10          for(int i = 15; i >=0; i --)
11          {
12              x = x + to_string(a >> i & 1);  //x表示a的二进制表示
13              y = y + to_string(b >> i & 1);  //y表示b的二进制表示
14          }
15          y += y; // 扩充成两倍长度
16           // 若两个数能够通过移位转换，则 A 中必然包含有等于 B 的子串
17          if(y.find(x) != -1) puts("YES");
18          else puts("NO");
19      }
20      return 0;
21  }
```

01背包问题

```
4   int w[105],v[105],n,W;
5   int main(){
6
7       scanf("%d%d", &W, &n);
8       for (int i = 0; i < n; i++)
9           scanf("%d%d", &w[i], &v[i]);
10      int ans = 0;
11      for (int i = 0; i < (1 << n); i++)
12      {   //i表示当前状态，枚举状态
13          int sumv = 0, sumw = 0;
14          for (int j = 0; j < n; j++)
15          {  //取出i的的所有1，如果当前位是1，表示取了这件物品
16              if (i&(1 << j)) sumv += v[j], sumw += w[j];
17          }
18          //如果当前状态的物品选择总量合法，比较是否是最大
19          if (sumw <= W) ans = max(ans, sumv);
20      }
21      printf("%d\n", ans);
22      return 0;
23  }
```

[6432] 二进制

给定一个长度为 32 位的二进制数 n，请你计算并输出 n+1 和 n+3 的二进制表示结果。
注意，结果不能忽略前导 0（不够 32 位的用前导 0 补足 32 位，超过 32 位的不用补前导 0）。

1 00000000000000000000000000000000	00000000000000000000000000000001 00000000000000000000000000000011
2 11111111111111111111111111111111 11111111111111111111111111111110	10000000000000000000000000000000 10000000000000000000000000000010 11111111111111111111111111111111 10000000000000000000000000000001

纯模拟即可，可以用字符串或数组处理。随后在指定的位置上加1。从后面往前面进位。

```
4   int n;
5   string s;
6   char a='0';
7   void jinwei(){
8       s[s.size()-1]++;//最低位+1
9       for(int i=s.size()-1;i>0;i--){
10          //从低位向高位一位一位处理
11          if(s[i]=='2'){//是2就要进位
12              s[i-1]++;//进位
13              s[i]='0';//进位后自己清零
14          }
15          if(s[0]=='2'){//最高位需要进位
16              s[0]='0';
17              a++;
18          }
19      }
20  }
```

```
22 int main(){
23     cin>>n;
24     for(int i=1;i<=n;i++){
25         cin>>s;
26         jinwei();//+1
27         if(a=='1')//最高位是1，超过 32 位
28             cout<<a;
29         cout<<s<<endl;
30         jinwei();
31         jinwei();
32         if(a=='1')
33             cout<<a;
34         cout<<s<<endl;
35         a='0';
36
37     }
38 }
```

状态	题号	题目编号	题目	分类	正确	提交
	6429	A	天宫课堂		13	44
	3081	B	[2004_p4]FBI树-数据结构	二叉树普及	7	7
Y	1213	C	二进制中1的个数	位运算	14	32
Y	6431	D	位操作练习	位运算	16	20
Y	2819	E	01背包问题	01背包	13	17
	6430	F	64位整数乘法	位运算 acwing	3	5
Y	6432	G	二进制	位运算	14	29
Y	1138	H	进制转换	进制模拟	4	15

[7391]格雷码

二进制转码：从最右边一位起，依次将每位与左边一位异或，作为该位格雷码，最左边一位不变。
解码：从左边第二
后的值异或，作为
不变。

某二进制格雷码为 $G_{n-1}G_{n-2}\cdots G_2G_1G_0$

其对应的自然二进制码为 $B_{n-1}B_{n-2}\cdots B_2B_1B_0$

异或运算：
相同为0
相异为1

其中：最高位保留—— $B_{n-1}=G_{n-1}$

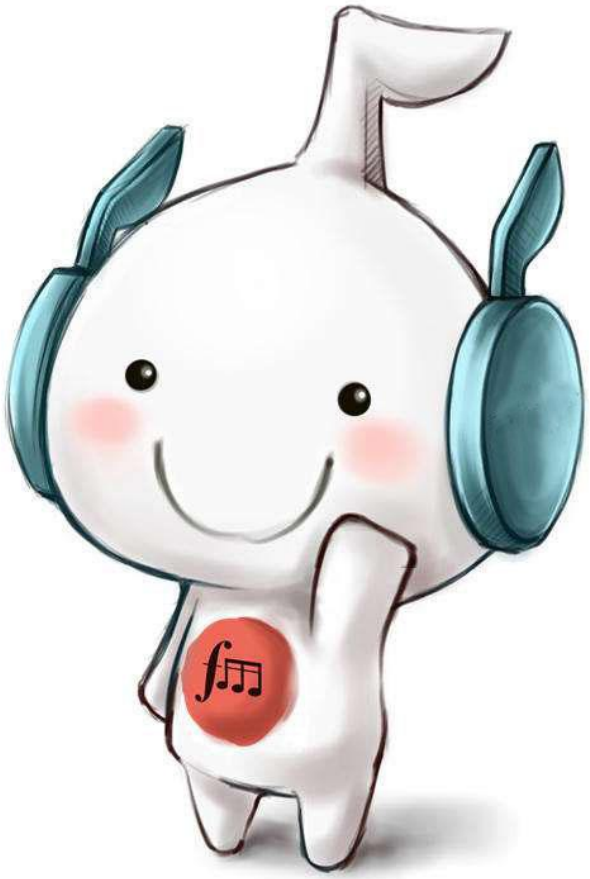
其他各位—— $B_{i-1}=G_{i-1}\oplus B_i$ $i=1, 2, \ldots, n-1$

例：二进制格雷码为 1 0 1 1 0

自然二进制码为 1 1 0 1 1

[6689] 等式

今天的课程结束啦……



下课了…
同学们再见！