位运算

黑猫编程
blackcat1995.com

任何信息在计算机中都是采用二进制表示的，数据在计算机中是以补码形式存储的，位运算就是直接对整数在内存中的二进制位进行运算。由于位运算直接对内存数据进行操作，不需要转换成十进制，因此处理速度非常快，在信息学竞赛中往往可以优化理论时间复杂度的系数。同时，一个整数的各个二进制位互不影响，利用位运算的一些技巧可以帮助我们简化程序代码。

黑猫编程
blackcat1995.com

位运算符

C++ 提供了按位与（&）、按位或（|）、按位异或（^）、取反（~）、左移（<<）、右移（>>）这 6 种位运算符。这些运算符只能用于整型操作数，即只能用于带符号或无符号的 char、short、int 与 long 类型。

按位与（&）

“a&b”是指将参加运算的两个整数a和b，按二进制位进行“与”运算。如果两个相应的二进制位数字都为 1 ，则该位的结果为1；否则为0。这里的1可以理解为逻辑中的true,0可以理解为逻辑中的false。“按位与”其实与逻辑上“与”的运算规则一致。

```
// 3 - 00000011
// 5 - 00000101
int a = 3;
int b = 5;
int c = a & b;

cout << c << endl;  // 00000001
```


按位或（|）

“a|b”是指将参加运算的两个整数a和b，按二进制位进行“或”运算。如果两个相应的二进制位数字有一个为 1 ，则该位的结果为1；否则为0。“按位或”其实与逻辑上“或”的运算规则一致。

```
// 48 - 00110000
// 15 - 00001111
int a = 48;
int b = 15;
int c = a | b;

cout << c << endl;  // 63 - 00111111
```


按位异或（^）

“a^b”是指将参加运算的两个整数a和b，按二进制位进行“异或”运算。如果两个相应的二进制位数字不相同，则该位的结果为1；否则为0。

```
// 52 - 00110100
// 15 - 00001111
int a = 52;
int b = 15;
int c = a ^ b;

cout << c << endl;  // 59 - 00111011
```

黑猫编程 blackcat1995.com 取反（~）

“~a”是指将整数a的各个二进制位都取反，即1变为0，0变为1。“~”是一元运算符。

```
cout << ~9 << endl;
/*
        9 - 00001001
     取反 - 11110110 补码
            11110101 反码
            10001010 原码
            最高位 1 表示负数  -10
*/
```

黑猫编程 blackcat1995.com 左移（<<）

“a<<b”是指将整数a的各个二进制位左移b位，高位丢弃，低位用0补齐。需要注意的是b必须是非负整数。在高位没有1丢弃的情况下，a<<1相当于a*2。

```
char a = (143<<2);
printf("%d", a);
/*
    143 - 10001111
  左移2 - 1000111100  60
*/
```

黑猫编程 blackcat1995.com 右移（>>）

“a>>b”是指将整数a的各个二进制位右移b位，低位丢弃。对于无符号数，高位补0。对于有符号数，某些机器将对左边空出的部分用符号位填补（即“算术移位”），而另一些机器则对左边空出的部分用0填补（即“逻辑移位”）。同样，b必须是非负整数。a>>1相当于a/2。

```
char a = (15 >> 2);
printf("%d", a);
/*
      15 - 00001111
   右移2 - 00000011  3
*/
```

```
char a = (-15 >> 2);
printf("%d", a);
/*
    15 - 10001111
  反码 - 11110000
  补码 - 11110001
 右移2 - 1111110001
  反码 - 11111011
  原码 - 10000100  -4
*/
```

整数幂

判断一个数 n 是不是 2 的整数幂，比如 64=2^6，所以输出“yes”，而 65 无法表示成 2 的整数幂形式，所以输出“no”。n 在 int 范围以内。

【问题分析】

我们考虑一个数如果是2的整数幂会有什么特殊性。观察发现64转换成二进制为01000000，只有一个位是1。将这个数减去1，就变成00111111的形式，我们将这2个数做按位与运算，发现结果为0。分析发现，如果一个数不能表示成2的整数幂形式，则以上过程的运算结果一定不为0。所以，可以利用位运算将算法的时间复杂度优化成O(1)。

黑猫编程 blackcat1995.com

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

int main() {
    int n;
    cin >> n;
    if (n & (n - 1))
        cout << "no";
    else
        cout << "yes";
    return 0;
}
```

文件

黑猫编程 blackcat1995.com

文件重定向（freopen）

【命令格式】

```
FILE * freopen ( const char * filename, const char * mode, FILE * stream );
```

【参数说明】

filename：要打开的文件名

mode：文件打开的模式，和fopen中的模式(r/w)相同

stream：文件指针，通常使用标准流文件(stdin/stdout/stderr),其中stdin是标准输入流，默认为键盘；stdout是标准输出流，默认为屏幕；

stderr是标准错误流，一般把屏幕设为默认。通过调用freopen，就可以修改标准流文件的默认值，实现重定向。

模式	描述
"r"	打开一个用于读取的文件。该文件必须存在。
"w"	创建一个用于写入的空文件。如果文件名称与已存在的文件相同，则会删除已有文件的内容，文件被视为一个新的空文件。
"a"	追加到一个文件。写操作向文件末尾追加数据。如果文件不存在，则创建文件。
"r+"	打开一个用于更新的文件，可读取也可写入。该文件必须存在。
"w+"	创建一个用于读写的空文件。
"a+"	打开一个用于读取和追加的文件。

黑猫编程 blackcat1995.com

使用方法

接下来我们使用freopen()函数以只读方式r(read)打开输入文件score.in。

格式：`freopen("score.in", "r", stdin);`

然后使用freopen()函数以写入方式w(write)打开输出文件score.out。

格式：`freopen("score.out", "w", stdout);`

接下来的事情就是使用freopen()函数的优点了，我们不再需要修改scanf，printf，cin和cout。而是维持代码的原样就可以了。因为freopen()函数重定向了标准流，使其指向前面指定的文件，省时省力。最后只要使用fclose关闭输入文件和输出文件即可。

格式：

```
fclose(stdin);
fclose(stdout);
```

memset 和 fill

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

const int INF = 0x3f3f3f3f;

int main() {

    int a[10] = {1, 2, 3, 4, 5, 6};
    // memset(a, 0, sizeof a);
    // memset(a, 0x3f, sizeof a);
    memset(a, -1, sizeof a);
```

```
    for(auto x : a)
        cout << x << " ";
    puts("");

    /*
    if(a[0] == INF) cout << "YES" << endl;
    else cout << "NO" << endl;
    */

    return 0;
}
```

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

int main() {

    int a[10] = {};
    fill(a + 0, a + 5 + 1, 123);
    for(int i = 9; ~i; i--)
        cout << a[i] << " ";

    return 0;
}
```

快乐刷题

- P142 二进制中1的个数
- P162 寻找独一无二的数