第九章 位运算与常用库函数

C++帮我们实现好了很多有用的函数，我们要避免重复造轮子。

——闫学灿

1. 位运算

& 与

| 或

~ 非

^ 异或

>> 右移

<< 左移

常用操作：

1. 求x的第k位数字 x >> k & 1
2. lowbit(x) = x & -x，返回x的最后一位1
3. 常用库函数、
4. reverse 翻转

翻转一个vector：

reverse(a.begin(), a.end());

翻转一个数组，元素存放在下标1~n：

reverse(a + 1, a + 1 + n);

1. unique 去重

返回去重之后的尾迭代器（或指针），仍然为前闭后开，即这个迭代器是去重之后末尾元素的下一个位置。该函数常用于离散化，利用迭代器（或指针）的减法，可计算出去重后的元素个数。

把一个vector去重：

int m = unique(a.begin(), a.end()) – a.begin();

把一个数组去重，元素存放在下标1~n：

int m = unique(a + 1, a + 1 + n) – (a + 1);

1. random\_shuffle 随机打乱

用法与reverse相同

1. sort

对两个迭代器（或指针）指定的部分进行快速排序。可以在第三个参数传入定义大小比较的函数，或者重载“小于号”运算符。

把一个int数组（元素存放在下标1~n）从大到小排序，传入比较函数：

int a[MAX\_SIZE];

bool cmp(int a, int b) {return a > b; }

sort(a + 1, a + 1 + n, cmp);

把自定义的结构体vector排序，重载“小于号”运算符：

struct rec{ int id, x, y; }

vector<rec> a;

bool operator <(const rec &a, const rec &b) {

return a.x < b.x || a.x == b.x && a.y < b.y;

}

sort(a.begin(), a.end());

1. lower\_bound/upper\_bound 二分

lower\_bound 的第三个参数传入一个元素x，在两个迭代器（指针）指定的部分上执行二分查找，返回指向第一个大于等于x的元素的位置的迭代器（指针）。

upper\_bound 的用法和lower\_bound大致相同，唯一的区别是查找第一个大于x的元素。当然，两个迭代器（指针）指定的部分应该是提前排好序的。

在有序int数组（元素存放在下标1~n）中查找大于等于x的最小整数的下标：

int I = lower\_bound(a + 1, a + 1 + n,. x) – a;

在有序vector<int> 中查找小于等于x的最大整数（假设一定存在）：

int y = \*--upper\_bound(a.begin(), a.end(), x);