Vector 只支持在尾部 O(1)增删元素 随机访问

Queue 首部O(1)删 尾部O(1)加 不支持随机访问

双端队列 首部尾部都支持O(1)增删 随机访问

优先队列 集合 map需要重载小于大于号

支持随机访问的stl容器 只有 deque vector array string map

Set map支持find set 还支持二分查找

求x的第k位数字 x >> k & 1

lowbit(x) = x & -x，返回x的最后一位1

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 + n + 1);

unique去重

返回去重（**只去掉相邻的相同元素**）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 + n + 1) – (a + 1);

random\_shuffle随机打乱

用法与reverse相同。

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 + n + 1, 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());

要先排序后 才能二分

lower\_bound 大于等于

upper\_bound 大于

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);

priority\_queue<int> q; // 大根堆

priority\_queue<int, vector<int>, greater<int>> q; // 小根堆

默认都是从小到大