

2. 向量

(e) 起泡排序

邓俊辉

deng@tsinghua.edu.cn

排序器：统一入口

❖ template <typename T>

```
void Vector<T>::sort(Rank lo, Rank hi) { //区间[lo, hi)

    switch (rand() % 5) { //可视具体问题的特点灵活选取或扩充

        case 1 : bubbleSort(lo, hi); break; //起泡排序

        case 2 : selectionSort(lo, hi); break; //选择排序 (习题)

        case 3 : mergeSort(lo, hi); break; //归并排序

        case 4 : heapSort(lo, hi); break; //堆排序 (第10章)

        default: quickSort(lo, hi); break; //快速排序 (第12章)

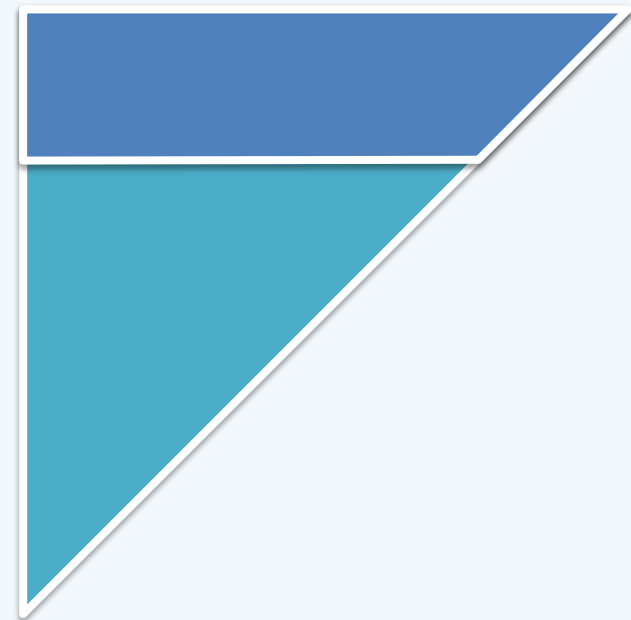
    }

} //在此统一接口下，具体算法的不同实现，将在后续各章节陆续讲解
```

起泡排序

```
❖ template <typename T> void Vector<T>::bubbleSort(Rank lo, Rank hi)
    { while (!bubble(lo, hi--)); } //逐趟做扫描交换，直至全序

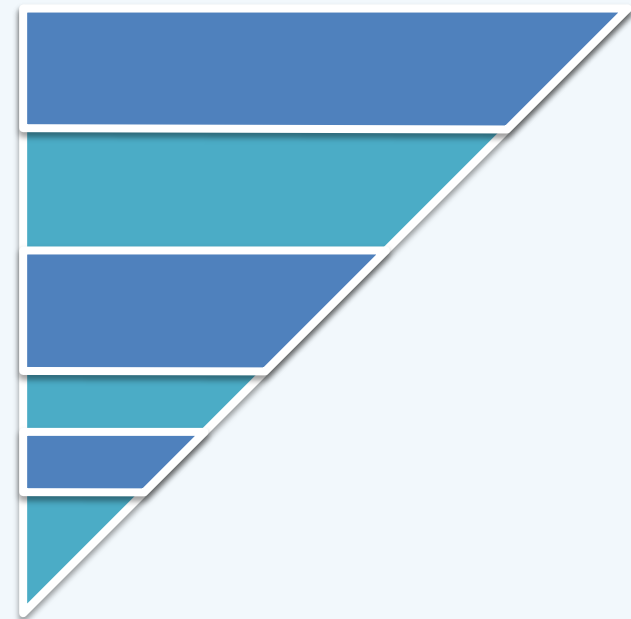
❖ template <typename T> bool Vector<T>::bubble(Rank lo, Rank hi) {
    bool sorted = true; //整体有序标志
    while (++lo < hi) //自左向右，逐一检查各对相邻元素
        if (_elem[lo - 1] > _elem[lo]) { //若逆序，则
            sorted = false; //意味着尚未整体有序，并需要
            swap(_elem[lo - 1], _elem[lo]); //交换
        }
    return sorted; //返回有序标志
} //乱序限于 $[0, \sqrt{n})$ 时，仍需 $O(n^{3/2})$ 时间——按理， $O(n)$ 应已足矣
```



再改进

```
❖ template <typename T> void Vector<T>::bubbleSort(Rank lo, Rank hi)
    { while (lo < (hi = bubble(lo, hi))); } //逐趟扫描交换，直至全序

❖ template <typename T> Rank Vector<T>::bubble(Rank lo, Rank hi) {
    Rank last = lo; //最右侧的逆序对初始化为[lo - 1, lo]
    while (++lo < hi) //自左向右，逐一检查各对相邻元素
        if (_elem[lo - 1] > _elem[lo]) { //若逆序，则
            last = lo; //更新最右侧逆序对位置记录，并
            swap(_elem[lo - 1], _elem[lo]); //交换
        }
    return last; //返回最右侧的逆序对位置
} //前一版本中的逻辑型标志sorted，改为秩last
```



综合评价

- ❖ 效率与第一章针对整数数组的版本相同，最好 $O(n)$ ，最坏 $O(n^2)$
- ❖ 输入含重复元素时，算法的**稳定性** (stability) 是更为细致的要求
重复元素在输入、输出序列中的相对次序，是否保持不变？

输入：6, 7_a , 3, 2, 7_b , 1, 5, 8, 7_c , 4

输出：1, 2, 3, 4, 5, 6, 7_a , 7_b , 7_c , 8 //stable

1, 2, 3, 4, 5, 6, 7_a , 7_c , 7_b , 8 //unstable

以上起泡排序算法是稳定的吗？是的！为什么？

- ❖ 在起泡排序中，元素a和b的相对位置发生变化，只有一种可能：
经分别与其它元素的交换，二者相互**接近**直至**相邻**
在接下来一轮扫描交换中，二者因**逆序**而交换位置
- ❖ 在if一句的判断条件中，若把 “>”换成 “>=”，将有何变化？