# Chapter 9 排序

## 基本概念

排序，就是根据排序码递增或递减的顺序，将数据元素依次排列起来，使一组任意排列的元素变成一组按其排序码线性有序的元素

稳定性：排序不改变排序码相同的元素的相对次序，即称排序是稳定的

内排序与外排序

排序的性能评估：根据算法执行时**数据比较次数**或**数据移动次数**来衡量

## 插入排序

核心思想：每步将一个待排序元素，按其排序码大小，插入到前面一组已经排好序的元素的适当位置

### 直接插入排序

当插入第 $i(i \geqslant 1)$ 个元素时，前面的 $V[0], V[1], \ldots V[i-1]$ 已经有序，用 $V[i]$ 的排序码与之顺序比较，找到插入位置后插入

直接插入排序的性能与待排序元素原始顺序密切相关，平均情况下时间复杂度为 $O(n^2)$

直接插入排序是一个稳定的排序方法

### 折半插入排序

与顺序插入排序不同的在于利用折半搜索寻找 $V[i]$ 的插入位置

性能好于直接插入排序的最差情况，但差于直接插入的最好情况，时间复杂度约为 $O(nlog_2n)$

折半插入排序是一个稳定的排序方法

### 希尔排序

设待排序元素有 $n$ 个，取整数 $gap < n$ 作为间隔，将所有元素分为 $gap$ 个子序列，所有距离为 $gap$ 的元素在同一个子序列中，在每个子序列中分别实施直接插入排序，然后缩小 $gap$，例如取 $gap = \lceil gap/2 \rceil$，重复直至 $gap$ 为 1，将所有元素放在同一序列排序。开始时 $gap$ 较大，每个序列中元素较少，排序较快，之后子序列元素逐渐增多，但大多数元素已基本有序，故排序速度仍然很快

希尔排序是一种不稳定的排序方法

# 快速排序

## 算法思想

分治法，选取某个元素作为基准，将整个序列分为左右两个子序列，左序列元素均小于基准元素，右序列元素均大于基准元素，然后对两个子序列递归排序

快速排序是不稳定的排序方法

```
QuickSort(List){
    if(List 长度大于 1){
        将序列划分为 LeftList, RightList
        QuickSort(LeftList);
        QuickSort(RightList);
        合并 LeftList, RightList
    }
}
```

算法代码如下

```
template<class T>
int partition(T a[], int start, int end) {
    int pivotpos = start;
    T pivot = a[start];                                  // 基准元素
    for (int i = start + 1; i < end; ++i) {
        if (a[i] < pivot) {
            pivotpos++;
            if (pivotpos != i) {
                exch(a[pivotpos], a[i]);          // 小于基准元素的交换到左侧
            }
        }
    }
```

```
    a[start] = a[pivotpos];
    a[pivotpos] = pivot;                          // 将基准元素就位
    return pivotpos;                              // 返回基准元素位置
}

template<class T>
void quicksort(T a[], int start, int end) {
    if (end - start > 1) {
        int pivotpos = partition(a, start, end);
        quicksort(a, start, pivotpos);
        quicksort(a, pivotpos + 1, end);
    }
}
```

## 性能分析

平均情况下时间复杂度为 $O(nlog_2n)$，最大递归调用深度为 $\lceil log_2(n+1) \rceil$，空间复杂度为 $O(log_2n)$，而在最差的情况下会退化为简单排序，递归树退化为单支树。

快速排序在元素较多的平均情况下性能较好，但在元素较少的情况下慢于简单排序

## 改进方法

改进短序列时的排序：在序列长度小于设定值 $M$ （一般取5~25）时采用直接插入排序，或是排序时跳过短序列，排序结束后得到整体基本有序的序列再使用直接插入排序

改进对基准值的寻找：取序列左端点，右端点，中点三者中的中间值，并将中间值交换到右端点的位置

## 三路划分的快速排序

目的：处理序列中有大量重复元素的情况

将序列划为三部分：小于基准元素，等于基准元素，大于基准元素

划分方法：将左序列中所有等于基准元素的放到最左边，右序列中所有等于基准元素的放到最右边，结束后依次交换

# 选择排序

第 $i$ 次排序时在后面 $n-i$ 个待排序元素中选出排序码最小的作为有序序列第 $i$ 个元素

## 直接选择排序

- 在 $V[i],\dots,V[n-1]$ 中寻找最小的元素
- 将其与 $V[i]$ 交换
- 对剩下的 $V[i+1],\dots,V[n-1]$ 重复直至完成排序

直接选择排序是一种不稳定的排序方法

时间复杂度为 $O(n^2)$，且与待排序元素无关，较为固定。直接选择排序在元素规模较大时效率较好，因为相比其他排序方法直接选择排序交换元素次数最少

## 锦标赛排序

构建胜者树，两两比较较小者优胜，最终的胜者是最小的元素，选出后将其的值改为最大值，然后在此基础上重构胜者树，重构的代价为 $O(log_2n)$，重构 $n-1$ 次，时间复杂度为 $O(nlog_2n)$ （参考书 P.414）

## 堆排序

构建最大堆，将堆顶元素交换到最后一个位置，然后对前面 $n-1$ 个元素重构最大堆，再交换堆顶和第倒数第二个位置，重复直至排序完成

堆排序的时间复杂度为 $O(nlog_2n)$

堆排序是一种不稳定的排序方法

# 归并排序

分治法，将序列分为两个长度相等的子序列，对两个子序列分别归并排序，然后再将两个子序列合并

归并排序的时间不依赖排序元素的初始排列，避免了快排的最差情况

时间复杂度为 $O(nlog_2n)$ 且与输入序列无关，即最好，最坏，平均的时间复杂度相同，但是需要一个和源数组一样大的辅助数组。

归并排序是一种稳定的排序方法