选择排序算法-Selection Sort

# 简单选择排序算法原理介绍

## 原理

**选择排序算法**：在一个长度为N的无序数组中，**在第一趟遍历N个数据，找出其中最小的数值与第一个元素交换，第二趟遍历剩下的N-1个数据，找出其中最小的数值与第二个元素交换**......**第N-1趟遍历剩下的2个数据**，找出其中最小的数值与第N-1个元素交换，至此选择排序完成。

**选择排序与打擂台排序有很大的相似之处**，打擂台排序是每一轮通过**多次交换数据**，使最大的或最小的数组到数组的顶端，而选择排序是通过记录最大值或最小值的索引，然后再将最大值或最小值数组与数组顶端数据交换，也就是一轮比较中只交换数据一次或不交换。

假定存在数组 array[0..n-1], 选择排序的核心思想是：

第 i 趟排序是从后面的 n - i + 1（i = 1，2，3，4，. . .，n - 1）个元素中选择一个值最小的元素与该 n - i + 1 个元素的最前门的那个元素交换位置，即与整个序列的第 i 个元素交换位置。如此下去，直到 i = n - 1，排序结束。

也可描述为：

每一趟排序从序列中未排好序的那些元素中选择一个值最小的元素，然后将其与这些未排好序的元素的第一个元素交换位置。

思想：在要排序的一组数中，选出最小（或者最大）的一个数与第1个位置的数交换；然后在剩下的数当中再找最小（或者最大）的与第2个位置的数交换，依次类推，直到第n-1个元素（倒数第二个数）和第n个元素（最后一个数）比较为止。

## 操作方法：

第一趟，从n 个记录中找出关键码最小的记录与第一个记录交换；

第二趟，从第二个记录开始的n-1 个记录中再选出关键码最小的记录与第二个记录交换；

以此类推.....

第i 趟，则从第i 个记录开始的n-i+1 个记录中选出关键码最小的记录与第i 个记录交换，直到整个序列按关键码有序。

## 特点：

1. 算法完成需要 n - 1 趟排序，按照算法的描述，n - 1 趟排序之后数组中的前 n - 1 个元素已经处于相应的位置，第 n 个元素也处于相应的位置上。

2. 第 i 趟排序，实际上就是需要将数组中第 i 个元素放置到数组的合适位置，这里需要一个临时变量 j 来遍历序列中未排好序的那些元素，另一临时变量 d 来记录未排好序的那些元素中值最小的元素的下标值，

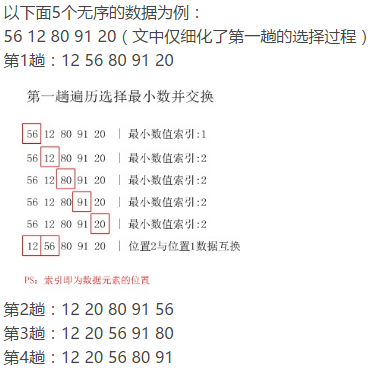
3. 一趟遍历开始时，令 d = i，假定未排序序列的第一个元素就是最小的元素，遍历完成后，变量 d 所对应的值就是值最小的元素，判断 d 是否是未排序序列的第一个元素，如果是，则不需要交换元素，如果不是，则需要交换array[d] 和 array[i]。

4. 此方法是不稳定排序算法，可对数组{a1 = 49，a2 = 38, a3 = 65, a4 = 49, a5 = 12, a6 = 42} 排序就可以看出，排序完成后 a1 和 a4的相对位置改变了。

5. 此方法移动元素的次数比较少，但是不管序列中元素初始排列状态如何，第 i 趟排序都需要进行 n - i 次元素之间的比较，因此总的比较次数为

1 + 2 + 3 + 4 +5 + . . . + n - 1 = n(n-1)/2, 时间复杂度是 O(n^2).

# 示例



# Java代码

**public** **static** **void** selectionSort(**int** a[]){

**int** N = a.length;

**int** temp = 0;

**for**(**int** i = 0;i<N-1;i++){//前面N-1个位置

**int** index = i;**//存放待排序列的最小元素的索引**

**for**(**int** j = i+1;j<N;j++){

**if(a[j] < a[index]){ //关键地方**

index = j;

}

}

**if**(index != i){//只有索引发生改变才交换

temp = a[i];

a[i] = a[index];

a[index] = temp;

}

}

}

# 算法分析

**平均时间复杂度**：O(n2)

**简单选择排序的比较次数与序列的初始排序无关**。 假设待排序的序列有 N 个元素，则比较次数总是**N (N - 1) / 2**。

而移动次数与序列的初始排序有关。

当序列正序时，移动次数最少，为 0；当序列反序时，**移动次数最多**，为**3N (N - 1) / 2**。

所以，综合以上，简单排序的时间复杂度为 **O(N2)**。

**空间复杂度**：O(1) (**用于交换和记录索引**)

简单选择排序需要**占用 1 个临时空间**，**在交换数值时使用**。

**稳定性**：**不稳定** （比如序列【5， 5， 3】第一趟就将第一个[5]与[3]交换，导致第一个5挪动到第二个5后面）

# 简单选择算法排序的改进-二元选择排序

简单选择排序，每趟循环**只能确定一个元素排序后的定位**。我们可以考虑改进为每趟循环确定**两个元素（当前趟最大和最小记录）的位置**,从而**减少排序所需的循环次数**。改进后对n个数据进行排序，最多**只需进行[n/2]趟循环**即可。

因为存在问题，所以至今没有完美的例子。

<http://blog.csdn.net/ye_scofield/article/details/39312717>