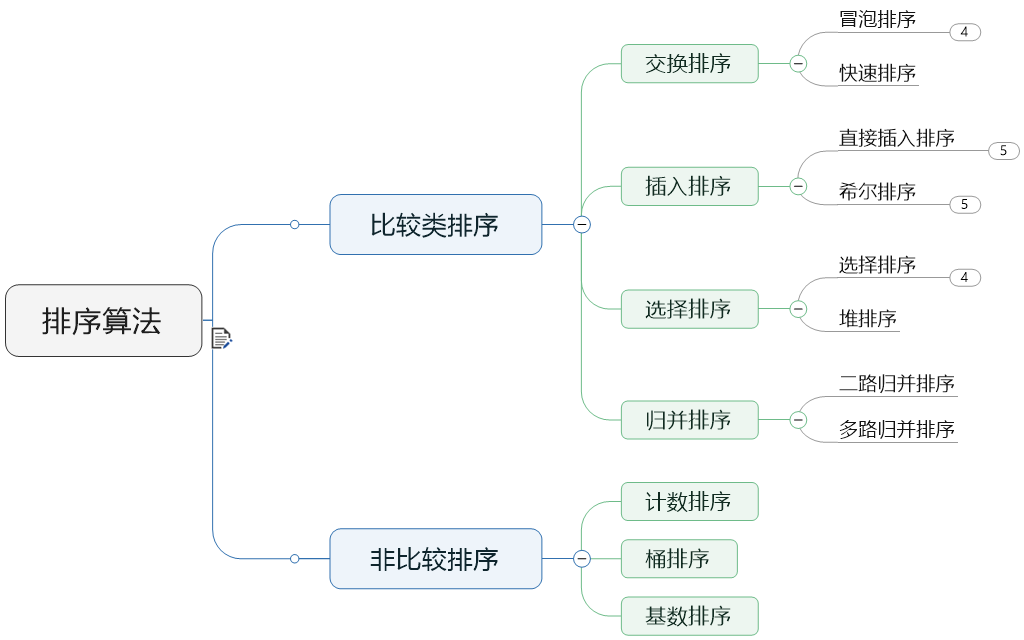
# 选择排序



选择排序算法是对冒泡排序算法进行优化后得到的一种排序算法，相比冒泡排序，选择排序减少了对序列进行排序的过程中元素移动的次数。

选择排序和冒泡排序十分相似，排序过程中其实存在一个待排序集合和已排序集合，每次遍历从待排序集合中找到一个元素，添加到已排序集合，不同之处在于，冒泡排序的极值元素是通过不断的比较和交换位置产生的，选择排序则是不断比较和一次交换位置产生，所以相对冒泡排序，选择排序在性能上占有。

## 原理

描述方式1：

以对序列进行升序排序为例：

1. 从待排序序列中选择一个待排序区间，假设待排序序列长度为n，则待排序区间由[0，n-1]、[0，n-2]、[0，n-3]逐渐减小到[0，1]。
2. 遍历待排序区间，找出待排序区间中的最大元素，遍历结束后将最大元素交换到待排序区间尾部。
3. 当所有待排序区间遍历结束，待排序序列有序。

描述方式2：

遍历序列，找出序列中的最小元素，然后将最小元素交换到序列第一个元素的位置；

遍历序列，找出剩下元素中的最小元素，然后将最小元素交换到序列第二个元素的位置；

依此类推，直到所有元素被交换到正确的位置，序列排序完成。

## 示例

使用选择排序算法对序列89 45 68 90 29 34 17 34 进行排序，排序过程如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 第1次遍历  待遍历区间：[89 45 68 90 29 34 17 34]  寻找最大值：[89 45 68 90 29 34 17 34]  移动最大值：[89 45 68 34 29 34 17 90] | 第2次遍历  待遍历区间：[89 45 68 34 29 34 17] 90  寻找最大值：[89 45 68 34 29 34 17] 90  移动最大值：[17 45 68 34 29 34 89] 90 | 第3次遍历  待遍历区间：[17 45 68 34 29 34] 89 90  寻找最大值：[17 45 68 34 29 34] 89 90  移动最大值：[17 45 34 34 29 68] 89 90 | 第4次遍历  待遍历区间：[17 45 34 34 29] 68 89 90  寻找最大值：[17 45 34 34 29] 68 89 90  移动最大值：[17 29 34 34 45] 68 89 90 |
| 第5次遍历  待遍历区间：[17 29 34 34] 45 68 89 90  寻找最大值：[17 29 34 34] 45 68 89 90  移动最大值：[17 29 34 34] 45 68 89 90 | 第6次遍历  待遍历区间：[17 29 34] 34 45 68 89 90  寻找最大值：[17 29 34] 34 45 68 89 90  移动最大值：[17 29 34] 34 45 68 89 90 | 第7次遍历  待遍历区间：[17 29] 34 34 45 68 89 90  寻找最大值：[17 29] 34 34 45 68 89 90  移动最大值：[17 29] 34 34 45 68 89 90 | 遍历结束  排序结果：17 29 34 34 45 68 89 90 |

## 分析

1. 选择排序是一种不稳定排序。

观察上面示例中的“第3次遍历“，能够发现，序列中两个34的相对位置在排序前后发生了改变： 17 45 68 34 29 34 89 90 → 17 45 34 34 29 68 89 90

2020年3月9日记：

从网上看了一篇文章，大致内容就是选择排序是否稳定与具体实现方式有关，如果是数组实现，那么就是不稳定排序，如果是链表实现，那么就是稳定排序。

1. 相比较冒泡排序，选择排序显著减少了元素移动的次数。

最坏情况下，对一个长度为n的倒序序列进行排序，使用冒泡排序需要移动元素的次数为n(n-1)/2，使用选择排序需要移动元素的次数为n-1。

3) 时间复杂度为O(n2)，空间复杂度为O(1)。

## 优化

使用以下方式对上文的选择排序算法进行优化：

1. 从待排序序列中选择一个待排序区间，假设待排序序列长度为n，则待排序区间为：[0，n-1]、[1，n-2]、[2，n-3]……
2. 遍历待排序区间，找出待排序区间中的最大元素和最小元素，遍历结束后将最大元素和最小元素分别交换到当前待排序区间的尾部和头部。
3. 当所有待排序区间遍历结束，待排序序列有序。

2020年3月9日，新的描述方式

遍历序列，找到序列中的最小元素和最大元素，然后将最小元素交换到序列第一个元素的位置，将最大元素交换到序列倒数第一个元素的位置；

遍历序列，找到剩下元素中的最小元素和最大元素，然后将最小元素交换到序列第二个元素的位置，将最大元素交换到序列倒数第二个元素的位置；

依此类推，直到所有元素被交换到正确的位置，序列排序完成。

需要注意：若待排序区间内的最大元素恰好在区间头部，最小元素恰好在区间尾部，则只能进行一次交换。

示例：使用优化后的选择排序算法对序列89 45 68 90 29 34 17 34 进行排序。

|  |  |  |  |
| --- | --- | --- | --- |
| 第1次遍历  遍历区间：[89 45 68 90 29 34 17 34]  寻找最值：[89 45 68 90 29 34 17 34]  移动最值：[17 45 68 34 29 34 89 90] | 第2次遍历  遍历区间：17 [45 68 34 29 34 89] 90  寻找最值：17 [45 68 34 29 34 89] 90  移动最值：17 [29 68 34 45 34 89] 90 | 第3次遍历  遍历区间：17 29 [68 34 45 34] 89 90  寻找最值：17 29 [68 34 45 34] 89 90  移动最值：17 29 [34 34 45 68] 89 90 | 第4次遍历  遍历区间：17 29 34 [34 45] 68 89 90  寻找最值：17 29 34 [34 45] 68 89 90  移动最值：17 29 34 [34 45] 68 89 90 |
| 第5次遍历  遍历区间：  寻找最值：  移动最值： | 第6次遍历  遍历区间：  寻找最值：  移动最值： | 第7次遍历  遍历区间：  寻找最值：  移动最值： | 遍历结束  排序结果：17 29 34 34 45 68 89 90 |

优化后的选择排序算法先比优化前，排序过程中遍历序列的次数减少了一半。