简单选择排序

- 选择排序
  - 每一趟两两比较大小，找出极值（极大值或极小值）并放置到有序区的位置

初始	1 9 8 5 6 7 4 3 2	
第一趟	9 1 8 5 6 7 4 3 2	选择出此轮最大数9，和索引0数交换
第二趟	9 8 1 5 6 7 4 3 2	选择出此轮最大数8，和索引1数交换
第三趟	9 8 7 5 6 1 4 3 2	以此类推
第四趟	9 8 7 6 5 1 4 3 2	
第五趟	9 8 7 6 5 1 4 3 2	
第六趟	9 8 7 6 5 4 1 3 2	
第七趟	9 8 7 6 5 4 3 1 2	
第八趟	9 8 7 6 5 4 3 2 1	

核心算法

- 结果可为升序或降序排列，默认升序排列。以降序为例
- 扩大有序区，减小无序区。图中红色部分就是增大的有序区，反之就是减小的无序区
- 相邻元素依次两两比较，获得每一次比较后的最大值，并记住此值的索引
- 每一趟都从无序区中选择出最大值，然后交换到当前无序区最左端

算法实现

```
m_list = [
    [1, 9, 8, 5, 6, 7, 4, 3, 2],
    [1, 2, 3, 4, 5, 6, 7, 8, 9],
    [9, 8, 7, 6, 5, 4, 3, 2, 1]
]
nums = m_list[0]
length = len(nums)
print(nums)

count_iter = 0
count_swap = 0
# 选择排序
for i in range(length-1):
    maxindex = i

    for j in range(i+1, length):
        count_iter += 1
        if nums[maxindex] < nums[j]:
            maxindex = j

    if maxindex != i:
        nums[maxindex], nums[i] = nums[i], nums[maxindex]
        count_swap += 1
```

```
print(nums)
print(count_iter, count_swap)
```

二元选择排序

- 同时选择出每一趟的最大值和最小值，并分别固定到两端的有序区
- 减少迭代的趟数

```
m_list = [
    [1, 9, 8, 5, 6, 7, 4, 3, 2],
    [1, 2, 3, 4, 5, 6, 7, 8, 9],
    [9, 8, 7, 6, 5, 4, 3, 2, 1]
]
nums = m_list[1]
length = len(nums)
print(nums)


count_iter = 0
count_swap = 0

for i in range(length//2): # 一次固定2个数，减半
    maxindex = i # 正索引，假设无序区第一个就是最大数，其索引记作最大
    minindex = -i-1 # 负索引，假设无序区最后一个就是最小数，其索引记作最小

    for j in range(i+1, length-i): # 每次左边加一个，右边也要减一个，表示无序区两端都减少
        count_iter += 1
        if nums[maxindex] < nums[j]:
            maxindex = j
        if nums[minindex] > nums[-j-1]:
            minindex = -j -1
    #print(maxindex, i, "|||", minindex, -i-1)

    if maxindex != i:
        nums[maxindex], nums[i] = nums[i], nums[maxindex]
        count_swap += 1
        # [1, 3, 2]为例，如果i位置上就是最小值，走到这里，说明最大值和最小值交换过了，要调整最小值索引为maxindex
        if i == length + minindex:
            minindex = maxindex - length

    if minindex != -i-1: # 负索引比较
        nums[minindex], nums[-i-1]= nums[-i-1], nums[minindex]
        count_swap += 1

print(nums)
print(count_iter, count_swap)
```

以上代码还有没有优化的可能?

如果一趟比较后，极大值、极小值的值相等，说明什么?

说明，剩余比较的数将全部相等，那么排序可以立即停止。

```
m_list = [
    [1, 9, 8, 5, 6, 7, 4, 3, 2],
    [1, 2, 3, 4, 5, 6, 7, 8, 9],
    [9, 8, 7, 6, 5, 4, 3, 2, 1]
]
nums = m_list[1]
length = len(nums)
print(nums)


count_iter = 0
count_swap = 0

for i in range(length//2): # 一次固定2个数，减半
    maxindex = i # 正索引，假设无序区第一个就是最大数，其索引记作最大
    minindex = -i-1 # 负索引，假设无序区最后一个就是最小数，其索引记作最小

    for j in range(i+1, length-i): # 每次左边加一个，右边也要减一个，表示无序区两端都减少
        count_iter += 1
        if nums[maxindex] < nums[j]:
            maxindex = j
        if nums[minindex] > nums[-j-1]:
            minindex = -j -1
    #print(maxindex, i, "|||", minindex, -i-1)
    if nums[maxindex] == nums[minindex]: # 元素全相同
        break

    if maxindex != i:
        nums[maxindex], nums[i] = nums[i], nums[maxindex]
        count_swap += 1
        # [1, 3, 2]为例，如果i位置上就是最小值，走到这里，说明最大值和最小值交换过了，要调整最小值索引为maxindex
        if i == length + minindex:
            minindex = maxindex - length

    if minindex != -i-1: # 负索引比较
        nums[minindex], nums[-i-1]= nums[-i-1], nums[minindex]
        count_swap += 1

print(nums)
print(count_iter, count_swap)
```

考虑一种特殊情况

[1, 1, 1, 1, 1, 1, 1, 1, 2] 这种情况，找到的最小值索引是-2，最大值索引8，上面的代码会交换2次，最小值两个1交换是无用功，所以，增加一个判断

```
m_list = [
    [1, 9, 8, 5, 6, 7, 4, 3, 2],
    [1, 2, 3, 4, 5, 6, 7, 8, 9],
    [9, 8, 7, 6, 5, 4, 3, 2, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1],
```

```
    [1, 1, 1, 2]
]
nums = m_list[4]
length = len(nums)
print(nums)


count_iter = 0
count_swap = 0

for i in range(length//2): # 一次固定2个数，减半
    maxindex = i # 正索引，假设无序区第一个就是最大数，其索引记作最大
    minindex = -i-1 # 负索引，假设无序区最后一个就是最小数，其索引记作最小

    for j in range(i+1, length-i): # 每次左边加一个，右边也要减一个，表示无序区两端都减少
        count_iter += 1
        if nums[maxindex] < nums[j]:
            maxindex = j
        if nums[minindex] > nums[-j-1]:
            minindex = -j -1
    #print(maxindex, i, "|||", minindex, -i-1)
    if nums[maxindex] == nums[minindex]: # 元素全相同
        break

    if maxindex != i:
        nums[maxindex], nums[i] = nums[i], nums[maxindex]
        count_swap += 1
        # [1, 3, 2]为例，如果i位置上就是最小值，走到这里，说明最大值和最小值交换过了，要调整最小值索引为maxindex
        if i == length + minindex:
            minindex = maxindex - length

    if minindex != -i-1 and nums[minindex] != nums[-i-1]: # 负索引比较，值不一样再交换
        nums[minindex], nums[-i-1]= nums[-i-1], nums[minindex]
        count_swap += 1

print(nums)
print(count_iter, count_swap)
```

总结

- 简单选择排序需要数据一趟趟比较，并在每一趟中发现极值
- 没有办法知道当前这一趟是否已经达到排序要求，但是可以知道极值是否在目标索引位置上
- 遍历次数1,...,n-1之和n(n-1)/2
- 时间复杂度O(n^2)
- 减少了交换次数，提高了效率，性能略好于冒泡法

马哥教育
IT人的高薪职业学院