选择排序（Selection sort）

Back to series (/series/algorithms-in-swift3)

字号
字号
默认主题
金色主题
暗色主题

SelectionSort

Selection sort是一种和insertion sort类似的排序方法，它同样只适用于对规模不大的集合进行排序。

它的核心思想是，在序列内部，把序列逻辑上分成已排序和未排序两部分，不断找到未排序部分中最符合排序规则的元素，添加进已排序部分，直到序列中所有元素都已经添加到了已排序部分，此时，整个序列就排序完成了。

我们用数组 `[1, 5, 7, 6]` 进行降序排列来举例，整个过程是这样的：

1. 我们用一个竖线来区分序列中“已排序”和“未排序”的部分，初始时，“已排序“部分为空，”未排序“部分是整个序列；

```
[ | 1, 5, 7, 6 ]
```

2. 然后，第一个应该从“未排序部分”取出的元素，是7。把7和紧挨竖线右侧的数字交换，并且把竖线向右移动一个位置；

```
[ | 1, 5, 7, 6 ]
    *<--->*
[ | 7, 5, 1, 6 ]
  ^-->
[ 7, | 5, 1, 6 ]
```

3. 然后，继续从“未排序部分”取出元素6，它和5交换，并且把竖线移动一个位置：

```
[ 7, | 5, 1, 6 ]
       *<--->*
[ 7, | 6, 1, 5 ]
     ^-->
[ 7, 6, | 1, 5 ]
```

4. 接下来是5，交换它和1的位置，然后把竖线向右移动一个位置：

```
[ 7, 6, | 1, 5 ]
          *  *
[ 7, 6, | 5, 1 ]
          ^-->
[ 7, 6, 5, | 1 ]
```

5. 最后，当序列中“未排序部分”只有一个元素的时候，实际上整个序列就已经全部排序完成了。

实现

理解和选择排序的核心思想之后，实现的部分就很简单了。首先是和插入排序类似的“套路声明”：

```
typealias Criteria<T> = (T, T) -> Bool
func SelectionSortOf<T: Comparable>(_ coll: Array<T>,
                     byCriteria: Criteria<T> = { $0 > $1 }) -> Array<T> {

    guard coll.count > 1 else { return coll }

    var result = coll

    // TODO: add implementation here
    return result
}
```

关于声明中和Swift 3相关的内容，大家可以参考插入排序中的相关说明(https://boxueio.com/series/algorithms-in-swift3/ebook/85)，我们就不再重复了。直接来看实现。

首先，遍历数组中的 0 - (N-1) 元素，每一次迭代，都表示把竖线右移，在“未排序部分”找下一个要交换的元素，在执行真正的交换前，我们先打印了序列的“已排序”和“未排序”部分；

```
// 1. Increase the sorted sub array
for x in 0 ..< coll.count - 1 {
    var candidateIndex = x

    print("--------------------------")
    print("Sorted:\t\(result[0 ..< candidateIndex])")
    print("Unsorted:\t\(result[candidateIndex ..< result.count])")
}
```

其次，我们再嵌套一个 for 循环，用于在“未排序部分”寻找下一个要交换的元素：

```
// 1. Increase the sorted sub array
for x in 0 ..< coll.count - 1 {
   var candidateIndex = x
   print("--------------------------")
   print("Sorted:\t\(result[0 ..< candidateIndex])")
   print("Unsorted:\t\(result[candidateIndex ..< result.count])")

   // 2. Find the next element to be moved into the sorted portion
   for y in x + 1 ..< coll.count {
       if byCriteria(result[candidateIndex], result[y]) {
           candidateIndex = y
       }
   }
}
```

第三，找到后，由于Swift不允许交换同一个位置的元素，我们需要判断下“待移进已排序数组”中的元素是否需要交换：

```
// 1. Increase the sorted sub array
for x in 0 ..< coll.count - 1 {
   var candidateIndex = x
   print("--------------------------")
   print("Sorted:\t\(result[0 ..< candidateIndex])")
   print("Unsorted:\t\(result[candidateIndex ..< result.count])")

   // 2. Find the next element to be moved into the sorted portion
   for y in x + 1 ..< coll.count {
       if byCriteria(result[candidateIndex], result[y]) {
           candidateIndex = y
       }
   }

   // 3. Swap the candidate into sorted sub array if needed
   print(">>> Move \(result[candidateIndex]) into the sorted portion")
   if (candidateIndex != x) {
       swap(&result[candidateIndex], &result[x])
   }
}
```

最后，当除了最后一个元素之外的其他元素都已经移进“已排序”部分时，整个序列就已经排序完成了，我们直接把 result 返回：

```
typealias Criteria<T> = (T, T) -> Bool
func SelectionSortOf<T: Comparable>(_ coll: Array<T>,
                    byCriteria: Criteria<T> = { $0 < $1 }) -> Array<T> {

    guard coll.count > 1 else { return coll }

    var result = coll

    // 1. Increase the sorted sub array
    for x in 0 ..< coll.count - 1 {
        var candidateIndex = x
        print("--------------------------")
        print("Sorted:\t\(result[0 ..< candidateIndex])")
        print("Unsorted:\t\(result[candidateIndex ..< result.count])")

        // 2. Find the next element to be moved into the sorted portion
        for y in x + 1 ..< coll.count {
            if byCriteria(result[candidateIndex], result[y]) {
                candidateIndex = y
            }
        }

        // 3. Swap the candidate into sorted sub array if needed
        print(">>> Move \(result[candidateIndex]) into the sorted portion"
)
        if (candidateIndex != x) {
            swap(&result[candidateIndex], &result[x])
        }
    }

    return result
}
```

测试

用一开始的 [1, 5, 7, 6] 来测试。

首先是默认的升序排列：

```
let a: Array<Int> = [1, 5, 7, 6]
SelectionSortOf(a)
```

```
--------------------------
Sorted:    []
Unsorted:  [1, 5, 7, 6]
>>> Move 1 into the sorted portion
--------------------------
Sorted:    [1]
Unsorted:  [5, 7, 6]
>>> Move 5 into the sorted portion
--------------------------
Sorted:    [1, 5]
Unsorted:  [7, 6]
>>> Move 6 into the sorted portion
```

然后是自定义的降序排列：

```
let a: Array<Int> = [1, 5, 7, 6]
SelectionSortOf(a, byCriteria: <)
```

```
------------------------
Sorted:    []
Unsorted:  [1, 5, 7, 6]
>>> Move 7 into the sorted portion
------------------------
Sorted:    [7]
Unsorted:  [5, 1, 6]
>>> Move 6 into the sorted portion
------------------------
Sorted:    [7, 6]
Unsorted:  [1, 5]
>>> Move 5 into the sorted portion
```

从这些结果里，我们就能看到选择排序的执行流程了。

⏮ 插入排序（Insertion sort）
(https://www.boxueio.com/series/algorithms-in-swift3/ebook/85)

使用SPM构建开发环境 ⏭
(https://www.boxueio.com/series/algorithms-in-swift3/ebook/123)