10.3 选择排序

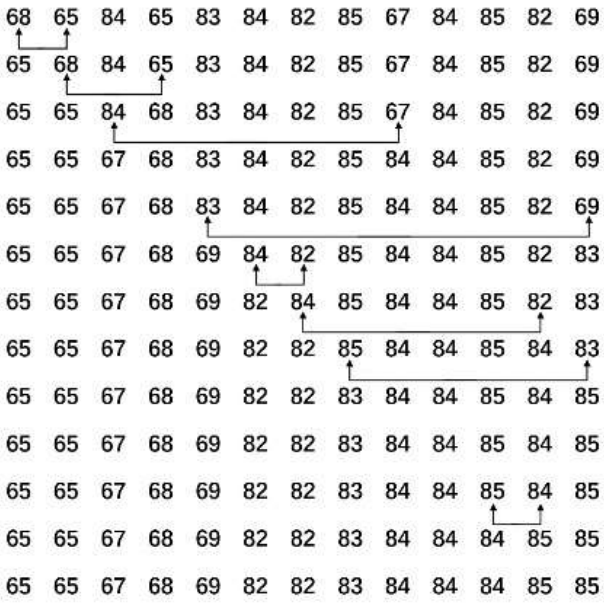
基本思想：首先选出键值**最小**的项，将它与**第一个项**交换位置；然后选出键值**次小**的项，将其与**第二个项**交换位置；...；直到整个序列完成排序。

步骤：

- 假设待排序的序列为$a_0, a_1, a_2, \cdots, a_{n-1}$
- 依次对$i = 0, 1, \cdots, n-2$执行如下步骤：
  - 在$a_i, a_{i+1}, \cdots, a_{n-1}$中选择一个键值最小的项$a_k$
  - 将a_i与a_k交换。

10.3.1 简单选择排序

简单选择排序在从待排序序列中选出键值最小的项时，所用的策略是简单的逐个枚举法。



简单选择排序过程

简单选择排序伪代码

算法10-3 简单选择排序 $SelectionSort(a,l,r)$

输入：序列a，左端点下标l，右端点下标r

输出：调整$a_l, a_{l+1}, \ldots, a_r$元素顺序，使元素按照非递减顺序排列

```
1     for i ← l to r − 1 do //依次从剩余未排序序列中选取一个最小的记录
2     | min ← i
3     | for j ← i + 1 to r do
4     | | if a_j < a_min then
5     | | | min ← j
6     | | end
7     | end
8     | t ← a_i //将当前的最小记录放入已排序好的队列的末尾
9     | a_i ← a_min
10    | a_min ← t
10    end
```

简单选择排序性能分析

时间复杂度：

最佳情况（有序）：$O(n^2)$

- $\frac{n(n-1)}{2}$ 次比较，0 次移动.

最坏情况（逆序）：$O(n^2)$

- $\frac{n(n-1)}{2}$ 次比较，$n-1$ 次移动.

平均情况：$O(n^2)$

- $O(n^2)$ 次比较，$O(n)$ 次移动

简单选择排序性能分析

空间复杂度：$O(1)$

排序时仅交换未排序序列中最值与未排序序列起始位置，不涉及额外空间的使用。

稳定性分析：不稳定

交换操作时可能会破坏具有相同key值元素的相对位置。因此选择排序为不稳定排序。

运行时间与记录顺序关系很小

交换次数很少

简单选择排序的应用：煎饼排序（LeetCode 969）

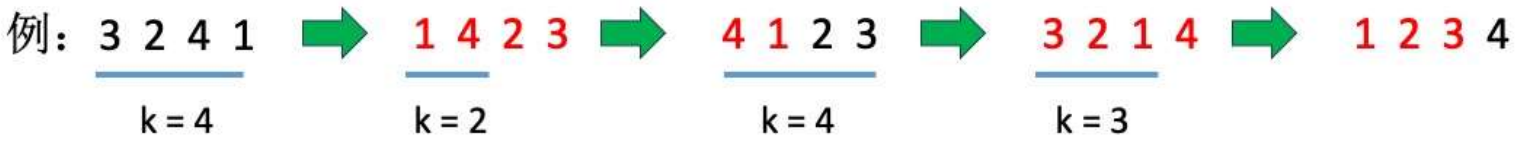
问题描述：

给定长度为n的正整数序列，且序列中所有元素互异。使用下面翻煎饼的手法(?)对序列排序。设当前序列为$a_1, a_2, \ldots, a_n$：

（1）选择一个整数k，满足$1 \le k \le n$

（2）反转序列前k个元素的子串 $a_1, \ldots, a_k$

持续上述操作，直到序列从小到大排好序为止，输出每次反转选择的k值。



简单选择排序的应用：煎饼排序（LeetCode 969）

问题描述：

给定长度为n的正整数序列，且序列中所有元素互异。使用下面翻煎饼的手法(?)对序列排序。设当前序列为$a_1, a_2, \ldots, a_n$：

（1）选择一个整数k，满足$1 \le k \le n$

（2）反转序列前k个元素的子串 $a_1, \ldots, a_k$

持续上述操作，直到序列从小到大排好序为止，输出每次反转选择的k值。

基于选择排序的算法思路：从后往前排(why？)，依次将序列前段（未排好序）中的最大值通过反转移到后段（已排序）的先头位置！

- 假设当前序列为：$a_1, a_2, \ldots, a_j, a'_{j+1}, \ldots, a'_n$，其中$a'_{j+1}, \ldots, a'_n$已排好序！

1) 查找$a_1, a_2, \ldots, a_j$中的最大值 $a_m = \max\{a_1, \ldots, a_j\}\ (1 \le m \le j)$

2) $\underbrace{a_1, \ldots, a_m}_{k=m}, \ldots, a_j, a'_{j+1}, \ldots, a'_n$ —反转→ $\underbrace{a_m, \ldots, a_1}_{k=j}, \ldots, a_j, a'_{j+1}, \ldots, a'_n$ —反转→ $a_j, \ldots, a_1, \ldots, a_m, a'_{j+1}, \ldots, a'_n$

a'_j = a_m

思考：什么情况下，不需要反转或只反转1次？

高等教育出版社

简单选择排序的应用：煎饼排序（LeetCode 969）

问题描述：

给定长度为n的正整数序列，且序列中所有元素互异。使用下面翻煎饼的方式(?)对序列排序。设当前序列为$a_1, a_2, \ldots, a_n$：

（1）选择一个整数k，满足$1 \le k \le n$

（2）反转序列前k个元素的子串 $a_1, \ldots, a_k$

持续上述操作，直到序列从小到大排好序为止，输出每次反转选择的k值。

基于选择排序的算法思路：从后往前排(why？)，依次将序列前段（未排好序）中的最大值通过反转移到后段（已排序）的先头位置！

例：3 2 4 1 ⇒ 4 2 3 1 ⇒ 1 3 2 4 ⇒ 3 1 2 4 ⇒ 2 1 3 4 ⇒ 1 2 3 4

k = 3　　k = 4　　k = 2　　k = 3　　k = 2

- 不是最优算法！！！

单选题 1分

煎饼排序长度n的序列，反转次数的最大值和最小值分别是：

A 2n 和 0

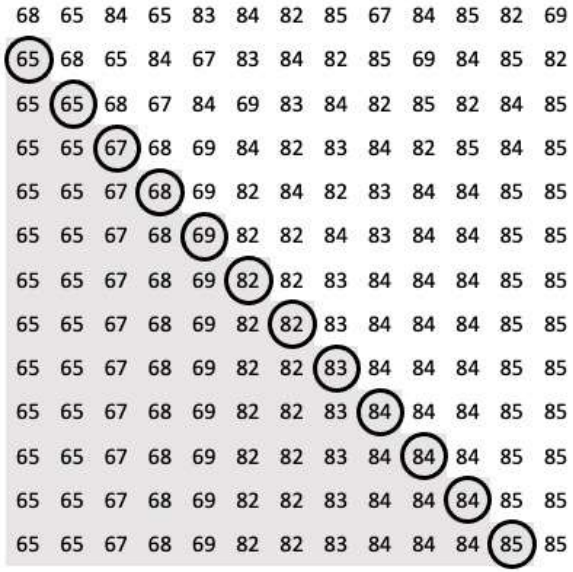
B 2n-1 和 0

C 2n-2 和 0

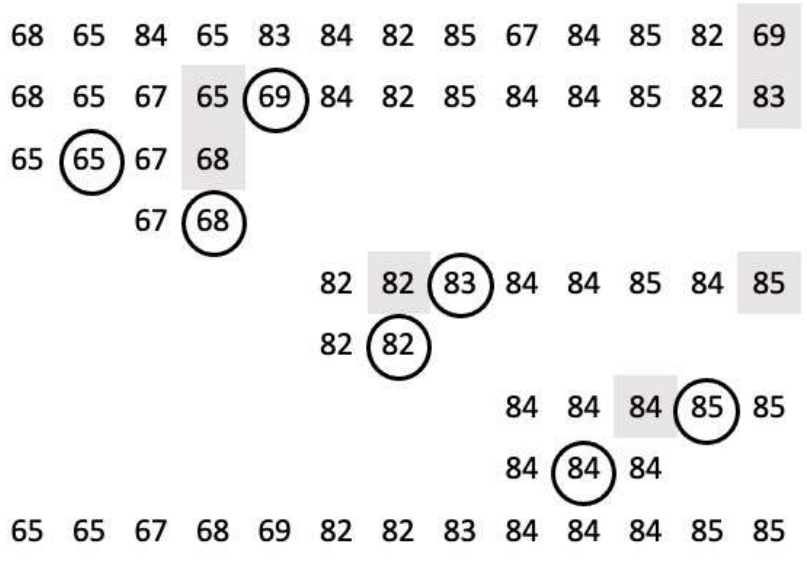
D 2n-3 和 0

10.4 交换排序

核心思路：对序列中的元素进行多次两两交换，从而使序列元素有序。



冒泡排序



快速排序

10.4.1 冒泡排序

基本思想：依次比较**相邻**的两个元素的顺序，如果顺序不对，则将两者交换，重复这样的操作直到整个序列被排好序。

通过比较与交换使得待排序列一个**最值**元素“上浮”到序列**一端**，然后缩小排序范围。

排序步骤：（以升序为例）

- 假设待排序序列为 $a_0, a_1, \ldots, a_{n-1}$。
- 起始时排序范围为 a_0 到 a_{n-1}。
- **从右向左**对相邻两元素进行比较，较大值向右移，较小值向左移。比较完当前排序范围后，键值**最小**的元素被移动到 a_0 位置。
- 下一次排序范围是 a_1 到 a_{n-1}。

冒泡排序示例

待排序数组	68	65	84	65	83	84	82	85	67	84	85	82	69
第一轮	(65)	68	65	84	67	83	84	82	85	69	84	85	82
第二轮	65	(65)	68	67	84	69	83	84	82	85	82	84	85
第三轮	65	65	(67)	68	69	84	82	83	84	82	85	84	85
第四轮	65	65	67	(68)	69	82	84	82	83	84	84	85	85
第五轮	65	65	67	68	(69)	82	82	84	83	84	84	85	85
第六轮	65	65	67	68	69	(82)	82	83	84	84	84	85	85
第七轮	65	65	67	68	69	82	(82)	83	84	84	84	85	85
第八轮	65	65	67	68	69	82	82	(83)	84	84	84	85	85
第九轮	65	65	67	68	69	82	82	83	(84)	84	84	85	85
第十轮	65	65	67	68	69	82	82	83	84	(84)	84	85	85
第十一轮	65	65	67	68	69	82	82	83	84	84	(84)	85	85
第十二轮	65	65	67	68	69	82	82	83	84	84	84	(85)	85


未排序
已排序
本轮排序

- 用蓝色标注每轮“冒”上去的数据

已无“泡”可“冒”

高等教育出版社

冒泡排序伪代码

算法10-5 冒泡排序 BubbleSort(a,l,r)

输入： 序列a，左端点下标l，右端点下标r

输出： 调整$a_l, a_{l+1}, \ldots, a_r$元素顺序，使元素按照非递减顺序排列

```
1  for i ← l to r do
2  |  for j ← r − 1 downto i do
3  |  |  if a_j > a_{j+1} then  //比较相邻位置的元素
4  |  |  |  Swap(a_j, a_{j+1})  //两个元素交换位置
5  |  |  end
6  |  end
7  end
```

冒泡排序性能分析

假设待排序的序列长度为n，那么冒泡排序共执行n轮，第i轮时最多需要执行$n-i$次比较和交换。因此，冒泡排序最多的比较和交换次数为：

$$\sum_{i=1}^{n} n-i = \frac{n(n-1)}{2}$$

所以，冒泡排序的时间复杂度为$\boldsymbol{O(n^2)}$。

- 冒泡排序仅对数组中的相邻元素进行比较和交换，因此关键字值相同的元素不会改变顺序。所以，冒泡排序是**稳定**的。
- 但注意，一旦将算法10-5中比较运算$a_j > a_{j+1}$改为$a_j \geq a_{j+1}$，算法就失去了稳定性。
- 另外，冒泡排序中的交换次数又称为反序数或逆序数，可用于体现数列的错乱程度。

冒泡排序性能分析

逆序对：设对序列$a_1, a_2, \ldots, a_n$按升序（非降序）排序，则序列中的任意一对元素a_i和a_j构成逆序对，当且仅当$i < j \wedge a_i > a_j$。

- 例如：序列2 4 3 1 包含的逆序对有(2,1)、(4,1)、(4,3)和(3,1)

- 当序列已经有序时，逆序对数量为0
- 当序列逆序时，逆序对数量最多，等于$\frac{n(n-1)}{2}$

- 排序算法设计的核心思想就是如何快速消除序列中的所有逆序对！

单选题 1分

设序列$a_1, \ldots a_i, \ldots, a_j, \ldots, a_n$中，$(a_i, a_j)$是逆序对，则交换这两个元素，对序列的逆序对总数有何影响？

A 一定减少1个逆序对

B 一定减少逆序对，且有可能数量大于1

C 不一定会减少，但一定不会增加逆序对

D 有可能会增加逆序对的数量

单选题 1分

设序列$a_1, \ldots a_i, a_{i+1}, \ldots, a_n$中，$(a_i, a_{i+1})$是逆序对，则交换这两个元素，对序列的逆序对总数有何影响？

A 一定减少1个逆序对

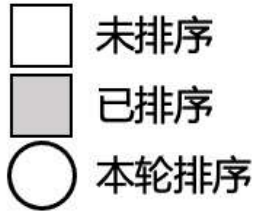
B 一定减少逆序对的总数，且有可能数量大于1

C 不一定会减少，但一定不会增加逆序对

D 有可能会增加逆序对的数量

冒泡排序示例

待排序数组	68	65	84	65	83	84	82	85	67	84	85	82	69
第一轮	(65)	68	65	84	67	83	84	82	85	69	84	85	82
第二轮	65	(65)	68	67	84	69	83	84	82	85	82	84	85
第三轮	65	65	(67)	68	69	84	82	83	84	82	85	84	85
第四轮	65	65	67	(68)	69	82	84	82	83	84	84	85	85
第五轮	65	65	67	68	(69)	82	82	84	83	84	84	85	85
第六轮	65	65	67	68	69	(82)	82	83	84	84	84	85	85
第七轮	65	65	67	68	69	82	(82)	83	84	84	84	85	85
第八轮	65	65	67	68	69	82	82	(83)	84	84	84	85	85
第九轮	65	65	67	68	69	82	82	83	(84)	84	84	85	85
第十轮	65	65	67	68	69	82	82	83	84	(84)	84	85	85
第十一轮	65	65	67	68	69	82	82	83	84	84	(84)	85	85
第十二轮	65	65	67	68	69	82	82	83	84	84	84	(85)	85



- 用蓝色标注每轮“冒”上去的数据

无“泡”可“冒”

序列中无逆序对！

*冒泡排序的改进

算法：　最优化冒泡排序 OptimizedBubbleSort(a, l, r)

输入： 序列a，左端点下标l，右端点下标r

输出： 调整$a_l, a_{l+1}, \ldots, a_r$元素顺序，使元素按照非递减顺序排列

```
1   sorted ← false   //排序结果的初始值
2   i ← l
3   while sorted = false 且 i < r do
4   |   sorted ← true   //假定已排好序（无逆序对）
5   |   for j ← r − 1 downto i do
6   |   |   if a_j > a_{j+1} then   //存在逆序对，需要继续排序
7   |   |   |   Swap(a_j, a_{j+1})   //两个元素交换位置
8   |   |   |   sorted ← false   //序列未排好序，继续
9   |   |   end
10  |   end
11  |   i ← i + 1
12  end
```

- 交换次数 = 逆序对总数

- 最好时间复杂度：O(n) --含比较次数
- 最坏时间复杂度：O(n^2)—序列逆序

最优化冒泡排序示例

待排序数组	68	65	84	65	83	84	82	85	67	84	85	82	69
第一轮	65	68	65	84	67	83	84	82	85	69	84	85	82
第二轮	65	65	68	67	84	69	83	84	82	85	82	84	85
第三轮	65	65	67	68	69	84	82	83	84	82	85	84	85
第四轮	65	65	67	68	69	82	84	82	83	84	84	85	85
第五轮	65	65	67	68	69	82	82	84	83	84	84	85	85
第六轮	65	65	67	68	69	82	82	83	84	84	84	85	85
第七轮													
第八轮													
第九轮													
第十轮													
第十一轮													
第十二轮													

未排序

已排序

本轮排序

- 用蓝色标注每轮“冒”上去的数据

- 无“冒泡”，排序结束

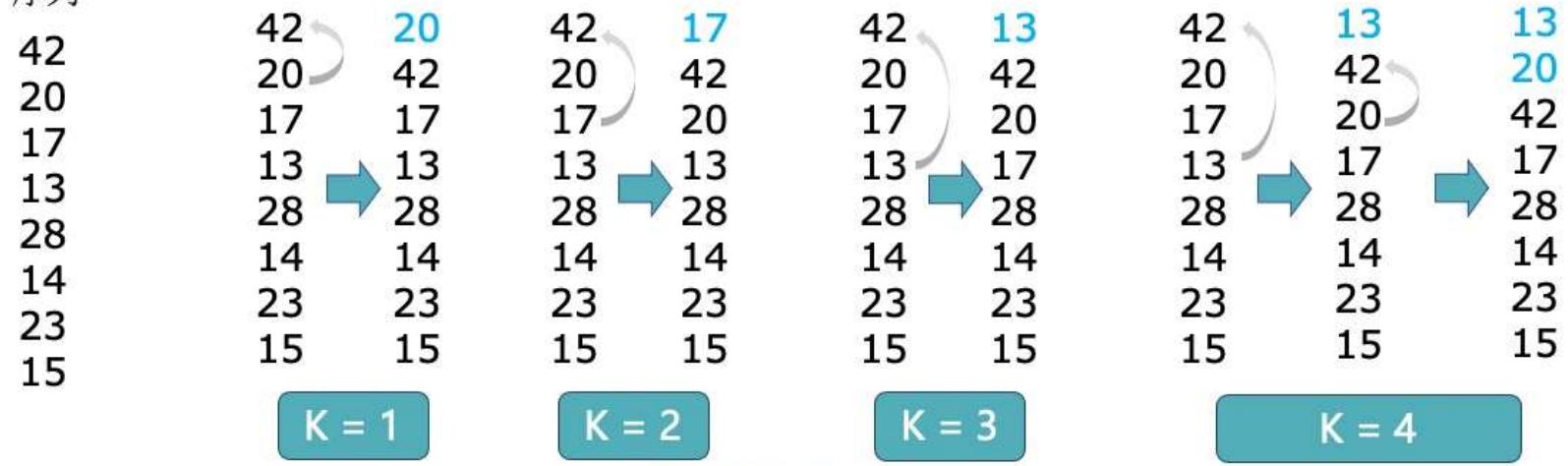
高等教育出版社

*冒泡排序的应用：最多交换K次冒泡排序

问题描述：

冒泡排序通过交换相邻元素来调整元素的顺序。限制相邻元素的交换次数不超过K，输出字典序最小的排序结果。

越靠前的元素越小越好！



*冒泡排序的应用：最多交换K次冒泡排序

问题描述：

冒泡排序通过交换相邻元素来调整元素的顺序。限制相邻元素的交换次数不超过K，输出字典序最小的排序结果。

越靠前的元素越小越好！

- 剩下的交换次数$0 \le k \le K$

target position

- 目标：使第i个位置的数据越小越好！

$< \underbrace{a_1, a_2, \dots, a_{i-1}}_{\text{Sorted}}, a_i, a_{i+1}, \dots, a_{i+k}, \dots, a_n >$

- 查找$a_i, a_{i+1}, \dots, a_{i+k}$中的最小值（？）

$a_j = \min\{a_i, .., a_{i+k}\}$

$< a_1, a_2, \dots, a_{i-1}, a_i, \dots, a_j, \dots, a_{i+k}, \dots, a_n >$

- 通过交换将a_j移到序列第i个位置

$< \underbrace{a_1, a_2, \dots, a_{i-1}, a_j}_{\text{Sorted}}, a_i \dots, a_{j-1}, \dots, a_{i+k}, \dots, a_n >$

target position

- 目标位置：i+1
- 剩下交换次数：k－j＋i

投票 最多可选1项

基本排序法：插入、选择和冒泡，都可对存放在单链表上的数据序列进行排序，其中时间效率受影响最大的算法是哪个？

A 插入排序

B 选择排序

C 冒泡排序

10.4.2 快速排序

快速排序是一种所需**比较次数较少、速度较快**的排序方法，由英国计算机科学家，图灵奖获得者东尼·霍尔于1961年发表。在基于比较的排序算法中，快速排序的**平均性能突出**。

基本思想：通过递归分治方法，基于轴点将待排序序列拆分成两个子序列并分别排序，直到序列有序。

排序步骤：

- 从待排序序列中**选取轴点**。
- 通过交换序列元素，将待排序序列**拆分**为左右两个子序列，左子序列元素**小于等于**轴点，右子序列元素**大于等于**轴点。
- 对两个子序列**递归**进行上述操作，直到子序列**元素个数小于等于1**。