Gwxjf-4.indd 27 2023/9/15 15:29:11

3.1 知识点定位

青少年编程能力“Python 四级”核心知识点 2：排序算法。

3.2 能 力 要 求

能够解释并实现冒泡排序、选择排序、直接插入排序算法，根据应用场景和数据存储结构选择合适的算法完成排序。

3.3 建议教学时长

本单元建议 6 课时。

3.4 教 学 目 标

1. 知识目标

本单元主要学习排序算法的算法思想和代码实现，帮助学习者理解冒泡排序、选择排序、直接插入排序算法的适用场景，掌握各排序算法的时间复杂度

Gwxjf-4.indd 28 2023/9/15 15:29:18

和空间复杂度。

2. 能力目标

学习者能够根据应用场景和数据存储结构选择合适的算法完成排序。

3. 素养目标

通过对各排序算法的学习，理解程序设计中的那些灵光一闪和精巧构思，感受算法在社会生活和工作中发挥的重要作用，通过对算法从感性到理性的认知提升过程，获得对计算机科学的探知欲望。

3.5 知识结构

本单元知识结构如图 3-1 所示。

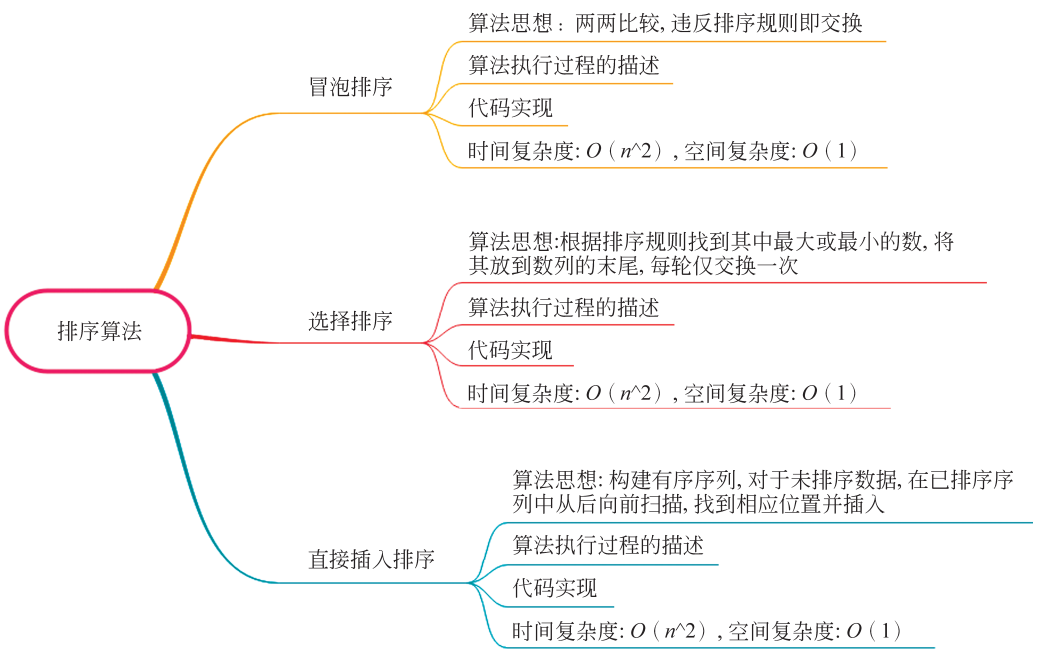

图 3-1　排序算法知识结构

Gwxjf-4.indd 29　2023/9/15 15:29:30

3.6 补充知识

1. 十种常见排序算法

除教材中提及的冒泡排序、选择排序、直接插入排序外，常见的排序算法还有：希尔排序、堆排序、快速排序、归并排序、计数排序、桶排序、基数排序等。十种常见排序算法可以分为以下两大类。

（1）比较类排序：通过比较来决定元素间的相对次序，由于其时间复杂度不能突破 $O(n\log n)$，因此也称为非线性时间比较类排序。

（2）非比较类排序：不通过比较来决定元素间的相对次序，它可以突破基于比较排序的时间下界，以线性时间运行，因此也称为线性时间非比较类排序。

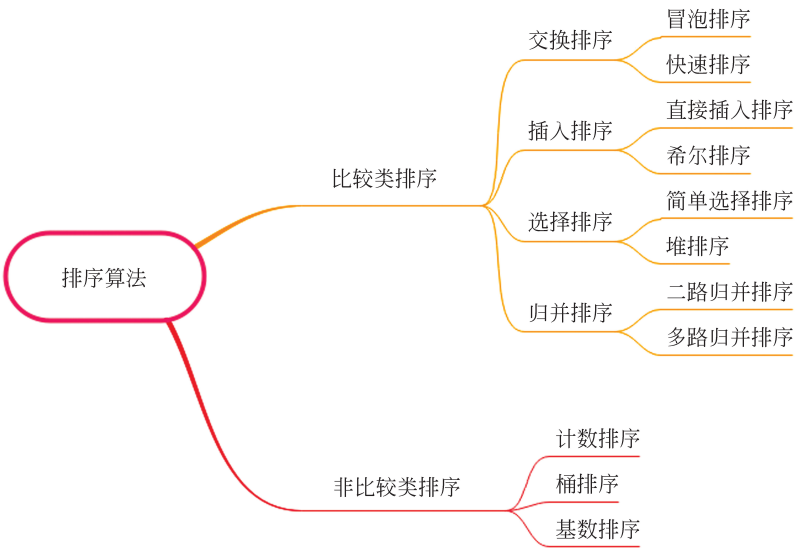

图 3-2　常见排序算法

2. 常见排序算法的复杂度

常见排序算法的复杂度如表 3-1 所示。

Gwxjf-4.indd 30　　2023/9/15 15:29:39

表 3-1　常见排序算法的复杂度

排序方法	时间复杂度（平均）	时间复杂度（最坏）	时间复杂度（最好）	空间复杂度	稳定性
插入排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定
希尔排序	$O(n^{1.3})$	$O(n^2)$	$O(n)$	$O(1)$	不稳定
选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
堆排序	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(1)$	不稳定
冒泡排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定
快速排序	$O(n\log n)$	$O(n^2)$	$O(n)$	$O(n\log n)$	不稳定
归并排序	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n)$	稳定
计数排序	$O(n+k)$	$O(n+k)$	$O(n+k)$	$O(n+k)$	稳定
桶排序	$O(n+k)$	$O(n^2)$	$O(n)$	$O(n+k)$	稳定
基数排序	$O(n\times k)$	$O(n\times k)$	$O(n\times k)$	$O(n+k)$	稳定

算法稳定性并未在学生用书中提及，其含义如下。

（1）稳定：如果 a 原本在 b 前面，而 $a=b$，排序之后 a 仍然在 b 的前面。

（2）不稳定：如果 a 原本在 b 的前面，而 $a=b$，排序之后 a 可能会出现在 b 的后面。

3. 希尔排序算法

希尔排序（Shell Sort）是插入排序的一种，又称“缩小增量排序”，是直接插入排序算法的一种更高效的改进版本。希尔排序是非稳定算法，因 DL. Shell 于 1959 年提出而得名。

算法思想：

设一个序列里有 n 个待排序的元素，将间隔相同距离的元素分为一组进行比较，这里的间隔称为增量（gap），增量通常为 $n/2$（奇偶数均可），随着算法的进行，增量慢慢缩小（通常为原增量的 1/2），当增量减至 1 时，所有元素被分成一组，直到相邻的元素比较完，结束排序。

排序过程演示：

假设有序列 8 1 5 4 6 2 3 7，现在要将它们按照升序排列。

Gwxjf-4.indd 31　2023/9/15 15:29:41

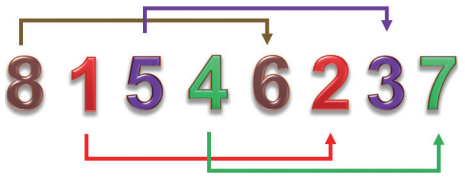

图 3-3　第一轮排序分组

第一轮排序：序列长度为 8，增量可为 4、2、1（每结束一轮排序，增量减半），首先增量为 4 时，将间隔相同距离的元素分为一组进行比较，如图 3-3 所示。

第一组：8 和 6 进行比较，8 > 6，两个元素交换位置（进行升序排序，小元素在前，大元素在后）。

第二组：1 和 2 进行比较，1 < 2，两个元素位置不变。

第三组：5 和 3 进行比较，5 > 3，两个元素交换位置。

第四组：4 和 7 进行比较，4 < 7，两个元素位置不变。

第一轮比较结束，结果如图 3-4 所示。

第二轮排序：接下来增量为 2 重新对序列进行分组排序，共分为两组，如图 3-5 所示。

图 3-4　第一轮排序结果

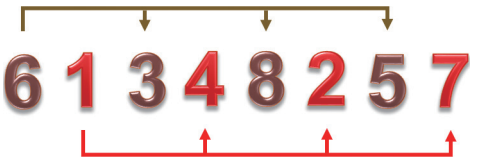

图 3-5　第二轮排序分组

第一组：6、3、8、5 进行比较，小的元素在前，大的元素在后，结果为 3、5、6、8。

第一组：1、4、2、7 进行比较，小的元素在前，大的元素在后，结果为 1、2、4、7。

第二轮比较结束，结果如图 3-6 所示。

第三轮排序：接下来增量为 1 重新对数组进行分组排序，所有元素被分为一组。可以看出，当增量为 1 时变为直接插入排序，最后一轮排序结束后得到一个升序数列，如图 3-7 所示。

图 3-6　第二轮排序结果

图 3-7　第三轮排序结果

希尔排序算法运行过程描述如下。

步骤 1：选定一个增量 h（一般是序列长度的一半），按照增量 h 作为数据分组的依据，对数据进行分组。

步骤 2：对分好组的每一组数据完成插入排序。

Gwxjf-4.indd 32　2023/9/15 15:29:43

步骤 3：减小增量（最小减为 1），按照增量对数据进行分组。

步骤 4：重复步骤 2、步骤 3 直至排序完成。

4. 归并排序算法

归并排序（Merge Sort）是利用归并的思想实现的排序方法，该算法采用经典的分治（divide-and-conquer）策略。分治法将问题分 (divide) 成一些小的问题然后递归求解，而治 (conquer) 的阶段则将分的阶段得到的各答案“修补”在一起，即分而治之。

算法思想：

把序列从中间划分成两个子序列；一直递归地把子序列划分成更小的子序列，直到子序列里面只有一个元素；依次按照递归的返回顺序，不断地合并排好序的子序列，直至整个序列完成排序。

排序过程演示：

假设有序列 8 4 5 7 1 3 6 2，现在要将它们按照升序排列。

根据分而治之的思想，整个算法的排序过程可以分解为“分”和“治”两个阶段，如图 3-8 所示。

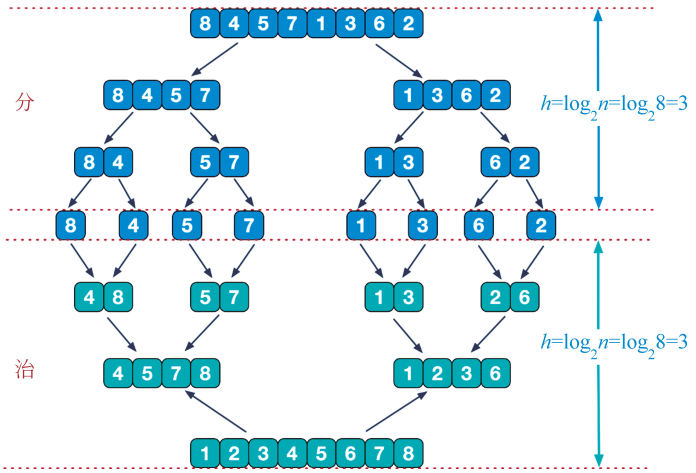

图 3-8　归并排序的分治演示

“分”阶段可以理解为递归拆分子序列的过程，递归深度为 $\log_2 n$。

在“治”阶段，需要将两个已经有序的子序列合并成一个有序序列，如图 3-8 中的最后一次合并，要将 [4,5,7,8] 和 [1,2,3,6] 两个已经有序的子序列，合并为最终序列 [1,2,3,4,5,6,7,8]，实现步骤如图 3-9 所示。

Gwxjf-4.indd 33　2023/9/15 15:29:49

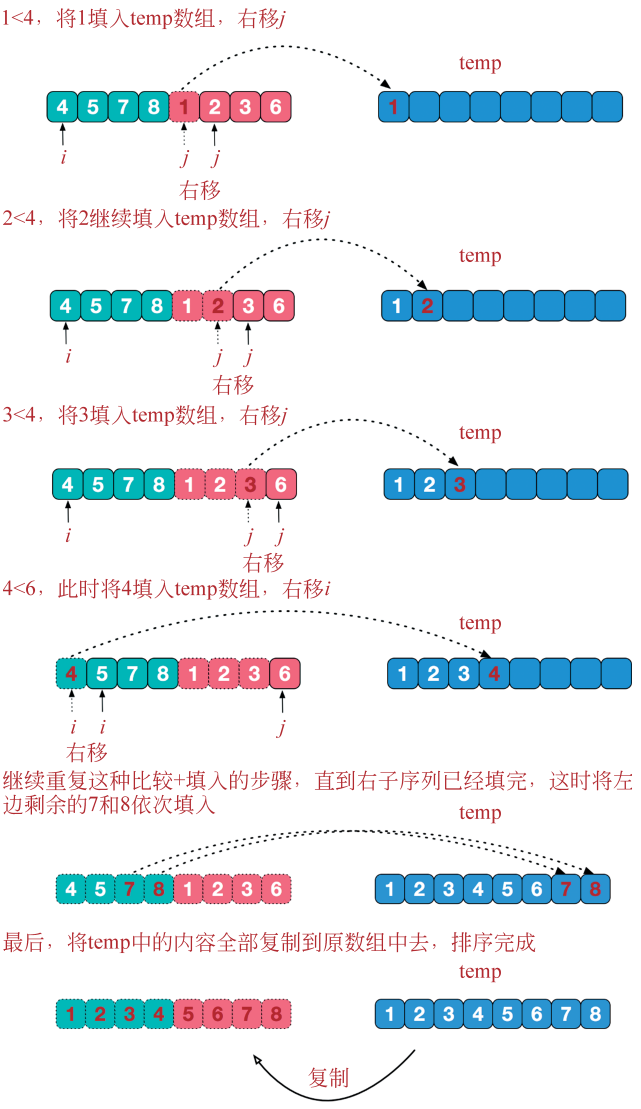

图 3-9　子序列合并

Gwxjf-4.indd 34　2023/9/15 15:30:01

3.7　教学组织安排

教 学 环 节	教 学 过 程	建议课时
知识导入	以虚拟人物间的对话为例，风趣地引出“排序”的概念，在此过程中注意点出以下两个重要的知识点。 （1）对于排序算法来讲，比较其优劣最直观的方式就是看完成排序时比较次数的多少。 （2）排序算法的执行效率和数据规模有关	2 课时
冒泡排序 （原理讲授）	（1）对算法的实现原理进行简要介绍，引发学生思考，通过对算法名称的形象化比喻，激发学生的学习兴趣。 （2）结合案例对算法的操作过程进行实例讲解，经过一轮排序后，让学生自主完成后续排序过程，引导其发现待排序数据个数和排序轮次之间的规律。 （3）对算法的操作步骤进行总结、凝练。 （4）引导学生思考升降序排列时算法比较规则的变化	
冒泡排序 （代码实现）	（1）对冒泡排序算法的代码实现进行讲解。 （2）布置课堂练习，对学生进行辅导，使其当堂完成代码编写	
冒泡排序 （算法复杂度分析）	对冒泡排序算法的时间复杂度和空间复杂度分析方法进行讲解	
选择排序 （原理讲授）	（1）对算法的实现原理进行简要介绍，通过与冒泡排序进行比较，指出二者间的区别：选择排序在每轮排序中只做一次交换。 （2）结合案例对算法的操作过程进行实例讲解，引导学生自主完成算法操作步骤的总结、凝练。 （3）对算法的操作步骤进行讲解。 （4）引导学生思考升降序排列时算法比较规则的变化	2 课时
选择排序 （代码实现）	（1）对选择排序算法的代码实现进行讲解。 （2）布置课堂练习，对学生进行辅导，使其当堂完成代码编写	

续表

教 学 环 节	教 学 过 程	建议课时
选择排序 （算法复杂度分析）	对选择排序算法的时间复杂度和空间复杂度分析方法进行讲解	
直接插入排序 （原理讲授）	（1）对直接插入算法的实现原理进行简要介绍。 （2）结合案例对算法的实现原理和操作过程进行实例讲解，引导学生自主完成算法操作步骤的总结、凝练。 （3）对算法的操作步骤进行讲解。 （4）引导学生思考升降序排列时算法比较规则的变化	2 课时
直接插入排序 （代码实现）	（1）对直接插入排序算法的代码实现进行讲解。 （2）布置课堂练习，对学生进行辅导，使其当堂完成代码编写	
直接插入排序 （算法复杂度分析）	对直接插入排序算法的时间复杂度和空间复杂度分析方法进行讲解	
单元总结	对本讲知识进行总结归纳，布置课后作业和拓展阅读内容	

3.8 教学实施参考

1. 讨论式知识导入

（1）以小萌和小帅的对话为例，引出排序的概念。

（2）抛开算法细节，给出讨论排序算法时两个重要的背景知识：①对于排序算法来讲，比较其优劣最直观的方式就是看完成排序时比较次数的多少；②排序算法的执行效率和数据规模有关。

2. 知识点一：冒泡排序的算法思想

（1）通过对算法名字由来的讨论，引发学生思考，对算法产生初步认识。

（2）结合示例对冒泡排序的算法思想进行讲解，冒泡排序的特点在于“两

Gwxjf-4.indd 36 2023/9/15 15:30:03

两比较，违反排序规则即交换”。

（3）对 n 个数进行排序所需的轮次为 n–1 轮。

3. 知识点二：冒泡排序的操作过程描述

冒泡排序的算法过程如下（以升序为例）。

步骤 1：比较相邻的元素。如果大的在前，就交换它们。

步骤 2：对每一对相邻元素做同样的操作，从开始第一对到结尾的最后一对，这样在最后的元素会是最大的数。

步骤 3：针对所有的元素重复以上的步骤，除了最后一个元素。

步骤 4：重复步骤 1~3，直到排序完成。

4. 知识点三：“升序排列”变“降序排列”时的规则更改

通过虚拟人物的提问引出实现“降序排列”时的比较规则变化，即“比较两个数时，如果小的在前，那么就交换位置，反之则不交换”。

5. 知识点四：冒泡排序的代码实现

对冒泡排序的核心关键代码进行讲解，配合练习题帮助学生完成知识内化吸收。

6. 知识点五：冒泡排序的算法复杂度计算

（1）冒泡排序总的比较次数是前 n–1 个数的和，即 $\frac{1}{2}n^2-\frac{1}{2}n$，所以时间复杂度为 $O(n^2)$。

（2）最好的情况下时间复杂度为 $O(n)$。

（3）冒泡排序不需要借助额外的存储空间，其空间复杂度为 $O(1)$。

7. 知识点六：选择排序的算法思想

（1）结合示例对选择排序的算法思想进行讲解，选择排序的特点在于“根据排序规则（升序或降序）找到其中最大或最小的数，将其放到序列的末尾”。

（2）选择排序每轮仅需交换一次。

（3）对 n 个数进行排序所需的轮次为 n–1 轮。

8. 知识点七：选择排序的操作过程描述

选择排序的运算过程可以总结如下（以升序为例）。

步骤 1：从当前序列中找出最大的元素，将它与序列的队尾元素交换，如其为队尾元素，则在原位置不动。

步骤 2：针对所有的元素重复以上的步骤，除了最后一个。

步骤 3：重复步骤 1、步骤 2，直至排序完成。

9. 知识点八：选择排序的代码实现

对选择排序的核心关键代码进行讲解，配合练习题帮助学生完成知识内化吸收。

10. 知识点九：选择排序的算法复杂度计算

（1）不管在最好或最坏情况下，选择排序的时间复杂度都是 $O(n^2)$。

（2）选择排序每轮只交换一次，所以其运行速度比冒泡排序更快。

（3）选择排序不需要借助额外的存储空间，其空间复杂度为 $O(1)$。

11. 知识点十：直接插入排序的算法思想

（1）结合示例对直接插入排序的算法思想进行讲解，直接插入排序的特点在于“通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入”。

（2）直接插入排序所谓“构建”有序序列，其实并非真的创建，而是将序列中位于队首的数据视为有序序列的第一个元素。

（3）对 n 个数进行排序所需的轮次为 n–1 轮。

12. 知识点十一：直接插入排序的操作过程描述

直接插入排序的运算过程描述如下（以升序为例）。

Gwxjf-4.indd 38 2023/9/15 15:30:09

步骤 1：首先将队首位置的元素视为“有序列表”中的第一个元素，其余元素视为“无序列表”。

步骤 2：将“无序列表”中的第一个元素 a 与“有序列表”中的元素从后向前依次进行比较。当“有序列表”中某个元素 b 比 a 大时，将元素 b 右移，继续向前进行比较，直到“有序列表”中没有比 a 大的元素时停止，将 a 插入；如“有序列表”中没有比 a 大的元素，将 a 插入到“有序列表”队尾；如果待插入的元素 a 与有序序列中的元素 b 相等，则将元素 a 插入到相等元素 b 的后面。

步骤 3：重复步骤 1、步骤 2，直至“无序列表”中元素为空，则排序完成。

13. 知识点十二：直接插入排序的代码实现

对直接插入排序的核心关键代码进行讲解，配合练习题帮助学生完成知识内化吸收。

14. 知识点十三：直接插入排序的算法复杂度计算

（1）直接插入排序的算法时间复杂度为 $O(n^2)$。

（2）最好的情况下时间复杂度为 $O(n)$。

（3）直接插入排序不需要借助额外的存储空间，其空间复杂度为 $O(1)$。

15. 单元总结

（1）小结本次课内容，布置课后作业。

（2）布置拓展阅读内容：希尔排序、归并排序、快速排序、堆排序。

3.9 拓 展 练 习

编写程序实现如下的功能。

（1）针对给定列表 `alist = [54, 26, 93, 17, 77, 31, 44, 55, 20]`，使用希尔排序算法编写函数 shellSort( ) 实现 alist 中元素的降序排列。

Gwxjf-4.indd 39　2023/9/15 15:30:11

（2）调用 shellSort( ) 函数，输出排序后的列表。

3.10 问题解答

【问题 3-1】 请使用冒泡排序实现如下列表中元素的降序排列。

```
alist = [55,26,83,17,88,34,47,59,22]
```

```
def bubbleSort(alist):
    # 每次遍历元素个数递减
    for num in range(len(alist)-1,0,-1):
        for i in range(num):
            if alist[i] < alist[i+1]:
                alist[i],alist[i+1] = alist[i+1],alist[i]
arr = [55,26,83,17,88,34,47,59,22]
bubbleSort(arr)
for i in range(len(arr)):
print("{} ".format(arr[i]))
```

【问题 3-2】 请使用选择排序实现如下列表中元素的降序排列。

```
alist = [55,26,83,17,88,34,47,59,22]
```

```
def selectionSort(alist):
     # 每次遍历元素个数递减
    for i in range(len(alist)-1,0,-1):
        # 指定初始的最小值下标为 0
        min_idx = 0
    # 遍历剩余元素，依次与当前最小值比较
        for j in range(1,i+1):
            if alist[j] < alist[min_idx]:
                # 更新最小值下标
                min_idx = j
        # 将最小值与末尾元素交换
```


Gwxjf-4.indd 40 2023/9/15 15:30:12

```
        alist[i],alist[min_idx] = alist[min_idx],alist[i]
arr = [55,26,83,17,88,34,47,59,22]
selectionSort(arr)
for i in range(len(arr)):
    print("{} ".format(arr[i]))
```

【问题 3-3】 请使用直接插入排序实现如下列表中元素的降序排列。

```
alist = [55,26,83,17,88,34,47,59,22]
```

```
def insertSort(alist):
  # 将第一个元素视为有序，其余元素为无序，依次遍历无序元素
    for idx in range(1,len(alist)):
        # 记录当前待插入无序元素的值和下标
        current = alist[idx]
        position = idx
        # 从后向前依次与有序元素比较，如果该元素比待插入值小，将其后移，
        # 直到有序列表中没有比其小的元素时停止
        while position >0 and alist[position-1] < current:
            alist[position] = alist[position-1]
            position = position - 1
        # 将待插入元素插入到当前位置
        alist[position] = current
arr = [55,26,83,17,88,34,47,59,22]
insertSort(arr)
for i in range(len(arr)):
    print("{} ".format(arr[i]))
```

3.11　第 3 单元习题答案

1. A　2. C　3. D　4. A　5. A　6. B　7. B　8. A　9. C

Gwxjf-4.indd 41　2023/9/15 15:30:12

10. 编程题

```
def bubbleSort(alist):
    # 每次遍历元素个数递减
    for num in range(len(alist)-1,0,-1):
        for i in range(num):
            if alist[i] < alist[i+1]:
                alist[i],alist[i+1] = alist[i+1],alist[i]
arr = [55,26,83,17,88,34,47,59,22]
bubbleSort(arr)
for i in range(len(arr)):
    print("{} ".format(arr[i]))
```

11. 编程题

```
def selectionSort(alist):
     # 每次遍历元素个数递减
    for i in range(len(alist)-1,0,-1):
        # 指定初始的最小值下标为 0
        min_idx = 0
        # 遍历剩余元素，依次与当前最小值比较
        for j in range(1,i+1):
            if alist[j] > alist[min_idx]:
                # 更新最小值下标
                min_idx = j
        # 将最小值与末尾元素交换
        alist[i],alist[min_idx] = alist[min_idx],alist[i]
arr = [55,26,83,17,88,34,47,59,22]
selectionSort(arr)
for i in range(len(arr)):
    print("{} ".format(arr[i]))
```

本单元资源下载可扫描下方二维码。

扩展资源

Gwxjf-4.indd 42 2023/9/15 15:30:15