

普通高中

教学参考资料

# 信息技术

选择性必修 1 数据与数据结构

普通高中  
教学参考资料

# 信息技术

选择性必修 1  
数据与数据结构

**总主编:** 李晓明

**副总主编:** 赵健 李锋

**本册主编:** 顾秋辉

**编写人员(按姓氏笔画排序):**

白晓琦 周刚 顾秋辉

**责任编辑:** 丁倩

**美术设计:** 卢晓红 储平

**普通高中 信息技术 选择性必修 1 数据与数据结构 教学参考资料**

**上海市中小学(幼儿园)课程改革委员会组织编写**

---

**出版发行** 华东师范大学出版社(上海市中山北路 3663 号)

**印 刷** 上海昌鑫龙印务有限公司

**版 次** 2022 年 8 月第 1 版

**印 次** 2025 年 8 月第 7 次

**开 本** 890 毫米×1240 毫米 1/16

**印 张** 14

**字 数** 318 千字

**书 号** ISBN 978-7-5760-2951-2

**定 价** 30.00 元

---

版权所有 • 未经许可不得采用任何方式擅自复制或使用本产品任何部分 • 违者必究

如发现内容质量问题,请拨打电话 021-60821714

如发现印、装质量问题,影响阅读,请与华东师范大学出版社联系。电话: 021-60821711

全国物价举报电话: 12315

# 说 明

《普通高中 信息技术 选择性必修 1 数据与数据结构 教学参考资料》根据教育部颁布的《普通高中信息技术课程标准(2017 年版 2020 年修订)》和高中信息技术教科书的内容和要求编写,与信息技术教科书配套,供高中二年级使用。

本书由华东师范大学、上海市信息技术教育教学研究基地(上海高校“立德树人”人文社会科学重点研究基地)主持编写,经上海市中小学教材审查委员会审查准予使用。

编写过程中,上海市中小学(幼儿园)课程改革委员会专家工作委员会、上海市教育委员会教学研究室、上海市课程方案教育教学研究基地、上海市心理教育教学研究基地、上海市基础教育教材建设研究基地等单位给予了大力支持。在此表示感谢!

欢迎广大师生来电来函指出书中的差错和不足,提出宝贵意见。出版社电话: 021 - 60821711。

**声明** 按照《中华人民共和国著作权法》第二十五条有关规定,我们已尽量寻找著作  
权人支付报酬。著作权人如有关于支付报酬事宜,可及时与出版社联系。

# 目 录



## 第一章 引言

---

一、本章学科核心素养的渗透	1
二、本章知识结构	2
三、本章项目活动设计思路	2
四、本章课时安排建议	2

### 第一节 数据 3

一、教学目标与重点	3
二、教学实施与评价	3
三、核心概念介绍	5
四、教学参考资源	5
五、教学参考案例	6

### 第二节 数据结构 9

一、教学目标与重点	9
二、教学实施与评价	10
三、作业练习与提示	14
四、核心概念介绍	15
五、教学参考资源	15
六、教学参考案例	16

## **第二章 数据类型**

---

一、本章学科核心素养的渗透	22
二、本章知识结构	23
三、本章项目活动设计思路	24
四、本章课时安排建议	24

### **第一节 基本类型 25**

一、教学目标与重点	25
二、教学实施与评价	25
三、作业练习与提示	27
四、核心概念介绍	27
五、教学参考资源	27
六、教学参考案例	28

### **第二节 常用类型 35**

一、教学目标与重点	35
二、教学实施与评价	35
三、作业练习与提示	39
四、核心概念介绍	40
五、教学参考资源	40
六、教学参考案例	41

### **第三节 组合类型 46**

一、教学目标与重点	46
二、教学实施与评价	46
三、作业练习与提示	47
四、核心概念介绍	48
五、教学参考资源	48
六、教学参考案例	48

#### **第四节 抽象类型 54**

一、教学目标与重点	54
二、教学实施与评价	55
三、作业练习与提示	57
四、核心概念介绍	57
五、教学参考资源	57
六、教学参考案例	58

### **第三章 基础数据结构**

---

一、本章学科核心素养的渗透	66
二、本章知识结构	67
三、本章项目活动设计思路	67
四、本章课时安排建议	67

#### **第一节 线性表 68**

一、教学目标与重点	68
二、教学实施与评价	68
三、作业练习与提示	79
四、核心概念介绍	81
五、教学参考资源	81
六、教学参考案例	84

#### **第二节 栈 89**

一、教学目标与重点	89
二、教学实施与评价	90
三、作业练习与提示	95
四、核心概念介绍	96
五、教学参考资源	97
六、教学参考案例	98

### **第三节 队列 105**

一、教学目标与重点	105
二、教学实施与评价	106
三、作业练习与提示	112
四、核心概念介绍	115
五、教学参考资源	115
六、教学参考案例	117

## **第四章 常用数据结构**

---

一、本章学科核心素养的渗透	123
二、本章知识结构	124
三、本章项目活动设计思路	124
四、本章课时安排建议	124

### **第一节 二叉树 125**

一、教学目标与重点	125
二、教学实施与评价	125
三、作业练习与提示	131
四、核心概念介绍	142
五、教学参考资源	142
六、教学参考案例	144

### **第二节 图 148**

一、教学目标与重点	148
二、教学实施与评价	149
三、作业练习与提示	151
四、核心概念介绍	153
五、教学参考资源	156
六、教学参考案例	158

### **第三节 哈希表 163**

- 一、教学目标与重点 163
- 二、教学实施与评价 163
- 三、作业练习与提示 165
- 四、核心概念介绍 167
- 五、教学参考资源 168
- 六、教学参考案例 169

## **第五章 数据结构应用**

---

- 一、本章学科核心素养的渗透 176
- 二、本章知识结构 177
- 三、本章项目活动设计思路 177
- 四、本章课时安排建议 177

### **第一节 排序 178**

- 一、教学目标与重点 178
- 二、教学实施与评价 178
- 三、作业练习与提示 183
- 四、核心概念介绍 187
- 五、教学参考资源 188
- 六、教学参考案例 191

### **第二节 查找 196**

- 一、教学目标与重点 196
- 二、教学实施与评价 197
- 三、作业练习与提示 200
- 四、核心概念介绍 204
- 五、教学参考资源 204
- 六、教学参考案例 205

# 第一章

## 引言

### 一、本章学科核心素养的渗透

数据和数据结构是计算机科学中两个重要的概念,本章在内容的选择和组织上结合“中学生健康数据”等生活中的实例,从一般性的数据含义到计算机科学中的数据类型,再到数据结构,逐步展开介绍。通过本章学习,学生能够认识数据、数据类型和数据结构。

本章是选择性必修1 数据与数据结构模块的起点。《普通高中信息技术课程标准(2017年版 2020年修订)》(以下简称《课程标准》)中相关内容要求包括:

1.1 通过列举实例,分析数据与社会各领域的关系,理解数字、数值和数据的基本含义。

1.2 通过列举实例,认识到数据作为新的原材料、生产资料和基础设施的价值与意义。

1.3 结合生活实际,理解数据结构的概念,认识数据结构在解决问题过程中的重要作用。

本章以“随时提供有序的报名信息”为主题,围绕方案讨论、设计数据类型、选择数据结构,再到程序实现等环节展开活动,落实《课程标准》要求。

本章从程序设计解决问题的需要和追求程序运行高效性角度分析多种数据类型和丰富数据结构的必要性。本章强调数据结构中讨论的数据主要是算法和程序直接处理的数据;强调数据结构的动态性及和算法的相互作用;分析数据类型与数据结构,二者一个在于事物的表示和数据属性的记录,一个在于过程的支持和数据之间关系的记录,在这些概念的辨析过程中加强对于数据类型和数据结构的意义和作用的认识,从而培养信息意识。

教科书中安排了“制造 TypeError”探究活动,在实践体验和解决问题的过程中提高计算思维。在整个章节中通过硬件原理分析、程序编写演示、案例性能对比等表达出如下一些思想和观念:数据类型,提高程序编写的效率;数据结构,提高程序执行的效率;程序设计中,数据类型与数据结构是数据具有的两个特征;程序运行中,数据结构与算法互动。同时,教科书中设计了作业练习和项目实践,在完成这些任务过程中提高数字化学习与创新能力。

## 二、本章知识结构

本章遵循《课程标准》，依据学分和课时规定，紧扣学科概念体系，将知识内容分为两节。从对数据认识的三个层面，引出程序中的数据，然后分析数据类型对于高效编写程序的必要性。从提高程序运行效率的角度分析导入数据结构的意义和作用。

第一节数据，主要包括数据的概念和数据类型。通过对各种数据的举例、对各种数据类型的类比加强对于概念的辨析。在查看不同形式的数据（Word 文件、程序中的变量）、修改文件扩展名的活动中体会数据类型的意义。

第二节数据结构，主要包括数据结构的概念。通过接收数据流等体验活动，感受数据结构在计算机问题求解中的基本作用。通过类比理解数据结构与算法的互动关系，通过实例体会数据结构与数据类型的相融性，理解其内在的关联和相互依存关系。

## 三、本章项目活动设计思路

“引言”作为本书的第一章，对于后面的学习有着重要的指导意义。在项目活动中，围绕“这门课是什么”“它为什么重要”这两个问题展开教学。本章对贯穿全书的数据、数据类型和数据结构这三个概念予以阐述，要求理解数据类型和数据结构这两个概念，建议从数据类型和数据结构的由来和作用引出这两个概念。

以项目活动为引领，用项目整合学习内容，整个项目分解为四个任务。任务 1 通过讨论记录报名信息的各种实现方法，激发学生学习兴趣。任务 2 首先分析“报名表”信息，揭示学生数据是有属性的，再引出设计某种数据类型实现这种属性，以记录单个学生信息，评估这种设计，感受数据类型的意义。任务 3 对于多个学生信息的记录，以“数据结构”的方式来体现数据之间的关系，分析采用数据类型和数据结构这两种实现方式，体会数据结构和数据类型的关系。任务 4 编写程序，实现“随时提供有序的报名信息”项目的存储和显示等基本功能，在调试和完善过程中发展计算思维。

本章第一节的学习以解决任务 1 和任务 2 的方式展开。其中任务 2 可以采用一种列表的方式呈现某一位报名者的信息记录。本章第二节的学习以解决任务 3 和任务 4 的方式展开。对于任务 3 可以用列表嵌套列表这种“数据结构”的方式实现多名报名者信息的记录。在完成项目的过程中，感受到数据结构可以通过数据类型来实现。在有了这种感受之后，教科书又重点分析抽象类和用线性表、二叉树等数据结构实现数据类型的案例，使学生体会数据类型和数据结构相互交融的关系，对数据类型和数据结构的概念形成清晰的认识。

## 四、本章课时安排建议

本章教学建议用 2 课时完成，具体参见表 1-1。

表 1-1 课时安排计划表

节名	建议课时
第一节 数据	1
第二节 数据结构	1

## 第一节 数 据

### 一、教学目标与重点

#### 教学目标:

- 通过对“数据”三个层面的辨析，理解“数据”概念的多义性，学会从计算机程序的角度理解数据的含义，能形成对数据内涵的正确认识。
- 在实践体验、探究活动的过程中，理解数据类型的概念和意义，能体会数据类型的作用，提高数据抽象的能力。

#### 教学重点:

理解数据的不同层面含义。

### 二、教学实施与评价

#### 1. 教学活动建议

本节以项目任务 1 和项目任务 2 的完成为引领，但因内容上比较抽象，建议结合案例、讨论、程序演示等展开教学活动。对于数据的概念，必修 1 从宽泛性方面已有所描述，本课在前期的基础上从计算机科学角度出发，以数据三个层面展开讨论，建议先呈现“关于数据含义的一种层次示意图”，然后让学生找寻并列举生活中、计算机中等不同情境下各种数据所在层面，在充分交流的基础上，完成项目任务 1，从而对数据的概念形成清晰的认识。

对于数据的第三个层面“程序中的数据”的介绍，建议演示一个编写好的程序，在编译环境下监视变量，通过单步运行程序，观察监视器中变量值变化的情况，在观察的过程中认识计算机解决问题以数据为中心的思想，通过这种形象的程序演示可以加深对本课程讨论的数据层面的理解，为后续的学习打下基础。

以项目任务 2 问题的解决引出关于“数据类型”的介绍，先通过演示“TypeError”程序观察出错现象，分析出错原因，然后从计算机硬件角度深入探究，这种出错提示其实是用

于防止计算机产生无效运行结果的,从而体会“数据类型”的必要性以及它在保证程序编写高效性方面的作用。建议在修改文件扩展名活动后,设置一个关于“数据类型”的形式和作用的讨论,进而完成项目任务 2。

## 2. 教学过程安排(见表 1-2)

表 1-2 教学过程设计表

课时	教学环节	教师活动	学生活动
1 课时 (数据)	1. 情境导入	以“随时提供有序的报名信息”项目介绍,引出本课的学习	倾听、思考
	2. 讨论交流:数据的含义	布置讨论: 介绍数据的宽泛性;围绕数据含义的层次示意图展开讨论并列举实例;布置项目任务 1	自主学习,讨论交流,完成项目任务 1
	3. 活动之一:运行“TypeError”程序	布置活动: 布置学生运行“TypeError”程序;介绍计算机硬件和程序语言的发展过程,讲解数据类型的必要性	倾听、思考、探究,形成对数据类型意义的理解
	4. 活动之二:修改文件扩展名	布置活动: 呈现不同扩展名的文件;引导学生对文件扩展名进行修改并讨论修改原因;归纳数据类型的作用	完成活动,分析修改原因
	5. 活动之三:完成项目任务 2	布置任务: 讲解各种数据类型的形式;布置完成项目任务 2;引导学生交流各自的设计方案	设计一种数据类型完成学生信息的记录,并讨论设计原因
	6. 总结	程序设计中数据和数据类型的作用	

本节在项目实践的过程中展开评价。项目任务 1 在纸上登记信息,并对传统方法带来的困难进行讨论,激发学生学习兴趣,使其能通过实例描述数据的三个层次;项目任务 2 设计报名表信息的数据类型,以此引导学生知道数据是有属性的,能评估这种设计并感悟数据类型的意义。在完成各项目任务的过程中,观察学生认知体系的发展与变化。教师也可设计如表 1-3 所示的实践评价量规。

表 1-3 实践评价量规参考示例

评价内容		是否达成	反思与建议
案例分析	认识数据的宽泛性和三个层次		
解析程序	能解释计算机利用数据来解决问题		
修改程序	能描述数据类型的意义		
交流分享	能设计一个数据类型来表示单个学生的信息		

### 3. 思考探究提示

探究活动参考答案如下：

在 Python 中运行如下程序，观察错误提示。

```
a= "1"  
b= "2"  
c= a * b
```

### 4. 项目实施提示

项目实施参考答案如下：

项目任务 1：可以采用表格、列表、文本等方式；会带来查询、排序等不方便。

项目任务 2：用 Python 中的列表定义学生信息，如[“学号”，“姓名”，“性别”]。

## 三、核心概念介绍

### 数据类型：

数据类型是计算机科学中的一个核心概念。几乎任何可编程的工具或系统（程序设计语言、数据库管理系统等）都有一套关于数据类型的规定，指明有哪些数据类型及其含义和操作等。

出于不同的应用需求和设计偏好，不同工具和系统所规定的数据类型不尽相同。然而它们都有一个共同的目的，就是使编程高效少错。

学习数据类型有关知识时，为突出某些要点，常会将种类繁多的数据类型分为几大类。教科书中采用了基本类型、常用类型、组合类型和抽象类型的划分方式。

## 四、教学参考资源

### ■ 阅读材料：与数据有关的术语

**数据**(data)是客观事物的符号表示，是所有能输入到计算机中并被计算机程序处理的符号的总称。如数学计算中用到的整数和实数，文本编辑中用到的字符串，多媒体程序处理的图形、图像、声音及动画等通过特殊编码定义后的数据。

**数据元素**(data element)是数据的基本单位，在计算机中通常作为一个整体进行考虑和处理。在有些情况下，数据元素也称为元素、记录等。数据元素用于完整地描述一个对象，如一名学生记录，树中棋盘的一个格局(状态)，以及图中的一个顶点等。

**数据项**(data item)是组成数据元素的、有独立含义的、不可分割的最小单位。例如，学生基本信息表中的学号、姓名、性别等都是数据项。

**数据对象**(data object)是性质相同的数据元素的集合，是数据的一个子集。例如：整数数据对象是集合  $N = \{0, \pm 1, \pm 2, \dots\}$ ，字母字符数据对象是集合  $C = \{'A', 'B', \dots, 'Z'\}, \{'a', 'b', \dots, 'z'\}$ ，学生基本信息表也可以是一个数据对象。由此可以看出，不论数据元素集合是无限集(如整数集)，或是有限集(如字母字符集)，还是由多个数据项组成的复合数

据元素(如学生表)的集合,只要集合内元素的性质均相同,都可称之为一个数据对象。

——摘自《数据结构》,严蔚敏、李冬梅、吴伟民,人民邮电出版社(有删改)

## 五、教学参考案例

### ■ 参考案例

#### 数 据

上海市闵行区教育学院 隋晓筱

(1课时)

##### 1. 学科核心素养

能够通过完成项目任务——“随时提供有序的报名信息”以及对数据三种层次的案例分析,正确认识和处理数据。在单步运行程序中理解数据的含义及数据之间的逻辑关系;愿意与团队成员共享信息,促进同伴协作解决问题,同时更深入理解数据的价值。(信息意识)

能够通过运行“`TypeError`”程序和修改扩展名活动,采用计算机可以处理的方式界定问题、抽象特征、建立结构模型、合理组织和存储数据。(计算思维)

能够通过设计报名信息记录,评估并选用数据类型,有效地管理学习过程与学习资源,创造性地解决问题,从而完成学习任务,形成创新作品的能力。(数字化学习与创新)

能够在项目实践过程中,感受和理解数据的重要性,合理利用数据,提升数据应用的社会责任。(信息社会责任)

##### 2. 《课程标准》要求

通过列举实例,分析数据与社会各领域的关系,理解数字、数值和数据的基本含义。

通过列举实例,认识到数据作为新的原材料、生产资料和基础设施的价值与意义。

##### 3. 学业要求

学生能够运用生活中的实例描述数据的内涵与外延,能够将有限制条件的、复杂生活情境中的关系进行抽象,用数据结构表达数据的逻辑关系。

能够针对限定条件的实际问题进行数据抽象,运用数据结构合理组织、存储数据。

能够分析数据与社会各领域间的关系,自觉遵守相应的伦理道德和法律法规。

##### 4. 教学内容分析

数据是教科书第一章第一节的内容。本节为1课时,教学内容也是本课程的第一课时。数据这一概念在必修1数据与计算课程中已有介绍。因此,本节课教学中的数据与必修1中介绍的数据之间的关系需进行辨析,本节课包括整个课程中的数据更应从数据结构的角度进行介绍。也就是说,这里的数据指的是程序设计中的数据。本节课为起始课,理解数据的含义有利于促进后续数据与数据结构课程的学习。

##### 5. 学情分析

本节课之前,学生在必修1数据与计算课程中已学习过有关数据方面的内容。对于学习本课内容有一定基础,但本节课中的数据和之前所学的又有所不同。数据与数据结

构是计算机方面的基础课程。本课从三个层面对数据展开介绍,特别在单步运行程序过程中,观察监视器中的数据变化,理解数据之间是有关系的,为后续数据结构学习做铺垫。同时,高中学生思维发展具有更高的抽象性、概括性,本节课关于数据概念的辨析能激发学生的学习兴趣。

## 6. 教学目标

- 通过解读、分析数据三个层面的案例和学生信息记录活动,理解数据的宽泛性,以及数据“程序中”“计算机中”和“计算机外”的三个层面,明确本课中数据的含义,提升信息意识。
- 通过单步运行程序,理解计算机以数据为中心解决问题的思想,加深数据层面的理解。
- 通过“`TypeError`”程序调试和修改扩展名的活动,理解数据类型的意义和体现形式,促进计算思维的养成。
- 通过小组活动,交流报名信息数据类型选取的设计方案,在实践体验中体现数字化学习与创新。

## 7. 教学重难点

- 教学重点:理解数据的不同层面含义。
- 教学难点:理解数据类型的意义。

## 8. 教学策略分析

本节课主要采用项目教学、案例分析、讨论交流、实践体验等教学策略。整节课将引导学生在体验感悟中理解数据及数据类型。

- (1) 以项目为引领开展教学,引出有关数据如何保存和数据类型如何设置的问题。
- (2) 单步运行程序,通过观察监视器中变量值变化的情况,了解计算机解决问题以数据为中心的思想,通过这种形象的程序演示加深理解本课程所讨论的数据的含义。
- (3) 选择不合适的数据类型导致程序运行出错,体会数据类型的重要性。

## 9. 教学过程设计(见图 1-1 和表 1-4)

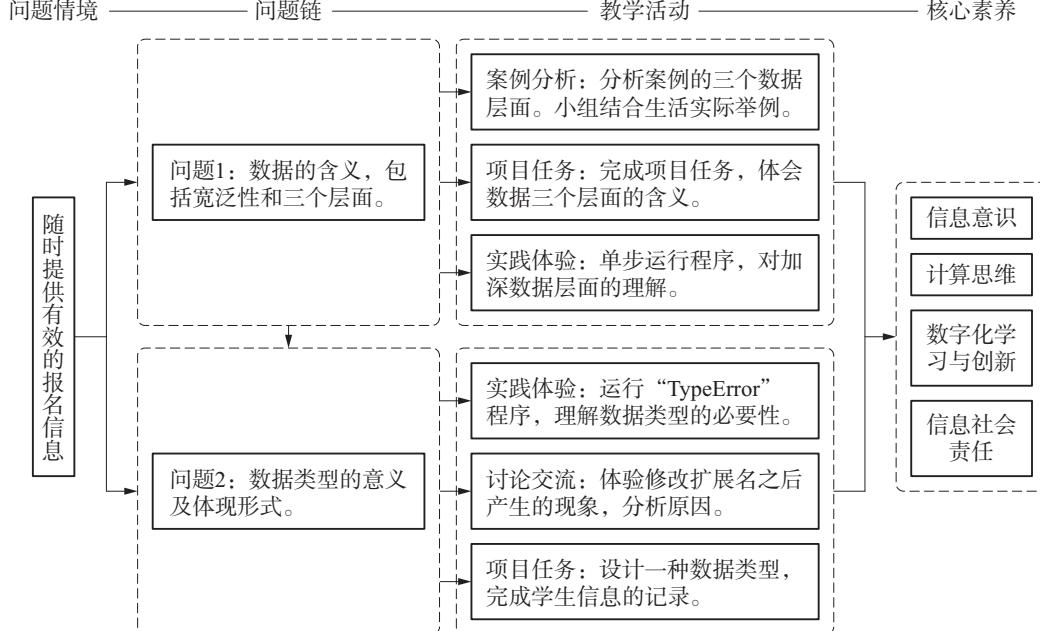


图 1-1 教学过程

表 1-4 教学过程设计表

教学环节	教师活动	学生活动	设计意图
情境导入	通过项目“随时提供有序的报名信息”,引出本课学习内容	结合自己的认知,思考、明确项目任务	创设情境,引出项目主题,体现问题解决的意识
数据含义的理解和实践	讲解分析: (1)介绍数据的宽泛性。 (2)引导学生围绕数据含义的层次示意图展开讨论,列举不同层面数据实例。 (3)引导完成项目任务1。 (4)实践体验,引导学生单步运行程序	小组讨论交流,列举生活中体现三个数据层面的实例;完成项目任务1;单步运行程序,观察监视器中的数据变化	结合生活实例,完成项目活动1,理解数据层面;通过实践体验,理解计算机以数据为中心解决问题的思想,加深数据层面的理解,培养信息意识
数据类型的程序运行	实践体验: (1)引导学生完成“TypeError”程序。 (2)教师分析计算机硬件和程序语言的发展过程,介绍数据类型的必要性	运行程序,观察现象,分析原因,理解数据类型的必要性	形成对数据类型意义的理解,促进计算思维的养成
数据类型作用的体现	实践体验: (1)呈现不同扩展名的文件。 (2)引导学生修改文件名和文件扩展名,并对比现象讨论原因。 (3)归纳数据类型的作用	完成活动,体验修改扩展名之后产生的现象,分析原因	理解数据类型的含义及作用,增强合理利用数据的意识,提升信息社会责任
数据类型不同形式的设计	实践体验: (1)讲解各种数据类型的形式。 (2)布置完成项目任务2。 (3)引导学生交流各自设计方案	设计一种数据类型完成学生信息的记录,小组讨论设计方案	通过项目任务2,总结各种实现方式的特点;在实践体验中体现数字化学习与创新
课堂总结	回顾总结: 教师回顾梳理本节课学习的内容,展示知识点——数据的三层含义、数据类型的作用	思考、归纳数据的三层含义、数据类型的作用	梳理本节课的知识点,体会学习数据结构的意义;促进对数据知识的学习,为后续学习奠定素养基础

## 附：“数据”学习单

活动1:完成项目任务1。

通过讨论项目情境中的要求,请你写下如果不用程序而是就在纸上做报名登记,可能有哪些方式,分别会带来哪些困难。请记录在表1-5中。

表 1-5 记录单

可以采用的方式	困难

活动 2: 单步运行如下程序, 并填空。

```
c=[1,1]
for i in range(9):
    c.append(c[-1]+c[-2])
for i in range(9):
    print(c[i])
```

- (1) 在编译环境下, 打开 Variable Explorer 窗口。
- (2) 点击 Debug 菜单下的 Debug 按钮, 然后继续点击 Step 按钮或按 Ctrl + F10 组合键。观察 Variable Explorer 窗口中变量值的变化情况。
- (3)  $c[i]$  的特点是 \_\_\_\_\_, 这段程序的功能是 \_\_\_\_\_。

活动 3: 运行如下程序, 观察现象。

```
a= "1"
b= "2"
c= a*b
```

活动 4: 修改文件扩展名, 然后说出现象。

活动 5: 完成项目任务 2。

假设每个报名者的信息包括(学号、姓名、性别)三项内容, 通过考察 Python 提供的丰富的语言机制, 设计一个数据类型来表示单个学生的信息, 并对自己的设计选择进行评估。

## 第二节 数据结构

### 一、教学目标与重点

#### 教学目标:

- 在性能分析、程序演示的过程中, 理解数据结构的概念和含义, 感悟数据结构对问题高效解决的支持, 形成数据结构和算法相辅相成的观念。
- 通过案例类比、原理分析, 了解数据结构和数据类型关系, 形成信息意识。
- 通过课堂练习和项目实践, 在对数据的合理使用中, 增强信息社会责任。

#### 教学重点:

理解数据结构的概念和含义。

## 二、教学实施与评价

### 1. 教学活动建议

本节内容在本课程中起着引领作用,在教学内容组织上可以从数据结构产生的背景、为什么需要等方面分析数据结构的意义,进一步分析数据结构与算法和数据类型的关系。建议从解决项目任务3、项目任务4出发开展教学。在具体实现上建议多设计一些程序演示、代码修改、案例分析、图解原理等活动。具体教学中,要注重讲解知识的背景,以问题贯穿教学过程,通过问题将理论知识和应用联系起来。对于数据结构的意义理解,从“数据流”活动出发,鼓励在学生动手填写线性表存储和二叉树存储的比较次数后,归纳什么是数据结构和为什么需要数据结构。

我们通常说数据结构是计算机存储、组织数据的方式。这样的概念太过抽象,对于高中生来说是很难理解的。因此可以设计两个问题来分析:(1)为什么在程序设计中需要数据结构? (2)用与不用数据结构会有什么区别?

对于问题(1),可以从计算机结构原理回答。计算机存储器中存储数据的方式比较简单,内部都是按线性顺序依次存储的(如图 1-2 所示),但是现实中的问题一般都比较复杂,特别是其中的数据具有复杂的关系,譬如导航地图(如图 1-3 所示),点与点之间有复杂的网状连接关系。如何把带有复杂关系的数据存储到只有简单线性顺序关系的存储器中呢? 为此人们设计了“图”这种数据结构,通过图结构将数据及数据之间的关系有效存储在计算机中并进行有效操作。这样,遇到类似比较复杂问题时也可以处理了。

图 1-2 线性存储

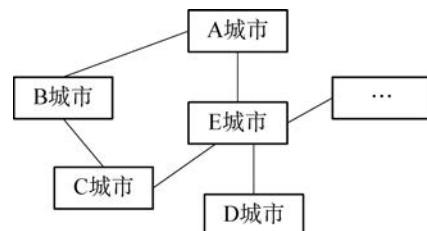
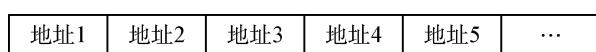


图 1-3 导航地图

对于问题(2),可以从选用不同的数据结构导致处理问题效率不同的角度考虑:譬如对于同样的查找问题,将数据组织成线性表(如图 1-4 所示)和组织成二叉树(如图 1-5 所示)在查找数据时效果不同,而数据结构的好处之一就是使问题解决更高效。

图 1-4 线性表

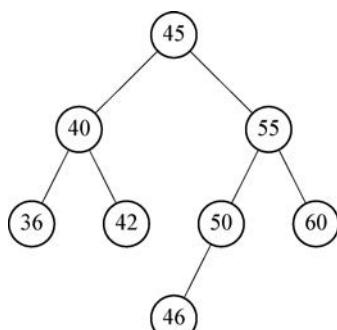
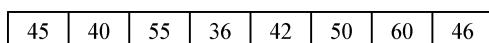


图 1-5 二叉树

通过运行如下程序,理解数据结构使程序运行更高效。

```
import time
import random
import numpy as np
a=[]
for i in range(1000000):
    a.append(random.random())
# 方法一 直接用内置函数求和
t1=time.time()
sum1=sum(a)
print(sum1,time.time()-t1)
# 方法二 用 numpy 转换为数组后求和
b=np.array(a)
t1=time.time()
sum1=np.sum(b) # 用 numpy 转换为数组后求和
print(sum1,time.time()-t1)
```

以上程序为将 100 万个随机数求和,如将数据按方法二数组方式组织比传统列表方式求和运算更快。

在理解了以上两个问题后,学生对于数据结构是什么也就比较容易理解了。这种讲来龙去脉、理解为什么的过程也有利于发展计算思维。

介绍数据结构与算法相辅相成的关系,建议从计算机解决问题的角度出发,设计算法,编写程序;在对数据操作过程中,从数据之间的自然关系和计算机内存储关系介绍数据结构的静态性;为了形象说明问题,可以演示按序存储有关的程序,如定时数据变化的案例,然后再演示构建二叉树的案例,介绍数据结构的动态性。

在上一节中接触到了数据类型,那么数据类型和数据结构有什么关系?厘清二者之间关系有利于明确它们各自的作用和理解融合设计的思想方法,同时可以将这种思想迁移并用于解决其他相关的问题。在教学过程中,建议以上节课构建的“报名者信息表”数据结构来说明里面的数据存在不同的数据类型,之后演示一个抽象数据类型的构建,以一种数据结构的方式构建学生数据类型,如下所示:

```
class Student:
    def __init__(self, name, height, weight): # 身高单位是米,体重单位是千克
        self.name=name
        self.height=height
        self.weight=weight
    def get_bmi(self):
        bmi= self.weight/(self.height* self.height)
        if bmi<=18.4:
```

```

        print('偏瘦')
    elif bmi <= 23.9:
        print('正常')
    elif bmi <= 27.9:
        print('超重')
    else:
        print('肥胖')

wang = Student("王同学", 1.7, 65)      # 将学生数据类型的数据赋予 wang 变量
zhang = Student("张同学", 1.7, 50)
wang.get_bmi()
zhang.get_bmi()

```

在具体分析时,要抓住数据类型表现的是一批数据具有相同的属性,数据结构表现的是一批数据之间的一种关系这一特点。而在某些时候,围绕具体的对象,数据类型会表现出双重性,如数组既可以看作数据类型,也可以看作数据结构。

## 2. 教学过程安排(见表 1-6)

表 1-6 教学过程设计表

课时	教学环节	教师活动	学生活动
1 课时 (数据结构)	1. 情境导入	以完成项目任务 3 记录一批报名者信息为例引出本课学习内容	思考解决问题方案
	2. 新知讲授	(1) 介绍“数据流”程序的两种实现方式。 (2) 分析各自的特点。 (3) 引导归纳数据结构的必要性和高效性	倾听、思考、体会数据结构的意义和作用
	3. 作业练习	(1) 布置完成教科书第 12 页作业练习。 (2) 组织学生交流完成情况。 (3) 引导学生归纳总结	完成作业,归纳巩固
	4. 新知讲授	(1) 以图示的方式介绍数据结构中的元素具有数据类型。 (2) 以程序演示实现一个抽象类的方式呈现数据类型可以通过数据结构来实现	倾听、思考、体会数据结构和数据类型的关系
	5. 编程实现	(1) 布置完成项目任务 3 和项目任务 4。 (2) 提供项目任务 4 中程序主体,引导学生完成关键代码填空	完成方案,编程实现
	6. 总结	数据结构的意义	

本节课在项目任务 3 和项目任务 4 完成过程中进行评价,在项目实践过程中考查运用数据结构解决实际问题的能力,可以对学生在以下几方面的表现进行评价:能够描述数据、数据结构及其相关概念;能够说明数据对信息社会的重要性;能够采用线性存储和非线性存储两种方案解决“数据流”问题,并能从性能角度描述数据结构的意义;能通过实例

解释数据结构与算法的相辅相成;能举例说明数据结构与数据类型的关系。

### 3. 项目实施提示

本章的项目活动以层层递进的方式展开,对于项目任务 1,通过讨论采用程序和纸上登记两种方式报名的特点,培养学生的信息意识。在完成项目任务 1 的基础上,开展项目任务 2,设计一个数据类型来表示一位学生的信息。从一位学生到多位学生的登记,引出项目任务 3,解决多位学生信息保存的问题,从而让学生体会数据结构的重要性,提升其计算思维能力。然后提出问题:对于多位学生的信息,采用怎样的存储方式可以方便按序打印?从而引出项目任务 4。在采用有序保存方式的输入过程中,让学生体验信息的合理利用,从而提升其信息社会责任意识。

(1) 项目任务 3:利用 Python 语言并使用列表嵌套设计用于存储多个报名者信息记录的数据结构。

(2) 项目任务 4:具体代码实现如下(比较采用和不采用 insert 的区别)。

```
import time

def studentappend():
    name_list= list()
    no= input('请输入学号')
    name= input('请输入姓名')
    sex= input('请输入性别')
    name_list.append(no)
    name_list.append(name)
    name_list.append(sex)
    j= 0; current_length= len(student)
    if current_length== 0:
        student.append(name_list)
    while j< current_length:
        if int(no)< int(student[j][0]):
            student.insert(j, name_list)
            break
        else:
            j= j+ 1
    if current_length== j and j!= 0:
        student.append(name_list)

def studentprint():
    print('当前已报到学生名单按学号递增输出')
    for j in range(len(student)):
        print(student[j])
    print()
```

```

# 以下为主程序
student= list()
while True:
    cho= input(
        """
        请选择功能:
        1.输入信息
        2.打印信息
        3.退出系统
        """
    )
    if cho== '3':
        break
    elif cho== '1':
        studentappend()
    elif cho== '2':
        studentprint()
    else:
        print(' 输入错误,请重新选择.')
    time.sleep(1)
    continue

```

### 三、作业练习与提示

#### ■ 题目描述

参照本章第二节第一部分中的例子,对 16 个数的序列: 5, 3, 5, 4, 2, 1, 6, 8, 3, 3, 4, 1, 7, 8, 4, 2, 给出类似于教科书表 1.1 和表 1.2 的结果。

#### ■ 作业提示

采用线性存储结构比较次数如表 1-7 所示。

表 1-7 线性存储结构下的比较次数

数据	5	3	5	4	2	1	6	8	3	3	4	1	7	8	4	2
比较次数	0	1	1	2	3	4	5	6	2	2	3	5	7	7	3	4

采用二叉树存储结构比较次数如表 1-8 所示。

表 1-8 二叉树存储结构下的比较次数

数据	5	3	5	4	2	1	6	8	3	3	4	1	7	8	4	2
比较次数	0	1	1	2	2	3	1	2	2	2	3	4	3	3	3	3

## 四、核心概念介绍

### 1. 数据结构

数据元素间的相互关系称为结构,数据结构是带结构的数据元素的集合。数据结构研究的是数据的存储和组织方式,以支持程序的高效执行,主要包括以下内容:逻辑结构、存储结构、数据的运算。逻辑结构描述数据元素之间的逻辑关系,与数据的存储无关,独立于计算机从具体问题中抽象出来的数学模型。存储结构是数据元素及其关系在计算机存储器中的结构,是数据结构在计算机中的表示。数据的运算,即对数据可以施加的操作以及这些操作在相应的存储结构上的实现。存储结构有顺序存储、链式存储、哈希存储等。逻辑结构有线性结构和非线性结构等。具体见图 1-6。

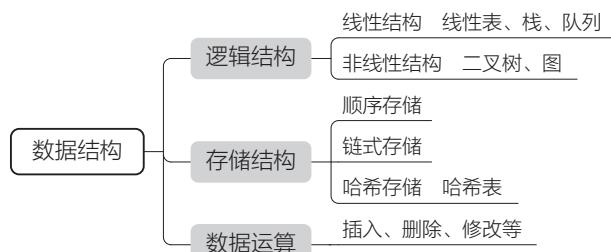


图 1-6 数据结构

### 2. 四种基本逻辑结构图(见图 1-7)

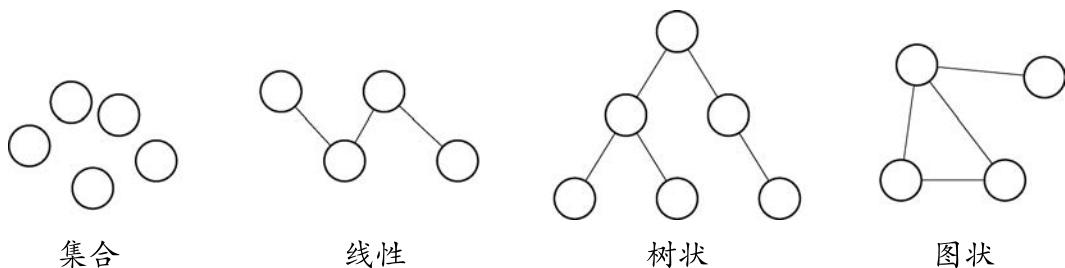


图 1-7 四种基本逻辑结构

## 五、教学参考资源

### ■ 阅读材料：数据结构课程的由来

“数据结构”在计算机科学中是一门综合性的专业基础课,是介于数学、计算机硬件和

计算机软件三者之间的一门核心课程。数据结构这一门课的内容不仅是一般程序设计(特别是非数值性程序设计)的基础,而且是设计和实现编译程序、操作系统、数据库系统及其他系统程序的重要基础。

1968年,数据结构作为一门独立的课程,在计算机科学的学位课程中开始出现。之后,20世纪70年代初,出现了大型程序,软件也开始相对独立,结构程序设计成为程序设计方法学的主要内容,人们也越来越重视“数据结构”。

一般认为,一个数据结构是由数据元素依据某种逻辑联系组织起来的。对数据元素间逻辑关系的描述称为数据的逻辑结构;数据必须在计算机内存储,数据的存储结构是数据结构的实现形式,是其在计算机内的表示;此外讨论一个数据结构必须同时讨论在该类数据上执行的运算才有意义。一个逻辑数据结构可以有多种存储结构,且存储结构的不同会影响数据处理的效率。在许多类型程序的设计中,数据结构的选择是一个基本的设计考虑因素。许多大型系统的构造经验表明,系统实现的困难程度和系统构造的质量都严重依赖于是否选择了最优的数据结构。许多时候,确定了数据结构,算法就容易得到了。

有些时候事情也会反过来,我们根据特定算法来选择数据结构与之适应。不论哪种情况,选择合适的数据结构都是非常重要的。“选择了数据结构,算法也随之确定,系统构造的关键因素是数据而不是算法。”在这种观念下产生了许多种软件设计方法和程序设计语言,面向对象的程序设计语言就是其中之一。

从20世纪70年代中期到20世纪80年代初,各种版本的数据结构著作相继出现。目前在我国,数据结构已成为计算机专业的核心课程之一,也是其他非计算机专业的主修课程之一。

——参考《数据结构》,严蔚敏、李冬梅、吴伟民,人民邮电出版社

## 六、教学参考案例

### ■ 参考案例

#### 数据结构

华东师范大学第二附属中学紫竹校区 严一滨  
(1课时)

##### 1. 学科核心素养

能够根据线性表和二叉树在解决问题中的效率差异,敏锐认识到数据结构在解决问题过程中所起的重要作用;在合作解决问题的过程中,愿意与团队成员共享信息,更好地形成数据结构的概念。(信息意识)

能够结合不同的问题情境进行分析,采用计算机可以处理的方式,考虑算法的运用,比较和选择合适的数据结构和数据类型,更好地解决问题。(计算思维)

能够评估并选用常见的数字化资源与工具,有效地管理学习数据结构的过程与相关资源。(数字化学习与创新)

## 2. 《课程标准》要求

结合生活实际,理解数据结构的概念,认识数据结构在解决问题过程中的重要作用。

### 3. 学业要求

学生能够运用生活中的实例描述数据的内涵与外延,能够将有限制条件的、复杂生活情境中的关系进行抽象,用数据结构表达数据的逻辑关系。

能够从数据结构的视角审视基于数组、链表的程序,解释程序中数据的组织形式,描述数据的逻辑结构及其操作,评判其中数据结构运用的合理性。

能够针对限定条件的实际问题进行数据抽象,运用数据结构合理组织、存储数据,选择合适的算法(如排序、查找、迭代、递归等)编程实现、解决问题。

能够分析数据与社会各领域间的关系,自觉遵守相应的伦理道德和法律法规。

### 4. 教学内容分析

数据结构是关于数据之间关系的描述,关于数据结构的知识是计算机学科的基础知识,运用数据结构的能力是开发高水平程序的要求。数据结构是教科书第一章第二节的内容。教科书在前一节里已经介绍过数据概念的宽泛性和层次观,学生对基本的数据类型已有一定的了解。本节将主要通过对两种数据结构查找次数的示例的比较和分析,展现数据结构在解决问题过程中所起的重要作用。

### 5. 学情分析

自主选择学习选择性必修1模块数据与数据结构的学生,一般对计算机学科具有一定的兴趣和能力,包括较强的逻辑思维能力和使用Python语言编写程序的基本技能。再加上第一章第一节数据部分的学习,学生对于基本的数据类型有所了解,但可能对数据结构这一计算机领域的专有名词还比较陌生,对其作用价值与独特魅力缺少认识。

### 6. 教学目标

- 从生活中的门禁识别问题开始,初步了解简单线性结构和二叉树结构,产生数据结构的概念雏形。能够通过对两种不同数据结构应用于门禁查找问题时结果的对比分析,认识到数据结构在解决问题过程中所起的重要作用。

- 通过问题情境创设和分组讨论,针对“连号录入”和“随机录入”两种情况选用合适的数据结构,知道数据结构可以按照天然关系进行设计,也可以按照算法逻辑进行设计。

- 结合数据类型的相关知识,尝试根据所选用的数据结构设计元素的数据类型,知道数据结构与数据类型之间的关系。

### 7. 教学重难点

- 教学重点:**理解数据结构的概念,认识数据结构在解决问题过程中所起的重要作用;知道数据结构与算法、数据结构与数据类型之间的关系。

- 教学难点:**理解数据结构的概念。

### 8. 教学策略分析

本节课作为本单元第二节课,引出了核心概念“数据结构”,主要以创设情境、实例对比、思考讨论、激发学生兴趣为主。拟通过问题链,引导学生一步步深入体验数据结构的作用和价值,知道数据结构与算法、数据类型之间的关系,从而体会到计算机科学领域中这

一特有概念的独特魅力。考虑到教科书的示例中采用的动态更新对学生而言可能有一定难度,这里稍作了一些调整,选用学生相对比较熟悉的查找问题,将数据结构的建立和数据结构的效率对比两个部分进行了拆分,等待学生对数据结构熟悉之后,再将两部分逐步融合在一起。

### 9. 教学过程设计(见图 1-8 与表 1-9)

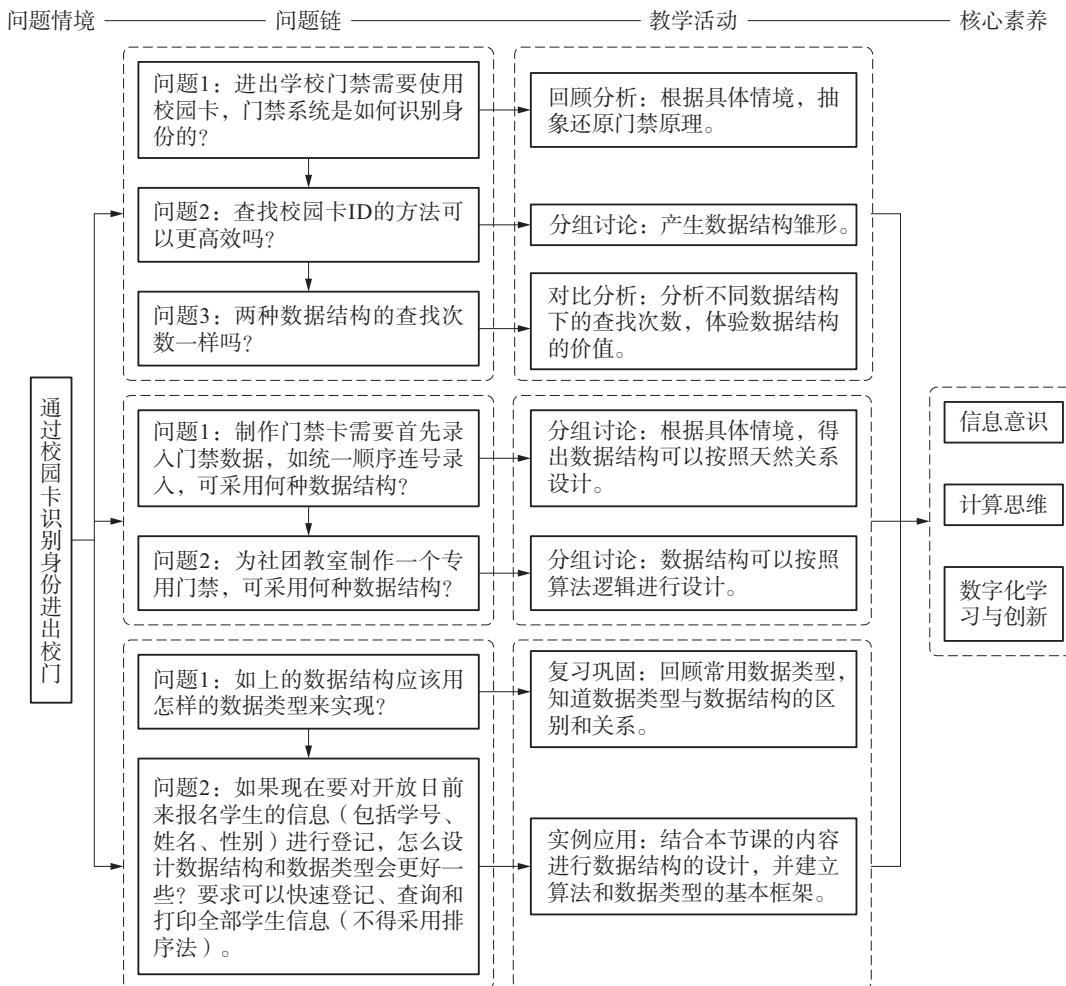
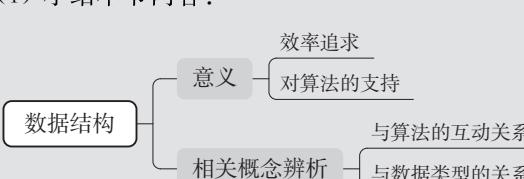


图 1-8 教学过程

表 1-9 教学过程设计表

教学环节	教师活动	学生活动	设计意图
数据结构的意义	<p>问题链一情境主题：大家现在上学进出校门，都要经过门禁，需要刷一下校园卡以识别是不是学校里的师生，你知道这个过程是怎么样的吗？</p> <p>细化问题 1：假设学校采用的是 ID 卡，卡里是不存储除卡号外的其他信息的，那么系统是如何判断这张卡是否属于我们学校师生的呢？比如现在系统里有 45、76、64、23、17、98、81、32 几个 ID 卡号，现在有位同学刷了一张卡……</p>	<p>思考，回答，进入情境。</p> <p>结合实际，思考判断卡号的方法。</p>	<p>根据具体情境，抽象还原门禁原理，产生数据结构概念雏形；分析不同数据结构下的查找次数，体验数据结构的价值，强化学生的信息意识</p>

续表

教学环节	教师活动	学生活动	设计意图
数据结构与算法	<p>细化问题2(小组讨论):我们可以把这种方法用线性图示的方式描绘出来。你觉得这种查找校园卡ID的方法好吗?还有更好,或者说更高效的方法吗?</p> <p>知识点引入与介绍:数据结构、线性存储、二叉树。</p> <p>细化问题3(对比分析):两种数据结构的查找次数一样吗?</p> <p>问题链二情境主题:刚才我们通过进出门禁的问题了解了数据结构的作用和价值,但实际生活中选用哪种数据结构还会受到很多方面的影响。请分组讨论以下两种情况,看适用的数据结构是否一样。</p> <p>细化问题1:如果在新生入学时统一录入校园卡ID,这些校园卡都是顺序连号的,可采用何种数据结构?</p> <p>细化问题2:老师想为社团教室制作一个专用门禁系统,为简化操作,在其中预设了录入模式,只要依次把社团成员的校园卡放上去,就可以录入其校园卡ID,且相同ID不能重复录入,可采用何种数据结构?你能计算其内部查找效率吗?(若放卡顺序是57、63、25、42、57、77、83、83、18、31)</p>	<p>讨论回顾查找数据的方法,引出数据结构。</p> <p>通过对比,感受数据结构的作用</p>	
数据结构与数据类型	<p>问题链三情境主题:以上问题中我们讨论的都是数据元素间的关系也就是数据结构,而在具体的算法实现中,我们需要用到具体的数据类型。试思考以下情况中,数据结构与数据类型应该如何设计。</p> <p>细化问题1:问题链二中的数据结构应该用怎样的数据类型来实现?</p> <p>细化问题2:如果现在要对开放日前来报名学生的信息(包括学号、姓名、性别)进行登记,怎么设计数据结构和数据类型会更好一些?要求可以快速登记、查询和打印全部学生信息(不得采用排序)</p>	<p>思考,回答,进入情境。</p> <p>结合实际,认识到数据结构按照天然关系设计会很方便。</p> <p>结合实例分析,认识到数据结构也可以按照算法逻辑进行设计,进一步强化数据结构对效率的影响</p>	<p>根据具体情境,得出数据结构可以按照天然关系设计,也可以按照算法逻辑进行设计,培养学生的计算思维</p>
小结	<p>(1) 小结本节内容:</p>  <p>(2) 布置课后作业:以小组为单位实现问题链三情境中开放日报名信息登记的相关程序</p>	<p>思考,回答,认识到数据结构和数据类型的区别。</p> <p>通过实例,知道数据类型与数据结构的关系。</p> <p>结合本节课的内容,进行数据结构的设计,并建立算法和数据类型的基本框架</p>	<p>回顾常用数据类型,知道数据类型与数据结构的区别和关系,更好地解决问题,提升学生的计算思维力</p>

## 附：“数据结构”学习单

### 一、数据结构的意义

#### 【基本概念】

不同于数据类型是多个学科都可能有的概念,数据结构是计算机学科的特有概念。

数据结构是关于数据之间关系的描述,及其形成与访问的方式。

### 【对比实例】

假设学校采用的是 ID 卡,卡里是不存储除卡号外的其他信息的,那么系统是如何判断这张卡是否属于我们学校师生的呢?比如现在系统里有 45、76、64、23、17、98、81、32 几个 ID 卡号,现在入校同学的刷卡 ID 依次是 98、45、56。

① 线性存储(将系统里的 ID 卡号填写在图 1-9 中):

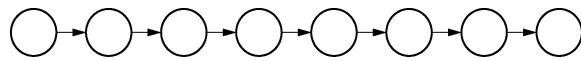


图 1-9 线性存储

将比较次数结果填写在表 1-10 中。

表 1-10 比较次数结果

数据	98	45	56
比较次数			

依次查找 45、76、64、23、17、98、81、32 的总比较次数:\_\_\_\_\_。

② 二叉树(见图 1-10):

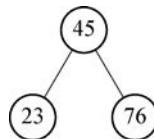


图 1-10 二叉树

此结构称为二叉树,它是由一些“节点”和“边”构成。顶端的节点称为根节点,根节点左下方整体称为左子树,右下方整体称为右子树。基于此结构的具体操作如下:当收到一个新的数(x),首先将此数和根节点相比较。如果相等就停止;如果较小,就与左子树比较,若左边没有子树,就将 x 放在左下,成为左子树的根;如果较大,就与右子树比较,若右边没有子树,就将 x 放在右下,成为右子树的根。依此类推,将结果填写在表 1-11 中。

表 1-11 比较次数结果

数据	98	45	56
比较次数			

依次查找 45、76、64、23、17、98、81、32 的总比较次数:\_\_\_\_\_。

## 二、数据结构与算法

刚才我们通过进出门禁的问题了解了数据结构的作用和价值,在实际生活中选用哪

种数据结构还会受到很多方面的影响。请分组讨论以下两种情况,看适用的数据结构是否一样。

1. 如果在新生入学时统一录入校园卡 ID,这些校园卡都是顺序连号的,可采用何种数据结构?

2. 如果老师想为社团教室制作一个专用门禁系统,为简化操作,在其中预设了录入模式,只要依次把社团成员的校园卡放上去,就可以录入其校园卡 ID,且相同 ID 不能重复录入,可采用何种数据结构?

若放卡顺序是 57 63 25 42 57 77 83 83 18 31,试使用图示方式画出其数据结构。

请将整个处理过程的比较次数结果填写在表 1-12 中。

表 1-12 比较次数结果

数据	57	63	25	42	57	77	83	83	18	31	总计
比较次数											

### 三、数据结构与数据类型

以上问题中我们讨论的都是数据元素间的关系也就是数据结构,而在具体的算法实现中,我们需要用到具体的数据类型。试思考以下情况中,数据结构与数据类型应该如何设计。

1. 上面“数据结构与算法”中的两种数据结构分别可以用怎样的数据类型来实现?

--	--

2. 如果现在要对开放日前来报名学生的信息(包括学号、姓名、性别)进行登记,怎么设计数据结构和数据类型会更好一些? 要求可以快速登记、查询和打印全部学生信息(不得采用排序)。

--

# 数据类型

### 一、本章学科核心素养的渗透

本章内容包括基本类型、常用类型、组合类型、抽象类型四节。通过案例分析,认识数字、数值和数据的关系,在体验思考、探究活动中理解各种数据类型的含义和作用。

本章是选择性必修1 数据与数据结构模块的基础概念。《课程标准》中相关内容要求包括:

1. 1 通过列举实例,分析数据与社会各领域的关系,理解数字、数值和数据的基本含义。

1. 2 通过列举实例,认识到数据作为新的原材料、生产资料和基础设施的价值与意义。

本章的项目主题为“用数据描述身边的同学”,由四个项目任务组成,在完成项目任务的过程中落实《课程标准》要求。

信息意识方面的培养,要使学生能够识别数据类型,并联系生活中的实例来描述数据类型的内涵和外延,能够将具体场景或情境中的数据类型加以归纳,用以恰当表达数据存储及其操作。

计算思维方面的培养,要使学生能够从数据元素和数据项的属性和存储方式的视角分析基于 Python 数值、字符(串)、数组、列表和自定义类的特点,解释数据类型的实现形式,描述数据类型与存储结构之间的关系,评判数据类型定义与使用的合理性;能够针对限定的条件和具体的场景进行数据类型抽象定义,实现简单操作。

数字化学习与创新方面的培养,要使学生能够较为准确地选择或创建数据类型来描述生活中真实的数据存储需求,并能较为熟练地对数据进行归类和抽象,运用“类”这一形式合理表达、存储数据。用数字技术实现对身边事物的描述。

信息社会责任方面的培养,要使学生能够较为深入地理解数据在合理的存储方式中所体现出的价值,能够正确评估数据类型对数据使用效率的影响,能够初步创建自定义数

据类型用于描述生活中的真实问题，并在此过程中进行自主和协作探究。

## 二、本章知识结构

本章遵循《课程标准》，依据学分和课时规定，紧扣学科概念体系，分为四节内容。

第一节基本类型，内容包括数值型与字符型、数值型数据的实现。

第二节常用类型，内容包括字符串、数组。

第三节组合类型，内容包括数据元素与数据项、组合数据类型的表示。

第四节抽象类型，内容包括数据抽象、抽象数据类型的定义与实现。

本章以“数据”作为引入，指明从生活模型的角度来看，数据作为表述事物的基本工具是呈现出多层次特征的，相应地也存在多种类型。程序通过在数据类型的区分维度之上加以规范结构和算法完成一定的逻辑功能，进而实现对某类客观问题的解决。在这里，数据类型具有两个层面上的涵义，即逻辑层面和物理层面：数据类型自身所反映出的“空间”结构特点称为逻辑结构，如家族谱关系是一个天然的树形逻辑结构。而逻辑结构在计算机存储中的具体实现则称为物理结构。我们知道在计算机中大部分数据都存储于外存储器和内存里，而CPU处理数据又必须将数据从外存储器读取到内存中，直接寻址和间接寻址是计算机组织数据的两种最基本的方式，正是通过这两种基本方式，数据才被存放到内存中，而存放的时候可能存于连续的地址空间中，也可能存于离散的地址空间中。正因为这样，才出现了计算机对数据的不同描述形式，如树形逻辑结构是用链式存储结构表示还是用数组（即顺序存储结构）实现。

早期人们把计算机作为仅仅用于数值计算的工具，可现实中更多问题的解决不仅需要进行数值计算，还需要一些更抽象和有效的手段（比如表、树和图等数据结构）的帮助。而数据类型涉及非数值计算的程序设计问题中的操作对象，以及它们之间的关系和操作等相关问题。

可以通过哪些基本的手段来描述数据类型呢？本章所指的数据类型也称为数据元素（data element），是用一组属性描述其定义、标识、表示和允许值范围的数据单元。这一概念包括但不限于：（1）数据，是对客观事物的符号表示，在计算机科学中指所有能够输入到计算机中并能被计算机程序处理的符号集合，包括数值、文字、图像、音频、视频等形式。（2）数据项，是数据中具有独立含义的、不可再分割的最小数据单位，是客观实体一种特征的数据表示。（3）数据元素，是一个或多个相关数据项的集，是一个客观实体一种或多种特征的数据描述，是计算机程序中加工处理的基本单位。数据元素按其组成可分为简单型数据元素和复杂型数据元素。简单型数据元素由一个数据项组成。复杂型数据元素由多个数据项组成，它通常携带着一个概念的多方面信息。数据类型可以理解为数据的基本单元，将若干具有相关性的数据元素按一定的次序组成一个整体结构即为数据类型。

### 三、本章项目活动设计思路

本章承接第一章,对其阐述的三个基本概念进行了初步展开和项目实现,在项目活动过程中确立数据类型这一核心概念,为后面章节的学习提供概念、知识和实现工具的基本准备。

本章项目活动基于一个真实情境对具体数据的属性进行描述、抽象和实现,提出如何将现实世界中具体事物(如人)所具备的特性表述为用数据描述的属性值,并为其在计算机内选择结构方式(存储),这一转换的过程是计算思维的重要体现,也即数据抽象的意义。

在本章学习过程中,通过将项目分解为四个任务,逐步落实针对生活中的实例描述数据类型的内涵和外延,在有限制条件的情况下对数据元素进行抽象,用合适的抽象类型实现数据元素之间的逻辑关系。任务1通过项目情境介绍,了解数据的价值是什么,数据元素的属性是什么,激发学生主动思考,提高其提炼归纳能力,即认识到“描”的必要性和“述”的现实依据,培养其信息意识。任务2通过自主探究、多人合作和讨论,能够使用概念图标识关键属性及其值,利用数据对事物特征进行描述,加深理解数据属性的含义,培养学生的积极思考和合作学习能力,以此发展其计算思维中的界定问题能力。任务3通过资源平台、自主学习和教师讲解,知道项目实施的规则,以及如何管理项目进程,能够进行项目选题、计划和步骤梳理,培养学生借助资源、选择工具的自主学习能力。任务4通过自主探究、展示交流、小组合作,能够进行准确的需求和条件分析,在项目实践的过程中发展学生的计算思维。

在完整的项目实施体验之后,那些比较难以理解的概念会在头脑中逐步清晰,对编程语言的熟悉程度会进一步提升,为后续章节的系统学习奠定基础。

### 四、本章课时安排建议

本章教学建议用6课时完成,具体参见表2-1。

表2-1 课时安排计划表

节名	建议课时
第一节 基本类型	1
第二节 常用类型	2
第三节 组合类型	1
第四节 抽象类型	2

## 第一节 基本类型

### 一、教学目标与重点

#### 教学目标:

- 在举例、对比过程中,辨明基本数据类型及其分类,了解定义数据类型的一般方法,树立信息规则意识。
- 通过对基本数据类型存储结构的分析,了解存储方案表述的基本方法,提高计算思维中抽象特征的能力。
- 通过对比浮点数与整型数的存储方式,掌握浮点数存储结构的具体实现过程,提高计算思维中问题界定的能力。
- 通过总结基本数据类型合理定义的有效性和必要性,形成积极辨析的态度,增强信息社会责任。

#### 教学重点:

- 基本类型的分类及其必要性。
- 浮点数的存储方式及表达。

### 二、教学实施与评价

#### 1. 教学活动建议

本节内容较为枯燥和琐碎,教学时如照本宣科,一定程度上会影响学生的学习兴趣。本节的重点主要是对基本概念的描述,因此学法以理解、识记为主。本节课互动以师生对话、讨论为主,辅以学生的识记和上机实践。对于这些基本概念,通过讨论,引导学生分析、理解概念的逻辑内涵,全面系统地归纳、识记和应用。在识记不同的内容时,为了避免枯燥,可采用不同的方法。在识记数据的类型和关键字时,可加入一些课堂练习——让学生当堂识记,然后用“我来问你来答”的方式刺激学生加强记忆;在理解并识记浮点数存储特征时,通过判断 S、M、E 的作用来加强识记和应用;字符编码可通过比较方案的异同来加强记忆和加深理解。

#### 2. 教学过程安排(见表 2-2)

表 2-2 教学过程设计表

课时	教学环节	教师活动	学生活动
1 课时 (基本类型)	1. 情境导入	列举生活中数据的呈现形式和保存方式。 列举必修一分册中 Python 语言定义数据类型的语句和作用。 引出基本数据类型的界定	思考数据类型的分类。 讨论数目与符号的呈现形式
	2. 基本类型的定义	以单变量赋值为例,将二进制、数值和字符编码概念关联。 展示数值型数据、非数值型数据分类的必要性	讨论: (1) 计算机内部数据的表述(基于二进制)。 (2) 列举字符编码的实例和技术规范,比较字符集异同。 上机实践: Python 单变量操作
	3. 整型数和浮点数的实现	整型数的表示和存储。 演示浮点数(二进制)的存储示意图,演示转换运算,引导学生厘清 S、E、M 的含义和作用	讨论: (1) 浮点数存储结构的特征。 (2) 浮点数转换的计算规则。 上机实践: 验证 Python 浮点数操作。 讨论: 浮点数计算误差产生的原因
	4. 完成项目任务 1	布置项目任务 1	讨论,实现项目任务 1
	总结	归纳各种基本数据类型	

本节课在项目任务 1 完成过程中进行评价。在项目实践过程中,考查运用基本类型解决实际问题的能力,可以对学生在以下几方面的表现进行评价:在完成各项目任务的过程中,观察学生对于数值型与字符型数据的理解;能通过比较数字、数值和数据的概念,找出现实生活中事物的特征属性,进而进行数据抽象;能描述基本数据类型的意义和作用。

### 3. 思考探究提示

#### (1) 本节体验思考参考答案:

在 Python2 中,默认代码中的内容采用 ASCII 编码,但 ASCII 编码中不存在汉字,因此在 Python2 中执行语句 `print('发展进步')` 会报错,而因 Python3 支持汉字,故执行此语句不会报错。解决问题之道就是要让 Python2 知道代码中使用的是什么编码形式,因此通过 `# coding = utf-8` 语句,指定此代码采用包含世界上大多数语言文字(包括汉字)的编码方案 UTF-8。

#### (2) 本节第 1 个探究活动参考答案:

$S=1$  表示  $x$  是负数,因为  $M=(0.0101)_2$ ,  $E=-2$ ,所以  $x=-(0.000101)_2$ ,换算成十进制为  $-\left(\frac{1}{16} + \frac{1}{64}\right) = -\frac{5}{64}$ 。

(3) 本节第 2 个探究活动参考答案：

绝对值最大值为： $2^{31} * (1 - 2^{-8})$ 。绝对值最小值为：0。此题还可以考虑非零的绝对值最小值为  $2^{-31} * 2^{-8} = 2^{-39}$ 。

#### 4. 项目实施提示

完成项目任务 1。可以选取性别、身高和体重这三个属性描述同学。

### 三、作业练习与提示

#### ■ 题目描述

试利用 Python 的复数类型写一个求解一元二次方程的程序，即对任何实数 a、b 和 c，都能按照求根公式给出方程  $ax^2 + bx + c = 0$  的两个根。（提示：Python 有支持复数运算的函数库 cmath，其中包含求平方根的函数 sqrt。）

#### ■ 作业练习参考答案

```
import cmath  
a= int(input("请输入 a"))  
b= int(input("请输入 b"))  
c= int(input("请输入 c"))  
d= cmath.sqrt(b*b- 4*a*c)  
x1= (- b+ d) / (2* a)  
x2= (- b- d) / (2* a)  
print(x1,x2)
```

### 四、核心概念介绍

#### 基本数据类型：

基本数据类型主要包括数值型和字符型。能够从二进制位的层面来讨论其含义是它们的共同点，这也就是“基本”所指。数值型和字符型数据所能进行的运算及含义都不相同，比如数值型数据可以进行算术运算，而字符型数据就不能。程序设计语言一般都会提供一组基本数据类型，例如 Python 语言中的基本数据类型就包括整型、浮点型、字符型、布尔型、复数型。从数据特性考虑，整型、浮点型和复数型属于数值型。

### 五、教学参考资源

Python Number 数据类型用于存储数值。数据类型是不允许改变的，这就意味着如果改变 Number 数据类型的值，将重新分配内存空间。

Python 支持四种不同的数值类型：

- 整型(**int**) 通常被称为整型或整数，是正或负整数，不带小数点。

- **长整型 (long integers)** 无限大小的整数, 整数最后是一个大写或小写的 L。
- **浮点型 (floating point real values)** 由整数部分与小数部分组成。浮点型也可以使用科学计数法表示 ( $2.5e2 = 2.5 \times 10^2 = 250$ )。
- **复数 (complex numbers)** 由实数部分和虚数部分构成, 可以用  $a + bj$  或者 `complex(a, b)` 表示。复数的实部 a 和虚部 b 都是浮点型。

### Python Number 类型转换

<code>int(x [,base ])</code>	将 x 转换为一个整数
<code>long(x [,base ])</code>	将 x 转换为一个长整数
<code>float(x)</code>	将 x 转换为一个浮点数
<code>complex(real [,imag ])</code>	创建一个复数
<code>str(x)</code>	将对象 x 转换为字符串
<code>repr(x)</code>	将对象 x 转换为表达式字符串
<code>eval(str)</code>	计算字符串中的有效 Python 表达式, 并返回一个对象
<code>tuple(s)</code>	将序列 s 转换为一个元组
<code>list(s)</code>	将序列 s 转换为一个列表
<code>chr(x)</code>	将一个整数转换为一个字符
<code>unichr(x)</code>	将一个整数转换为一个 Unicode 字符
<code>ord(x)</code>	将一个字符转换为它的整数值
<code>hex(x)</code>	将一个整数转换为一个十六进制字符串
<code>oct(x)</code>	将一个整数转换为一个八进制字符串

——参考《Python 程序设计》, 董付国, 清华大学出版社

## 六、教学参考案例

### ■ 参考案例

#### 数据的基本类型

上海市控江中学 董 姣

(1课时)

#### 1. 学科核心素养

在运用数据描述生活中对象的过程中, 自觉主动地从数据的视角观察生活, 分析数据中所承载的信息, 感知数据的作用与价值。(信息意识)

在对生活中数据的数字化表达过程中, 通过界定问题、抽象特征, 区分基本数据类型, 从存储和计算的角度, 合理组织数据, 由概念形成到编程实现。(计算思维)

借助在线学习平台, 开展自主学习, 与小组同学开展协作学习, 进一步认识数字化学习环境的优势。(数字化学习与创新)

## 2. 《课程标准》要求

通过列举实例,分析数据与社会各领域的关系,理解数字、数值和数据的基本含义。

## 3. 学业要求

能够运用生活中的实例描述数据的内涵与外延,能够将有限制条件的、复杂生活情境中的关系进行抽象,用数据结构表达数据的逻辑关系(信息意识、计算思维)。

## 4. 教学内容分析

基本类型是教科书第二章第一节的内容,是理解计算的一个重要基础。深读并分析教科书,可以看到,教科书强调数据结构与生活的联接,强调对学生信息意识的培养。教学内容以学生熟识的生活中的数目与文字符号为起点,在区分数目与文字符号的基础上,认识计算机中相对应的数值型和字符型两种基本数据类型。从存储和计算的角度引导对数值型的子类型的范围和精度的理解,并通过浮点数的精度问题进一步理解数值型数据的实现,从概念理解逐步到编程实现,以问题的形式逐步推进学习的深入。

## 5. 学情分析

本课的教学对象是高二选修“数据与数据结构”模块的学生。在知识基础上,学生已经学完必修课程的两个模块,已掌握基本的数据与编码的知识,具备一定的数据分析和程序编写的能力,对于数据的基本类型,有初步的了解和编程应用体验。在认知心理上,其逻辑思维趋于严密,具备一定的探究能力,有自主解决问题的主观意愿。因此,在本节课的教学设计中,需要引导学生进一步从数据结构的视角理解数据的基本类型,通过项目情境和编程应用中出现的实际问题来推动学习的展开。

## 6. 教学目标

- 通过对生活中实例的讨论与分析,区分生活中的常见数据,认识数目与数字的关系,在数据抽象的过程中培养信息意识。
- 通过实例中数据的数字化表达,学会使用数据的基本类型来表示数据,在编程实践中理解基本数据类型的含义以及对存储的需求和应用的影响,促进学生计算思维的养成。
- 借助在线学习平台完成个人“数据画像”,开展自主探究和小组合作交流,提升学生数字化学习与创新能力。

## 7. 教学重难点

- 教学重点:理解数值型与字符型数据类型的概念和特点。
- 教学难点:数值型数据的实现。

## 8. 教学策略分析

教法:以问题引领学生探究,以任务串联课堂教学。

学法:数字化自主学习,小组合作探究,在学习过程中形成合力,组内交流、分享。

## 9. 教学过程设计(见图 2-1 和表 2-3)

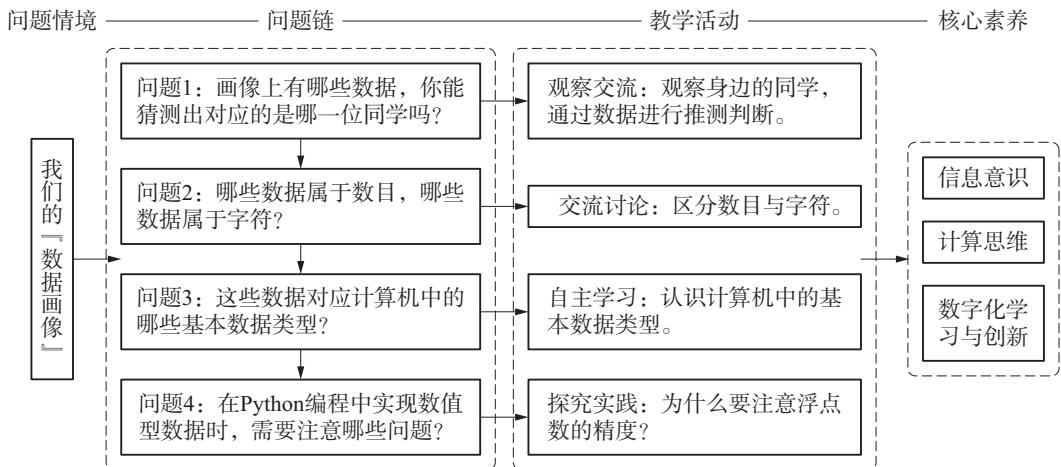


图 2-1 教学过程

表 2-3 教学过程设计表

教学环节	教师活动	学生活动	设计意图
情境导入	<p>情境主题: 我们的“数据画像”</p> <p>在班级中,我们每一个人都有自己的鲜明特点。课前,我们通过在线学习平台,请同学们尝试完成了自己的“数据画像”。  <b>【问题1】</b>画像上有哪些数据,你能猜测出对应的是哪一位同学吗?</p>	<p><b>【观察、交流】</b></p> <p>观察身边的同学,通过数据(性别、身高、年龄、体重等)进行推测判断</p>	引导学生从数据的视角观察身边同学,从生活中的体验到数据的使用过程,感知数据的作用与价值
区分数目与文字符号	<p>生活中的数值与文字符号  <b>【问题2】</b>哪些数据属于数目,哪些数据属于字符?</p> <p>(1) 提供自主学习材料“知识链接1”。  (2) 交流讨论。  (3) 思考: 画像中,班级一栏的“202202”属于文字符号还是数目? 请说出你的理由</p>	<p><b>【交流、讨论】</b></p> <p>(1) 自主学习“知识链接1”。  (2) 讨论,列举数目和文字符号。(预设答案:表示性别的F、M是文字符号,表示年龄的16是数目等)  (3) 进一步区分数目与文字符号。(预设答案:202202代表2022届2班,属于文字符号)</p>	通过自主学习,区分生活中的常见数据,理解数目与文字符号的区别,逐渐形成数据类型的概念
认识计算机中的基本数据类型	<p>数值型与字符型  <b>【问题3】</b>这些数据对应计算机中的哪些基本数据类型?</p> <p>(1) 提供自主学习材料“知识链接2”。  (2) 讨论、分析数值型数据的子类型,列出相应子类型所能表示的数值的个数、存储需求,并写出各整数型的范围,填写在表格中。  (3) 为画像上的数值型数据选择子类型,并说明依据</p>	<p><b>【分析、讨论】</b></p> <p>(1) 自主学习“知识链接2”,举例说出数目和文字符号分别对应的基本数据类型。(预设答案:性别、班级等属于字符型,年龄、身高属于数值型)  (2) 完成练习。  (3) 交流,选择子类型并说明依据。(预设答案:年龄为整型,身高为浮点型等)</p>	通过自主学习,将有限制条件的、复杂生活情境中的数据进行区分和抽象,在概念形成的过程中发展学生的计算思维

教学环节	教师活动	学生活动	设计意图
数值型数据的实现	<p>数值型数据的实现  <b>【问题 4】</b>在 Python 编程中实现数值型数据时,需要注意哪些问题?</p> <p>(1) 上机执行学习材料中的语句代码,观察执行结果,说出你观察到的问题。  (2) 提供自主学习材料“知识链接 3”,完成“练一练”。  (3) 观察“练一练”中的浮点数在数轴上的分布,思考为什么 <math>x == 0.3</math> 的值为 False。  (4) 编程验证解决问题的方案。  (5) 结合编程体验,谈一谈为什么要注意浮点数的精度</p>	<p><b>【探究活动】</b></p> <p>(1) 执行语句代码,观察执行结果,提出观察到的问题。  (预设答案:执行语句 <math>x = 0.1 + 0.2</math> 后,表达式 <math>x == 0.3</math> 的值为 false)</p> <p>(2) 自主学习“知识链接 3”,小组交流讨论,完成“练一练”。</p> <p>(3) 小组交流,回答问题。  (预设答案:由于二进制位是离散的,浮点数在数轴上的分布并不均匀,0.1 和 0.2 在计算机中只有近似值)</p> <p>(4) 编程实现,使用 round() 函数,避免因精度问题产生的错误。</p> <p>(5) 结合编程体验,交流分享自己的答案。  (预设答案:避免因精度问题产生的计算错误,满足问题的精度需要等)</p>	<p>在自主学习和小组交流的基础上,探究浮点数在数轴上的分布,分析问题产生的原因,提出解决问题的方法,提高解决实际问题的能力。借助数字化学习环境促进小组开展协作,调试修正解决问题的方案,感受数字化学习环境的优势</p>
课后实践与挑战	尝试编写程序,使用复数类型求解一元二次方程		通过课后挑战,引导学生进一步获得更丰富的学习经验,实现知识迁移

## 附：“数据的基本类型”学习单

**【问题 1】**画像上有哪些数据,你能猜测出对应的是哪一位同学吗?

**【问题 2】**哪些数据属于数目,哪些数据属于字符?

### 知识链接 1: 数目与文字符号

数目(number)与文字符号(character, word, symbol)不仅是记载人类文明,还是促进人类文明交流互鉴的两种最基本方式。数目让我们能够描述距离的远近、时间的长短、规模的大小;文字符号让我们能够描述形态、状态、情感、思想和信念。

数目通过数字(digit, 也称“数目字”)来表示,例如一、二、三、四……十、百、千……;1, 2, 3, …, 9 和 0; I, II, III, IV, …, X, C, M, …。它们发端于不同的文明,追求同样的目的(即表示数目)。经过历史长河的洗礼,阿拉伯数字成为主流,其根本原因是基于它形成的进位制能够表达任意大的数目。

进位制思想的一个典型运用——二进制(binary system),则成为当下信息社会的一个基础。

二进制,只需要两个数字——0 和 1,就能表示任意大的数目,以至于我们现在讲的数

字主要指的就是它们。<sup>①</sup>而且,0 和 1 不仅能表示数目,还能用于对文字符号进行编码,常见编码系统如 ASCII 码、汉字国标码 GB1830 - 2005、UTF - 8 等,从而构成了信息处理的数据基础。这些内容我们在本课程必修模块的学习中已多有认识。

数目与文字的一个重要区别,是前者具有“数值”。每个数对应数轴上的一个点,从而具有自然的大小关系,可以对它们有意义地进行算术运算。后者则没有这种特性,它们更强调灵活组合(例如构成任意字符串),以表达现实世界中无比丰富的数值无法刻画的意义。

阅读“知识链接 1”,完成以下练习:

1. 观察同学们提供的数据,区分哪些数据属于数目,哪些数据属于文字符号。

属于数目的是:\_\_\_\_\_。

属于文字符号的是:\_\_\_\_\_。

2. 班级一栏的“202202”属于文字符号还是数目?为什么?

【问题 3】这些数据对应计算机中的哪些基本数据类型?

#### 知识链接 2:数值型和字符型

计算机作为能够处理现实社会各种信息的通用工具,必然要能够有效地表示数目与文字这两个基本概念。在程序设计语言中,与数目和文字相对应的就是数值型和字符型两种基本数据类型。在数据类型的区别下,程序中语句  $x = 1$  和  $x = '1'$  的含义是不一样的,前者是将变量  $x$  赋值为数值 1,后者是将变量  $x$  赋值为字符 1。按照二进制编码,数值 1 的二进制码是 00000001,字符 1 则是 00110001。另一方面,同一个二进制码在不同数据类型中有可能表示不同的含义。例如二进制码串“01000001”既可以表示十进制整数 65,也可以表示 ASCII 码表中对应的字符 A。

在计算机程序语言中,为了应对各种数值计算的需求,按数据表示范围和精度要求的不同(对应存储需求与编码方式的不同),数值型数据通常又分为多种不同的子类型,图 2-2 所示是常见子类型关系示意图<sup>②</sup>,其中也标注了相应的存储需求。

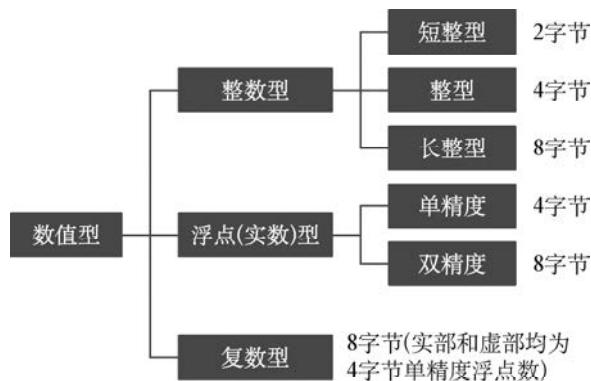


图 2-2 常见子类型关系示意图

<sup>①</sup> 例如,作为信息社会的基础,万物数字化(digitization)就是指将事物在最基础的层面都用 0 和 1 编码,而不是用 0~9 编码。

<sup>②</sup> 不同程序设计语言的具体情况会有些差别。

关于字符型数据,我们需要了解的基础概念首先是字符集,然后是编码方案。字符集规定了其所要表示的字符。一般来说,字符集的大小决定了编码所需要的位数。

数值型数据的范围和精度,是关系到其存储和计算的两个基本问题。图中指出了不同子类型所需的字节数,本质上,也就给出了相应子类型所能表示的数值的个数。例如,短整型最多能表示 $2^{16}$ 个数,整型则能表示 $2^{32}$ 个数,单精度浮点型也最多能表示 $2^{32}$ 个数,等等。

阅读“知识链接2”,完成以下练习:

列出相应子类型所能表示的数值的个数、存储需求,并写出各整数型的范围,填写在表2-4和表2-5中。

表2-4 整数型

	子类型	个数	存储需求	范围
整数型	短整型			
	整型			
	长整型			

表2-5 浮点型

	子类型	个数	存储需求
浮点型	单精度		
	双精度		

### 探究活动

【问题4】在Python编程中实现数值型数据时,需要注意哪些问题?

1. 执行代码,观察执行结果,说出你观察到的问题。

```
>>> x= 0.1+ 0.2
```

```
>>> x== 0.3
```

2. 自主学习“知识链接3”,完成“练一练”。

知识链接3:

日常生活中说的“小数”,在计算机中称为浮点类型数据。为什么用“浮点”这个有点奇怪的名称呢?下面我们会看到一种巧妙的格式设计,它使得小数点的位置可以“浮动”(不同于上面讲的十进制例子,明确规定小数点后是2位),从而使范围和精度达到更好的平衡。这样一种设计,本质上就是对可用的二进制位做一个合适的使用安排和解释。下面我们以8个二进制位为例,解释浮点数表示的基本原理。为简单起见,其中提到的一些“规则”与计算机中实际采用的不尽相同,但不妨碍我们对核心原理的理解。

数据存储时可用的二进制位被分为三个部分:符号(S),指数(E)和尾数(M)。图

2-3 展示的例子中, S 占最左边第 1 位; E 占第 2~4 位; M 则占第 5~8 位。



图 2-3 在一种浮点数格式下  $\frac{11}{4}$  的存储

按照现在给的二进制位的值,这个图中所表达的是什么数(x)呢?

S=0,表示 x 为正数。M 部分字面上是 1011,我们规定将它看成最左边隐含一个小数点的小数,即  $M = (0.1011)_2$ 。E 部分则指出为得到所表达的数 x,需要在 M 的基础上移动小数点的位数。不妨假设 E 中的第 1 位也表示其正(0)负(1),正则右移,负则左移。

现在,  $E = +2$ ,于是  $x = (10.11)_2$ ,换算成十进制数为  $2 + 0 + \frac{1}{2} + \frac{1}{4} = \frac{11}{4}$ ,即 2.75。

练一练:

(1) 如图 2-4 所示,按照给定的浮点数格式,该数字换算成十进制数为\_\_\_\_\_。



图 2-4 二进制浮点数

(2) 按照图 2-5 所示的格式,任意写出三个浮点数,将其换算成十进制数写在旁边的方框内,并尝试在数轴上标出它们的位置。

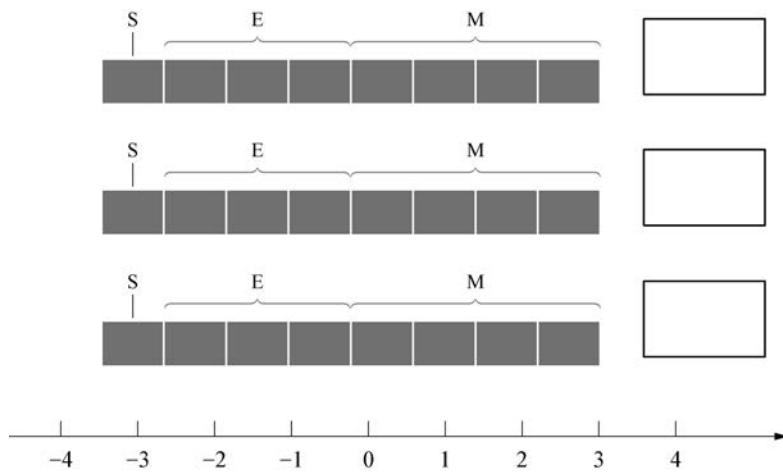


图 2-5 二进制浮点数转换

3. 观察“练一练”中的浮点数在数轴上的分布,思考为什么  $x == 0.3$  的值为 False。
4. 写出解决这一问题的编程方案。
5. 结合编程体验,谈一谈为什么要注意浮点数的精度。

## 第二节

## 常用类型

### 一、教学目标与重点

#### 教学目标:

- 在学习字符串类型存储结构的过程中,掌握字符串类型的基本操作,了解字符串访问方式的特性,形成信息意识。
- 在对比过程中,体会数组存储结构的特点,掌握数组元素的基本操作方法,能正确评估数组的优势和适用场合,形成信息工具选择意识。
- 在学习数组和列表的操作过程中,了解“迭代”的思想方法,提高抽象特征的能力。
- 在对比串和数组特点的过程中,体会数据组织的有效性,提高问题解决方案的迁移能力。

#### 教学重点:

- 字符串的基本操作。
- 数组的基本操作。

### 二、教学实施与评价

#### 1. 教学活动建议

数组是学生接触到的较为形象的涉及逻辑存储和物理存储相关联的数据结构,是其他数据结构的基础,也是熟练掌握 Python 列表操作的实践途径。引导学生结合生活实际,理解使用常用数据类型(字符串和数组)在解决特定问题上的有效性,培养根据现实问题选择并使用数据结构的能力,如可举例字符移位加密、顺序索引同一属性的一批数据等。通过设计范例,比较字符串和数组的异同。从数组可索引的角度出发,引入排序、查找的概念。要使学生充分掌握数组(对应为 Python 列表实现)的操作和特性及上机编程的实际应用。虽然数组是依靠列表具体实现的,但数组逻辑结构涉及的原理还是有必要精讲、细讲的。

#### 2. 教学过程安排(见表 2-6)

表 2-6 教学过程设计表

课时	教学环节	教师活动	学生活动
第 1 课时 (常用类型 1)	1. 情境导入	举例有序同一属性批量数据的存储,举例字符串、一组可索引的相关数据,导出字符串和数组的概念	讨论: (1) 连续存放数据的存储效率。 (2) 引用某一位置元素的方法
	2. 活动 1:字符串的基本操作,分析使用字符串组织和存储数据的方法,体会字符串的作用	讲解、演示: (1) 解释字符串的定义和表示。 (2) 列举用 Python 列表实现字符串的操作	上机实践: 掌握字符串的基本操作,如提取、比较。 讨论: 字符串偏移量加密的实现
	3. 活动 2:数组的基本操作,分析使用数组组织和存储数据的方法,体会数组的作用	讲解、演示: (1) 数组下标运算和数值元素赋值(结合循环语句)。 (2) 数组的特性,数组元素的空间分配	讨论: (1) 数组元素存储访问地址的计算。 (2) 数组访问的效率。 上机实践: 用 Python 列表实现一维数组
	4. 总结	归纳各种常见的数据类型	
第 2 课时 (常用类型 2)	1. 活动:数组与迭代	(1) 列举图像灰度平滑实例,解释迭代的含义。 (2) 二维数组的基本操作	讨论: (1) 一维、二维数组在模拟存储实现上的区别。 (2) 用 Python 列表模拟数组的便利性。 上机实践: 验证 Python 图像平滑实例
	2. 完成项目任务 2	布置项目任务 2	讨论,实现
	3. 总结	归纳数组的应用	

在完成各项目任务的过程中,观察学生对于字符串与数组的理解和实现。在项目实践过程中,考查运用常用类型解决实际问题的能力,可以对学生在以下几方面的表现进行评价:能从存储和访问角度归纳字符串和数组的共同点;能实现字符串和数组中元素的访问;能通过列表和数组访问的性能比较,认同数据结构的重要性;能基于图像平滑计算实践活动阐述迭代的作用和意义。

### 3. 思考探究提示

(1) 本节第 1 个体验思考第 1 小题参考答案:

```
s= "I am a Python learner."
# 方法一
s1= s[2:4]
print(s1)
# 方法二
```

```

s2= list(s)
y= slice(2, 4)
y1= s2[y]
y2= "".join(y1)
print(y2)

```

(2) 本节第 1 个体验思考第 2 小题参考答案:

```

import numpy as np
a= np.random.randint(100,1000,500)
list1= list(a)
b= {}
for i in set(list1):
    b[i]= list1.count(i)
c= sorted(b.items(), key= lambda x: x[0])
print(c)

```

(3) 本节第 2 个体验思考参考答案:

这里每个元素均占 4 个字节, 存储访问以字为单位。因此计算公式为  $L(A) + i * n_2 + j$ 。其中  $n_2$  表示原数组的列,  $i$  和  $j$  表示需求元素的行数和列数。

对于  $A[2,2]$  地址为:  $0AH + 2 * 3 + 2 = 0AH + 08H = 12H$

对于  $A[4,1]$  地址为:  $0AH + 4 * 3 + 1 = 0AH + 0DH = 17H$

(4) 本节第 1 个探究活动参考答案:

```

def kaisa(Py_string, k):
    lower_instr= "abcdefghijklmnopqrstuvwxyz"
    upper_instr= "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    instr= lower_instr+ upper_instr
    outstr= lower_instr[k:]+ lower_instr[:k]+ upper_instr[k:]+
    upper_instr[:k]                                # 密文
    transtable= str.maketrans(instr,outstr)          # 形成映射表
    aftertrans= Py_string.translate(transtable)
                                            # 根据映射表翻译字符串
    return aftertrans
print(kaisa("Hello", 4))

```

(5) 本节第 2 个探究活动参考答案:

```

import copy
import numpy as np
image= np.zeros((9,16))
for j in range(4,12):
    image[2,j]= 1

```

```

    image[6,j]=1
for i in range(3,6):
    image[i,4]=1
    image[i,5]=1
    image[i,10]=1
    image[i,11]=1
imaget= copy.copy(image)
print(image)
for j in range(15):
    for i in range(8):
        image[i,j]=(imaget[i-1,j]+imaget[i+1,j]+imaget[i,j-1]+
        imaget[i,j+1])/4
        imaget[i,j]=image[i,j]
print(image) # 第1次迭代后数据
imaget= copy.copy(image)
for j in range(15):
    for i in range(8):
        image[i,j]=(imaget[i-1,j]+imaget[i+1,j]+imaget[i,j-1]+
        imaget[i,j+1])/4
        imaget[i,j]=image[i,j]
print(image) # 第2次迭代后数据
imaget= copy.copy(image)
for j in range(15):
    for i in range(8):
        image[i,j]=(imaget[i-1,j]+imaget[i+1,j]+imaget[i,j-1]+
        imaget[i,j+1])/4
        imaget[i,j]=image[i,j]
print(image) # 第3次迭代后数据

```

#### 4. 项目实施提示

完成项目任务2。可以采用距离公式计算。如同学1体重和身高为( $w_1, h_1$ )，同学2体重和身高为( $w_2, h_2$ )，那么按欧氏距离公式计算相似度即为 $(w_1 - w_2)^2 + (h_1 - h_2)^2$ 。进一步思考，体重数据差和身高数据差对结果的影响不同，因此一般我们把数据进行归一化或标准化。这里介绍一种线性归一化，也称为离差标准化，是对原始数据的线性变换，使得结果映射到0~1之间。公式为 $x' = (x - X_{\min}) / (X_{\max} - X_{\min})$ 。其中： $X_{\max}$ 为样本数据的最大值， $X_{\min}$ 为样本数据的最小值， $x$ 为原始数据， $x'$ 为归一化后0~1之间的一个值。

### 三、作业练习与提示

#### ■ 题目描述

用数组实现大整数的乘法。使用两个一维数组分别记录两个十进制大整数(数组的每个元素对应一位数),并做这两个大整数的乘法。(注:由于Python中的整数可以任意大,实际应用中没有必要在Python中专门实现大整数的乘法。此活动的目的是熟悉数组的操作。)

#### ■ 作业练习参考答案

第一个数的第*i*位和第二个数的第*j*位相乘,一定要累加到结果的第*i+j*位上,这里是<0位置开始算的。

简单来说,就是将两数按位相乘,乘积与结果对应位累加,累加时先不算任何进位,最后再计算进位。

例如:计算 $98 \times 21$ ,步骤如下

$$\begin{array}{r} 9 & 8 \\ \times & 2 & 1 \\ \hline (9) & (8) & \leftarrow \text{第1趟: } 98 \times 1 \text{ 的每一位结果} \\ (18) & (16) & \leftarrow \text{第2趟: } 98 \times 2 \text{ 的每一位结果} \\ \hline (18) & (25) & (8) \leftarrow \text{这里就是相对位的和,还没有累加进位} \end{array}$$

这里唯一要注意的便是进位问题,可先不考虑进位,当所有位对应相加,产生结果之后,再考虑。从低位向高位依次运算,如果该位的数字大于10,用取余运算,在该位上只保留取余后的个位数,而将十位数进位(通过模运算得到)累加到高位,循环直到累加完毕。

参考程序如下:

```
def list2str(li):
    while li[0]==0:
        del li[0]
    res=''
    for i in li:
        res+=str(i)
    return res

def multi(inta, intb):
    stra=str(inta) # 将数字a转变成字符串a
    strb=str(intb) # 将数字b转变成字符串b
    aa=list(stra) # 将字符串a转变成数组aa
    bb=list(strb) # 将字符串b转变成数组bb
    lena=len(stra)
    lenb=len(strb)
```

```

result=[0 for i in range(lena + lenb)]
for i in range(lena):
    for j in range(lenb):
        # 将两数按位相乘,乘积与结果对应位累加
        result[lena- i - 1+ lenb- j - 1] += int(aa[i]) * int(bb[j])
for i in range(len(result) - 1): # 计算进位
    if result[i] >= 10:
        result[i+ 1] += result[i] // 10
        result[i] = result[i] % 10
return list2str(result[::-1])

```

## 四、核心概念介绍

### 常用数据类型：

常用数据类型包括数组、字符串等。它们的第一个特性是以一个名称代表多个相同数据类型的数据；第二个特性是它们中的数据元素被存储在一段连续的数据空间中，我们可以以索引的方式访问其中的数据元素。

在实践意义上，数组和字符串是“常用”的，基本数据类型也是“常用”的。

## 五、教学参考资源

Python 提供一组可以在列表或数组上使用的内建方法，如表 2-7 所示。

表 2-7 Python 内建方法

方法	描述
append()	在列表的末尾添加一个元素
clear()	删除列表中的所有元素
copy()	返回列表的副本
count()	返回具有指定值的元素数量
extend()	将列表元素(或任何可迭代的元素)添加到当前列表的末尾
index()	返回具有指定值的第一个元素的索引
insert()	在指定位置添加元素
pop()	删除指定位置的元素
remove()	删除具有指定值的项目
reverse()	颠倒列表的顺序
sort()	对列表进行排序

注释:Python 没有内置对数组的支持,但可以使用 Python 列表代替数组。

——参考《Python 程序设计》,董付国,清华大学出版社

## 六、教学参考案例

### ■ 参考案例

#### 数    组

上海市杨浦区教育学院 白晓琦

(1课时)

##### 1. 学科核心素养

创设运用数组存储照片标注信息的情境,尝试运用生活中的实例描述数据的内涵与外延,能够将有限制条件的、复杂生活情境中的关系进行抽象,用数据结构表达数据的逻辑关系。(信息意识)

依托数组存储结构,编写程序实现图像的灰度平滑,体验采用计算机可以处理的方式界定问题、抽象特征、建立结构模型、合理组织数据。(计算思维)

在项目活动中选择适合的存储方式,尝试评估常见的数字化资源与工具对学习数据结构的价值,根据需要进行合理选择。(数字化学习与创新)

##### 2. 《课程标准》要求

通过案例分析,理解数组、链表等基本数据结构的概念,并能编程实现其相关操作。比较数组、链表的区别,明确上述两种数据结构在存储不同类型数据中的应用。

##### 3. 学业要求

能够从数据结构的视角审视基于数组、链表的程序,解释程序中数据的组织形式,描述数据的逻辑结构及其操作,评判其中数据结构运用的合理性。

##### 4. 教学内容分析

数组是教科书第二章数据类型第二节“常用类型”中的内容。教科书中常用类型主要包括字符串和数组,因此本节内容分为 2 课时,本节课数组是第 2 课时。字符串中的每个字符和数组中的元素在逻辑关系上都是线性结构,在存储上也通常采用顺序存储结构。从原理上分析,上节课字符串的学习为本节课数组的学习打下了一定基础。数组是一种数据结构,使用数组还可以构造其他数据结构,如栈、队列等,因此数组这节课起着承上启下的作用。

##### 5. 学情分析

在高一必修课中学生学习了 Python 的基本语法及基本操作,也掌握了 Python 中列表的概念、特性等。数组和列表有相似之处。因为数组是学生接触的第一种数据结构,所以学生更有必要掌握其原理及存储方式。充分理解数组的特性及物理存储,不仅有利于了解其和列表的区别,更可为其他数据结构的学习打下良好的基础。另外,高中生思维活跃,具有很强的探究意识,使用数组解决实际问题能激发学生的学习兴趣。

## 6. 教学目标

- 通过案例分析,理解数组的含义、特性和操作方式,并尝试用生活中的实例描述其内涵和外延。
- 通过程序效率的对比,理解数组的基本原理,总结选择恰当数据结构实现算法的依据。
- 通过图像的灰度平滑,理解数组、循环与迭代的含义,实践解决问题的过程。
- 通过编程实践,体会数组在顺序数据组织上的有效性,体验存储结构选择的必要性,经历用计算机思想解决问题的过程。

## 7. 教学重难点

- 教学重点:理解数组基本原理。
- 教学难点:理解数组、循环与迭代的含义。

## 8. 教学策略分析

本节课主要采用案例分析和编程实践形式,围绕数组的概念、特性及应用三个部分进行教学开展。

(1) 设置情境导入,引出核心概念。

以计算多个数的平均值为情境,通过需求分析,引出数组的概念。

(2) 编程对比实践,揭示数组特性。

分别使用数组和列表编程解决相同的问题,通过统计程序运行所需的时间,比对各自效率,分析使用数组效率高的原因。

(3) 解决真实问题,实现数组应用。

以图像灰度计算为情境,灵活使用二维数组解决实际问题。

## 9. 教学过程设计(见图 2-6 和表 2-8)

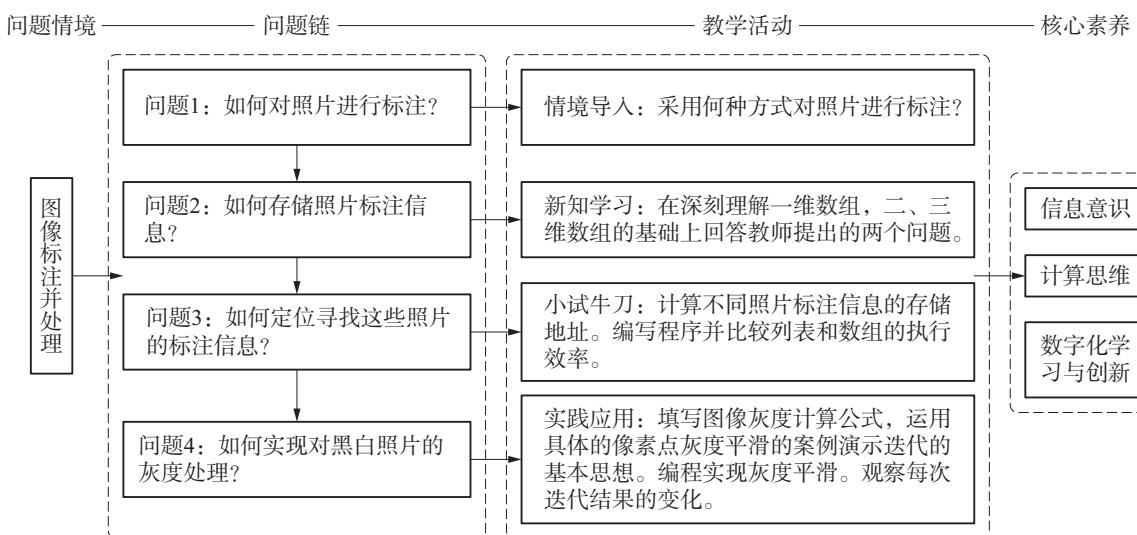


图 2-6 教学过程

表 2-8 教学过程设计表

教学环节	教师活动	学生活动	设计意图
情境导入	<p>问题 1: 如何对照片进行标注?</p> <p>小申收集了爷爷奶奶年轻时的很多黑白照片,准备重新制作整理成视频送给爷爷奶奶作为金婚纪念日的礼物。由于照片太多,小申希望能够对照片进行编号标注并进一步对照片进行处理与完善。但是他遇到了一些难题,例如:如何对收集到的照片进行编号?如何对这些照片进行灰度处理?</p>	<p>交流讨论:采用何种方式对照片进行标注,运用何种数据类型存储这些标注信息?</p>	创设情境,提出问题,引发学生思考,并激发学习新知的渴望
新知学习	<p>问题 2: 如何存储照片标注信息?</p> <p>布置自主学习任务。</p> <p>提供知识链接:</p> <p>(1) 数组的概念、下标;</p> <p>(2) 一维数组和二维数组的区别。</p> <p>提供知识链接:</p> <p>黑白图像的灰度值以及彩色图像的 RGB 值的概念及原理。</p> <p>提出问题:</p> <p>(1) 照片的名称标注、黑白图像的灰度值和彩色图像的 RGB 值分别适合用哪种数组来表示?</p> <p>(2) 假设用数组 A 存储照片的名称标注信息,数组 B 存放某张照片的灰度值,那么如何访问第 n 张照片,如何访问该照片的第三行第四列的灰度值?</p>	<p>(1) 学生自主学习,并讨论交流;</p> <p>(2) 在深刻理解一维数组、二维数组、三维数组的基础上,回答老师提出的两个问题</p>	理解数组的概念、数组元素的访问,提升学生的信息意识
小试牛刀	<p>问题 3: 如何定位寻找这些照片的标注信息?</p> <p>提供知识链接:</p> <p>数组的性质。</p> <p>提出问题:</p> <p>假设这些照片的标注信息存储在数组 A 中,其中第一张照片的地址是 L[0],每个元素占用 m 个字节,那么 A[i]照片标注信息的存储地址是多少?请用公式表示。</p> <p>提供代码片段:</p> <p>数组访问和列表访问的代码片段,并设置填空</p>	<p>(1) 思考计算不同照片标注信息的存储地址;</p> <p>(2) 在空白处补充代码,分别实现数组访问和列表访问照片的标注信息;</p> <p>(3) 对比数组访问和列表访问的执行效率</p>	引导理解数组的特性和存储方式,启发学生寻找不同的方式解决问题并进行比较分析
实践应用	<p>问题 4: 如何实现对黑白照片的灰度处理?</p> <p>提供知识链接:</p> <p>图像灰度平滑及迭代的相关材料。</p> <p>提出问题:</p> <p>(1) 请写出图像灰度计算公式;</p> <p>(2) 基于针对具体的像素点实现灰度平滑的过程阐述迭代的思想;</p> <p>(3) 运用循环结构编程实现图像灰度平滑</p>	<p>(1) 自主学习图像灰度平滑的相关知识,并选择和写出适合的图像灰度计算公式;</p> <p>(2) 通过针对具体的像素点灰度平滑的案例演示迭代的基本思想;</p> <p>(3) 编程实现图像灰度平滑;</p> <p>(4) 运行程序,并观察每次迭代结果的变化</p>	通过具体实例应用,深刻理解数组的基本应用及迭代思想,注重学生计算思维的养成

续表

教学环节	教师活动	学生活动	设计意图
拓展延伸	布置任务： 通过查阅相关资料和网站,进行数字化学习,寻找更多的解决方法,创造性解决图像灰度平滑问题,并阐述该方法的优势	课下开展数字化学习图像灰度平滑的相关知识,设计解决方案并阐述优势	依托数字化资源和工具,设计最优方案,创造性解决问题

## 附：“数组”学习单

- 如何对照片进行标注?

\_\_\_\_\_。

- 如何存储照片标注信息?

1. 以下信息分别适合用哪种数组存储? (一维数组、二维数组、三维数组)

照片的名称标注: \_\_\_\_\_;

黑白图像的灰度值: \_\_\_\_\_;

彩色图像的 RGB 值: \_\_\_\_\_。

2. 假设用数组 A 存储照片的名称标注信息,数组 B 存放某张照片的灰度值,请在横线处填写相应的数组元素。

访问第 n 张照片的名称标注信息: \_\_\_\_\_;

访问该照片的第三行第四列的灰度值: \_\_\_\_\_。

- 如何定位寻找这些照片的标注信息?

1. 假设这些照片的标注信息存储在数组 A 中,其中第一张照片的地址是 L[0],每个元素占用 m 个字节,那么 A[i] 照片的标注信息的存储地址是多少? (请用公式表示)

\_\_\_\_\_。

2. 在空白处补充代码,分别实现数组访问和列表访问照片的标注信息。

```
import time
import random
import numpy as np
a = []
for i in range(20):# 假设有 20 张照片
    a.append(random.randint(0,100))# 假设用随机数表示图片的标注
# 方法一 列表求和
t1= time.time()
for i in range(20):
    print(_____)# 访问每项列表的值
print('访问列表花费的时间:',time.time()- t1)
# 方法二 数组求和
b= np.array(a)
```

```
t1= time.time()  
for i in range(20):  
    print(_____)      # 访问每个数组元素  
print('访问数组花费的时间:', time.time() - t1)
```

3. 根据调试结果,比较并说明数组访问和列表访问的执行效率。

- 如何实现对黑白照片的灰度处理?

1. 自主学习图像灰度平滑的相关知识,并选择和写出适合的图像灰度计算公式:

\_\_\_\_\_。

2. 通过针对具体的像素点灰度平滑的案例演示迭代的基本思想。

3. 编程实现灰度平滑。

4. 运行程序,并记录每次迭代结果的变化。

5. 课下自学灰度平滑的相关知识,寻找还有哪些图像灰度平滑算法。

- 知识链接

### 数组的概念、下标:

数组(array)数据类型,用一个符号名来统一表示这一组数据,而用下标(index)的方式区别其中的数据元素,如  $h[0], h[1], \dots, h[19]$ 。

### 图像的存储:

人们关心的图像数据呈现为阵列的形式,例如一幅  $1024 \times 768$  像素的黑白图像,其每个像素的灰度值自然地排成一个二维阵列,做图像处理就要涉及一个像素与周围像素之间的关系。如果是一幅彩色图像,每个像素则由红、绿、蓝(RGB)3种颜色表示,每种颜色分别对应一个二进制数,于是涉及的数值有  $1024 \times 768 \times 3$  个,我们可以在脑海中将其想象为一个三维的数据体。

### 灰度值:

由于景物各点的颜色及亮度不同,拍摄的黑白照片上或电视机重现的黑白图像上各点呈现不同程度的灰色。把白色与黑色之间按对数关系分成若干级,称为“灰度等级”,范围一般从  $0 \sim 255$ ,白色为 255,黑色为 0,故黑白图片也称灰度图像。灰度值在医学、图像识别领域有很广泛的用途。

### 彩色图像的 RGB:

每一种颜色在视觉效果上都可以用不同比例的红、绿、蓝三种颜色来合成,彩色图像的每一个像素,都是用红、绿、蓝这三种颜色合成的,这样的图像即为 RGB 图像。RGB 分量:R、G、B 三个颜色对应的数值是 RGB 图像的分量,每一个分量的取值范围为  $0 \sim 255$ 。

### 数组的性质:

数组作为一种数据类型,具有以下最主要的特性:一是其中的数据元素具有相同的基础类型;二是在程序中一旦声明,其规模就不再变化。

### 图像灰度平滑:

所谓图像的灰度平滑,一个简单的方法就是用像素点  $[i, j]$  周围 4 个像素点的灰度值的平均值取代原灰度值,程序为两重循环结构,循环内进行如下操作:

```
image[i, j]=(image[i-1, j]+ image[i+1, j]+ image[i, j-1]+ image[i, j+1])/4
```

### 迭代：

迭代就是重复做相同的操作,操作对象往往是数组中的不同元素,或者同一个数据元素的不同取值。为实现迭代,采用循环语句是最自然有效的方式。在循环语句的控制下,数组元素不仅会依序得到处理,而且会得到反复处理。

## 第三节 组合类型

### 一、教学目标与重点

#### 教学目标:

- 通过举例和类比,掌握正确表述对象属性的方法,提高界定问题的能力。
- 通过对数据集、数据项,了解数据元素的关联形式,形成对结构模型的归纳意识。
- 通过学习组合类型及列表的实现方式,掌握构建数据项的基本方法,形成信息工具选择意识。
- 通过辨析复杂数据类型的结构,体会数据结构抽象性的特点,树立规范表达的意识。

#### 教学重点:

- 数据集、数据项的关系及其联系与区别。
- Python 列表嵌套实现的组合数据类型构建和基本操作。

### 二、教学实施与评价

#### 1. 教学活动建议

本节内容是由基本数据类型向“结构体”为代表的抽象数据类型的过渡,也是帮助学生熟练掌握 Python 列表操作的重要环节。教师要注意引导学生在上节使用列表存储、索引同一属性数据的基础上,区分本节不同属性的数据作为列表的项。教学所涉及实例可侧重于列表的嵌套和元素赋值、访问(遍历)。

#### 2. 教学过程安排(见表 2-9)

表 2-9 教学过程设计表

课时	教学环节	教师活动	学生活动
第 1 课时 (组合类型)	1. 情境导入	举例现实生活中数据对象包含众多相关联的数据项目,其数据类型不同	讨论: 单一数据类型在表述对象时的不足
	2. 活动 1:认识数据元素和数据项	讲解、演示 (1) 数据集的含义和表示。 (2) 数据项属性之间的关系	讨论: 用基本数据类型和数组实现数据集的可行性
	3. 活动 2:组合数据类型的表示和实现	讲解、演示 (1) 组合类型的各数据元素及其属性界定。 (2) Python 列表的实现	讨论: 有效确定数据集内各数据项的属性。 上机实践: 使用 Python 列表嵌套实现组合数据类型的赋值、索引和遍历
	4. 项目实践:完成项目任务 1	布置项目任务 1	讨论,实现
	5. 总结	归纳组合类型的特点	

在完成各项目任务的过程中,观察学生对于组合类型的理解。在项目实践过程中,考查运用组合类型解决实际问题的能力,可以对学生在以下几方面的表现进行评价:能够解释数据元素和数据项;能举例说明组合数据类型中包括不同类型数据项的数据元素;能描述组合数据类型的表示方式;能结合实例用组合数据类型表示数据。

### 3. 思考探究提示

#### (1) 本节体验思考参考答案:

药材类似各种数据项。药柜存储着各种药材,类似数据存储。各种药材通过药方组织起来成为各种元素。

#### (2) 本节探究活动参考答案:

对于具体某位学生的描述,要实现以学校、班级、学生这种顺序按层次访问,就可以用三重嵌套列表来组织数据。

## 三、作业练习与提示

### ■ 题目描述

结合本章项目情境,设计一个描述学生的数据(元素)表示方案,即需要哪些数据项、各数据项分别是什么类型、可以取哪些值,并完成计算两个数据元素相似度的函数。

### ■ 作业练习参考答案

学生的数据(元素)包括姓名、性别、身高、体重、兴趣爱好等数据项,分别为字符型、字符型、数值型、数值型、字符型。相似度可以采用欧氏距离公式来计算。

## 四、核心概念介绍

组合数据类型：

区别于上述常用数据类型,组合数据类型强调的是一个数据(在程序中即一个名称)可由多个不同类型的数据分量构成。组合数据类型的描述和访问方式在不同的程序语言中有所不同,经典的如C语言中由struct所定义的数据。现在也有将Python中的列表、元组、集合、字典看作是组合数据类型,因为它们也可以包含不同类型的数据元素。其中,特别值得一提的是列表,它功能强大,除了可以有不同类型的数据元素之外,其可嵌套性和通过下标访问数据元素的特点,使得它可以方便地作为数组使用。

## 五、教学参考资源

### ■ 阅读材料：Python 中的组合数据类型

在 Python 中通过列表(list),元组(tuple),集合(set),字典(dict)可以方便地构建组合数据类型。

#### 1. 列表(list)

列表可以包容多种类型数据,不管是字符串(str)、数字(int、float),还是布尔(bool)值,都可以放进列表中,成为我们可以随时更改、查阅以及调用的数据组合。

#### 2. 元组(tuple)

元组也可以存放多种类型的数据,但是,它一旦被声明出来以后就无法更改里面的內容,所以元组可以理解为特殊的列表。

#### 3. 集合(set)

集合是一个无序的、元素不可重复的数据类型,集合中各元素之间用逗号隔开,可以使用大括号{}或者set()函数创建集合。

#### 4. 字典(dict)

字典是由若干个“键:值”组成的无序可变序列。每个键都与一个值相关联,可以使用键来访问与之相关联的值,与键相关联的值可以是数字、字符串、列表乃至字典。

——参考《Python 语言程序设计基础》,嵩天、礼欣、黄天羽,高等教育出版社

## 六、教学参考案例

### ■ 参考案例

#### 制作“个人数据画像”表——组合类型

上海市中原中学 张汉玉

(1课时)

#### 1. 学科核心素养

能够根据解决问题的需要,自觉、主动地寻求恰当的方式获取与处理信息;在小组交

流讨论的过程中,愿意与团队成员共享信息,完善个人数据画像的数据,实现信息的更大价值。(信息意识)

在编程实现个人数据表的过程中能够按照计算机可以处理的方式界定问题、抽象特征、建立结构模型、合理组织数据。运用 Python 语言编写程序,能够读入个人画像的组合数据,并支持访问数据项及做简单处理。(计算思维)

通过评估并选用常见的数字化资源与工具,有效地学习数据相似度的相关知识,创造性地解决同学间的相似问题。(数字化学习与创新)

## 2.《课程标准》要求

结合生活实际,理解组合类型的基本概念,认识组合类型在解决问题过程中的重要作用。

能够对个人数据画像问题进行分析,选择组合类型,并能够运用 Python 语言编程实现。

## 3. 学业要求

学生能够运用生活中的实例描述数据的内涵与外延,能够将有限制条件的、复杂生活情境中的关系进行抽象,用数据结构表达数据的逻辑关系。

能够针对限定条件的实际问题进行数据抽象,运用数据结构合理组织、存储数据,选择合适的算法编程实现、解决问题。

## 4. 教学内容分析

组合类型是教科书第二章第三节的内容,涵盖数据元素与数据项的基本概念及组合数据类型的表示。教科书创设“谈一谈身边同学”的项目情境,介绍了数据集、数据元素、数据项,以及运用 Python 语言表示组合数据类型,并进一步引导学生计算数据项间的相似度。

## 5. 学情分析

学生具备一定的 Python 语言编程基础,通过前面章节内容的学习了解了程序设计中的基本数据类型,并能够根据现实需求选择不同的数据类型描述客观事物。同时,学生具备较强的思维逻辑能力和数学计算能力,能够运用数学公式解决现实问题。

## 6. 教学目标

- 通过小组讨论、分析交流,设计“个人数据画像”表,厘清数据集、数据元素、数据项的基本概念,提升信息意识。

- 在制作“个人数据画像”表过程中,按照计算机可以处理的方式,认识组合数据类型,并能够运用 Python 语言表示组合类型。

- 通过数字化学习获取相应的学习资源,并能够根据实际情况,具体问题具体分析,选择合适的函数计算数据项的相似度,注重计算思维的养成。

## 7. 教学重难点

- 教学重点:用 Python 语言表示组合类型。

- 教学难点:计算数据项的相似度。

## 8. 教学策略分析

- (1) 本节课立足“谈一谈身边同学”的单元项目情境,制订了制作“个人数据画像”

表的学习任务，并围绕该任务设计了系列学习活动：设计“个人数据画像”表→介绍“个人数据画像”表→表示“个人数据画像”表→完善“个人数据画像”表→计算“个人数据画像”表。为保障各项学习活动顺利开展，设置了层层递进的学习环节，逐步引导学生厘清基本概念，掌握如何运用 Python 语言表示组合类型，并能够运用函数计算数据项的相似度。

(2) 通过学习单提供“参考示例”，将抽象问题形象化，启发学生从列表的嵌套扩展到三重嵌套列表，促进学习迁移的发生。

### 9. 教学过程设计(见图 2-7 和表 2-10)

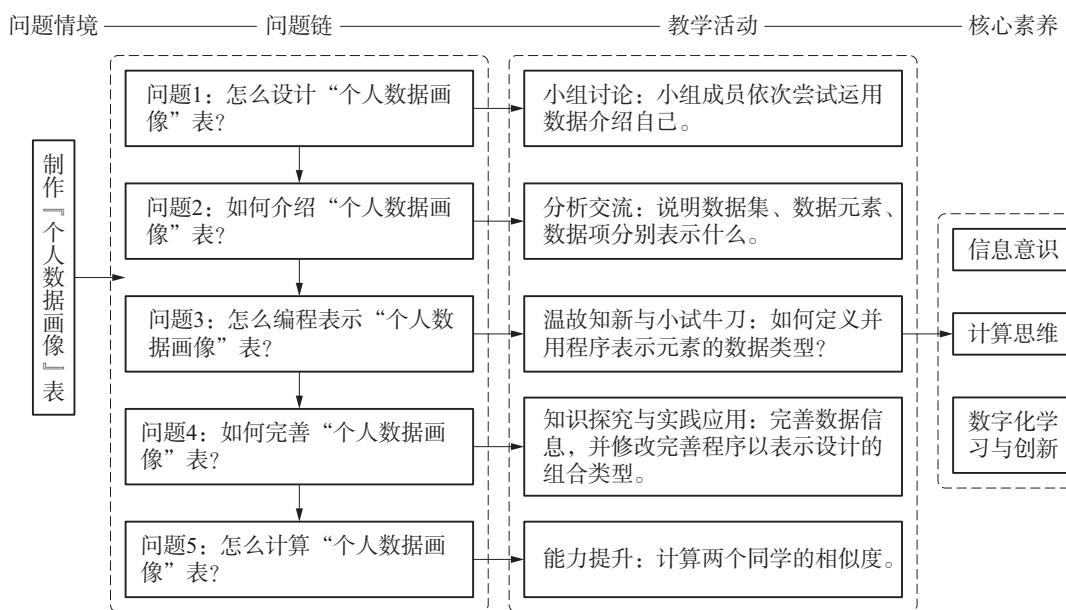


图 2-7 教学过程

表 2-10 教学过程设计表

教学环节	教师活动	学生活动	设计意图
热身活动	问题 1: 怎么设计“个人数据画像”表? 引导同学们分小组活动，尝试用数据介绍自己	小组成员依次尝试运用数据介绍自己，各小组制作一张数据表格并展示分享	加深对数据的理解，并为组合类型的学习提供素材支撑
分析交流	问题 2: 如何介绍“个人数据画像”表? 提供数据集、数据元素、数据项基本概念的相关材料。 提问：结合数据表格，举例说明数据集、数据元素、数据项分别表示什么。	自主学习数据集、数据元素、数据项的基本概念，并结合小组制作的数据表格，举例说明数据集、数据元素、数据项分别表示什么。 数据集：_____。 数据元素：_____。 数据项：_____。 数据集的基本单位是_____； 最小可处理单位是_____。	结合实例厘清数据集、数据元素与数据项的基本概念，提升学生的信息意识

教学环节	教师活动	学生活动	设计意图																		
温故知新	问题3:怎么编程表示“个人数据画像”表? 提供组合数据类型的基本概念。 提问:如何定义每个数据元素(每位同学)的数据类型?	说一说“个人数据画像”表中每个数据项的数据类型。 思考如何定义每个数据元素(每位同学)的数据类型。 <b>【小试牛刀】</b> 在空白处补充代码; 运用Python语言编写程序,能够读入表格数据,并支持访问数据项及做简单处理	回顾基本的数据类型,并学习新授内容——组合数据类型;学会运用Python语言表示组合类型																		
知识探究	问题4:如何完善“个人数据画像”表? 提供参考示例如下:  <table border="1" style="margin-left: auto; margin-right: auto; text-align: center;"> <tr><th colspan="6">喜欢的电影类型</th></tr> <tr><td>爱情片</td><td>动作片</td><td>科幻片</td><td>喜剧片</td><td>战争片</td><td>恐怖片</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td></tr> </table> 提问: 请根据参考示例,尝试添加新的数据项描述个人信息,让“个人数据画像”表更加丰富。(提示:可运用列表嵌套表示数据)	喜欢的电影类型						爱情片	动作片	科幻片	喜剧片	战争片	恐怖片	0	1	1	0	0	0	根据参考示例,通过添加新的数据项进一步完善数据信息,让人物信息更加丰富。 小组交流分享修改后的数据表包含哪些数据项,每个数据项的类型及取值如何	引导学生思考运用列表的嵌套描述数据,表达极其丰富的数据对象,促进学生计算思维的养成
喜欢的电影类型																					
爱情片	动作片	科幻片	喜剧片	战争片	恐怖片																
0	1	1	0	0	0																
实践应用	请同学们编写程序表示设计的组合类型	<b>【实践应用】</b> 请参照你设计的数据表,进一步修改完善以上程序,并简要说明其能够实现的功能																			
能力提升	问题5:怎么计算“个人数据画像”表? 提供相似度计算的相关资料。 提问:某两位同学的数据相似度如何?	查阅资料,完成计算两个数据元素相似度的函数。 根据“个人数据画像”表中的数据,计算两个同学的相似度	通过数字化学习选择合适的函数,计算两个数据元素在某方面的相似度																		
作业	查阅资料,自主学习不同的相似度计算方法,并说明每种方法的适用条件		通过数字化学习获取相应的学习资源,并能够根据实际情况,具体问题具体分析,选择合适的函数解决问题;注重培养学生的数字化学习能力并启发其创造性地解决问题																		

## 附：“组合类型”学习单

- 怎么设计“个人数据画像”表?

小组成员依次尝试运用数据介绍自己,各小组制作一张数据表格并展示分享。


- 如何介绍“个人数据画像”表？

自主学习数据集、数据元素、数据项的基本概念，并结合小组制作的数据表格，回答以下问题：

数据集：\_\_\_\_\_。

数据元素：\_\_\_\_\_。

数据项：\_\_\_\_\_。

数据集的基本单位是\_\_\_\_\_；最小可处理单位是\_\_\_\_\_。

- 怎么编程表示“个人数据画像”表？

运用 Python 语言，读入表格数据，并能够访问数据项及做简单处理。

请在空白处补充代码：

```
student= _____ # 创建 student 列表
for i in range(2):
    name,id,weight= eval(input('enter next student info:'))
    student.append(_____) # 添加列表的数据元素
print('name',_____) # 输出每个学生的姓名
print('id',_____) # 输出每个学生的 id
print('average of weights:',_____) # 输出平均体重
```

测试数据可参考表 2-11。

表 2-11 测试数据表

name	id	weight
"zhang"	101	45
"li"	102	55

- 如何完善“个人数据画像”表？

参考示例见表 2-12。

表 2-12 参考示例表

喜欢的电影类型					
爱情片	动作片	科幻片	喜剧片	战争片	恐怖片
0	1	1	0	0	0

请根据参考示例,尝试添加新的数据项描述个人信息,让“个人数据画像”表更加丰富  
(提示:可运用列表嵌套表示数据)。

完善后的数据表:


编程实现:

参考代码:

```
student= list()
for i in range(2):

    name,id,weight,[romance,action,science,comedy,war,horror]= eval
    (input('enter next student info:'))

    student.append([name,id,weight,[romance,action,science,comedy,
    war,horror]])

print('name:',student[0][0],student[1][0])
print('id:',student[0][1],student[1][1])
print('average of weights:',(student[0][2]+ student[1][2])/2)
print('film',student[0][3],student[1][3])
```

- 怎么计算“个人数据画像”表?

运用相似度函数,计算“个人数据画像”表中两个同学的相似度。

- 知识链接:

1. 数据集、数据元素、数据项

“数据集”,可能存放在一个文件中,是某个电子表格处理软件的输入或输出。此数据集中的每一个“数据元素”(有时简称“元素”),对应表格中的每一行,而每个数据元素中每一列对应的称为“数据项”。数据元素是数据(集)的基本单位,可由若干个数据项组成,数据项则是最小的可处理单位。

## 2. 组合数据类型

数据元素的数据项不一定是同类型的。用第二章第一节的基本数据类型来描述的话,姓名是字符型,身高和体重是浮点型(可以进行求平均值等统计运算),性别是字符型,学号可以是字符型也可以是整型,这样的数据元素在计算机程序中如何表示?此内容正是本节的中心话题——组合数据类型。

## 3. 相似度计算方法

欧氏距离(euclidean distance):

欧氏距离全称是欧几里得距离,是最易于理解的一种距离计算方式,源自欧氏空间中两点间的距离公式。

平面空间内的 A(x<sub>1</sub>, y<sub>1</sub>)与 B(x<sub>2</sub>,y<sub>2</sub>) 间的欧氏距离:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2};$$

三维空间里的欧氏距离:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}.$$

# 第四节 抽象类型

## 一、教学目标与重点

### 教学目标:

- 通过归纳,理解抽象数据类型的组成要素,形成准确表达信息的意识。
- 通过举例和对比,了解抽象数据类型的性质和封装的基本方法,形成信息工具的选择意识。
- 通过操作自定义数据类型,掌握其实现的基本方法,树立工具使用的有效性意识。
- 通过归纳数据封装的抽象方法,体会利用合理的数据表达来提升解决问题效率的优势。

### 教学重点:

- 数据集与数据封装的联系与区别。
- 用 Python 类实现抽象数据类型表示和基本操作。

## 二、教学实施与评价

### 1. 教学活动建议

本节应从生活中有关对象及其属性的类比入手,引导学生建立起“分析对象特征——抽象数据属性——实现数据操作”的流程处理意识。通过对身边熟悉的对象的属性、属性关联的分析,帮助学生熟悉“结构体”的含义。要结合 Python 类中具体的操作,包括创建自定义类、属性和方法,以实现具体应用。

### 2. 教学过程安排(见表 2-13)

表 2-13 教学过程设计表

课时	教学环节	教师活动	学生活动
第 1 课时 (抽象类型 1)	1. 情境导入	举例说明现实中数据对象的复杂性和属性的多样性,以及由此产生的对数据操作的特殊性	回顾基本数据类型和组合数据类型相关内容
	2. 活动 1: 了解数据抽象的含义	讲解、演示: (1) 数据抽象过程; (2) 使用自定义方式实现抽象数据类型的可行性	讨论: 用基本数据类型和数组实现数据集的可行性。 上机实践: 验证大数乘法例子
	3. 活动 2: Python 类的定义与使用	讲解、演示 (1) Python 类的一般构成要素; (2) Python 类的定义和调用	讨论: 对比必修 1 分册中的函数调用的方式。 上机实践: 用 Python 实现自定义类型的赋值和访问
	4. 总结	归纳抽象数据类型的定义和特点	
第 2 课时 (抽象类型 2)	1. 活动: 使用类封装,实现抽象数据类型的基本功能	讲解、演示: (1) 分析具体问题中所需数据类型及其属性; (2) 使用 Python 的实现方法	讨论: 由列表方式实现组合数据类型。 上机实践: 用 Python 实现自定义类型的创建和基本操作
	2. 项目实践: 完成项目任务 3、项目任务 4	布置项目任务 3、项目任务 4	讨论分析, 编程实现各任务
	3. 总结	归纳抽象数据类型的实现和应用	

在完成各项目任务的过程中,观察学生对于抽象数据类型的理解。在项目实践过程中,考查运用抽象类型解决实际问题的能力,可以对学生在以下方面的表现进行评价:能结合实例解释数据抽象的含义和作用;能描述抽象数据类型的定义;能结合实例使用类的方法设计并实现抽象数据类型;能通过案例分析、评价数据封装的意义。

### 3. 思考探究提示

本节探究活动参考答案:

```

class Triple():
    def __init__(this,data1,data2,data3):
        this.data=[data1,data2,data3]
    def win(this):
        if this.data[0]>=this.data[1]:
            if this.data[0]>=this.data[2]:
                return(this.data[0])
            else:
                return(this.data[2])
        if this.data[1]>=this.data[2]:
            return(this.data[1])
        else:
            return(this.data[2])
    def __getitem__(this,index):
        t=this.data
        return t[index]
    def __setitem__(this, index, value):
        this.data[index]=value
    def add_triple(x,y):
        z=Triple(0,0,0)
        z[0]=x[0]+y[0]
        z[1]=x[1]+y[1]
        z[2]=x[2]+y[2]
        return z
    a,b,c=eval(input('a,b,c:'))
    x=Triple(a,b,c)
    a,b,c=eval(input('a,b,c:'))
    y=Triple(a,b,c)
    z=add_triple(x,y)
    print(z.win())

```

#### 4. 项目实施提示

完成项目任务3,设计同学类。参考如下:

```

class Student:
    def __init__(self, name, height,weight): # 身高单位是米,体重单位是千克
        self.name= name
        self.height= height
        self.weight= weight

```

完成项目任务 4,可以采用项目任务 2 中介绍的欧氏距离计算相似度。

### 三、作业练习与提示

#### ■ 题目描述

用 Python 语言的类结构来定义“复数”的抽象数据类型。

```
class ComplexA(object):
```

#### ■ 作业练习参考答案

```
class ComplexA:  
    def __init__(self, real, imag): # real 表示实部,imag 表示虚部  
        self.real= real  
        self.imag= imag  
    def getreal(self):          # 得到实部  
        return self.real  
    def getimag(self):          # 得到虚部  
        return self.imag  
    def printcomplex(self):  
        print(str(self.real)+"+"+ str(self.imag)+"i")      # 显示虚数  
        return  
a= ComplexA(5, 6)  
a.printcomplex()
```

### 四、核心概念介绍

#### 抽象数据类型:

抽象数据类型(Abstract data type, ADT),从实践的角度,可以看成是让用户可自定义复杂数据类型的一种机制。它让程序员指出该数据包含的成员(或元素、分量)及其操作。那些操作通常由函数(也称方法)定义,可以是任意复杂的。用户在使用这样的数据时,只需要知道其中有哪些数据成员,理解其对应的操作,无需了解具体实现细节。

前面提到的由 struct 定义的组合数据类型,也可以看成是用户自定义的,但它只可以说明包含的数据分量,而不能定义操作,其操作取决于各分量的类型。

### 五、教学参考资源

#### 抽象数据类型的表示与实现:

运用抽象数据类型描述数据结构,有助于在设计一个软件系统时,不必首先考虑其中包含的数据对象以及操作在不同处理器中的表示和实现细节,而是在构成软件系统的每

个相对独立的模块上定义一组数据和相应的操作,把这些数据的表示和操作细节留在模块内部解决,在更高的层次上进行软件的分析和设计,从而提高软件的整体性能和利用率。

抽象数据类型的概念与面向对象方法的思想是一致的。抽象数据类型独立于具体实现,将数据和操作封装在一起,使得用户程序只能通过抽象数据类型定义的某些操作来访问其中的数据,从而实现了信息隐藏。在 C++ 中,我们可以用类的声明表示抽象数据类型,用类的实现来实现抽象数据类型。因此,C++ 中实现的类相当于数据的存储结构及其在存储结构上实现的对数据的操作。

抽象数据类型和类的概念实际上反映了程序或软件设计的两层抽象:抽象数据类型相当于在概念层(或称为抽象层)上描述问题,而类相当于在实现层上描述问题。此外,C++ 中的类只是一个由用户定义的普通类型,可用它来定义变量(称为对象或类的实例)。因此,在 C++ 中,最终是通过操作对象来解决实际问题的,所以我们可将该层次看做是应用层。例如,main 程序就可看作是用户的应用程序。

——摘自《数据结构》,严蔚敏、李冬梅、吴伟民,人民邮电出版社

## 六、教学参考案例

### ■ 参考案例

#### 抽象类型

复旦大学附属中学 吴玉静  
(2课时)

##### 1. 学科核心素养

在学生信息统计任务中能够将有限制条件的、复杂生活环境中的关系进行抽象,并用抽象类型 student 类来表达数据的逻辑关系。(信息意识)

通过对比使用不同数据结构的程序代码,体会数据抽象的过程与意义,从数据结构的视角解释数据的组织形式,描述数据的逻辑结构及其操作,评判其中数据结构运用的合理性;了解 Python 中抽象数据类型的定义与使用方法。(信息意识、计算思维)

针对复数的四则运算和学生信息统计两个实际问题进行数据抽象,运用数据结构合理组织、存储数据,选择合适的算法编写程序,解决实际问题。(计算思维)

在自主学习和实践体验环节,在设定的情境中能够运用数字化学习资源完成相应的自主学习协同工作和程序编写。(数字化学习与创新)

##### 2. 《课程标准》要求

结合生活实际,理解数据结构的概念,认识数据结构在解决问题过程中的重要作用。

通过列举实例,认识到抽象数据类型对数据处理的重要性,理解抽象数据类型的概念。

### 3. 学业要求

学生能够运用生活中的实例描述数据的内涵与外延,能够将有限制条件的、复杂生活情境中的关系进行抽象,用数据结构表达数据的逻辑关系。

能够从数据结构的视角解释程序中数据的组织形式,描述数据的逻辑结构及其操作,评判其中数据结构运用的合理性。

能够针对限定条件的实际问题进行数据抽象,运用数据结构合理组织、存储数据,选择合适的算法编程实现、解决问题。

### 4. 教学内容分析

数据结构是计算机存储、组织数据的方式。通常情况下,精心选择的数据结构可以使程序运行得更快,并提升数据在计算机中存储的效率。数据结构对培养学生的信息意识与计算思维并形成学科核心素养具有非常重要的作用。抽象类型是教科书第二章第四节的内容。教科书从数据抽象的概念、抽象数据类型的定义与实现、数据封装的一个实例来介绍抽象数据类型。

### 5. 学情分析

学生已经完成了必修 1 的学习,然后自主选择数据与数据结构课程。学生具有较强的逻辑思维能力,并具备了一定的使用 Python 语言编写程序的基础。

通过数据与数据结构课程前面内容的学习,已经了解了基本数据类型、常用数据类型和组合数据类型,掌握了数组及其特征。

### 6. 教学目标

- 用 Python 编写一个“闹钟”程序,对比使用不同数据结构的算法设计,体会数据抽象的过程与意义。
- 观察使用抽象数据类型的程序代码,总结 Python 中抽象数据类型“类”的定义与使用方法。
- 运用 Python 中的类,合理组织、存储数据,选择合适的算法编程解决问题。

### 7. 教学重难点

- 教学重点:理解数据抽象的概念,运用正确的抽象数据类型解决实际问题。
- 教学难点:掌握 Python 中类的定义与使用方法。

### 8. 教学策略分析

本单元主要采用实践体验、小组协作、讨论交流的教学策略。从解决复数的四则运算和学生信息统计两个实际问题出发,体会数据抽象的过程与意义,提升学生的信息意识;锻炼学生数字化学习与创新的能力;培养学生的计算思维。

(1) 第 1 课时,先体验再总结,自然生成新的学习内容,借助学习单,提供知识链接,培养自主学习能力。

- 本课中通过对比使用不同数据结构的算法设计,体会数据抽象的过程与意义。
- 通过阅读教科书和资料,小组讨论协作完成 Python 中抽象数据类型“类”的定义与使用方法的总结。
- 完成由“复数的加法运算”拓展至多种运算的任务,巩固已经构建的新知识。

(2) 第2课时,实战演练学以致用。通过四个环节的讨论,引导学生进一步掌握Python中抽象数据类型“类”的概念、定义与使用的方法。

- 分析问题:明确所需创建类的属性和方法。
- 选择合适的数据结构:类的定义。
- 编写程序:进一步熟练类的使用。
- 运行调试:提升用计算机解决问题的能力。

9. 教学过程设计(见图2-8、图2-9、表2-14和表2-15)

## 第1课时:认识抽象类型

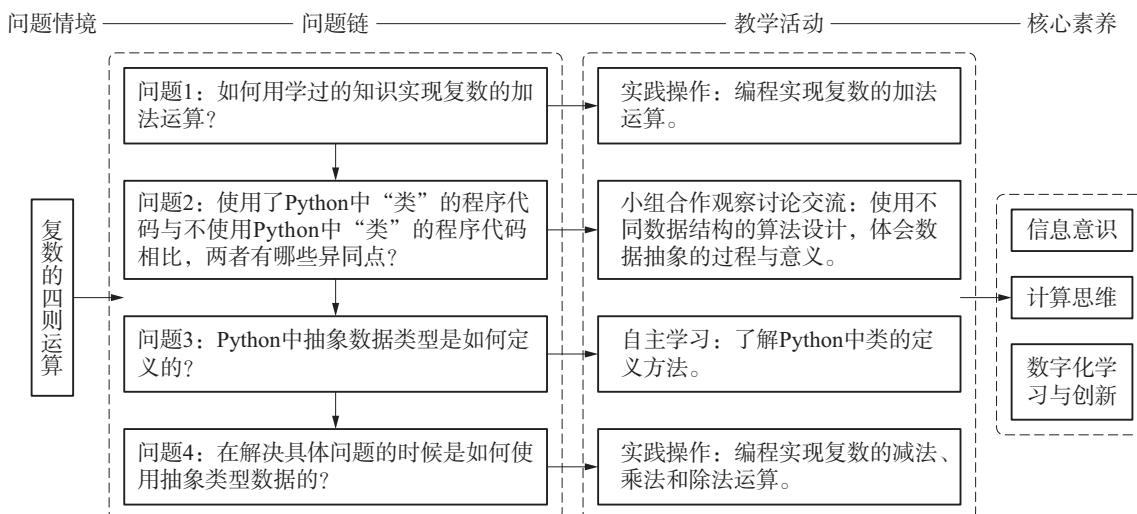
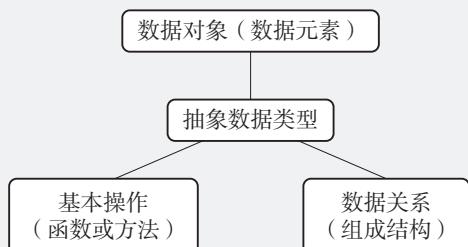


图2-8 第1课时教学过程

表2-14 第1课时教学过程设计表

教学环节	教师活动	学生活动	设计意图
情境导入	问题1: 如何用学过的知识实现复数的加法运算? 布置任务: 编程实现“复数的加法运算”	【实践体验】 完成任务一: 编写程序	熟悉Python语法,巩固已有知识
自主探究	问题2: 使用了Python中“类”的程序代码与不使用Python中“类”的程序代码相比,两者有哪些异同点? (1) 带着学生阅读使用了Python中“类”的程序代码。 (2) 组织引导学生小组讨论。 (3) 总结。 数据抽象鼓励程序设计者以数据对象作为程序设计的中心。抽象数据类型把数据定义为抽象的对象集合,只为它们定义可用的操作,而不用暴露具体的实现细节	【交流、讨论】 (1) 完成任务二: 阅读教师提供的程序代码。 (2) 小组交流讨论异同点	通过对比使用不同数据结构的算法设计,体会数据抽象的过程与意义,提升学生的信息意识

教学环节	教师活动	学生活动	设计意图
教师讲解	<p>问题3: Python中抽象数据类型是如何定义的? (学生自主学习完成相应练习后,开始讲解)</p> <p>(1) ADT:指一个数据模型以及定义在该模型上的一组操作。</p>  <p>(2) Python中抽象数据类型“类”的定义。 (3) Python中抽象数据类型“类”的使用</p>	<p><b>【自主学习】</b></p> <p>(1) 自主学习教科书第38页内容和阅读材料。 (2) 完成任务三:student类的定义</p>	通过阅读教科书和资料,小组讨论协作完成Python中抽象数据类型“类”的定义与使用方法的总结。锻炼数字化学习与创新的能力
巩固拓展	<p>问题4:在解决具体问题的时候是如何使用抽象类型数据的?</p> <p>布置任务:编程实现复数的减、乘、除运算</p>	<p><b>【实例应用】</b></p> <p>完善程序</p>	完成由加法运算拓展至多种运算的任务,在培养计算思维的同时建构新的知识
课堂总结	引导学生根据学习任务单和课堂笔记进行本课内容的总结	<p>(1) 类定义的格式。 (2) 对类中的属性的访问,采用圆点记法。 (3) 实例对象初始化时自动调用<code>__init__()</code>,第一个参数<code>self</code>表示正在初始化的对象自身。 (4) 在定义类时,应当避免属性名和方法名相同</p>	再次梳理数据抽象的概念、抽象数据类型的定义与使用;提升对“数据”和“算法”两个学科大概念的认识

## 第2课时：抽象数据类型实例

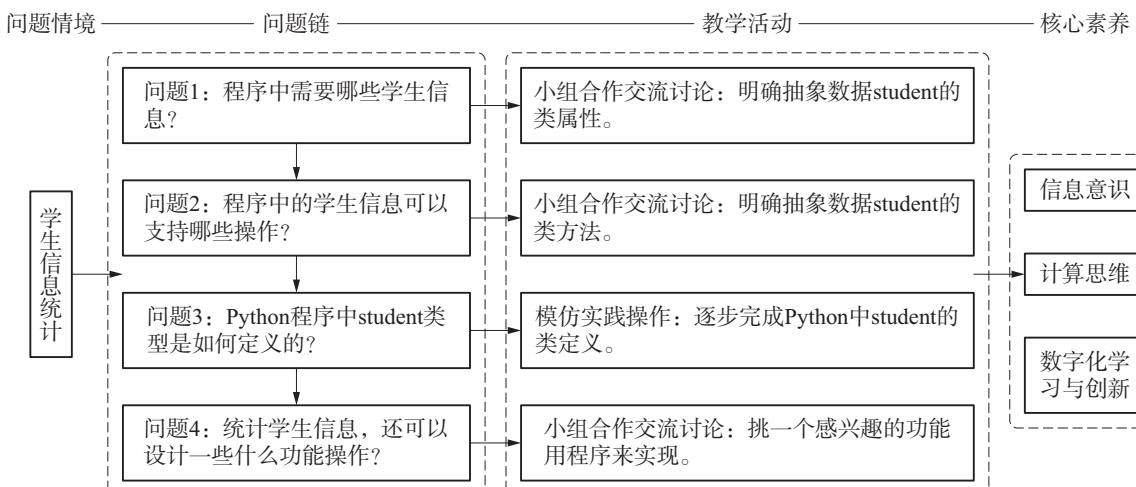


图 2-9 第2课时教学过程

表 2-15 第 2 课时教学过程设计表

教学环节	教师活动	学生活动	设计意图
提出问题	布置任务:编写一个统计学生信息的程序。 (1) 学生信息包括学号、姓名、性别、出生年月、年龄。 (2) 学号自动生成。 (3) 能够根据出生年月计算学生的年龄。 (4) 可以修改学生相应的信息。 (5) 能够打印学生的全部信息	<b>【交流、讨论】</b> 根据上述任务要求可知,需要一个抽象数据 student	一个学生熟知的问题,更容易引起学生的共鸣,培养其信息意识
分析问题	问题 1:程序中需要哪些学生信息? 组织引导学生小组讨论明确 student 类的属性。 问题 2:程序中的学生信息可以支持哪些操作? 组织引导学生小组讨论明确 student 类的方法	<b>【交流、讨论】</b> 完成任务一: (1) 明确抽象数据 student 类的属性有:学号、姓名、性别、出生年月、年龄。 (2) 明确抽象数据 student 类的方法有:学号、姓名、性别、出生年月、年龄	通过对实际问题进行数据抽象,并用抽象类型 student 类来表达数据的逻辑关系,从而实现对学生计算思维核心素养的培养
选择合适的数据结构	问题 3:Python 程序中的 student 类是如何定义的? (1) 从输入直接获得姓名、性别、出生年月。 (2) 学号生成。 (3) 年龄计算。 (4) 属性修改(以姓名为例,其他同理)。 (5) 打印全部属性	<b>【实例应用】</b> 逐步完成 Python 中的类定义	定义 student 类,运用 Python 中的类合理组织、存储数据,强化“数据”这一学科大概念
编写程序	(1) 巡视学生的程序编写情况。 (2) 随时答疑	<b>【实例应用】</b> 完善程序	选择合适的算法编程实现、解决实际问题,注重学生计算思维的养成
课堂拓展	问题 4:统计学生信息,还可以设计些什么功能操作?	<b>【交流讨论自主探究】</b> (1) 交流还可以实现的功能。 (2) 挑一个感兴趣的功能,用程序来实现	提升学生用计算机解决问题的能力,锻炼其数字化学习与创新能力

## 附 1: “认识抽象类型” 学习单

任务一:编程实现复数的加法运算。

任务二:阅读以下程序代码。

```
class mycomplex:  
    def __init__(self,realpart,imagpart):  
        self.realpart= realpart  
        self.imagpart= imagpart  
    def display(self):  
        print(str(self.realpart)+" "+ str(self.imagpart)+ "i")  
    def plus(self,another):  
        realpart= self.realpart+ another.realpart  
        imagpart= self.imagpart+ another.imagpart  
        return mycomplex(realpart,imagpart)  
c1= mycomplex(1,2)  
c2= mycomplex(3,4)  
c3= c1.plus(c2)  
c3.display()
```

任务三:阅读材料。

在任务二的程序代码中, class 是关键字, 表示类定义的开始, mycomplex 就是这个类的名字。每个类中我们都会定义\_\_init\_\_函数, 称为初始化方法, 用于构造一个该类的新对象, 我们以类名作为函数创建实例化对象, 如 c1 = mycomplex(1,2), 在调用的时候, 应当给出除 self 外的其他参数。realpart 和 imagpart 都是复数类的属性。调用其他方法时, 仍用 self 表示本实例, 应当给出其他参数, 比如 c3 = c1. plus(c2), 此时 c1 的值作为 plus 方法的第一个参数, c2 的值为第二个参数。

同理, display 函数中只有一个参数 self, 所以调用的时候写作 c3. display()。执行以上语句后我们可以得到如下输出: 4 + 6i。

依据教科书和阅读材料, 用 Python 类定义的方法模仿如下示例设计一个数据结构 (ADT 表示这是一个抽象数据类型)。

```
ADT student:  
    id      # 学号  
    name    # 姓名  
    gender   # 性别  
    chinese_sc  # 语文成绩  
    maths_sc    # 数学成绩  
    english_sc   # 英语成绩  
    sum(student)    # 三门总分
```

完善程序：实现复数的减、乘、除运算。

```
class mycomplex:  
    def __init__(self, realpart, imagpart):  
        self.realpart = realpart  
        self.imagpart = imagpart  
    def display(self):  
        print(str(self.realpart) + " + " + str(self.imagpart) + "i")  
    def plus(self, another):  
        realpart = self.realpart + another.realpart  
        imagpart = self.imagpart + another.imagpart  
        return mycomplex(realpart, imagpart)  
    _____ # 实现减法运算函数  
  
    _____  
    _____ # 实现乘法运算函数  
  
    _____  
    _____ # 实现除法运算函数  
  
c1 = mycomplex(1, 2)  
c2 = mycomplex(3, 4)  
c3 = c1.plus(c2)  
c3.display()
```

## 附 2：“抽象数据类型实例” 学习单

- 任务一：根据目标任务的需求，不使用抽象数据类型也可以完成这个程序，但是使用抽象数据类型，可以更有效地组织管理相关数据。

抽象数据类型 student 的类属性：                        ；

抽象数据类型 student 的类方法：                        。

- 任务二：使用 Python 编程，定义学生类 student，完成下划线处代码，实现修改学生姓名、计算学生年龄等功能。

```
import datetime  
class Student: # 定义学生类 student  
    _id = 0
```

```

def __init__(self, name, gender, brithday):
    self.name= name
    Student._id += 1
    self.id= Student._id
    self.gender= gender
    self.brithday= brithday

def age(self):          # 计算年龄
    today = _____ # 获取当天的日期
    return (
        today.year
        - self.brithday.year
        - ((today.month, today.day) < (self.brithday.month, self.
brithday.day))
    )

def display(self):      # 显示学生的所有属性
    print(
        "id: {}, name: {}, gender: {}, brithday: {} (age: {})".
format(
            self.id, self.name, self.gender, self.brithday,
self.age()
        )
    )
    def rename(self,newname):   # 修改姓名属性
        _____
Student1= Student("张三", "男", datetime.date(2001, 1, 1))
Student1.display()
Student1.rename("李四")
Student1.display()

```

# 基础数据结构

## 一、本章学科核心素养的渗透

本章内容包括线性表、栈、队列三节。整章通过案例分析，引导学生理解数组、链表等基本数据结构的概念，并能编程实现相关操作；通过问题解决，引导学生理解包括栈、队列在内的线性表的概念和基本操作，并能编程实现。

本章是选择性必修1“数据与数据结构”模块的核心内容。《课程标准》中相关内容要求包括：

1. 4 通过案例分析，理解数组、链表等基本数据结构的概念，并能编程实现其相关操作。比较数组、链表的区别，明确上述两种数据结构在存储不同类型数据中的应用。

1. 5 通过问题解决，理解包括字符串、队列、栈在内的线性表的概念和基本操作，并编程实现。

本章以“送餐机器人的行走路线”为主题，分析关键步骤，设计多种方案，选择合适的数据结构进行组织和存储并编程实现，归纳总结机器人送餐方案，探究在生活中更广泛的应用等活动，落实《课程标准》要求。

本章的学习重点围绕计算思维和数字化学习与创新能力的培养，以项目整合课堂教学，将线性表、栈、队列整体考虑，同时要注意对这三个概念的辨析。在内容的选取上注重学科本体知识和高中生实际情况，如对于一些比较抽象的概念，通过实例和原理展开描述，如从学生较熟悉的“进制转换”程序中的数据关系分析引出栈的概念。对于栈的基本操作，从“车辆调度”问题体会栈对于数据组织的意义。加强概念的辨析，如对栈概念要点（一端操作，后进先出）和队列概念要点（两端操作）的辨析，同时以队列假溢出处理体现出循环的意义。

本章设置了大量的体验思考和探究活动，强调在思考和实践的基础上对数据抽象、数据结构的思想和方法的初步认识。本章还提供了丰富的编程实践案例，在案例中强调聚焦真实问题，尝试用数据结构视角审视问题。在解决问题过程中，鼓励学生运用学过的理

论知识大胆尝试,勇于创新,发现问题解决的新思路和新办法。通过解释程序中数据的组织形式,描述数据的逻辑结构及其操作,评判程序中数据结构运用的合理性,体现出对于信息意识的培养。在解决实际问题的过程中引导学生经历抽象数据、建立数据模型、选择数据结构、算法实现、上机调试、问题解决的全过程,培养计算思维。

## 二、本章知识结构

本章遵循《课程标准》,依据学分和课时规定,紧扣学科概念体系,分为三节内容。

第一节线性表,内容包括线性表的概念、基本操作、实现与应用、性能分析。

第二节栈,内容包括栈的概念、基本操作、实现与应用。

第三节队列,内容包括队列的概念、基本操作、实现与应用。

教学过程中,可以将教科书中的体验思考、探究活动合理分配到教学环节中。对于项目实践、作业练习,建议先进行方案设计、图解过程,之后进行编程实践。学生可在解决问题的过程中,自行优化或设计某种数据结构,体会数据结构和算法的相互作用。

## 三、本章项目活动设计思路

现实生活中经常会出现“机器人”助手,因此本章以“送餐机器人的行走路线”为主题开展项目活动,理解线性表、栈和队列这三种基础数据结构之间的关系,掌握这三种数据结构的基本操作,在程序设计过程中运用这三种结构解决实际问题。

在项目实现过程中,通过不同的实现方案将线性表、栈、队列作为整体一起来学习。在线性表的学习过程中,将“地图”抽象并构建成可处理的数据模型,同时建立“机器人”各方向移动的数据模型。在栈的学习过程中,实现“机器人”行走的任意一条路线方案。在队列的学习过程中实现“机器人”行走的最短路线方案。通过项目的归纳总结,将实现路径规划的方法迁移到其他应用场景中。

## 四、本章课时安排建议

本章教学建议用 10 课时完成,具体参见表 3-1。

表 3-1 课时安排计划表

节名	建议课时
第一节 线性表	4
第二节 栈	3
第三节 队列	3

## 第一节 线 性 表

### 一、教学目标与重点

#### 教学目标：

- 在案例分析、方案设计过程中,理解线性表的概念和结构,掌握线性表在两类存储结构上的基本操作,认同线性表在问题解决中的支持作用。
- 能运用线性表解决实际问题,在使用数组、链表对数据进行存储和处理的过程中感受不同数据结构的逻辑内涵,体验用计算机思想方法解决问题的过程。
- 在项目活动实践中通过问题的解决,归纳适用线性表两类存储结构实现方式的场景及处理数据的特点,体会迭代思想的逻辑内涵。

#### 教学重点：

掌握线性表的两类存储结构及其基本操作。

### 二、教学实施与评价

#### 1. 教学活动建议

教师在讲解线性表概念的同时,需要结合基本操作及相关练习,还应了解时间复杂度的重要性。数据结构为算法服务,高效、简洁的数据结构设计是好算法的基础。

在进行教学设计时,不妨先将本课的知识点一一列出,找出它们的联系。应从整体考虑分课时的活动设计,遵循先易后难,先了解、再模仿、后操作的学习规律。

教学中碰到的第一个重要知识点是线性表的概念及其最具标志性的基本操作——插入和删除。在讲解概念时一定要强调其中的逻辑关系,从而引出顺序表和链表。通过教科书中的实例分析,引发学生寻找生活中的实例描述线性表的内涵,尝试在学习了线性表的实现模型后,将有限制条件的、生活情境中的关系进行抽象。

为了理解顺序表和链表内涵,学习、领会基于顺序表和链表的一些基本操作成为掌握线性表本质的重要途径。教科书基于实际问题即数据流检测,将线性表的基本操作贯穿其中,教师可以将其分解为阶段性任务逐一击破。如,在教学过程中,将教科书中出现的完整流程图拆分成若干功能模块,每次揣摩一个模块的功能、实现方法并理解其中涉及的基本操作,组织学生分组用不同的方法进行探索并解决问题,从而完成搭建知识脚手架的工作。

对于时间复杂度的学习,是帮助学生建立评价基础的关键。因为只有在恰当的数据结构支持下,计算机算法才能够实现为高效运行的程序,而用时间复杂度评估数据结构是

否恰当是最直接,也是最有效的方法。带领学生对不同数据结构计算其时间复杂度可以帮助学生掌握这种方法,以便今后能灵活有效地进行评估。

Python 语言封装了一些功能,在选用时应建立在学生已经理解并掌握基本操作的基础上。

教科书主要从线性表的概念,在顺序存储和链接存储上的基本操作,以及实现数据流检测三部分进行了介绍。

线性表有以下两种基本的实现模型:

(1) 将表中数据元素顺序地存放在一大块连续的存储区里,这样实现的表称为顺序表(sequential list)。在这种实现中,数据元素间的顺序关系由它们的存储顺序自然表示。

(2) 将表中数据元素存放在通过链接构造起来的一系列存储块里,这样实现的表称为链接表,简称链表(linked list)。

教科书中详细介绍了顺序表以及单链表的查找、插入和删除等操作的实现方法。在实际操作中,还需要了解循环单链表的实现。

循环单链表与单链表的区别仅仅在于:循环单链表中最后一个节点链接域的值不是None,而是指向表头节点,整个链表形成一个环。由此,从表中任一节点出发均可找到表中的其他节点。

循环单链表上的操作实现和单链表基本一致,仅需在算法中出现None处改为头指针head,如图3-1(a)所示。仔细考虑,我们会发现,让头指针指向表尾节点[如图3-1(b)所示]会让部分操作更简单,譬如表头或表尾的插入以及表头的删除。当然,由于循环链表里的节点连成一个圈,哪个节点是表头或表尾,主要是概念问题,从表的内部形态上无法区分。

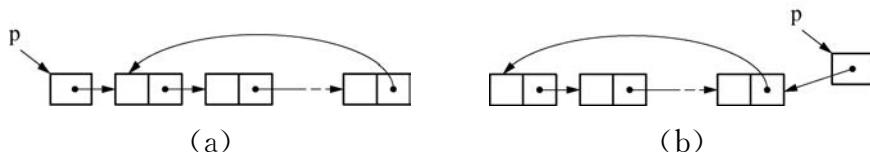


图 3-1 循环单链表

下面考虑实现一个循环单链表类 Py\_CircularLinkedList,如图3-1(b)所示,此类派生自Py\_LinkedList。这种表对象只需要一个数据域rear,在逻辑上始终引用这表的表尾节点。前端加入节点,就是在尾节点和首节点之间加入新的首节点,尾节点引用不变。通过尾节点引用很容易实现这个操作。另一方面,尾端加入节点也是在原尾节点之后(在首节点之间)插入新节点,只是插入后要把它作为新的尾节点,因此需要更新尾节点引用。这两个操作都要考虑空表插入的特殊情况。对于输出表元素的操作,关键在于循环结束的控制。

```
from Py_LinkedList import Py_LinkedList
class Py_ListNode :
    def __init__(self, data, next_=None):
        self.data = data      # 数据域
        self.next = next_     # 链接域
```

```
class Py_CircularLinkedList(Py_LinkedList): # 循环单链表
    def __init__(self):      # 初始化循环单链表
        Py_LinkedList.__init__(self)
        self.rear= None

    def isEmpty(self):       # 判断循环单链表是否为空
        return self.rear is None

    def prepend(self, data):  # 前端插入
        p= Py_ListNode(data, None) # 构造一个将要插入的节点并赋给 p
        if self.isEmpty():      # 判断循环单链表是否为空
            p.next= p  # 建立一个节点的环
            self.rear= p
        else:
            p.next= self.rear.next
            self.rear.next= p

    def append(self, data):   # 尾端插入
        self.prepend(data)
        self.rear= self.rear.next

    def pop(self):           # 前端弹出
        if self.isEmpty(): # 判断是否为空表
            raise ValueError
        p= self.rear.next  # 把头节点赋给 p
        if self.rear is p: # 判断是不是只有一个节点的循环单链表
            self.rear= None # 置空
        else:
            self.rear.next= p.next
        return p.data  # 返回弹出的数据

    def printall(self):      # 输出表元素
        p= self.rear.next # 把头节点赋给 p
        while True:
            print(p.data, end=' ') # 输出数据
            if p is self.rear:    # 判断是不是尾节点
                break             # 如果是,说明数据输出完毕
```

```

    p= p.next          # 寻找下一个结点
    print()

```

教科书中分别使用顺序表和链表来实现数据流的检测,让大家对相关数据结构的精髓有更加直观的理解,并使用 Python 提供的列表来直接实现数据流的检测。对比发现,Python 列表似乎没有长度上界的限制,而且并没有提及此处使用的线性表到底是顺序表还是链表!这就是抽象数据类型的魅力——程序员只需要知道操作接口,不用管具体的实现方式!

为了对程序执行时间有一个客观的分析和判断,通常将程序的执行时间看成问题规模(一般用整数  $n$  表示)的函数。问题规模在具体的问题中含义不同,它可以是线性表的长度,图的顶点数,矩阵的阶,等等。

一般情况下,程序中基本操作重复执行的次数是问题规模  $n$  的某个函数  $f(n)$ ,程序执行的时间量度记作  $T(n) = O(f(n))$ ,它表示随问题规模  $n$  的增大,程序执行时间的增长率和  $f(n)$  的增长率相同,称作算法的渐近时间复杂度,简称时间复杂度。「 $O$ 」的形式定义为:若  $f(n)$  是正整数  $n$  的一个函数,则  $x_n = O(f(n))$  表示存在一个正的常数  $M$ ,使得当  $n \geq n_0$  时都满足  $|x_n| \leq M|f(n)|$

现有如下四段程序:

```

(1) x= x+ 1

(2) for i in range(n):
    x= x+ 1

(3) for i in range(n):
    for j in range(n):
        x= x+ 1

(4) for i in range(n):
    for j in range(i,n):
        x= x+ 1

```

在这四段程序中,语句  $x = x + 1$  的执行次数分别为  $1, n, n^2$  和  $\frac{n(n+1)}{2}$ ,则这四段程序的时间复杂度分别是  $O(1), O(n), O(n^2)$  和  $O(n^2)$ 。

下面详细地分析一下顺序表和单链表在进行查找、插入和删除操作时的时间复杂度。

对于查找操作,按位序查找[对应的函数为 `getItem(self, i)`]时,顺序表可以随机访问,访问任一元素的时间相同,即任意元素的地址都可以通过一个统一的公式(即首元素地址加上适当偏移量)得到,于是可称顺序表按位序查找的时间复杂度为  $O(1)$ ,即“常数时间”。而链表不支持随机访问,在单链表中无法直接获得第  $i + 1$  个数据元素的值,只有从表头指针出发,通过链接域往下搜索,直到找到第  $i + 1$  个节点为止。这个算法的基本操作是指针后移及计数,执行次数与所给的  $i$  值有关。当  $i = 0$  时,执行次数为 0;当  $i = 1$  时,执行次数为 1;当  $i = n - 1$  时,执行次数为  $n - 1$ ,从而平均执行次数约为  $\frac{n}{2}$ ,所以可称单链表按位序查找的时间复杂度为  $O(n)$ 。

按值查找[对应的函数为 `getLoc(self, x)`]时,如顺序表无序,顺序表和单链表都需要从前向后依次比较各数据元素的值是否等于  $x$ ,直到找到值等于  $x$  的数据元素或表结束为止。这个算法的基本操作是比较元素的值是否等于  $x$ ,执行次数与所给的  $x$  值有关。若第一个元素值为  $x$ ,则比较次数为 1,若最后一个元素值为  $x$  或未找到,则比较次数为  $n$ ,从而平均比较次数大约为  $\frac{n}{2}$ ,所以顺序表和链表的时间复杂度都为  $O(n)$ 。但是当顺序表有序时,可以采用对分查找(见第五章),将时间复杂度降为  $O(\log_2 n)$ 。

对顺序表及单链表进行插入操作[对应的函数为 `insert(self, i, x)`]时,顺序表的基本操作是移动数据元素,当  $i = n$  时,移动次数为 0;当  $i = 0$  时,移动次数为  $n$ ,从而平均移动次数为  $\frac{n}{2}$ ,所以顺序表插入操作的时间复杂度为  $O(n)$ 。而单链表执行插入操作的时候虽然不需要移动数据元素,但为了找到表中第  $i$  个节点,仍需要执行时间复杂度为  $O(n)$  的查找操作。当然如果单链表在得到某位置的指针后,插入操作的时间复杂度为  $O(1)$ 。

同样,对顺序表及单链表进行删除操作[对应的函数为 `delete(self, i)`],顺序表平均需要移动表长一半的数据元素,时间复杂度为  $O(n)$ ,单链表在得到某位置的指针后,删除时间仅为  $O(1)$ 。

教科书从数据存储方式、时间性能、空间性能几方面对线性表的两种实现方式进行对比,总结不同的使用场景。如果线性表需要频繁查找且很少进行插入和删除操作,宜采用顺序表。如果需要频繁插入和删除,宜采用单链表结构。当线性表中的元素个数变化较大或者数量难以预判时,采用单链表结构可能更有利。而如果事先知道线性表的大概长度,使用顺序表会更有效率。

## 2. 教学过程安排(见表 3-2)

表 3-2 教学过程设计表

课时	教学环节	教师活动	学生活动
第 1 课时 (线性表 和顺序表 的实现)	1. 导入	列举 2003 年到 2018 年参加全国高考的人数	思考生活中与线性表有关的现象
	2. 线性表的概念	引导学生认识什么是线性表。 (1) 线性表的特点有哪些? (2) 线性表如何表示? (3) 线性表的基本操作有哪些?	阅读教科书,回答问题
	3. 体验思考	引导思考:完成本节第 1 个体验思考	通过重新排队活动,体验线性表特点
	4. “线性表基本操作”活动	布置活动:对某个数据序列进行插入操作	图解线性表的操作
	5. “顺序表”的实现	讲解线性表的顺序存储实现	倾听、思考
	6. “顺序表”的基本操作	布置活动:实现顺序表的查找、插入和删除等操作	编程实现
	7. 总结	线性表结构特点、实现和基本操作	

续表

课时	教学环节	教师活动	学生活动
第 2 课时 (链表的实现)	1. 导入	回顾上节课内容:线性表的第二种存储实现,引出链表	倾听、思考
	2. 链表概念及各种形式	引导学生认识什么是链表。 (1) 链表的特点是什么? (2) 链表的形式有哪些?	自我阅读、思考分析
	3. 单链表的实现	讲解单链表的各种实现和常见操作	倾听、思考
	4. 单链表的各种操作	布置活动:单链表的建立、查数据元素、删除元素、取数据元素	思考分析,编程实现
	5. 体验思考	布置体验思考:完成本节第 2 个体验思考	思考并编程实现
	6. 总结	链表的特点和实现	
第 3 课时 (线性表的应用)	1. 导入	回顾第二章,引出数据流检测问题	倾听、思考
	2. 活动:数据流检测的实现	布置活动:用顺序表和链表两种方式分别实现数据流检测	编程实现、对比分析
	3. 活动:用数组实现顺序表	布置任务:用数组实现顺序表,应用于数据流检测问题	思考分析、编程实现
	4. 体验思考	布置活动:完成本节第 3 个体验思考	倾听、思考
	5. 活动:用数组实现链表	布置任务:用数组实现链表,应用于数据流检测问题	思考分析、编程实现
	6. 探究活动	布置作业:完成本节探究活动	编程实现
	7. 总结	用数组实现顺序表和链表的方法和各自特点	
第 4 课时 (线性表性能分析与完成项目任务 1)	1. 导入	回顾第二章第二节数组部分的“列表和数组访问的性能比较”程序,引出时间性能问题	倾听、思考
	2. 性能分析	围绕数据流检测问题分析时间性能、空间性能	思考,分析
	3. 活动:项目任务 1 分析	布置任务:分析项目中的关键步骤。寻找特征,建立数据模型	思考,完成
	4. 活动:项目任务 1 实现	布置活动: (1) 地图的抽象及可处理的数据模型的构建。 (2) 抽象建立向各方向移动的数据模型	编程实现
	5. 作业练习 1	安排项目实践:本节项目任务	完成方案、编程实现
	6. 作业练习 2	布置作业:本节作业练习 1	编程实现
	7. 总结	性能分析、抽象特征、建立数据模型	

本节的评价可采用过程性评价和终结性评价结合的方式,在完成各项目任务的过程中观察学生对于线性表的理解和应用:能够针对实际问题进行数据抽象;能运用线性表等数据结构合理组织、存储数据;能选择合适的算法编程实现,解决问题。教师也可设计实践评价量规,重点考查学生在本节项目活动中运用线性表解决问题的能力,如表 3-3 所示。

表 3-3 实践评价量规参考示例

	评价内容	是否达成	反思与建议
项目主题	(1) 主题明确,能列举主题特征。 (2) 能描述生活中各种线性表应用的情境		
项目分析	(1) 明确数据分析的目标,分解任务。 (2) 分析从起点到终点的路径设计方案		
方案设计	(1) 能描述餐桌和通道、地图的关系。 (2) 将地图抽象并构建成可处理的数据模型,能找寻特征,抽象出相关数据,建立合适的数据模型。 (3) 抽象建立各方向移动的数据模型		
探究实践	(1) 根据方案,选择合适的数据结构,实现数据组织和存储。 (2) 能用 Python 语言编程实现系统		
成果交流	(1) 分享各自的作品,讨论数据处理方式和执行流程是否合理、正确。 (2) 讨论各自方案,并修改完善		

### 3. 思考探究提示

(1) 教科书第 45 页体验思考示例如表 3-4 和图 3-2 所示:

表 3-4 数据表

	数据	下一个
0	4	6
1	5	7
2	6	1
3	8	-1
4	1	2
5	7	4
6	2	8
7	9	3
8	3	9
9	10	5

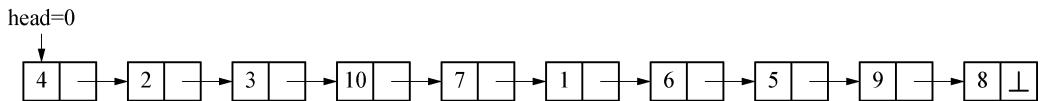


图 3-2 线性表

(2) 教科书第 52 页体验思考提示如下：

第一个定义的是顺序表，第二个定义的是链表，其余代码都一样，运行结果也都一样，这就是抽象数据类型的魅力——程序员只需要知道操作接口，不用管具体的实现方式！

(3) 教科书第 56 页体验思考程序如下：

1	import numpy as np
2	datastream= [2, 3, 6, 1, 3, 8, 1, 2, 4, 5]
3	print ('data stream:', datastream)
4	linear_list= np.zeros((100))
5	length= 0
6	while datastream:
7	x= datastream.pop(0); no= True
8	for i in range(length):
9	if x== linear_list[i]:
10	print (x,'Found it ! ')
11	no= False; break
12	if x> linear_list[i]:
13	for j in range(length,i,- 1):
14	linear_list[j]= linear_list[j- 1]
15	linear_list[i]= x; length= length + 1
16	no= False; break
17	if no:
18	linear_list[length]= x; length= length + 1
19	print ('final list:', linear_list[range(length)])
20	exit()

我们在第 17 和 18 行之间插入一个 print("append... ", x)

我们在第 12 和 13 行之间插入一个 print("insert... ", x)

运行程序，结果为：

data stream: [2, 3, 6, 1, 3, 8, 1, 2, 4, 5]

```
append... 2
insert... 3
insert... 6
append... 1
3 Found it !
insert... 8
1 Found it !
2 Found it !
insert... 4
insert... 5
final list: [8. 6. 5. 4. 3. 2. 1.]
```

可以知道：第 12 和 13 行之间插入一个 print 语句，将被执行 5 次。

(4) 教科书第 58 页探究活动提示：在程序中增加打印 memory 的内容。

```
import numpy as np
memory= np.zeros((10,2),dtype= np.int32)
datastream=[4,3,5,1,4,2,3,1]
print ('data stream:',datastream);print(memory)
x= datastream.pop(0)
memory[0,0]= x; memory[0,1]= - 1
linked_list_head= 0; new= 1;print(linked_list_head,memory)
while datastream:
    x= datastream.pop(0); no= True
    search_ptr= linked_list_head; pre_node= - 1
    while search_ptr != - 1:
        if x== memory[search_ptr,0]:
            print (x,'Found it ! ')
            no= False; break
        if x> memory[search_ptr,0]:
            memory[new,0]= x
            memory[new,1]= search_ptr
            if search_ptr== linked_list_head:
                linked_list_head= new
            else:
                memory[pre_node,1]= new
            new= new + 1;print(linked_list_head,memory)
            no= False; break
        else:
```

```

pre_node= search_ptr
search_ptr= memory[search_ptr, 1]

if no:
    memory[new, 0]= x; memory[new, 1]= - 1
    memory[pre_node, 1]= new
    new= new + 1; print(linked_list_head, memory)

while linked_list_head != - 1:
    print (memory[linked_list_head, 0])
    linked_list_head= memory[linked_list_head, 1]
exit()

```

程序运行过程如表 3-5 所示。

表 3-5 程序运行过程

memory 初始	第 1 次循环	第 2 次循环	第 3 次循环	第 4 次循环	第 5 次循环
0[[0 0]	0[[ 4 -1]	0[[ 4 1]	2[[ 4 1]	2[[ 4 1]	2[[ 4 1]
[0 0]	[ 0 0]	[ 3 -1]	[ 3 -1]	[ 3 3]	[ 3 4]
[0 0]	[ 0 0]	[ 0 0]	[ 5 0]	[ 5 0]	[ 5 0]
[0 0]	[ 0 0]	[ 0 0]	[ 0 0]	[ 1 -1]	[ 1 -1]
[0 0]	[ 0 0]	[ 0 0]	[ 0 0]	[ 0 0]	[ 2 3]
[0 0]	[ 0 0]	[ 0 0]	[ 0 0]	[ 0 0]	[ 0 0]
[0 0]	[ 0 0]	[ 0 0]	[ 0 0]	[ 0 0]	[ 0 0]
[0 0]	[ 0 0]	[ 0 0]	[ 0 0]	[ 0 0]	[ 0 0]
[0 0]	[ 0 0]	[ 0 0]	[ 0 0]	[ 0 0]	[ 0 0]
[0 0]]	[ 0 0]]	[ 0 0]]	[ 0 0]]	[ 0 0]]	[ 0 0]]

#### 4. 项目实施提示

餐厅中有餐桌和通道,分析这些特征,抽象餐厅的布局并建立成可处理的地图数据模型。在机器人沿着通道前行过程中,建立各方向移动的数据模型。

分析项目实践任务:进行地图抽象并构建出可处理的数据模型、抽象建立机器人向各方向移动的数据模型、合适数据结构的选择、队列中存放数据的设计、算法设计、过程模拟、编程实现、迁移更多的解决方案和应用场景。本节项目任务完成具体如下。

(1) 进行地图抽象并构建出可处理的数据模型:

地图通常由一些障碍物和通路组成,现有地图如图 3-3 所示。从每个点可以移动到相邻的东、南、西、北四个点。

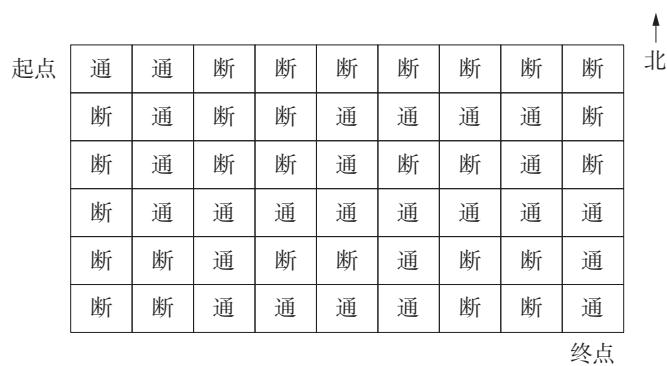


图 3-3 现有地图

在计算机内部,可以用一个二维数组表示这个简单的地图,地图抽象后(如图 3-4 所示),用 0 表示通路,用 1 表示阻断,设计一个算法,找到从起点到终点的一条最短路径。

0	0	1	1	1	1	1	1	1
1	0	1	1	0	0	0	0	1
1	0	1	1	0	1	1	0	1
1	0	0	0	0	0	0	0	0
1	1	0	1	1	0	1	1	0
1	1	0	0	0	0	1	1	0

图 3-4 抽象地图

在现实中向东、南、西、北前进可以分别用在二维数组中向右、下、左、上移动来表示,因此机器人走地图的过程可用二维数组坐标( $x, y$ )位置的改变来表示。首先对 6 行 9 列的地图用 6 行 9 列的二维数组进行编码。为了使机器人在开始的时候可以向四个方向前进,所以在上述二维数组的四周加上了一圈数值为 1 的元素,形成一个 8 行 11 列的二维数组。若规定从第 0 行和第 0 列开始标记,则机器人实际的入口坐标为(1, 1),对应第 1 行第 1 列的位置,出口坐标为(6, 9)。最终地图如图 3-5 所示,在图 3-5 中用 1 表示阻断,那么内部通路的坐标为(1, 1), (1, 2), (2, 2), (2, 5), (2, 6), (2, 7), (2, 8), (3, 2), (3, 5), (3, 8), (4, 2), (4, 3), (4, 4), (4, 5), (4, 6), (4, 7), (4, 8), (4, 9), (5, 3), (5, 6), (5, 9), (6, 3), (6, 4), (6, 5), (6, 6), (6, 9)。

1	1	1	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1	1
1	1	0	1	1	0	0	0	0	1	1
1	1	0	1	1	0	1	1	0	1	1
1	1	0	0	0	0	0	0	0	0	1
1	1	1	0	1	1	0	1	1	0	1
1	1	1	0	0	0	0	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1

图 3-5 最终地图

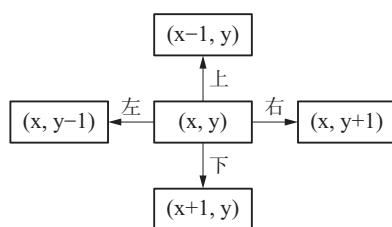


图 3-6 数据模型

(2) 抽象建立机器人向各方向移动的数据模型。

站在任意一个坐标( $x, y$ )位置,按照向右[原坐标 +  $(0, 1)$ ],下[原坐标 +  $(1, 0)$ ],左[原坐标 +  $(0, -1)$ ],上[原坐标 +  $(-1, 0)$ ]的顺序将四个方向全部试探一遍(如图 3-6 所示)。注意这里的坐标( $x, y$ )指的是第  $x$  行第  $y$  列,并不是数学中二维坐标系  $x$  轴和  $y$  轴的坐标。

### 三、作业练习与提示

#### ■ 题目描述

1. 假设有一个含 8 个元素的线性表: 2, 3, 6, 8, 1, 7, 4, 5。设计一个用二维数组实现的链表,并画出其展开形式。

2. 线性表的核心操作是“插入”与“删除”。在本节的数据流检测例子中,展现了插入操作的意义和实现的方式,但没有涉及线性表的删除操作。下面“报数选拔问题”的求解则是体验在线性表中做删除操作的经典例子。

报数选拔问题: 设想  $n$  个同学坐成一圈,按照一个事先确定的报数最大值  $m$ ,从任意一个指定的同学开始,沿逆时针方向(可循环)报数:1, 2, ...,  $m$ ,第  $m$  个同学出列;然后从他右边的那位同学开始报数:1, 2, ...,  $m$ ,也是第  $m$  个同学出列;如此下去,直到剩下最后一个同学。

以图 3-7 中的设定为例,有 7 名同学(A, B, C, D, E, F, G)围坐一圈,假设  $m = 3$ ,从 C 开始。上述报数选拔的出列顺序就是: A, E, B, D, F, C,最后剩下 G。此过程可以用一个循环链表来实现,其中的“出列”行为对应于在链表中做“删除”操作。试完成此程序设计。

#### ■ 作业练习参考答案

1. 参考教科书第 45 页的体验思考。
2. 参考程序如下:

# 用循环链表解决报数选拔问题

```
class Node:  
    def __init__(self, data):  
        self.data= data  
        self.next= None  
  
class LoopLinkedList:  
    def __init__(self):  
        self.root= None  
  
    # 在循环链表尾部添加节点,尾部即头节点上一个  
    def addNode(self, data):  
        newNode= Node(data)  
        if self.root== None:  
            self.root= newNode
```

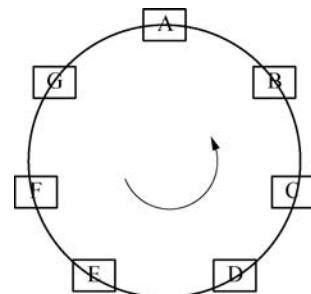


图 3-7 报数选拔问题示意图

```

        else:
            cursor= self.root
            while cursor.next! = self.root:
                cursor= cursor.next
                cursor.next= newNode
            newNode.next= self.root      # 链接成环

# 链表长度
def size(self):
    if self.root== None:
        return 0
    cursor= self.root
    i= 1
    while cursor.next ! = self.root:
        i+= 1
        cursor= cursor.next
    return i

# 报数选拔问题仿真函数
def circle(num, nameList):
    linkedList= LoopLinkedList()
    for i in range(len(nameList)):
        linkedList.addNode(nameList[i])
    i= 1
    pre= linkedList.root
    cursor= linkedList.root
    while linkedList.size()! = 1:
        if i!= num:
            pre= cursor
            cursor= cursor.next
            i+= 1
        else :
            print(cursor.data)
            pre.next= cursor.next      # 删除当前节点需要用上一个节点链接下一个节点
            cursor= pre.next
            linkedList.root= cursor  # 重新选择头节点是为了计算链表长度

```

```
i=1  
return cursor.data  
  
# 主函数  
if __name__ == '__main__':  
    nameList = ["C", "B", "A", "G", "F", "E", "D"]  
    print(circle(3, nameList))
```

运行结果：

A  
E  
B  
D  
F  
C  
G

## 四、核心概念介绍

**顺序存储:**在内存中,一个元素紧接着另一个元素存储,占用的是完整连续的空间。因此,顺序存储在申请内存时是申请一个定长的空间。优点是读写速度快,缺点是不易扩容。

**链接存储:**通常每一个元素存储的位置相对随机,每一个元素附带一个指针来锁定下一个元素的位置,占用的是相对零散的空间。优点是容易扩容,缺点是读写速度慢。

**时间复杂度:**定性描述一个算法的运行时间。因为不同计算机运行速度不一样,所以需要一个统一的指标来描述一个算法的快慢。时间复杂度估计的是一个算法的操作单元数量。时间复杂度主要讨论影响算法快慢的主要因素。

**空间复杂度:**定性描述一个算法占用的存储空间。空间复杂度考虑存储算法本身占用的空间、输入输出数据占用的空间和运行过程中占用的空间。空间复杂度主要讨论影响算法占用空间的主要因素。

**位序与值:**值是线性表中某个元素存储的数据,位序是一个值存储在线性表中的位置。每个位序都对应一个值,每一个存储在不同位置的元素也都对应一个位序。应好好体会在顺序存储和链接存储中位序与值的关系。

**插入与删除:**插入与删除是线性表最重要的操作。插入用于解决人们增加数据的需要,删除用于解决人们减少数据的需要。

## 五、教学参考资源

学校假期招募青少年志愿者,某班 45 名同学都踊跃报名,但名额只有一个,那么该

选哪位同学担任志愿者呢？同学们各抒己见，提出多种筛选方案，其中有如下两个方案：方案一，让 45 名同学排成一列，从第一名开始报数，逢奇数的人落选退出队列，偶数的人不动，整个队列报数结束后再继续从头开始，如此下去，谁能成为最后的幸运儿，谁就能担任志愿者；方案二，让 45 名同学围成一圈，任意指定一名同学为 1 号，开始报数，报到 3 的同学出列，之后令下一名同学作为 1 号开始继续报数，直到最后剩下一名同学，这名同学当选。某同学非常想成为学校志愿者，那么对于这两种不同的筛选方案，他在排队时应分别站在队列的什么位置？

“争做志愿者”项目中，方案一是从头报数到尾，报到奇数出列，之后再从头报数到尾，需要经过多轮报数才能得到最后一个。方案二是所有同学围成一圈，任意指定一名同学为 1 号，开始报数，报到 3 的同学出列，之后令下一名同学作为 1 号开始继续报数，直到最后剩下一名同学。

我们把项目活动中的方案一、方案二进行更一般的推广：

某班有  $n$  个同学围成一圈，从第  $k$  个人开始报数，报到第  $m$  个数的人退出，然后下一个人开始继续报数，并按同样规则退出，直至所有人退出，最后一个退出的人即为幸运儿，担任志愿者。

下面给出基于数组、顺序表、循环单链表的三种设计思路。

### 思路 1 基于数组概念的解法：

基于 Python 的列表和固定大小的“数组”概念，也就是说，在这里把列表看作元素个数固定的对象，只修改元素的值，不改变表的结构（不进行加入或删除元素的操作）。这相当于将  $n$  把椅子摆了一圈，人可以走但椅子在那里且位置不变。

下面给每个人赋予一个编号，没有人的情况用 0 表示，各列表的元素记录这些编号。

算法思路：

- (1) 建立一个包含  $n$  个人（的编号）的表。
- (2) 找到第  $k$  个人，从那里开始。
- (3) 处理中如果把相应的表元素修改为 0 则表示已出列，反复执行：
  - ① 数  $m$  个（还坐在椅子上的）人，遇到表的末端转回下标 0 继续。
  - ② 把表示第  $m$  个人的表元素修改为 0。
- (4)  $n$  个人出列即结束。

### 思路 2 基于顺序表的解法：

我们考虑：把保存人员编号的列表按表的方式处理，一旦确定了应该退出的人，就将表示其编号的表元素从表中删除。这样，随着计算的进行，所用的表将变得越来越短。

下面用  $num$  表示表的长度，每退出一人，表的长度  $num$  减 1，直到表长度为 0 时计算结束。使用这种方法，表中的元素都是有效元素（不再出现表示没人的 0），元素计数与下标计数得到统一，所以下标更新可以用  $i = (i + m - 1) \% num$  统一描述。

### 思路3 基于循环单链表的解法：

从形式上看，循环单链表可以很直观地表示围坐一圈的人。顺序数人头可以自然地反映为在循环表中沿着 next 链扫描，一个人退出可以用删除相应节点的操作模拟。在删除以后，又可以继续沿着原方向继续数人头了。

据此，算法分为两个阶段：

- (1) 建立包含指定个数(和内容)的节点的循环单链表。
- (2) 循环计算，找到并删除应该退出的节点。

下面我们基于之前的循环单链表类 Py\_CircularLinkedList 派生出一个专门的类 Py\_elect\_DLL。我们考虑把计数过程看作人圈的转动(节点环的转动)。这个类里定义了新方法 turn，它负责将循环表对象的 rear 指针沿 next 方向移动 m 步(相当于节点环旋转)。

在这个类的初始化函数中，首先调用 Py\_CircularLinkedList 的初始化函数建立一个空表，然后通过一个循环建立包含 n 个节点和相应数据的初始循环表。最后的循环反复调用 turn 方法，找到并逐个弹出节点，输出节点里保存的编号。

下面，我们给出基于数组、顺序表、循环单链表的三种参考程序。

基于数组的参考程序如下：

```
def Py_elect_A(n, k, m):  
    # 创建一个 n 个人(的编号)的列表，编号从 1 到 n  
    people = list(range(1, n+1))  
    i = k - 1  
    for num in range(n):  
        count = 0  
        while count < m:  
            if people[i] > 0:  
                count += 1  
            if count == m:  
                print(people[i], end=" ")  
                people[i] = 0  
            i = (i + 1) % n  
    return
```

基于顺序表的参考程序如下：

```
def Py_elect_L(n, k, m):  
    people = list(range(1, n+1))  
  
    num = n  
    i = k - 1  
    for num in range(n, 0, -1):
```

```
i= (i+m-1) % num
print(people.pop(i), end= "," if num> 1 else "\n")
return
```

基于循环单链表的参考程序如下：

```
from Py_CircularLinkedList import Py_CircularLinkedList
class Py_elect_CLL(Py_CircularLinkedList):
    def turn(self,m):
        for i in range(m):
            self.rear= self.rear.next

    def __init__(self,n,k,m):
        Py_CircularLinkedList.__init__(self) # 创建循环单链表(空表)
        for i in range(1,n+1):# 插入 n 个节点
            self.append(i)
        self.turn(k-1)
        while not self.isEmpty():
            self.turn(m-1)
        print(self.pop(),end= "\n" if self.isEmpty() else ",")
```

## 六、教学参考案例

### ■ 参考案例

#### 线性表初探

上海市崇明中学 顾 娟

(1课时)

##### 1. 学科核心素养

能够根据选拔志愿者项目活动中解决问题的需要,自觉、主动地寻求恰当的方式获取与处理信息;愿意与团队成员共享信息,实现信息的最大价值。(信息意识)

在报数选拔志愿者游戏中,能够对复杂生活情境中的关系进行抽象,采用合适的数据结构组织数据,表达数据的逻辑关系。(计算思维)

##### 2.《课程标准》要求

结合生活实际,理解数据结构的概念,认识数据结构在解决问题过程中的重要作用。

通过案例分析,理解数组、链表等基本数据结构的概念,并能编程实现其相关操作。比较数组、链表的区别,明确上述两种数据结构在存储不同类型数据中的应用。

### 3. 学业要求

知道数据结构对于数据处理的重要性,能够辨别简单的基于线性表的程序设计中数据的组织形式,描述数据的逻辑结构、存储结构和运算。

### 4. 教学内容分析

线性表是教科书第三章基础数据结构第一节的内容。通过对教科书内容进行单元设计,线性表教学分为 4 课时,本节课线性表初探是第 2 课时。在前一课时,学生已初步了解了什么是数据和数据结构、数据结构的意义、什么是线性表。本节课具有承上启下的作用,从整体上学习线性表的两种实现方式——顺序存储实现和链式存储实现,以及它们分别如何存储数据元素、如何访问数据元素、如何进行数据元素的插入和删除,还会学习链表的分类(单链表、双向链表、循环链表)和它们的区别。在后一节课,需通过编程实现这些存储方式。

### 5. 学情分析

本节课之前,学生已经学习了算法的三种基本结构和数组,对于本节课学习的内容,已具备了一定的知识基础。由于数据结构的内容比较抽象,本节课只进行一些概念和基本思想方法的学习,不涉及用具体的程序实现,同时,本节课从情境出发,采用报数选拔志愿者游戏的方式,分析关键步骤,总结特征,逐步开展学习。

### 6. 教学目标

- 通过用不同的报数方案选拔志愿者,分析、总结顺序表和链表存储数据元素的区别,学会在顺序表和链表中删除数据元素的方法,提升学生的信息意识,发展其计算思维。
- 通过列出志愿者报数顺序活动,理解链表中指针的作用。
- 通过增加志愿者名单活动,学会在顺序表和链表中插入数据元素,进一步理解两种实现方式的区别,发展学生的计算思维。
- 通过搭乘地铁案例,初步了解单链表、双向链表、循环链表的区别,体会数据结构对提高程序效率的作用,体会学习数据结构的意义。

### 7. 教学重难点

- 教学重点:顺序表和链表的特点。
- 教学难点:链表中的指针。

### 8. 教学策略分析

- (1) 采用项目活动教学法,通过选拔志愿者项目活动,学习顺序表和链表的实现思想和基本操作。
- (2) 采用游戏教学法,通过直观的游戏活动,实现不同的志愿者选拔方案,游戏完成后,再进行分析,寻找特征。
- (3) 采用情境教学法,利用小明搭乘地铁 4 号线的问题,引出双向链表和循环链表。

### 9. 教学过程设计(见图 3-8 和表 3-6)

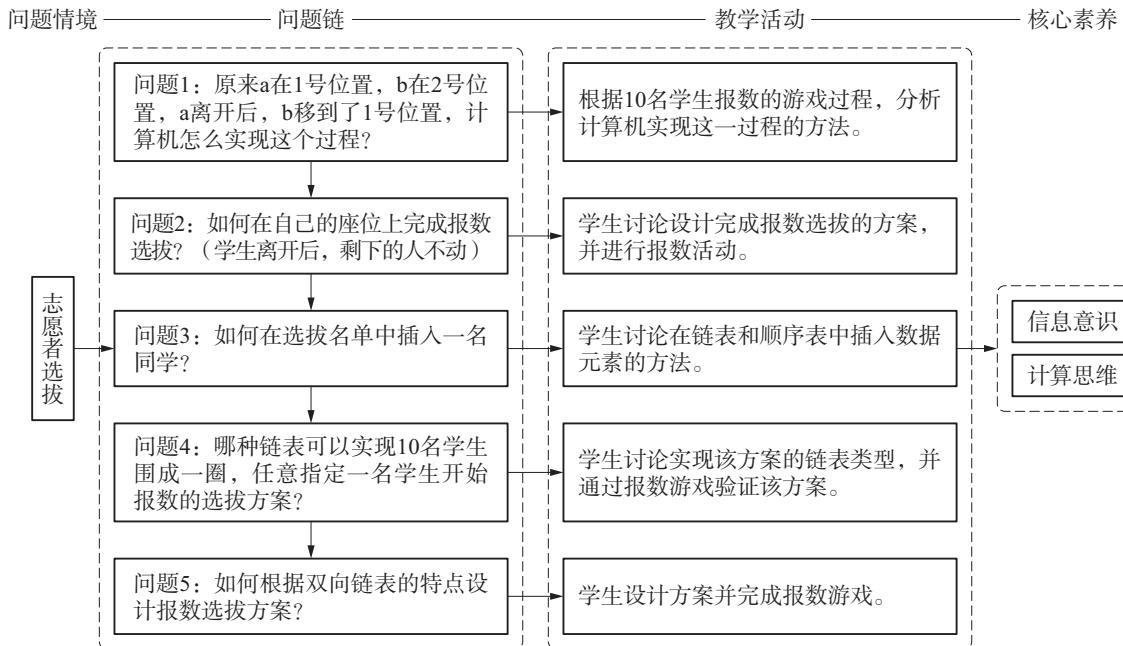
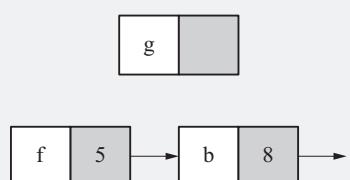
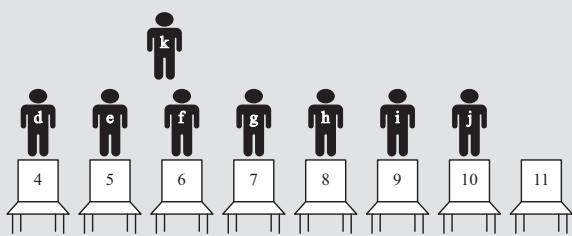


图 3-8 教学过程

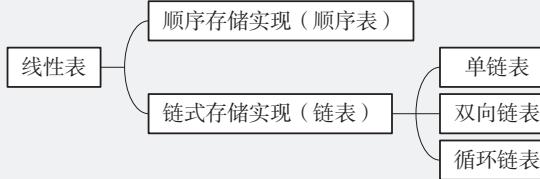
表 3-6 教学过程设计表

教学环节	教师活动	学生活动	设计意图
创设情境	<p>提出问题:同学们都当过志愿者吗?有人知道青年志愿者精神是什么吗?</p> <p>引出主题:学校假期招募青少年志愿者,某班同学踊跃报名,经过初步筛选,产生了一份 10 人名单,但志愿者招募名额只有一个,那么该选哪名学生做志愿者呢?</p> <p>a,b,c,d,e,f,g,h,i,j</p>	学生交流	引出本课主题,激励学生争当志愿者
顺序存储实现	<p>方案一:让 10 名学生排成一列,从第一名开始 1, 2, 1, 2……报数,报到 1 的人出列,剩下的人向前移动,重新形成一个新的队列,继续报数</p> <p>教师回顾演示几轮报数后的状态。</p> <p>教师分析:</p> <p>人——数据元素(a,b,c,d,e,f,g,h,i,j)。</p> <p>椅子——数据元素的存储位置(物理位置)。</p> <p>需要在教室前面把椅子放好——在计算机中需要一块连续的空间。</p> <p>每一轮报数结束后,重新排队——相邻的数据元素,物理位置相邻</p>	10 名学生到教室前面完成游戏	通过完成游戏并分析游戏中的关键步骤,寻找特征,总结顺序存储实现的特点,提升信息意识,发展学生的计算思维
	<p>提出问题:原来 a 在 1 号位置,b 在 2 号位置,a 离开后,b 移到了 1 号位置,计算机怎么实现这个过程?</p> <p>教师复习回顾线性表并总结:用这种方式存储的线性表,称为顺序存储实现(简称顺序表)</p>	学生交流讨论:用赋值	

续表

教学环节	教师活动	学生活动	设计意图
链式存储实现	提出问题:如果全班学生都参加选拔,还用刚才的方式,有什么问题?	学生交流:需要放几十把椅子,移动速度很慢	
	提示:是否可以不用到教室前面,而是在你们自己的座位上完成报数选拔?(学生离开后,剩下的人不动) 提出问题:需要解决哪些问题?	学生回答:从谁开始?按什么顺序报?	通过完成游戏并分析游戏中的关键步骤,寻找特征,总结链式存储实现的特点,提升信息意识,发展学生的计算思维
	方案二:10名学生手拉手,从第一名开始1,2,1,2……报数,报到1的人出列,剩下的人重新手拉手,继续报数。 教师回顾演示几轮报数后的状态。 教师分析: 第一轮报数后剩下b、d、f、h、j,他们的椅子不相邻——逻辑上相邻的数据元素,物理位置不一定相邻。 用手拉手确定报数的顺序——用指针。 报1的同学离开后,剩下的同学重新拉手——改变指针的值。 教师总结:用这种方式存储的线性表,称为链式存储实现(简称链表)。	学生完成报数选拔游戏	
	任务1:有6名学生a、b、c、d、e、f,物理位置分别是1、5、3、8、12、6,如果按照c、e、f、b、d、a的顺序报数,写出各数据元素中指针的值	学生完成任务1,填写各数据元素中指针的值	通过该任务,进一步理解链表中指针的作用
	提出问题:如果要在名单中加入一名学生,在b(物理位置5)前面插入g(物理位置7),怎么插入?  	学生交流回答:把g的指针值设置成5,把f的指针值改为7	
数据元素的插入	提出问题:在下面的顺序表中,如何在e前面插入k?  	学生交流回答:j开始依次向后移动	学会在顺序表和链表中插入数据元素,进一步理解两种实现方式的区别,并能够注意指针修改的顺序

续表

教学环节	教师活动	学生活动	设计意图
课堂小结	任务 2:总结顺序表和链表的特点,将正确选项前的字母填入表格中(表格见学习单)	学生完成任务	通过完成该任务,梳理前面的知识点,总结两种实现方式的区别
循环链表 和双向链表	小明搭乘地铁 4 号线(环线)从宜山路站到金沙江路站,结果他在车上睡着了,醒来的时候已经到了南浦大桥站,他该怎么办?	学生讨论: 方法 1:继续往前乘。 方法 2:下车后,往相反的方向乘	
	教师分析:可用循环链表实现方法 1。 提出问题:怎样将单链表修改成循环链表? 教师给出循环链表示意图	学生交流讨论:把最后一个数据元素的指针指向第一个数据元素	通过该活动,引出循环链表和双向链表的内容,比较三种链表的区别,发展学生的计算思维
	提出问题:单链表的头指针必须是第一个数据元素,循环链表可以任意指定指针起始位置,为什么?	学生讨论:单链表和循环链表的区别	
学生活动	教师分析:可用双向链表实现方法 2。 提出问题:怎样将单链表修改成双向链表? 教师给出双向链表示意图	学生讨论:再增加一个指针,指向前面的数据元素	
	方案三:10 名学生围成一圈,任意指定一名学生开始报数(1,2,3,1,2,3……),报到 3 的学生出列,之后令下一名学生从 1 开始继续报数,直到剩下最后一名学生当选。 提出问题:该方案用哪种链表实现? 提出问题:如果要根据双向链表的特点设计报数选拔方案,怎么设计?	学生回答:可用循环链表设计方案。 学生讨论并完成游戏	通过该活动,进一步理解循环链表的特点,发展学生的计算思维
	教师回顾梳理本节课学习的内容,展示知识结构图。 	思考、倾听	梳理本节课的知识点,体会学习数据结构的意义
课堂总结	教师总结:我们从生活中遇到的实际问题中抽象出数据结构,运用合适的数据结构,能更好地解决生活中的问题		

## 附：“线性表初探”学习单

任务 1:有 6 名学生按照如图 3-9 所示链表的顺序报数,写出各数据元素中指针的值。

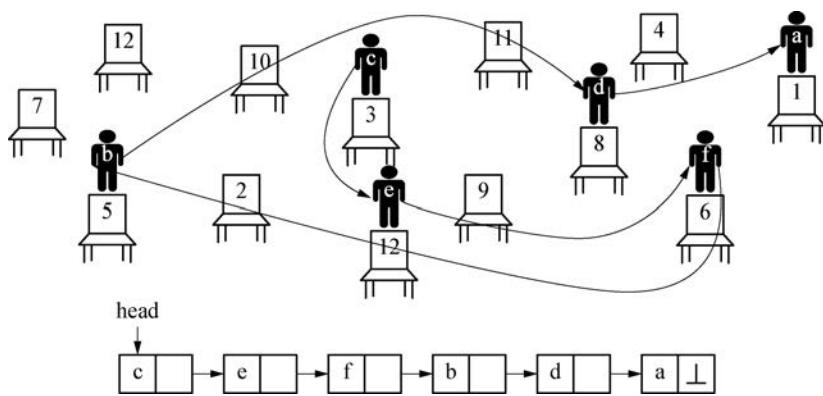


图 3-9 完成链表

任务 2: 总结顺序表和链表的特点, 将正确选项前的字母填入表 3-7 中。

表 3-7 顺序表和链表的特点

特点	顺序表	链表	选项
存储方式			A. 按顺序存放在一段连续的空间中 B. 存放在不连续的空间中 C. 相邻的数据元素物理位置不一定相邻 D. 相邻的数据元素物理位置相邻
空间性能			A. 不需预分配存储空间 B. 需要预分配存储空间 C. 需要存储链接信息
时间性能 (插入和删除)			A. 效率高 B. 效率低

任务 3: 完成方案三的志愿者选拔, 即 10 名学生围成一圈, 任意指定一名学生开始报数(1, 2, 3, 1, 2, 3……), 报到 3 的学生出列, 之后令下一名学生从 1 开始继续报数, 直到剩下最后一名学生当选。

## 第二节 栈

### 一、教学目标与重点

#### 教学目标:

- 通过体验思考和原理分析, 理解栈结构的含义、基本操作及应用场景。

- 在探究活动和编程实践过程中,掌握实现栈结构的一种方法,了解不同实现方法的差异。

- 在项目实践的过程中,能够运用栈结构解决实际问题,发展计算思维,提高数字化学习与创新能力。

**教学重点:**

运用栈结构解决实际问题。

## 二、教学实施与评价

### 1. 教学活动建议

本节内容包括栈的概念、基本操作、实现和应用。教学中对于栈的介绍应从为什么需要栈这种结构开始,通过“数制转换”活动观察程序中数据之间的关系,这种关系在程序运行中自然存在,再通过设计一种数据结构来体现这种关系,从而体会数据结构自然存在和人为设计这两层含义。在学生了解栈结构是怎么出现后提出栈的概念。通过对栈结构特点分析和生活中例子的类比,促进学生对栈这一概念的理解。对于栈后进先出的特点,建议在教学中图解演示一个出栈过程(当然在具体教学中图解一个入栈过程也可以)。

熟练掌握栈的基本操作,不仅能从中体会到栈的本质,还可以为后续学习栈的实现和应用打下基础。因此在教学活动设计上,建议通过解决真实的问题(如教科书中“列车调度”等学习活动)练习对栈结构的灵活操作,同时,引导学生去归纳栈结构对于改变序列的一般性规律。在这里可以从生活中的例子(如“超市手推车”的依次存放等)入手,激发学生的学习兴趣。

在栈的应用部分,需要知道栈的一般应用场景,举例说明栈作为数据结构中一种重要的线性结构,其广泛应用在编译软件和程序设计中(如递归实现、浏览器中的回退操作)。在解决“符号匹配”问题时,教师可以先图解过程,帮助学生观察理解,之后再请学生自己画图分析过程,通过演示结合讲解来实现问题的解决。对于示例程序的教学,可以先让学生尝试读懂程序,再让学生对关键性代码进行修改,最后会独立编写“符号匹配”的程序。

### 2. 教学过程安排(见表 3-8)

表 3-8 教学过程设计表

课时	教学环节	教师活动	学生活动
第 1 课时 (栈及栈的基本操作)	1. 情境导入	以生活中往箱子里面堆放东西和取东西的过程为例,引导分析该过程的特点。 在程序运行过程中经常需要记录一些中间状态,然后以一种相反的顺序再对这些状态一一进行处理(比如文字处理软件中“撤销”操作的实现),由此引出栈的概念	思考生活中后进先出现象和程序设计中的栈结构

续表

课时	教学环节	教师活动	学生活动
第2课时 (栈的实现)	2. 活动一: 数制转换	布置活动: 将十进制数N转换成八进制数, 需要不断用8除N及其商, 保留余数, 直至商为0	学生编程实现数制转换, 尝试发现规律——需要的结果中各位数字的次序正好与保留时的次序相反
	3. 栈的概念	讲解案例: 在浏览网页时, 点击超链接可以访问一个新的页面(压入栈中), 你可以不断点击新的超链接来访问新的页面, 你也总是可以点击回退按钮去访问以前访问的页面(从栈中弹出)。 引导学生认识什么是栈, 请学生回答以下问题: (1) 栈是不是线性表? (2) 对栈的操作遵循什么原则?	阅读“案例”, 结合栈示意图回答老师问题: (1) 栈是线性表中的一种。 (2) 栈遵循“后进先出(LIFO)”的原则。 名词: 栈顶、栈底、入栈(压栈)、出栈(弹栈)
	4. 活动二: 车辆调度	布置活动: 假设一列火车有四节车厢, 顺序编号为1、2、3、4, 从右边来, 现通过尽头线(相当于栈)实现1、4、3、2顺序从左边出去	根据要求完成“火车调度”活动; 听老师分析
	5. 栈的基本操作	提供: 栈的操作学习材料。 1. 顺序编号为1、2、3、4, 通过栈操作无法实现1、4、2、3这样的出栈序列, 为什么? 2. 对于任意的栈输入序列1, 2, 3, …, n, 怎样判断可行的输出序列?	小组讨论, 根据栈操作改变顺序的特点, 归纳一般规律
	6. 体验思考	引导思考: 1、2、3共有6种可能的排列, 即1, 2, 3; 1, 3, 2; 2, 1, 3; 2, 3, 1; 3, 1, 2; 3, 2, 1。利用栈操作, 不可能从1, 2, 3得到3, 1, 2, 但其他5种都能得到, 例如, 执行操作push, pop, push, push, pop, pop, 将实现从1, 2, 3到1, 3, 2。请给出从1, 2, 3到其他4个序列的栈操作过程	讨论、分析问题的解决方法。 完成方案
	7. 作业练习	布置作业: 作业练习第1题	完成方案
	8. 总结	结构特点和基本操作	
	1. 导入	在程序设计中栈结构是如何体现出来的?	倾听、思考
	2. 活动一: 数制转换1	布置活动: 在“数制转换”程序中, 通过变量更新体现栈操作	思考分析, 编程实现
	3. 探究活动	布置活动: 本节探究活动	思考, 回答问题
	4. 活动二: 数制转换2	布置活动: 在“数制转换”程序中, 先通过定义函数方法实现一个栈结构, 再通过对栈结构的调用在程序中实现栈操作	思考分析, 编程实现
	5. 总结	(1) 数据结构的两方面含义。 (2) 如何通过函数实现一种栈结构	

课时	教学环节	教师活动	学生活动
第3课时 (栈的应用)	1. 导入	列举栈结构在编译软件和程序设计中的广泛运用	倾听、思考
	2. 活动: 符号匹配	布置活动:“符号匹配”的要求 思路分析:主要步骤讲解 图解过程:数据出栈入栈绘图	倾听、思考 倾听、填写
		巡视,指导	编程实现
	3. 项目实践	安排项目实践:找出任意一条从起点到终点的路径,设计方案并编程实现	完成方案、编程实现
	4. 作业练习	布置作业:作业练习第2题	编程实现
	5. 总结	栈结构的应用场景、程序设计中栈结构的配合使用	

在完成各项目任务的过程中,观察学生对于栈的理解和应用。可以对学生在以下一些方面的表现进行评价:能描述栈结构的特点;能够使用栈操作、组织数据并能推广到一般规律;会使用类的方法构建栈结构;能够针对模型较为直观的实际问题,合理选用栈数据结构组织、存储数据,设计合适的算法,编程解决实际问题。在项目实践过程中,重点考查学生运用栈结构解决问题的能力及知识迁移能力。

### 3. 思考探究提示

(1) 本节体验思考参考答案见表3-9。

表3-9 体验思考参考答案

1, 2, 3—1, 2, 3	push, pop, push, pop, push, pop
1, 2, 3—2, 1, 3	push, push, pop, pop, push, pop
1, 2, 3—2, 3, 1	push, push, pop, push, pop, pop
1, 2, 3—3, 2, 1	push, push, push, pop, pop, pop

(2) 本节探究活动提示: top 是从 -1 开始计数的。(编程略)

### 4. 项目实施提示

从厨房到餐桌往往有多条路径,从通用性角度考虑,找出任意一条从起点到终点的路径,设计方案并编程实现。

(1) 选择合适的数据结构。

机器人从厨房出发,按某一方向(如向右)向前探索,如果路能走通(未走过的),向前到达一个新点,否则换一个方向试探;如果四个方向都不通,则沿原路返回前一点,换一个方向继续试探,直到所有可能的通路都搜索到,或找到一条通路,或无路可走又返回到出发点。因为只有先试探过的路,后面才能退回来,所以这里可以用栈来保存机器人走过的

路径,每走一步,将该位置压入栈中,若该点无路可走,则出栈返回上一位置。

### (2) 栈中存放数据的设计。

栈中所存放的数据元素包含所到达的每个点的坐标以及从哪个方向到达该点。在 Python 中可用一个元组(x,y,d)来表示。其中 d 的取值为“R”(右),“D”(下),“L”(左),“T”(上),d 的起始值取为“\_”。

### (3) 算法设计。

机器人走到某一新坐标位置,判断该处坐标值是 1 还是 0,如是 1 就表示不通,应换一个方向。如是 0 就表示通,把当前坐标位置入栈,同时把该坐标位置的值置为 1,表示已走到过这个坐标位置。然后按顺序判断该位置往 4 个方向是否存在通路,若找到通路,则走到该方向对应坐标位置,重复之前的操作。假如走到某一位置,往 4 个方向坐标都是 1,表示都走不通,那么就出栈前一个坐标,退回到前一坐标位置,再尝试 4 个方向,如还是都不通,再出栈前一个位置,这样一直往回退,直到 4 个方向中有可以前进的坐标位置,再继续走,再入栈。某一次入栈的位置等于餐桌坐标位置,就表明找到餐桌。流程如下:

① 用一个栈来记录机器人从厨房到餐桌探寻过程中的路径。

② 走到某点后,将该点坐标入栈,并把该点坐标值置成 1,表示走过了。

③ 按照临近的右,下,左,上方向选取前进位置往前走一步。

④ 如果在到达某点后临近的 4 个方向都不能走,说明已经走入了死胡同。此时出栈,退回一步尝试其他点。

⑤ 反复执行步骤②③④,直到找到餐桌。

### (4) 过程模拟。

机器人从厨房到餐桌送餐过程中实质是用栈暂存可以走的路径。在上一节线性表的项目实践中得到的最终地图上,模拟机器人送餐的过程,并在表 3-10 中填写入栈出栈过程中栈内数据元素变化情况。注意按右、下、左、上方向尝试。

表 3-10 栈内数据元素变化情况

① 读取到起点坐标(1,1)后,因为(1,1)坐标值为 0,入栈

(1,1,_)									
---------	--	--	--	--	--	--	--	--	--

② 向右读取(1,2)后,因为(1,2)坐标值为 0,入栈

(1,1,_)	(1,2,R)								
---------	---------	--	--	--	--	--	--	--	--

③ 向右读取(1,3)后,因为(1,3)坐标值为 1,表示不通;向下读取(2,2)坐标值为 0,表示通,入栈

(1,1,_)	(1,2,R)	(2,2,D)							
---------	---------	---------	--	--	--	--	--	--	--

④

...

(5) 参考程序。

```
import numpy as np
maze= np.ones(shape= (8,11))
maze[1,1]= 0;maze[1,2]= 0;maze[2,2]= 0;maze[2,5]= 0
maze[2,6]= 0;maze[2,7]= 0;maze[2,8]= 0;maze[3,2]= 0
maze[3,5]= 0;maze[3,8]= 0;maze[4,2]= 0;maze[4,3]= 0
maze[4,4]= 0;maze[4,5]= 0;maze[4,6]= 0;maze[4,7]= 0
maze[4,8]= 0;maze[4,9]= 0;maze[5,3]= 0
maze[5,6]= 0;maze[5,9]= 0;maze[6,3]= 0
maze[6,4]= 0;maze[6,5]= 0;maze[6,6]= 0;maze[6,9]= 0
print(maze)
start= (1,1)                                # 起点厨房坐标
end= (6,9)                                   # 终点餐桌坐标
i,j= start
ei,ej= end
Py_stack= list()                             # 创建一个空栈
Py_stack.append((i,j,"_"))                  # 将厨房坐标入栈
maze[i][j]= 1                                # 厨房坐标置为 1,表示机器人已走过
while Py_stack:                               # 当栈内不为空时
    i,j,d= Py_stack[-1]                      # 将栈顶坐标赋值给 i,j 。成为厨房新起点
    if (i,j)== (ei,ej):                      # 如果新起点坐标和终点坐标相同,说明到达餐桌
        break
    for di,dj,d in [(0,1,"R"),(1,0,"D"),(0,-1,"L"),(-1,0,"T")]:
        # 每次按右,下,左,上依次试探四个方向
        if maze[i+ di][j+ dj]== 0:           # 如果某一方向坐标对应的值为 0,
            # 说明路通
                maze[i+ di][j+ dj]= 1          # 将该方向坐标对应的值置 1
                Py_stack.append((i+ di,j+ dj,d)) # 将坐标和走过来的方向入栈
                break                         # 退出去往下一位置走
    else:
        Py_stack.pop()                      # 遍历四个方向后,没有路,则出栈一个坐标
if Py_stack:                                  # 如果栈内不为空
    print("路径是:",Py_stack)                 # 显示路径
else:
    print("没有出口")
```

### 三、作业练习与提示

#### ■ 题目描述

- 完成一个程序,令其输入是从 $1 \sim n$ 的n个数字的任意排列,功能是检查输入的排列是否能在序列 $1, 2, \dots, n$ 的基础上通过栈操作得到。如果能,就给出栈操作序列(例如push, push, pop, push, pop, …);如果不能,则指出出现失败的位置。进一步思考:上述情况隐含地假设了栈是无穷大的,当加入栈的大小限制条件时,例如m,情况会有什么变化?
- 多括号匹配:假设有一个表达式是由三种括号()[]{}嵌套组成的,并且嵌套顺序是任意的,但左右括号必须一一匹配,例如{[]}(( ))这样的格式是正确的,{[](( ))}这样的格式是不正确的。请编写一个程序,判断一个表达式字符串 $\text{expr} = "[[1 + 2] * [3 / (1 - 2) * (3 + 2)]]"$ 的括号匹配是否正确。

#### ■ 作业练习参考答案

1. 题意为判断任意一个序列B是否可以用一个从小到大的序列A通过栈操作得到。因此,本问题的解决首先需要创建一个栈stack。因为从小到大的序列入栈后,栈中的数据元素从栈顶来看的话是按从大到小排序的,所以我们将序列B中第一个数据元素和序列A中的数据元素依次比较,当序列B中的数据元素大于等于序列A中的数据元素时,就将序列A中的数据元素加入栈stack,否则结束比较。然后,序列B中第一个数据元素和栈顶数据元素比较,如果相等,则栈顶数据元素出栈,说明这个数据元素可以通过前面的几次入栈和一次出栈得到。如果不相等,则说明这个序列不能通过栈操作得到。上面是对序列A中第一个数据元素的判断,同理,重复上面的操作,可以实现对序列B中所有数据元素的判断。

参考程序如下:

```
# 输入任意需判断序列:如可输入 4,5,3,1,2
b= list(map(int, (input('请输入序列:')).split(',')))
a= list(range(1, len(b) + 1))
print('原来序列:', a)
print('任意序列:', b)
operate= []
stack= []
for i in b:
    while a and i >= a[0]:
        stack.append(a.pop(0))
        operate.append("push")
    if i == stack[-1]:
        operate.append("pop")
```

```

        print (stack.pop())
else:
    print ('不能实现'); break
else:
    print(operate)

```

2. 参考答案见表 3-11。

表 3-11 作业练习第 2 题参考答案

代码(lx4.py)	注释
<pre> left= {"(", "[", "{"} right= {")", "]", "}"} expr= "{[1+2]* [3/(1-2)* (3+2)]}" stack= [] for c in expr:     if c in left:         stack.append(c)     elif c in right:         if not stack:             print("缺左括号,不匹配")             break         if not 1&lt;= ord(c) - ord(stack[-1]) &lt;= 2:             print("括号错误,不匹配")             break         stack.pop()     else:         if not stack:             print("匹配")         else:             print("缺右括号") </pre>	<pre> # 创建一个包括左括号的集合类变量 left # 创建一个包括右括号的集合类变量 right # 需匹配的表达式 expr # 创建一个栈 stack # 依次遍历表达式中的每个字符到 c 中 # 如果 c 中内容是左括号 # 入栈 # 如果 c 中内容是右括号 # 如果是空栈 # 说明右括号无法与左括号匹配,所以显示结果“缺左括号,不匹配” # 退出循环,结束遍历 # 如果 c 和栈顶中字符的 ASCII 值差不为 1 或 2 # 显示“括号错误,不匹配” # 退出循环,结束遍历 # 出栈 # 循环正常结束 # 如果是空栈 # 显示“匹配” # 否则 # 显示“缺右括号” </pre>

## 四、核心概念介绍

**栈的概念:**一种特殊的线性表,只允许在一端进行插入或删除操作。允许进行插入和删除的一端称为栈顶;不允许进行插入和删除的一端称为栈底。

**举例:**假设一个栈里面存放了 4 个数据,数据 A 位于栈底,数据 D 位于栈顶,要想取出数据 B,首先要令数据 D 出栈,再令数据 C 出栈,然后才能令需要的数据 B 出栈,整个过程如图 3-10 所示。

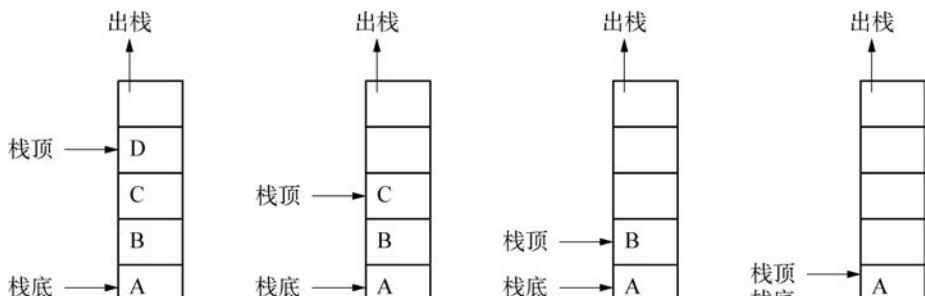


图 3-10 栈操作过程

## 五、教学参考资源

### 1. 栈的抽象数据类型

事实上,栈的抽象数据类型定义可以如下所示:

ADT 栈(stack)

(1) 数据对象。

数据对象为集合 $\{a_1, a_2, \dots, a_n\}$ ,每个元素的类型相同。

(2) 数据关系。

数据元素之间的关系是一对一的关系。其中除第一个元素 $a_1$ 外,每个元素有且只有一个直接前驱元素,除最后一个元素 $a_n$ 外,每个元素有且只有一个直接后继元素。

(3) 数据操作(见表 3-12)。

表 3-12 栈数据操作方法

方法	描述
stack()	创建一个栈区
push(x)	将 x 元素入栈
pop()	出栈
isEmpty()	判断是否为空栈
length()	获取栈的长度
getTop()	取栈顶的元素,元素不出栈
empty()	清空一个栈
free()	释放该栈所占据的内存空间

endADT

### 2. 栈的顺序存储和链接存储

栈是一种线性表,因此可以采用顺序存储结构或链式存储结构来实现。

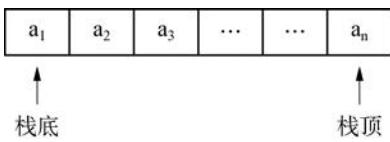


图 3-11 顺序栈

### (1) 顺序存储结构及其实现。

按顺序存储思想来实现的栈,称为顺序栈,见图 3-11。

实质上,借助 Python 的高度封装特点,可将列表(list)对象作为一个栈结构来使用,将列表的末尾看作栈顶,使用列表方法 append 能将元素压入栈中,而使用列表方法 pop 会删除并返回栈顶的元素。因此直接通过 stack = list() 或 stack = [] 就可以创建一个空栈。常用方法如表 3-13 所示。

表 3-13 顺序栈常用方法

stack 的方法	描述
append(x)	入栈,向栈内添加一个元素 x
pop()	出栈,删除栈顶一个元素
not stack	判断是否为空栈
len(stack)	获取栈内元素的数量
stack [-1]	获取栈顶的元素
stack = []	清空一个栈
del stack	释放该栈所占据的内存空间

### (2) 链式存储结构及其实现。

因只在栈顶进行插入和删除操作,所以只需构建一个单链表就能实现栈的功能。可以把单链表的首节点作为栈顶,这样单链表头部就是栈顶。通常这种栈也称为“链栈”,见图 3-12。

通常情况下,如果栈的使用过程中元素变化不可预料,一般用链栈,如果变化不大,就用顺序栈。

——参考《大话数据结构》,程杰,清华大学出版社

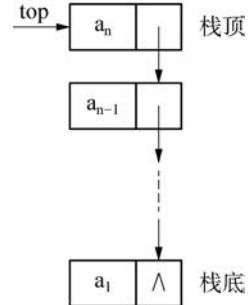


图 3-12 链栈

## 六、教学参考案例

### ■ 参考案例

#### 栈及栈的基本操作

同济大学第一附属中学 郎 樱

(1 课时)

##### 1. 学科核心素养

能够运用生活中的实例理解栈的概念和特点,能够将有限制条件的生活情境中的关系进行抽象,合理运用栈这类线性表来表达数据的逻辑关系。(信息意识)

针对模型较为直观的实际问题,能进行数据抽象,合理选择并熟练运用栈的特性存储数据,在此过程中进行自主或协作探究,通过编程操作来解决问题。(计算思维、数字化学习与创新)

## 2. 《课程标准》内容要求

结合生活实际,理解数据结构的概念,认识数据结构在解决问题过程中的重要作用。

通过问题解决,理解包括字符串、队列、栈在内的线性表的概念和基本操作,并编程实现。

## 3. 学业要求

能够针对限定条件的实际问题进行数据抽象,运用数据结构合理组织、存储数据,选择合适的算法(如排序、查找、迭代、递归等)编程实现、解决问题。

## 4. 教学内容分析

栈是教科书第三章基础数据结构第二节的内容。通过对教科书内容进行单元设计,栈教学分为3课时,本节课栈及栈的基本操作是第1课时。数据结构呈现的是计算机按照怎样的方式存储数据、组织数据之间的逻辑关系。栈作为数据结构中一种重要的线性结构,其广泛运用在各类软件系统里。栈的本质是受限的线性表,其基本操作依然属于线性表操作的子集,因此教科书中栈排在线性表的后面介绍。同时将栈和后续出现的队列相比较,从定义、操作方法、Python实现及应用实例等方面展示该数据结构的存储特点及在解决实际问题中的高效性。

## 5. 学情分析

学生应已熟练掌握Python的基本语法及基本操作,包括输入输出、分支循环、列表及自定义函数等,并已理解线性表的概念、掌握线性表(包括数组、链表等)的相关操作。

## 6. 教学目标

- 从观察生活实例出发,研讨实例中元素进出的过程,厘清栈的概念,明晰其“后进先出”的特点,并能在适当的限定场景中合理使用。

- 通过设计进制转换算法,寻找符合栈特性的关键数据结构模型,并利用栈的基本操作编程解决问题。在问题解决过程中,形成对数据抽象、数据结构的思想与方法的初步认识,提升学生的计算思维。

## 7. 教学重难点

- 教学重点:理解栈的概念和结构特点。
- 教学难点:掌握栈结构的基本操作。

## 8. 教学策略分析

本节课主要采用情境教学策略和问题教学策略。围绕栈的概念、基本操作及基础应用案例的实现三大环节展开教学过程。

(1) 创设情境、激发兴趣,引出核心概念。

教师利用生活中取用机场行李推车的案例,通过观察,归纳出“后进先出”的特点,从而引出数据结构——栈的核心概念。

(2) 设计问题链,由浅入深提供学习支架,突破难点。

设计层层递进的任务支架：从进行简单的行李推车管理员小实验来模拟出入栈任务到完成难度提升的进制转换任务，设计层层递进的问题支架：从询问“生活中有哪些‘后进先出’的实例”到询问“推车管理员小实验可能有几种情况”及“进制转换问题中哪部分数据具备栈的特性”，引导学生学会在遇到问题时选择恰当的数据结构，并通过算法实现解决问题。

### 9. 教学过程设计(见图 3-13 和表 3-14)

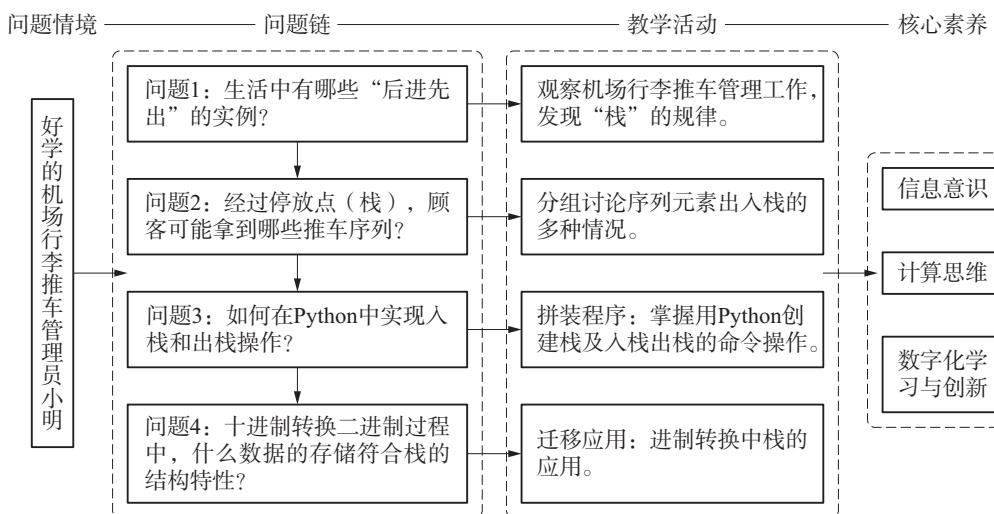
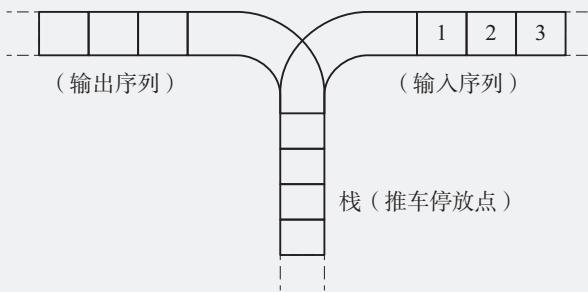
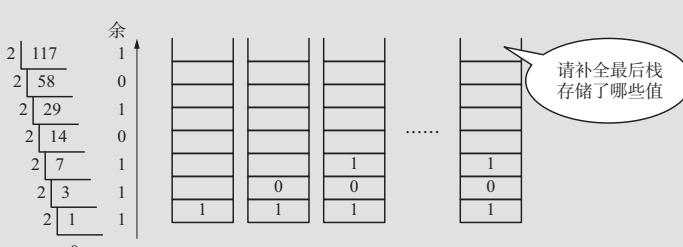


图 3-13 教学过程

表 3-14 教学过程设计表

教学环节	教师活动	学生活动	设计意图
情境导入明晰概念	<p>什么是栈 【情境：好学的机场行李推车管理员小明】 小明是个推车管理员，他每天负责将零散的推车整理到专用停放区内。久而久之，他渐渐发现了一个规律——最后被推进去的推车，往往是最先被顾客拿走的。这种“后进先出”的结构在现实生活中很普遍。 问题 1：生活中有哪些情况与此“后进先出”的情境类似？ 【栈的概念】 栈(stack)是一种遵循后进先出(last in first out)操作限定的线性表，此结构简称 LIFO 结构，该结构只允许在一端进行插入或删除操作</p>	<p>思考生活中的“后进先出”情境。 (预设可能的答案：收纳箱、软件的撤销和还原、字母车位等……)</p>	<p>在观察各类生活实例的过程中，明晰栈的内涵和外延，促进学生信息意识的提升</p>

教学环节	教师活动	学生活动	设计意图															
新知学习合作探究	<p>栈的基本操作——入栈(push)和出栈(pop)</p> <p><b>【机场推车的探究实验】</b></p> <p>小明是个爱探究的人,他通过观察行李推车的取用发现了栈的“后进先出”规律后,又做了个小实验。他给三辆行李推车编号(1,2,3),问题2:经过停放点(栈),顾客可能拿到哪些行李推车序列?</p>  <p>小明通过观察,记录了第一种情况——“321”。 请同学分组讨论,除了“321”之外,顾客还可能以哪些序列取得推车? (根据学生探究情况,追问为什么“312”不行) <b>【巩固练习】</b> 若小明提供的行李推车输入序列为“1234”,则借助栈操作,下列输出顺序不可能的是( )。(多选) A. “1234” B. “2341” C. “4123” D. “1342” E. “4321” F. “3412” G. “3241” 答案:CF。 规律:对于出栈序列中的每一个数字,在它后面的、比它小的所有数字,一定是按递减顺序排列的</p>	<p>学生分组讨论,提出剩余4种情况(123, 132, 213, 231)</p>	从数据结构的视角审视较复杂的生活场景,表达其逻辑关系															
小试牛刀	<p>Python中栈的实现</p> <p><b>【顺序栈】</b>按顺序存储思想来实现栈。</p> <p>问题3:如何在Python中实现入栈和出栈操作?</p> <p>Python的列表作为栈结构使用时,将列表的末尾看作栈顶。例如使用<code>py_stack = list()</code>可以创建一个空栈。常用栈操作方法如下表所示。</p> <table border="1"> <thead> <tr> <th>py_stack的方法</th> <th>功能描述</th> </tr> </thead> <tbody> <tr> <td><code>append(x)</code></td> <td>入栈,向栈内添加一个元素</td> </tr> <tr> <td><code>pop()</code></td> <td>出栈,删除栈顶一个元素</td> </tr> <tr> <td><code>not py_stack</code></td> <td>判断是否为空栈</td> </tr> <tr> <td><code>len(py_stack)</code></td> <td>获取栈内元素的数量</td> </tr> <tr> <td><code>py_stack[-1]</code></td> <td>获取栈顶的元素</td> </tr> <tr> <td><code>py_stack = []</code></td> <td>清空一个栈</td> </tr> <tr> <td><code>del py_stack</code></td> <td>释放该栈所占据的内存空间</td> </tr> </tbody> </table> <p>教师演示操作:输入序列“123”,用Python进行栈操作,输出序列“321”。</p>	py_stack的方法	功能描述	<code>append(x)</code>	入栈,向栈内添加一个元素	<code>pop()</code>	出栈,删除栈顶一个元素	<code>not py_stack</code>	判断是否为空栈	<code>len(py_stack)</code>	获取栈内元素的数量	<code>py_stack[-1]</code>	获取栈顶的元素	<code>py_stack = []</code>	清空一个栈	<code>del py_stack</code>	释放该栈所占据的内存空间	<p>各学习小组抽签得到上一环节的四类情况(123, 132, 213, 231)之一,根据教师提供的出入栈指令,完成程序的拼装</p> <p>编写程序,实现栈的逻辑结构及操作</p>
py_stack的方法	功能描述																	
<code>append(x)</code>	入栈,向栈内添加一个元素																	
<code>pop()</code>	出栈,删除栈顶一个元素																	
<code>not py_stack</code>	判断是否为空栈																	
<code>len(py_stack)</code>	获取栈内元素的数量																	
<code>py_stack[-1]</code>	获取栈顶的元素																	
<code>py_stack = []</code>	清空一个栈																	
<code>del py_stack</code>	释放该栈所占据的内存空间																	

教学环节	教师活动	学生活动	设计意图
	<pre>py_stack= list() py_stack.append('1') py_stack.append('2') py_stack.append('3') print (py_stack.pop()) print (py_stack.pop()) print (py_stack.pop())</pre>		
迁移应用	<p>栈的应用 栈最大的特点是后进先出,所以逆序输出是栈经常用到的一个应用场景。 【进制转换】 输入一个十进制数 N,将 N 转换成二进制数后输出。例如输入 117,输出 1110101。 分析:“采用除 2 取余,逆序排列”的方法。 问题 4:十进制转换二进制过程中,什么数据的存储符合栈的结构特性? 活动:请补全最后栈存储了哪些值,并完成程序填空。</p> 	<p>学生思考回答问题,观察栈的存放规律,推断出最后存储的结果,完成程序填空和调试</p>	<p>寻找实际问题中关键数据的组织方式,将栈的特点迁移到该问题的解决中,促使学生计算思维的初步养成</p>
进阶挑战	<p>课后作业:进制转换升级(可选择完成)。 挑战 1:设计一个十进制数 n 转换成 m 进制的函数 convert(n,m),其中 n 为输入的十进制数,m(<math>1 &lt; m &lt; 10</math>)为转换成的进制。(难度:☆☆) 例如: convert(183,2) = 10110111 convert(183,8) = 267 挑战 2:设计将一个十进制数 n 转换成 m 进制的函数 convert(n,m),其中 n 为输入的十进制数,m(<math>m &gt; 1</math>)为转换成的进制。(提示:请把十六进制考虑进去)(难度:☆☆☆) 例如: convert(183,2) = 10110111 convert(183,8) = 267 convert(183,16) = B7</p>	<p>学生可根据自身学习水平,选择性完成个性化作业</p>	<p>依据不同学情及资源,开展个性化学习、知识分享与创新创造</p>
	<p>课时教学板书设计:</p> <ol style="list-style-type: none"> <li>1. 栈的概念 {线性表 后进先出}</li> <li>2. 栈的基本操作(Python 为例) 入栈——列表.append(x) 出栈——列表.pop()</li> <li>3. 栈的应用之一——逆序输出</li> </ol>		

## 附：“栈及栈的基本操作”学习单

### 【栈的概念】

栈(stack)是一种\_\_\_\_\_的线性表,此结构简称 LIFO 结构。

### 【情境:好学的机场行李推车管理员小明】

小明是个爱探究的人,他通过观察行李推车取用发现了栈的“后进先出”规律后,又做了个小实验。他给三辆行李推车编号(1,2,3),经过停放点(栈)操作后,观察顾客可以拿到哪些推车序列,如图 3-14 所示。

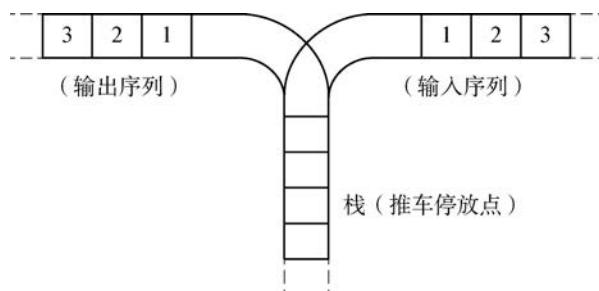


图 3-14 推车序列

顾客还可能拿到哪些行李推车序列?

若小明提供的行李推车输入序列为“1234”,则借助栈操作,下列输出顺序不可能的是( )。(多选)

- A. “1234”
- B. “2341”
- C. “4123”
- D. “1342”
- E. “4321”
- F. “3412”
- G. “3241”

### 【Python 中栈的实现】

使用 `py_stack = list()` 可以创建一个空栈。常用栈操作方法如表 3-15 所示。

表 3-15 常用栈操作方法

py_stack 的方法	功能描述
<code>append(x)</code>	入栈,向栈内添加一个元素
<code>pop()</code>	出栈,删除栈顶一个元素
<code>not py_stack</code>	判断是否为空栈
<code>len(py_stack)</code>	获取栈内元素的数量
<code>py_stack[-1]</code>	获取栈顶的元素
<code>py_stack = []</code>	清空一个栈
<code>del py_stack</code>	释放该栈所占据的内存空间

### 任务一:出入栈语句拼装,完成行李推车序列的其他几种情况(难度:☆)

代码区	输入 123
① py_stack = list()	输出 231 ,则语句排序为_____;
② py_stack.append('1')	输出 132 ,则语句排序为_____。
③ py_stack.append('2')	
④ py_stack.append('3')	
⑤ print(py_stack.pop())	
#可多次选择语句进行拼装	

### 【栈的应用】

#### 任务二:进制转换(难度:☆☆)

输入一个十进制数 N,将 N 转换成二进制数后输出。例如输入 117,输出 1110101。  
请在图 3-15 中补全最后栈存储了哪些值,并完成程序填空。

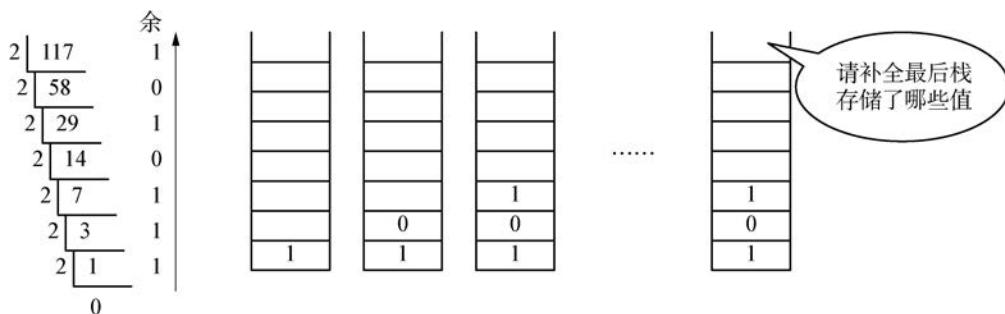


图 3-15 进制转换过程

程序区	测试数据
py_stack = list() # 创建一个栈	
N = int(input("number"))	
while N: # 到 N 为 0 停止循环	输入 389
<div style="border: 1px dashed black; padding: 5px; width: fit-content;"># 计算余数入栈 # 继续除 2</div>	输出 _____
out = ""	
while py_stack: # 所有元素依次出栈,栈不空就一直循环	
out = out + _____	
print(out)	

### 【进阶挑战】进制转换升级(选择其中之一完成)

- 设计一个十进制数 n 转换成 m 进制数的函数 convert(n, m),其中 n 为输入的十进制数,m( $1 < m < 10$ )为转换成的进制。(难度:☆☆)

例如:

convert(183,2) = 10110111

convert(183,8) = 267

2. 设计一个将十进制数 n 转换成 m 进制数的函数 convert(n,m), 其中 n 为输入的十进制数, m(m>1) 为转换成的进制。(提示: 请把十六进制考虑进去)(难度: ☆☆☆)

例如:

convert(183,2) = 10110111

convert(183,8) = 267

convert(183,16) = B7

**参考答案:**

```
def convert(n,m):  
    py_stack= list()    # 创建一个栈  
    while n:           # 余数入栈, 到 n 为 0 停止循环  
        py_stack.append(n% m)  
        n= n//m  
    out= ""  
    while py_stack: # 栈不空就一直循环  
        num= py_stack.pop()  
        if num< 10:  
            out= out+ str(num)  
        else:  
            out= out+ (chr(num+ 55))    # 16 进制时转换为 ABCDEF 字符  
    return out  
  
n= int(input("n= "))  
m= int(input("m= "))  
print (convert(n,m))
```

### 第三节 队列

#### 一、教学目标与重点

##### 教学目标:

- 通过体验思考和原理分析, 理解队列结构的含义、基本操作及应用场景。

• 在探究活动和编程实践过程中,掌握实现队列结构的一种方法,了解不同实现方法的差异,感悟队列解决问题的高效性。

• 在项目实践的过程中,能够运用队列结构解决实际问题,发展计算思维,提高数字化学习与创新能力。

**教学重点:**

运用队列结构解决实际问题。

## 二、教学实施与评价

### 1. 教学活动建议

本节内容主要包括队列的概念、基本操作、实现和应用。在队列的概念部分注意关注知识的来龙去脉,建议从计算机领域一些常见问题的解决中引出队列这一数据结构,可以从子任务、多程序协调管理等案例开始介绍,凸显出队列结构的重要性。对于队列结构的基本操作建议以活动的形式开展,如采用教科书中“编码解密”这个活动让学生体会“先进先出”这种特点是如何体现的。当然这种“先进先出”和生活中排队现象可以做类比但也应注意一些区别。

在教学中可以采用图解的方法表示“入队”“出队”这种常见操作,在队列的实现部分,用函数的方式加深对队列操作的理解。建议结合函数实现要点进行逐步分析,也可以让学生先自主学习,再听教师讲解。对实现过程中出现的多种方法进行比较,体现代码的通用性。在应用部分,可以采用 Python 自带的库构建一个队列。在解决实际问题的过程中,需要保存的数据满足“先进先出”的规则时,采用队列结构是一种高效简便的解决方案。在队列的应用部分可以增加“挑选数字”的活动,如下所示。

(1) 问题描述。

随机产生 10 个小于 100 的正整数,将其中的奇数保存并输出。

(2) 设计思路。

问题中求解的数据可以采用队列结构来保存。首先,用 random.randint(1, 99) 产生 10 个随机整数全部入队,然后依次出队,出队一个判断一个,如果是奇数就重新添加到队尾。依此类推。

(3) 参考程序。

```
import queue          # 导入队列库
import random         # 导入随机数库
que= queue.Queue()    # 新建一个空队列
for i in range(10):      # i 从 0 到 9 循环
    que.put(random.randint(1, 99))  # 10 个随机数添加到队列中
for i in range(10):      # i 从 0 到 9 循环
    if que.get()%2==1:       # 判断是否为奇数
        que.put(que.get())  # 是奇数则重新添加到队列尾
```

```

temp= que.get()                      # 出队,并赋值给变量 temp
if  temp % 2== 1:                     # 如果是奇数
    que.put(temp)                     # 添加到队尾
while not que.empty():
    print(que.get(),end= " ")        # 显示队列

```

为了与栈结构做比较学习,在队列的应用部分建议增加“符号匹配”的活动,如下所示。

- (1) 建立一个空队列,用来存储尚未匹配的左括号。
- (2) 遍历字符串,遇到左括号则入队,遇到右括号则出队一个左括号进行匹配。若遇到右括号时,出现了队列为空的情况,则说明“缺左括号”,程序结束。
- (3) 在第 2 步遍历结束后,队列为空,则输出“匹配”;队列不为空,则输出“缺右括号”,程序结束。
- (4) 参考程序。

```

import queue
string= "(6+ (2+ 4) * (3/ (1+ 2) - (3+ 2)))"
que= queue.Queue()
for a in string:
    if a== "(":
        que.put(a)
    elif a== ")":
        if que.empty():
            print ("缺左括号")
            break
        que.get()
else:
    if que.empty():
        print ("匹配")
    else:
        print("缺右括号")

```

## 2. 教学过程安排(见表 3-16)

表 3-16 教学过程设计表

课时	教学环节	教师活动	学生活动
第 1 课时 (概念和基 本操作)	1. 导入 2. 队列概念	列举生活中的排队现象、计算机中处理数据的排队如打印,引入队列的概念 引导学生认识什么是队列。 (1) 队列是不是线性表? (2) 队列的操作遵循什么原则?	思考生活中队列和程序设计中的队列现象 阅读教科书,回答问题

续表

课时	教学环节	教师活动	学生活动
第 2 课时 (队列的实现)	3. “编码解密”活动	布置活动:按照一定的规律对微信号解密	完成解密,对数据重新组织
	4. “队列操作”活动	布置活动:对某个数据序列进行入队出队操作	图解队列的操作
	5. 活动归纳	提炼分析队列操作的一般规律	参与归纳
	6. 体验思考	引导思考:本节体验思考	讨论、分析问题的解决方法
	7. 总结	结构特点和基本操作	
	1. 导入	提出问题: (1) 队列结构特点; (2) 队列基本操作	倾听、思考
	2. 队列实现方案	(1) 布置自我阅读(函数实现队列要点); (2) 按步骤图解队列实现的要点	自我阅读、思考分析
第 3 课时 (队列的应用)	3. 函数实现队列结构	引导学生读流程图、完成修改关键代码、完成程序	思考分析,编程实现
	4. 库实现队列结构	讲解如何在 Python 中调用 queue 库实现队列结构,引导学生学会使用 help(queue)自主学习	调用,编程
	5. 作业练习	布置作业:作业练习 2	编程实现
	6. 总结	用函数实现队列结构	
	1. 导入	列举队列在计算机系统工作的过程中有多种用途,如打印任务管理	倾听、思考
	2. 探究活动	引导:本节探究活动	交流讨论
	3. 活动:“挑选数字”	布置任务:随机产生 10 个小于 100 的正整数,将其中的奇数保存并输出	思考,完成
	4. 活动:“符号匹配”	布置活动:“符号匹配”的要求,用队列实现	倾听
		思路分析:主要步骤讲解,可以和栈的实现做比较	倾听、思考
		图解过程:绘制数据出队入队过程示意图	倾听、填写
		巡视,指导	编程实现
	5. 项目实践	安排项目实践:本节项目任务	完成方案、编程实现
	6. 作业练习	布置作业:本节作业练习 1	编程实现
	7. 总结	队列结构的应用场景、程序设计中队列结构的配合使用	

在完成各项目任务的过程中观察学生对于队列的理解和应用。可以对学生在以下一些方面的表现进行评价:能描述队列结构特点及队列在计算机系统工作过程中的多种用

途;能够使用队列操作、组织数据;能描述采用数组作为队列储存空间的关键要素;会使用函数的方法构建队列结构;能够针对实际问题,合理选用队列数据结构组织、存储数据,设计合适的算法,编程解决实际问题。在项目实践过程中,重点考查学生运用队列结构解决问题的能力及知识迁移能力。

### 3. 思考探究提示

#### (1) 本节体验思考参考答案:

out, out, in, in, in, in, in, in, in, out

#### (2) 本节探究活动参考答案:

计算机内打印队列等。

### 4. 项目实施提示

#### (1) 选择合适的数据结构。

在地图中,从起点到终点往往有多条路径。从起点出发,寻找所有下一个能继续走的点,根据下一个点继续寻找所有能走的点,直到该点等于终点。可以用队列暂存所有能走的点。首先将起点位置入队。在队列不为空的时候循环:出队一次并存储数据值至列表中,如果当前位置为终点,则结束,否则找到当前相邻的4个位置(右、下、左、上)中可走的位置,加入队列。最后列表中的数据就是走过的路径。

#### (2) 队列中存放数据的设计。

队列中所存放的元素包含所到达的每个点的坐标以及是从哪个点移动至该点的。在Python中使用一个元组(x,y,d)来表示。其中,d的值为当前列表中的路径步数减去1,表示当前坐标是由前面哪个点移动过来的,d起始值取为-1。

#### (3) 算法设计。

走到某点后,按右、下、左、上方向向前探索,走到某一新坐标位置,并判断该坐标值是1还是0。如是1就表示不通,换一个方向。如是0就表示通,把当前坐标位置入队,同时把该坐标位置的值置为1,表示已走到过这个坐标。将相通的坐标位置和列表中前点位置全部入队。出队的位置等于终点坐标位置,就表明找到终点。流程如下:

① 创建一个队列来记录机器人走过的路径,创建一个列表来保存出队的数据,将起始坐标入队。

② 当队列不为空时,出队一个数据保存到列表中,如果出队的数据和终点相同,去显示列表中的最短路径并结束。

③ 走到某点后,按右、下、左、上方向向前探索,将相通的位置坐标值设为1,并将相通的坐标位置和列表中前点位置全部入队。

④ 反复执行步骤②、步骤③,直到找到终点。

#### (4) 过程模拟。

模拟机器人行走过程,观察表3-17中队列和列表中数据变化。注意按右、下、左、上方向尝试。队列中起始坐标为(1,1,-1)。

表 3-17 机器人行走过程中数据变化

① 出队,将数据保存到列表中;从当前出队的坐标位置往右、下、左、上试探读取坐标后,将相通的坐标和列表元素总数减 1 入队。

列表: 

(1,1,-1)						
----------	--	--	--	--	--	--

队列: 

(1,2,0)						
---------	--	--	--	--	--	--

② 出队,将数据保存到列表中;从当前出队的坐标位置往右、下、左、上试探读取坐标后,将相通的坐标和列表元素总数减 1 入队。

列表: 

(1,1,-1)	(1,2,0)					
----------	---------	--	--	--	--	--

队列: 

(2,2,1)						
---------	--	--	--	--	--	--

③ 出队,将数据保存到列表中;从当前出队的坐标位置往右、下、左、上试探读取坐标后,将相通的坐标和列表元素总数减 1 入队。

列表: 

(1,1,-1)	(1,2,0)	(2,2,1)				
----------	---------	---------	--	--	--	--

队列: 

(3,2,2)						
---------	--	--	--	--	--	--

④ 出队,将数据保存到列表中;从当前出队的坐标位置往右、下、左、上试探读取坐标后,将相通的坐标和列表元素总数减 1 入队。

列表: 

(1,1,-1)	(1,2,0)	(2,2,1)	(3,2,2)			
----------	---------	---------	---------	--	--	--

队列: 

(4,2,3)						
---------	--	--	--	--	--	--

⑤ 出队,将数据保存到列表中;从当前出队的坐标位置往右、下、左、上试探读取坐标后,将相通的坐标和列表元素总数减 1 入队。

列表: 

(1,1,-1)	(1,2,0)	(2,2,1)	(3,2,2)	(4,2,3)		
----------	---------	---------	---------	---------	--	--

队列: 

(4,3,4)						
---------	--	--	--	--	--	--

⑥ 出队,将数据保存到列表中;从当前出队的坐标位置往右、下、左、上试探读取坐标后,将相通的坐标和列表元素总数减 1 入队。

列表: 

(1,1,-1)	(1,2,0)	(2,2,1)	(3,2,2)	(4,2,3)	(4,3,4)	
----------	---------	---------	---------	---------	---------	--

队列: 

(4,4,5)	(5,3,5)					
---------	---------	--	--	--	--	--

⑦ ...

### (5) 参考程序。

```
import queue # 导入队列库
maze= np.ones(shape= (8,11))
maze[1,1]= 0; maze[1,2]= 0; maze[2,2]= 0; maze[2,5]= 0
maze[2,6]= 0; maze[2,7]= 0; maze[2,8]= 0; maze[3,2]= 0
```

```

maze[3, 5]= 0;maze[3, 8]= 0;maze[4, 2]= 0;maze[4, 3]= 0
maze[4, 4]= 0;maze[4, 5]= 0;maze[4, 6]= 0;maze[4, 7]= 0
maze[4, 8]= 0;maze[4, 9]= 0;maze[5, 3]= 0
maze[5, 6]= 0;maze[5, 9]= 0;maze[6, 3]= 0
maze[6, 4]= 0;maze[6, 5]= 0;maze[6, 6]= 0;maze[6, 9]= 0
print(maze)

def shortest_path(path):                                # 通过 path 找到最短路径
    shortestpath= []
    curNode= path[- 1]
    while curNode != path[0]:
        shortestpath.append(curNode)
        curNode= path[curNode[2]]                      # 根据第三个元素找到前一个点
    shortestpath.append(path[0])
    shortestpath.reverse()
    return shortestpath

start= (1, 1)                                         # 起点厨房坐标
end= (6, 9)                                           # 终点餐桌坐标
i, j= start
ei, ej= end
Py_queue= queue.Queue()                               # 新建一个空队列
Py_queue.put((i, j, - 1))                           # 将起点坐标入队
path= []                                              # 创建列表
maze[i][j]= 1                                         # 起点坐标置为 1, 表示机器人已走过
while not Py_queue.empty():
    temp= Py_queue.get()                            # 出队一个元素, 成为机器人新起点
    path.append(temp)                             # 将出队的元素添加到列表中
    i, j= temp[0], temp[1]
    if (i, j)== (ei, ej):           # 若新起点坐标和终点餐桌坐标相同, 则到餐桌
        print(shortest_path(path))                # 显示最短路径
        break
    for di, dj in [(0, 1), (1, 0), (0, - 1), (- 1, 0)]:
        # 每次走的坐标增量值 di, dj, 依右、下、左、上遍历四个方向
        if maze[i + di][j + dj]== 0:
            # 如果某一个方向上坐标对应的值为 0, 说明路通
            maze[i + di][j + dj]= 1                  # 将该坐标对应的值置 1
            Py_queue.put((i + di, j + dj, len(path) - 1)) # 将该坐标入队

```

### (6) 解决方案的迁移。

机器人送餐可以有多种实现方案,除了上面的方案外,还有沿着地面上贴的轨迹线送餐,或者利用在餐桌上放置的信号发射源辅助送餐等多种方案,请结合不同应用场景,设计和分析尽可能多的方案,归纳总结特点,并探究类似方案还可以应用在生活中的哪些场景中。

## 三、作业练习与提示

### ■ 题目描述

- 对于任意给定的一对正整数  $a$ 、 $b$ ,假设可以在  $a$  基础上进行加 1、减 1 或乘 2 的操作。那么,对  $a$  最少进行多少次操作后,能得到  $b$ ? 例如:  $a = 3$ ,  $b = 11$ ,可以通过  $3 \times 2 \times 2 - 1$ ,3 次操作得到 11。 $a = 5$ ,  $b = 8$ ,可以通过  $(5 - 1) * 2$ ,2 次操作得到 8。试给出解决这个问题的一般思路,并编程实现。
- 在本章第三节第二点讨论的队列实现中,用到计数器变量(counter)来指示队列中元素的个数,并通过对其值的判断,决定出队和入队操作是否可能。事实上,也可省去这个变量,直接通过队列头指针 head 和尾指针 tail 之间的关系来做相应判断。试讨论总结出相关规律,并尝试编程实现。
- 有一种资源管理问题,抽象来讲,设有  $n$  个单位的资源, $r_1$ ,  $r_2$ , ...,  $r_n$ ,程序(资源管理器)面对一个请求(req)和释放(rel)序列,如:

req, rel(i), ..., req, req, ..., rel(j), ..., rel(i), ...

其中 req 表示请求,可被任一单位的资源满足,rel(i)表示释放,会指出释放哪一个(i),且不一定按照请求发生的顺序释放。于是,在请求和释放序列的作用下,我们看到两个队列“当前使用队列”(USE)和“当前可用队列”(AVL)的动态变化——“此消彼长”。资源是不连续存储的,在这种情形下,队列的实现就需要使用链表。试用一个二维数组来表示资源集合  $r_1$ ,  $r_2$ , ...,  $r_n$ ,每行代表一个资源,其内容包括指向相关队列下一个资源的指针。试讨论实现这样一个资源管理器所涉及的问题,并尝试编程实现。

### ■ 作业练习参考答案

- 对  $a$  分别进行加 1、减 1、乘 2 得到 3 个数并保留,依次提取这三个数判断是否为  $b$ ;如果 3 个数都不是,则每个数再进行加 1、减 1、乘 2 得到新的 9 个数并保留下,之后再依次判断,如此反复,直到找到为止。这个保留和提取数据的过程,就是一个“先进先出”的过程,因此可以用队列结构来暂存这些数据。

- ① 设置一个计数器  $i$ , $i$  的初始值为 0。
- ② 建立一个队列,把初始数据( $a$ ,  $i$ )入队; $a$  表示变化的初始数据, $i$  是操作次数。
- ③ 出队,并将其中的两个数据分别赋值给  $x$  和  $i$ ,判断  $x$  是否为  $b$ ,如是则计算结束并输出  $i$ 。

④ 如果  $x$  不为  $b$ , 那么再依次进行  $x+1$ 、 $x-1$ 、 $x*2$  操作, 对变化后的数据依次判断是否为  $b$ , 如不是则新数据和  $i+1$  作为一个整体元素入队。

⑤ 重复步骤③④。

根据需求, 进一步细化入队规则: ①因为求操作次数, 所以每次入队的内容除了数据外还应有操作次数。具体实现可以用元组如  $(x, i)$  的形式作为入队的内容。 $x$  表示数据,  $i$  表示操作次数, 每次入队操作数加 1。②为防止重复变换而进入死循环, 因此每次变换后产生的新数据若之前入队过, 它就不要入队了。具体实现可以把每次产生的新数据添加到一个集合中, 下次产生新数后判断集合里有否相同数, 有的话, 就不入队。

参考程序如下:

```
import queue # 导入队列库
def change(a, b): # 创建一个函数, a 表示初始数据, b 表示最终数据
    Py_queue = queue.Queue() # 新建一个空队列
    Py_queue.put((a, 0)) # 初始值 a 入队
    checked = {a} # 初始值添加到集合
    x, i = Py_queue.get() # 将数据和操作次数出队
    while x != b: # 数据不为最终数据
        if x + 1 not in checked: # 数据加 1 后, 不在集合中
            Py_queue.put((x + 1, i + 1)) # 数据加 1, 操作次数加 1, 然后添加到队尾
            checked.add(x + 1) # 数据加 1 后, 进入集合中
        if x - 1 not in checked: # 若数据减 1 后, 不在集合中
            Py_queue.put((x - 1, i + 1)) # 数据减 1, 操作次数加 1, 然后添加到队尾
            checked.add(x - 1) # 数据减 1 后, 进入集合中
        if x * 2 not in checked: # 若数据乘 2 后, 不在集合中
            Py_queue.put((x * 2, i + 1)) # 数据乘 2, 操作次数加 1, 然后添加到队尾
            checked.add(x * 2) # 数据乘 2 后, 进入集合中
        x, i = Py_queue.get() # 将数据和操作次数出队
    return i # 返回函数值 i (操作次数)
if __name__ == "__main__":
    print(change(5, 8))
```

2. 可通过  $head$  和  $tail$  变量的比较来实现队列, 如  $head$  和  $tail$  相等时, 就表示队列空。

参考程序如下:

```
import numpy as np
n = 4
q = np.zeros((n), dtype=np.int32)
print(q)
head = 0; tail = 0;
```

```

def queue(op, para):
    global q, head, tail
    if op== 'out':
        if head== tail:
            return('empty! ')
        else:
            para= q[head% n]
            head= head+ 1
            return(para)
    elif op== 'in':
        if tail- head== 4:
            print('full! ')
            return()
        else:
            q[tail% n]= para
            tail= tail+ 1
            return()
    else:
        print('invalid operator')
        return()
x= 'x'
h= 0
test= [1, 2, x, 4, x, x, x, 2, x, x, 3, 5, 1, x, 3, 5, 7, x]
while test:
    e= test.pop(0)
    if e== x:
        print(queue('out', h)),
    else:
        queue('in', e)
print('')
while not head== tail:
    print(q[head% n]),
    head= head+ 1
exit()

3. 略。

```

## 四、核心概念介绍

队列的概念：队列(queue)是一种特殊的线性表，其插入操作在表的一端进行，而删除操作在表的另一端进行。插入端称为队尾(tail)，删除端称为队头(head)。

举例：假设一个队列里存放了4个数据，数据A位于队头，数据D位于队尾，要想取出数据C，首先要将数据A出队，然后将数据B出队，最后才能将需要的数据C出队。整个过程如图3-16所示。

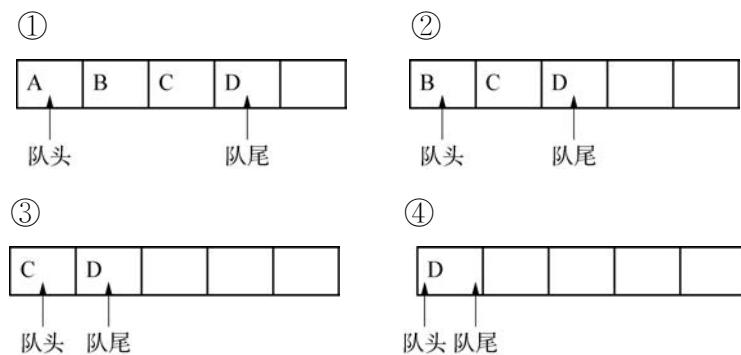


图3-16 队列数据操作过程

## 五、教学参考资源

### 1. 队列的抽象数据类型

事实上，队列的抽象数据类型定义可以如下所示：

ADT 队列(Queue)

(1) 数据对象。

数据对象集合为 $\{a_1, a_2, \dots, a_n\}$ ，每个元素的类型相同。

(2) 数据关系。

数据元素之间的关系是一对一的关系。其中除第一个元素 $a_1$ 外，每个元素有且只有一个直接前驱元素，除最后一个元素 $a_n$ 外，每个元素有且只有一个直接后继元素。

(3) 数据操作(见表3-18)。

表3-18 队列数据操作方法

方法	描述
Queue()	创建一个队列
Enqueue(x)	插入新元素x到队尾
Dequeue()	取出队头元素，并将队头元素删除
IsEmpty()	判断队列是否为空

方法	描述
Length()	获取队列当前的长度
Clear()	清空一个队列
Delete()	释放队列空间

endADT

## 2. 队列的顺序存储和链式存储

队列是一种线性表,因此可以采用顺序存储结构或链式存储结构来实现。

### (1) 顺序存储和实现。

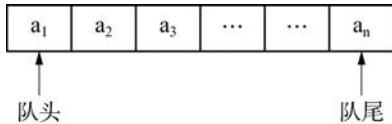


图 3-17 顺序队列

按顺序存储思想来实现的队列,称为顺序队列,如图 3-17 所示。

在 Python 中,可以使用列表来模拟队列,但在出队时,从队头删除元素会涉及列表中大量后继元素的移动,效率较低。因此一般采用 Python 自带的标准队列库——queue,使用时先通过 import queue 载入队列库,再通过 que = queue.Queue() 命令创建一个空队列。Python 队列库提供了如表 3-19 所示的一些常用方法。

表 3-19 队列库常用方法

queue 的方法	描述
put(x)	入队,向队列右边添加一个元素 x
get()	出队,删除并返回队列左边一个元素
empty()	判断队列是否为空
full()	判断队列是否为满
qsize()	统计队列中元素的个数

在 Python 的 IDLE 解释器环境下,运行 help(queue),可查看其他操作。

### (2) 链式存储和实现。

队列只能采用队头删除、队尾插入的操作,所以只需构建一个单链表就能实现。可以把单链表的首节点作为队头,单链表的尾节点作为队尾。通常这种队列也可称为“链队”,见图 3-18。可以使用链表实现“链队”。

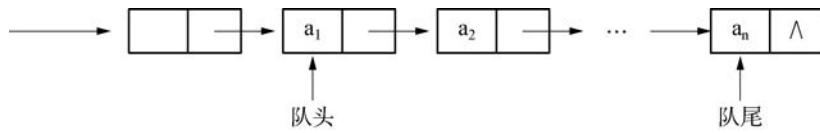


图 3-18 链队

通常情况下,如果队列的使用过程中元素变化不大,可以采用顺序队列;队列元素变化不可预料,则使用链队。

——参考《大话数据结构》,程杰,清华大学出版社

## 六、教学参考案例

### ■ 参考案例

#### 队列初探

上海市控江中学 覃 怡

(1课时)

##### 1. 学科核心素养

通过对解决篮球队成员选拔问题的思考和分析,感知生活实际问题中数据所承载的信息,并能根据解决问题的需求,寻找恰当的方式处理信息;在合作解决问题的过程中,愿意与团队成员共享信息,实现信息的最大价值。(信息意识)

在解决篮球队成员选拔问题的过程中,采用计算机可以处理的方式界定问题、抽象特征、建立结构模型、合理组织数据。(计算思维)

在用计算机解决篮球队成员选拔问题的过程中,能运用数字化学习资源与学习工具开展自主学习、协同工作、知识分享与创新创造。(数字化学习与创新)

从篮球队成员选拔的生活情境出发,进一步思考生活中用队列处理数据的过程,具有积极的学习态度和理性判断的能力。(信息社会责任)

##### 2.《课程标准》要求

结合生活实际,认识数据结构在解决问题过程中的重要作用。

通过问题解决,能对简单数据问题进行分析,选择恰当的数据结构。

通过问题解决,理解队列的概念和基本操作,并编程实现。

##### 3. 学业要求

学生能够运用生活中的实例描述数据的内涵与外延,能够将有限制条件的、复杂生活情境中的关系进行抽象,用数据结构表达数据的逻辑关系。

能够解释程序中数据的组织形式,描述数据的逻辑结构及其操作,评判其中数据结构运用的合理性;能够针对限定条件的实际问题进行数据抽象,运用数据结构合理组织、存储数据,选择合适的算法编程实现、解决问题。

能够分析数据与社会各领域间的关系,自觉遵守相应的伦理道德和法律法规。

##### 4. 教学内容分析

数据结构是计算机存储、组织数据的方式,是相互之间存在一种或多种特定关系的数据元素的集合。通常情况下,选择合理的数据结构,配合算法,可以使得程序的运行更快,并提升数据在计算机中存储和处理的效率。队列是教科书第三章中的内容,该章介绍了线性表、栈和队列三种数据结构。教科书中从队列的概念、实现与应用展开,本节课是队

列这一节的第 1 课时,从生活情境入手,认识队列的概念及其特征,理解队列的结构特点及操作方法,并运用队列来解决篮球队成员的选拔问题。

### 5. 学情分析

学生已经完成了必修 1 数据与计算课程的学习,自主选择学习选择性必修 1 数据与数据结构课程。学生已经具有较强的逻辑思维能力,并具备了一定的使用 Python 语言编写程序的基础。通过对本课程前面内容的学习,已经理解线性表、栈的概念、特征及其在计算机中的实现。

### 6. 教学目标

- 从生活中的排队、篮球队成员的选拔,感知生活中数据所承载的信息,采用计算机可以处理的方式抽象特征,知道队列及其特点,用队列合理组织数据。
- 通过比较队列和栈的结构特点、比较队列的入队(出队)操作和栈的入栈(出栈)操作,理解队列的操作方法。
- 在对篮球队成员选拔问题的解决过程中,从分析问题到解决问题,掌握队列在实际问题中的应用。
- 在用计算机处理篮球队成员身高数据的过程中,愿意与同伴互助,共享信息,合作解决问题,理解栈与队列的异同之处。
- 在解决问题的过程中,思考用队列来处理生活中的问题,能以积极学习的态度和理性判断的能力负责任地使用信息。

### 7. 教学重难点

- 教学重点:理解队列的概念和结构特点。
- 教学难点:掌握队列的基本操作。

### 8. 教学策略分析

本节课主要采用实践体验、小组协作、讨论交流的方式进行教学。整节课进一步深入对队列的学习,通过解决问题感受队列特征及其应用。

#### (1) 从栈到队列,感受队列和栈的异同。

本课通过三处比较——比较队列和栈的结构特点、比较队列和栈的基本操作、比较利用队列和栈解决同一问题,感受队列和栈的异同点,让学生能更好地理解队列的概念和操作方法。

#### (2) 设计问题链,提供学习材料,由浅入深层层破解。

本课以情境为主线,通过由三个问题组成的问题链,引导学生由浅入深地掌握队列的概念、结构特点及其操作方法。借助学习单及提供的知识链接,培养学生自主学习能力。

### 9. 教学过程设计(见图 3-19 和表 3-20)

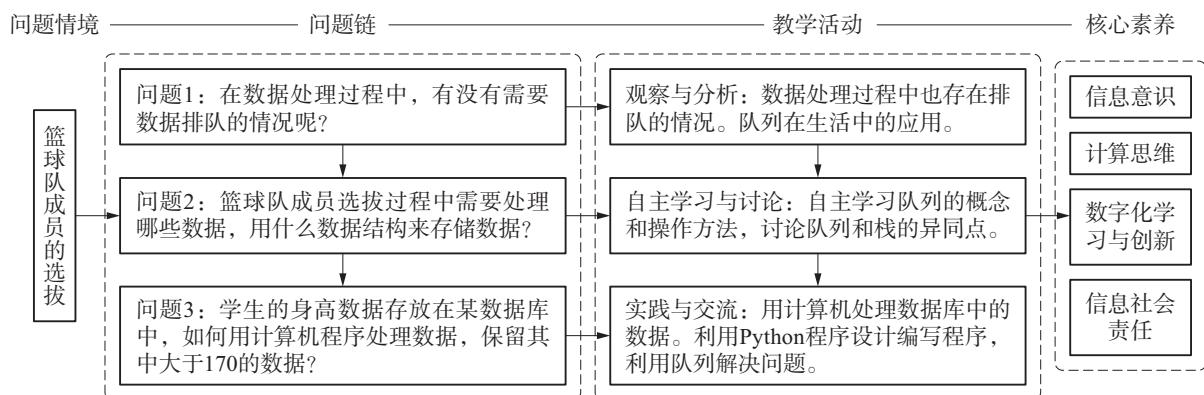


图 3-19 教学过程

表 3-20 教学过程设计表

教学环节	教师活动	学生活动	设计意图
情境导入	<p>篮球队成员选拔</p> <p>问题情境：新学期，学校将为校篮球队选拔新成员。篮球队成员的基本要求之一是：身高大于170cm(注意不包含170cm)。体育课上，有意向参加篮球队成员选拔的学生排队测量身高，根据身高要求进行第一轮选拔。</p> <p>问题1：生活中有很多类似情况需要排队。在数据处理的过程中，有没有需要数据排队的情况呢？</p>	<p><b>【观察、分析】</b></p> <p>数据处理过程中也存在排队的情况。（预设答案：选课系统、车牌拍卖、数据筛选……）</p>	<p>从生活中的排队、篮球队成员的选拔，感知生活中数据所承载的信息；初步感知队列在数据处理中的作用</p>
认识队列	<p>队列的概念、结构特点</p> <p>问题2：篮球队成员选拔过程中需要处理哪些数据，用什么数据结构来存储数据？</p> <p>(1) 生活中排队人流的运动方向是怎样的？ 数据处理过程中，数据如何实现排队呢？</p> <p>(2) 阅读教科书中“一、什么是队列”，提供自主学习材料“知识链接1”。</p> <p>(3) 提问：队列是不是线性表？队列的插入、删除操作遵循什么原则？</p> <p>(4) 提问：队列和栈有哪些异同点？</p> <p>(5) 提问：篮球队成员选拔过程中，有哪些数据需要用队列进行存储？</p>	<p><b>【自主学习、讨论】</b></p> <p>(1) 分析生活中的排队人流运动方向。（预设答案：从队尾入队，从队头出队） 思考：数据排队也与此类型。</p> <p>(2) 自主学习教科书相关内容和学习单中的“知识链接1”。</p> <p>(3) 小组交流讨论。（预设答案：队列是线性表；队列的插入在队尾、删除在队头，队列遵循“先进先出”的原则）</p> <p>(4) 小组交流讨论。（预设答案：队列和栈都是线性表，队列遵循先进先出原则，栈遵循后进先出原则）</p> <p>(5) 小组交流和分享在篮球队成员选拔过程中所需要处理的数据</p>	<p>思考和讨论采用计算机可以处理的方式抽象特征；通过自主学习，知道队列及其特点；交流讨论队列和栈的异同，深入理解队列的操作方法；在此过程中与同伴互助，共享信息</p>

教学环节	教师活动	学生活动	设计意图
队列的实例	<p>队列的实现 问题3:学生的身高数据存放在某数据库中,如何用计算机程序处理这些数据,保留其中大于170的数据?</p> <p>(1) 讨论:队列中存储什么数据?如何保留大于170的数据?哪些数据入队?哪些数据出队? 建议:可以绘制流程图,并与同学分享。</p> <p>(2) 提供自主学习材料“知识链接2”,请学生完善程序,并上机调试</p>	<p>【讨论、实践】 (1) 小组交流,绘制流程图:取出队列中的一个数据。判断其是否大于170,若大于170则入队。 (2) 展示与分享解决方法。 (3) 完善程序,并上机调试</p>	在对篮球队成员选拔问题的解决过程中,从分析问题到解决问题,掌握队列在实际问题中的应用。在解决问题的过程中,与同伴互助,共享信息,合作解决问题
课后思考	<p>思考:用线性表、栈能不能解决这个问题?如果用线性表、栈能解决,那么和用队列相比有没有差异?</p> <p>拓展学习(可选):请利用相关书籍或者网络,收集和学习栈的应用(可以收集一个栈的应用实例)</p>		通过使用不同数据结构,掌握不同数据结构的异同点,更好地理解队列的结构特点

## 附:“队列初探”学习单

新学期,学校将为校篮球队选拔新成员。篮球队成员的基本要求之一是:身高大于170 cm(注意不包含170 cm)。体育课上,有意向参加篮球队成员选拔的学生排队测量身高,根据身高要求进行第一轮选拔。

【问题1】生活中有很多类似情况需要排队。在数据处理的过程中,有没有需要数据排队的情况呢?

---



---

【问题2】篮球队成员选拔过程中需要处理哪些数据,用什么数据结构来存储数据?

---



---

### • 知识链接1:队列的概念

队列的数据对象集合( $a_1, a_2, a_3, \dots, a_n$ )中,每个元素的类型是相同的。队列中数据元素之间的关系是一对一的关系。其中,除了第一个元素  $a_1$  外,每个元素有且只有一个直接前驱元素,除了最后一个元素  $a_n$  外,每个元素有且只有一个直接后继元素。

队列的插入操作在一端进行,删除操作在另一端进行。插入端称为队尾,删除端称为队头。向队列中插入新元素称为进队或入队,新元素进队后就成为新的队尾元素;从队列中删除元素称为出队或离队,元素出队后,其后继元素就成为队头元素。

阅读“知识链接 1”，请回答以下问题：

(1) 队列是不是线性表？\_\_\_\_\_。

(2) 队列的插入、删除操作遵循什么原则？\_\_\_\_\_。

(3) 请在“知识链接 1”的第二段中划出关键名词。

(4) 交流与分析队列和栈的异同点：

\_\_\_\_\_。

**【问题 3】**学生的身高数据存放在某数据库中，如何用计算机程序处理这些数据，保留其中大于 170 的数据？

- **知识链接 2：队列的实现与操作**

(1) 队列可采用顺序存储结构、链式存储结构来实现。

(2) 队列的顺序存储结构，Python 中自带标准队列库 queue 和标准双端队列库 deque(见表 3-21)。

表 3-21 标准双端队列库 deque 的一些常用操作方法

方法	描述
from collections import deque	头函数
deque()	创建队列
append()	入队，在队列右边添加一个元素
extend()	入队，在队列右边添加多个元素
popleft()	出队，返回队列左边的第一个元素，并在队列中删除该元素
clear()	清空队列
copy()	复制队列
len()	返回队列中元素的个数

完善以下程序：某数据库中存放了 100 位学生的身高数据，保留并输出其中大于 170 的数据。

```
from collections import deque
import random
q= deque()
for i in range(100):
    q.append(random.randint(160,190))
# 通过随机函数产生 100 位学生的身高数据
print('原始数据：')
print(q)
for i in range(100):
```

\* 请在上述区域填写程序 \*

```
print('筛选后数据：')
print(q)
```

# 常用数据结构

### 一、本章学科核心素养的渗透

本章介绍了二叉树、图、哈希表三种数据结构，在将有限制条件的、复杂生活情境中的关系进行抽象，用数据结构表达数据的逻辑关系过程中体现信息意识。

本章是选择性必修 1 数据与数据结构模块的重点内容。《课程标准》中相关内容要求包括：

1.6 通过列举实例，认识到抽象数据类型对数据处理的重要性，理解抽象数据类型的概念，了解二叉树的概念及其基本操作方法。

本章的项目以完成“心算游戏好帮手”并服务生活为目标渗透了社会责任意识，本章提供的思考体验和探究活动引导逐步掌握二叉树的特性。在特性理解基础上，学会合理选择二叉树来解决实际问题，落实《课程标准》要求。

在二叉树一节中，以支持算法的高效实现为出发点，教科书中通过体验思考和探究活动进一步阐析二叉树的概念及其基本操作方法。以同学们所熟悉的集合作为二叉树的一种表现形式来进行构建，其中包含了数据之间的逻辑关系。

在二叉树实现部分，教科书以二维数组的方式实现二叉树，通过相应的设置如用链表方法实现逻辑关系，体现出物理的顺序存储和逻辑结构之间的灵活性，在具体应用时教科书中以类的方式构建二叉树的节点，关注问题解决过程中的数据抽象，通过列举实例认识到抽象数据类型对数据处理的重要性，理解抽象数据类型的概念。

整章设置了一定数量的编程实践活动，在对复杂问题（如表达式二叉树、最短路径求解）的解决中体会数据结构对算法的支持，引导学生根据需要通过建立数据模型、抽象数据、合理选择二叉树等数据结构、算法实现、上机调试来解决生活中的真实问题，从而进行计算思维培养。

## 二、本章知识结构

本章遵循《课程标准》，依据学分和课时规定，紧扣学科概念体系，将内容分为三节。

第一节二叉树，主要包括二叉树的概念、实现、基本操作和应用。

第二节图，主要包括图的概念、实现和应用。

第三节哈希表，主要包括哈希表的概念、实现和基本操作。

本章开始，数据结构从一对一线性关系进入到一对多、多对多关系。内容更抽象了，因此在教学中建议多举一些实例，多利用画图来演示问题解决过程中的每一个基本操作。其中二叉树是一种一对多的结构，也是本章的一个学习重点。图和哈希表可以作为选学内容。图结构作为自然现象的一种抽象表示，有着广泛的应用，对于这一节的教学，首先应重点关注如何将一些现实中的问题抽象成图结构，然后在计算机中表示出来，再进行编程实现。对于哈希表，可在理解其高效搜索原理的基础后，尝试编程实现及解决实际问题。在完成本章的项目实践和作业练习过程中体会数据结构与算法相辅相成的关系。

## 三、本章项目活动设计思路

从学生生活实际出发，以“心算游戏好帮手”作为主题，引起学生兴趣。这其中涉及二叉树的一些思想、方法和主要知识点。注意先让学生思考讨论，之后教师引导，总结出规律后再编程实现，这其中的实现方法和思想应该是重点，特别是对于根节点的处理方法，可以通过图解加模拟过程的方式帮助学生理解掌握。在完成项目任务过程中学习常见的数据结构，参与基于真实问题的项目活动，经历建立数据模型、抽象数据、选择数据结构、算法实现、上机调试、问题解决的全过程。

在编程实现过程中逐步理解二叉树、图、哈希表三种结构，掌握它们的基本操作。通过案例学习、实践活动等环节掌握如何运用这些结构来解决实际问题。在实践过程中鼓励团队协作，在完成挑战性强的核心活动中，提高创新精神和实践能力。

## 四、本章课时安排建议

本章教学建议用8课时完成，具体参见表4-1。

表4-1 课时安排计划表

节名	建议课时
第一节 二叉树	4
第二节 图	2
第三节 哈希表	2

### 一、教学目标与重点

#### 教学目标:

- 通过体验思考和原理分析,理解二叉树的含义、基本操作及应用场景。
- 在探究活动和编程实践过程中,掌握实现二叉树的一种方法,了解不同实现方法的差异,感悟运用二叉树解决问题的高效性。
- 在项目实践的过程中,能够运用二叉树解决实际问题,发展计算思维,提高数字化学习与创新能力。

#### 教学重点:

运用二叉树结构解决实际问题。

### 二、教学实施与评价

#### 1. 教学活动建议

在栈这一节,我们感觉到在程序运行中数据之间自然存在一些关系,根据这种关系,通过数据抽象,可建立相应的数据结构。事实上,为了解决问题的需要,还可以人为构造一些数据结构,将这些数据结构和程序配合使用。通常,构造这些数据结构的目的是使程序运行更高效。二叉树就是一种人为构造的数据结构。

二叉树是一种逻辑关系分层次的非线性结构。在教学过程中,教师可提供一些生活中的实例如家族图谱、文件系统树、生物分类树、超市的商品管理图、图书馆的藏书归类图等,引导学生讨论、分析数据元素,并针对需要解决的问题进行不同层次的分析,感受数据抽象在问题解决中的作用。对于二叉树的概念和表示形式,注意在理解数据之间的逻辑关系后,引导学生尝试用集合来表示二叉树。本节涉及的概念比较多,如满二叉树、完美二叉树等,建议在图示和活动的过程中介绍这部分内容。

教科书在二叉树之前介绍的几种数据结构都是一对一的线性结构,二叉树是一种一对多的非线性结构。对于二叉树的实现,教科书中以数组这种顺序存储的线性表来实现,这其中的实现方法和操作过程需让学生理解和掌握,这也体现出数据结构的灵活性和构建方法的多样性。

在二叉树的基本操作中,先结合案例掌握一种先序遍历,在课堂中再分组讨论实现中序和后序遍历,揭示各种遍历的特点,如通过中序遍历可以对一些特殊的二叉树(二叉搜

索树)实现数据的有序排列。为了能更好地理清二叉树中的数据之间的关系,建议以探究原理方式展开。比如在顺序存储方式下,对二叉树中某个位置增加或删除节点,会使数组中其他元素移动,效率会比较低,因此对二叉树一般采用链式存储的方式。

“心算游戏好帮手”项目具体实施时,需要对项目进行分解,讨论分析项目中的一些关键信息。要把一个表达式翻译成正确求值的机器指令序列,或者直接对表达式求值,首先要能正确解释表达式。任何一个表达式都是由运算符、运算数、基本界限符组成的。人们在书写表达式时通常采用“中缀”表达形式,也就是将运算符放在两个运算数中间,用这种“中缀”形式表示的表达式称为中缀表达式。组织学生探究运算符和运算数之间一对多的关系,通过数据抽象,建立求值的数学模型。一般地,运算数既可以是常数,也可以是被说明为变量或常量的标识符;运算符可以分为算术运算符、关系运算符和逻辑运算符三类;基本界限符有左右括号和表达式结束符等,这里我们仅仅讨论简单算术表达式的求值问题。这种表达式只包括加、减、乘、除四种运算符。采用表达式求值的方式可以将过程记录下来,比较哪一种数据结构更适合组织表达式中的逻辑关系,结合二叉树的思想和基本操作进行对比分析后,为了方便存储,构造一棵“中缀”表达式二叉树。最后,只要后序遍历这棵“中缀”表达式二叉树,就能将其转化成后缀表达式从而实现计算。进行编程实现,对于树节点的处理可以采用类的方式来实现。

## 2. 教学过程安排(见表 4-2)

表 4-2 教学过程设计表

课时	教学环节	教师活动	学生活动
第 1 课时 (概念和 表示方式)	1. 导入	回顾“数据流”的案例,引出二叉树	倾听、思考
	2. 二叉树的概念	引导阅读教科书,寻找二叉树数据之间的关系及二叉树定义	阅读、思考
	3. 二叉树的表示	讲解演示:用集合表示法描述二叉树	倾听、思考
	4. 体验思考	布置任务:本节体验思考 1	思考、分享
	5. 二叉树的各种类型	引导学生一起分析、归纳各类二叉树	分析、归纳
	6. 探究活动	布置任务:本节探究活动 1	讨论、交流
	7. 作业练习	布置作业:本节作业练习 1	完成方案
	8. 总结	概念和表示形式	
第 2 ~ 3 课时 (实现、遍 历)	1. 导入	回忆二叉树的形式	倾听、思考
	2. 二叉树表示	讲解用数组表示二叉树	思考分析,编程实现
	3. 体验思考	布置任务:本节体验思考 2	思考、分享
	4. 二叉树实现	布置活动:引导编程实现二叉树	思考分析,编程实现
	5. 递归	讲解递归的概念和形式	倾听、思考
	6. 二叉树遍历	讲解三种遍历,引导编程	思考,编程实现

课时	教学环节	教师活动	学生活动
第4课时 (应用)	7. 探究活动	本节探究活动2	交流讨论
	8. 作业	本节作业练习2、作业练习3	分析、实现
	9. 总结	(1) 概念和实现。 (2) 二叉树的遍历方式	
	1. 导入	二叉树在计算机中的应用	倾听、思考
		布置活动：“表达式二叉树”的要求	分析、编程
	2. 活动：“表达式二叉树”	思路分析：关键步骤讲解	倾听、思考
		图解过程：逐步绘图构建二叉树	倾听、填写
		巡视，指导	编程实现
	3. 项目实践	安排项目实践：本节项目实践	完成方案、编程实现
	4. 作业练习	布置作业：本节作业练习4	编程实现
	5. 总结	二叉树结构的应用场景、程序设计中二叉树结构的配合使用	

在完成各项目活动的过程中观察学生对于二叉树的理解和应用。可以对学生在以下一些方面的表现进行评价：能用集合的形式表示二叉树；能描述满二叉树、完全二叉树、完美二叉树及其特性；能用二维数组描述数据之间的关系，进而根据二叉树数据结构进行编程存储数据；会进行二叉树的三种遍历；能够针对表达式计算问题进行数据抽象，运用二叉树合理组织、存储数据，利用递归的思想编程以解决实际问题。在项目活动中，考查学生运用二叉树解决问题的能力。

### 3. 思考探究提示

(1) 本节第1个体验思考参考答案：先确定根节点，之后按{父节点,{左子节点},{右子节点}}表示。

```
{a, {d, {}, {}}, {b, {}, {c, {}, {}}}} }  

{a, {b, {c, {}, {}}, {d, {}, {}}}, {}}, {e, {f, {}, {}}, {g, {}, {}}} } }  

{a, {b, {c, {}, {}}, {d, {}, {}}}, {e, {f, {}, {}}, {g, {h, {}, {}}, {i, {}, {}}}}} } }
```

(2) 本节第2个体验思考参考答案见表4-3与表4-4。

表4-3 参考答案1

索引	数据	左链	右链
0	a	3	6
1	c	-1	5
2	f	-1	-1
3	b	1	-1
4	g	-1	-1
5	d	-1	-1
6	e	2	4

表4-4 参考答案2

索引	数据	左链	右链
0	f	-1	-1
1	g	-1	-1
2	c	-1	3
3	d	-1	-1
4	e	0	1
5	b	2	-1
6	a	5	4

(3) 本节第 1 个探究活动参考答案:完美二叉树有一些非常有用的特性,如表 4-5 所示。

表 4-5 完美二叉树特性

性质 1	在非空二叉树第 $i$ 层中最多有 $2^i$ 个节点
性质 2	高度为 $h$ 的二叉树最多有 $2^{(h+1)} - 1$ 个节点,也就是说要实现 $n$ 个数据的存储只需构造一棵高度为 $\log_2(n+1) - 1$ 的二叉树即可,从这里也可以看到高度与节点个数成对数关系的特点

对于给定的一个顺序编号  $k$ ,其与层序编号  $i$  之间的关系为: $k = 2^i + j - 1$ 。反过来的关系为  $i = \text{取整}(\log_2(k+1+1)) - 1$ ,  $j = k + 1 - 2^{\text{取整}(\log_2(k+1+1))}$ 。

(4) 本节第 2 个探究活动参考答案:

```
class TreeNode:
    def __init__(self, data, left=None, right=None):
        self.data = data
        self.left = left
        self.right = right

class BinaryTree:
    def __init__(self):
        self.head = None

    def create(self, triple): # 从一个集合定义(中根序)构建一棵二叉树,由于序的问题,实际用元组实现
        root = None
        if triple: # not empty
            root = TreeNode(triple[0], None, None)
            if self.head == None:
                self.head = root
            print(root.data, end=' ')
            root.left = self.create(triple[1])
            root.right = self.create(triple[2])
        return root

    def preroot_traversal(self, root):
        if root == None:
            return
        else:
            print(root.data, end=' ')
            self.preroot_traversal(root.left)
```

```

        self.preroot_traversal(root.right)
    return

def midroot_traversal(self, root):
    if root == None:
        return
    else:
        self.midroot_traversal(root.left)
        print(root.data, end=' ')
        self.midroot_traversal(root.right)
    return

def postroot_traversal(self, root):
    if root == None:
        return
    else:
        self.postroot_traversal(root.left)
        self.postroot_traversal(root.right)
        print(root.data, end=' ')
    return

t=BinaryTree()
tree= (1, (2, (3, (), (4, (), ())), (), (5, (6, (), ()), (7, (), ()))))
t.create(tree)
print()
t.preroot_traversal(t.head)
print()
t.midroot_traversal(t.head)
print()
t.postroot_traversal(t.head)
print()

```

#### 4. 项目实施提示

先分析项目任务 1 中的问题,分组讨论各种评判方案,尝试使用手机上的计算器,看看能否解决这一问题。对于项目任务 2,需根据讨论结果,引出设计一个计算机程序解决评判问题。根据所学知识,可以采用表达式二叉树来实现。在完成项目任务 3 的过程中,先读取文件中的表达式,然后对表达式采用二叉树的结构来存储,最后通过树的合并操作完成表达式的计算。具体程序代码如下:

```

class TreeNode():

    def __init__(self, data, left= None, right= None):
        self.data= data

```

```

        self.left= left
        self.right= right
    def CreateTree(expre):
        root= None
        for c in expre:
            node= TreeNode(c)
            if root== None:
                root= node
            elif root.data.isdigit(): # 如果根节点是数字
                node.left= root
                root= node
            elif node.data.isdigit(): # 如果当前节点是数字
                tempNode= root # 当前节点沿右路插入最右边成为右子节点
                while (tempNode.right != None):
                    tempNode= tempNode.right
                tempNode.right= node
            else:
                if (GetPriority(node.data) <= GetPriority(root.data)):
                    node.left= root
                    root= node
                else:
                    node.left= root.right
                    root.right= node
        return root
    def calculate(root): # 后序遍历函数求值
        if root== None:
            return
        else:
            calculate(root.left)
            calculate(root.right)
            print(root.data, end= " ")
            if root.data! = "+" and root.data! = "-" and root.data! = "* "
and root.data! = "/":
                stackcal.append(int(root.data))
            else:
                x= stackcal.pop()
                y= stackcal.pop()

```

```

result= 0
if root.data== "+":
    result= y+ x
elif root.data== "-":
    result= y- x
elif root.data== "* ":
    result= y* x
elif root.data== "/":
    result= y/x
stackcal.append(result)

def GetPriority(c):
    if c in ["+", "-"]:
        return 0
    if c in ["*", "/"]:
        return 1

if __name__ == "__main__":
    # 主程序段
    expression= "2 + 3 × 4 - 5 + 1 × 6 × 7 - 9"
    tree= CreateTree(expression)
    stackcal= list()
    calculate(tree)
    print(stackcal)

```

### 三、作业练习与提示

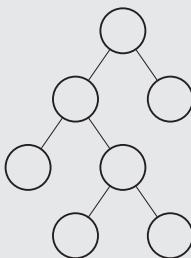
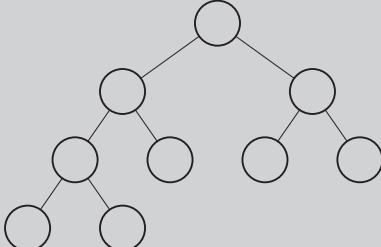
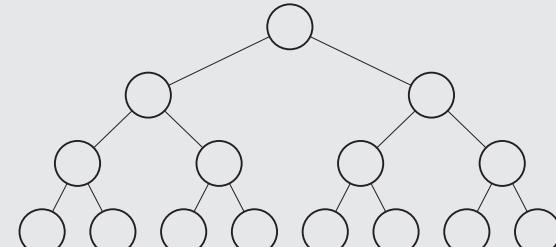
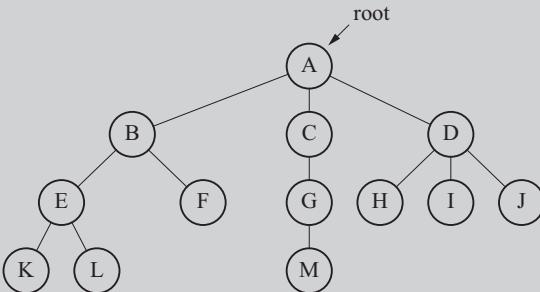
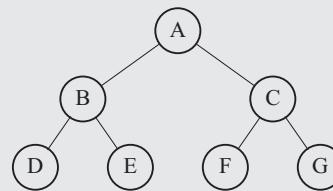
#### ■ 题目描述

1. 本章第一节中提到了关于二叉树种类或特征性质的一些术语概念,如满二叉树、完全二叉树、高度、叶子节点等,试做归纳总结。
2. 针对一个二叉树具体例子,做三种遍历。
3. 写出二叉树的中序和后序遍历程序。
4. 完善表达式求值程序(允许多位整数,允许括号和负数,能判断除数为 0 的错误)。

#### ■ 作业练习参考答案

1. 答案见表 4-6。

表 4-6 二叉树种类及特征性质

名称	描述	图示
满二叉树	除了叶子节点之外的每一个节点都有两个子节点	
完全二叉树	除了最后一层之外的其他每一层都被完全填充，并且所有节点都保持向左对齐	
完美二叉树	除了叶子节点之外的每一个节点都有两个子节点，每一层都被完全填充	
高度	从根节点到树中某节点所经路径上的边数称为该节点的层次，根节点为第0层，依次往下递增。如右图中M的层次为3。所有节点中层次最大的值称为树的高度。如右图中树的高度是3	
叶子节点	没有子节点的节点，如右图中D、E、F、G	

归纳：

完美二叉树一定是完全二叉树，但完全二叉树不一定是完美二叉树。

完美二叉树一定是满二叉树，但满二叉树不一定是完美二叉树。

完全二叉树可能是满二叉树,满二叉树也可能是完全二叉树。

既是完全二叉树又是满二叉树也不一定就是完美二叉树。

2. 遍历二叉树就是按照一定的顺序访问二叉树的所有节点,使得每个节点被访问一次,而且只访问一次的过程。遍历是最基本的运算,是树中所有其他运算的基础。

根据对于根节点的访问次序,二叉树遍历分为先序、中序、后序。先序是先访问根节点,然后访问左子树,左子树访问完后,再访问右子树,依次遍历。中序是先访问左子树,然后访问根节点,最后访问右子树,依次遍历。后序是先访问左子树,再访问右子树,最后访问根节点,依次遍历。

对教科书第 80 页图 4.1(d)中所示二叉树的各种遍历结果如表 4-7 所示。

表 4-7 二叉树遍历结果

先序遍历	中序遍历	后序遍历
abcdefg	cdbafeg	dcbfgea

3. 编程对教科书第 80 页图 4.1(d)实现三种遍历。事实上:先序、中序、后序遍历是一种深度优先遍历,遍历时先顺着一条路径尽可能向前探索,直到检查完叶子节点,由于无路可走,只能回头到上一个节点(这种回头的过程也称为回溯),再继续沿另一路径向前探索,如此反复,直到所有节点都访问到。

对于先序、中序、后序遍历,因为是顺着路径不断探索、不断回头(碰到叶子节点就回溯)的过程,先访问到的会压在下面,所以可以通过入栈、出栈的方法,来模拟各种遍历的过程。

以先序为例,分析入栈、出栈的过程,如表 4-8 所示。

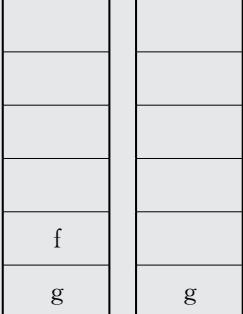
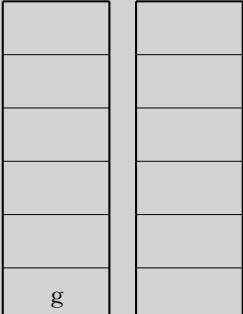
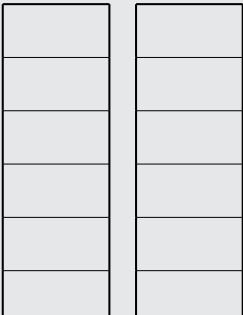
表 4-8 入栈、出栈过程

步骤	内容	图示	遍历得到元素
1	先创建一个空栈		无

续表

步骤	内容	图示	遍历得到元素
2	先将根节点 a 入栈,之后出栈		a
3	刚出栈的节点的右子节点为 e,将 e 入栈;左子节点为 b,将 b 入栈,之后出栈		b
4	刚出栈的节点无右子节点,将其左子节点 c 入栈,之后出栈		c
5	将刚出栈的节点的右子节点 d 入栈,该节点无左子节点,将右子节点 d 出栈		d
6	刚出栈的节点无右子节点,无左子节点,没有数据入栈。出栈一个栈内数据		e

续表

步骤	内容	图示	遍历得到元素
7	将刚出栈的节点的右子节点 g 入栈,再将其左子节点 f 入栈之后出栈		f
8	刚出栈的节点无右子节点,无左子节点,没有数据入栈,出栈一个栈内数据		g
9	刚出栈的节点无右子节点,无左子节点,没有数据入栈,栈空,遍历结束		无

参考程序如下：

```
import numpy as np
import time
memory= np.zeros((100,3),dtype= np.int32)      # 100 行 3 列数组
expr= "{a,{b,{c,{},{}},{}},{}},{e,{f,{},{}},{g,{},{}},{}},{}"
change= list()
stack= list()
p= 0
for i in range(len(expr)-1):
    pos= expr[i]                      # 读取当前字符
```

```

nex= expre[i+1]          # 读取下一个字符
if pos== "{}":
    if nex != "}":
        memory[p][0]= p      # 生成节点
        stack.append(memory[p]) # 节点入栈
        p= p+1                # 指针位置下移
        change.append(nex)
    else:
        if stack[-1][1]== 0:
            stack[-1][1]= -1      # 将-1赋值给栈顶元素的左链
        else:
            stack[-1][2]= -1      # 将-1赋值给栈顶元素的右链
elif pos== "}":
    pre= expre[i-1]          # 读取前一个字符
    if pre != "{":
        aa= stack.pop()       # 出栈一个节点元素
        if stack[-1][1]== 0:
            stack[-1][1]= aa[0]  # 将出栈元素地址给父节点左链
        else:
            stack[-1][2]= aa[0]  # 将出栈元素地址给父节点右链
for k in range(p):
    print(change[k],memory[k][1],memory[k][2])

stack1= list()  # 建立一个栈,存放各节点
stack1.append(memory[0])
while stack1:
    temp= stack1.pop()  # 出栈
    print(change[temp[0]],end= " ")
    if temp[2]!= -1:
        stack1.append(memory[temp[2]]) # 右子树入栈
    if temp[1]!= -1:
        stack1.append(memory[temp[1]]) # 左子树入栈
print()

stack1= list()  # 建立一个栈,存放各节点
stack1.append(memory[0])
result= []

```

```

while stack1:
    temp=stack1.pop() # 出栈
    result.append(change[temp[0]])
    if temp[1]! = - 1:
        stack1.append(memory[temp[1]]) # 左子树入栈
    if temp[2]! = - 1:
        stack1.append(memory[temp[2]]) # 右子树入栈
print(list(reversed(result)))

print()
stack2= list() # 建立一个栈,存放各节点
stack2.append(memory[0])
result= []
no= np.array([0,0,0])
stack2[0]= np.array([0,1,4])
while stack2:
    while memory[0][1]! = - 1 and not (memory[0]== no).all():
        stack2.append(memory[memory[0][1]]) # 左子树入栈
        memory[0]= memory[memory[0][1]] # 左子树作为下一次的父节点
        temp= stack2.pop() # 出栈一个数据
        result.append(change[temp[0]]) # 该数据已访问,保存到列表 result 中
    if temp[2]! = - 1:
        stack2.append(memory[temp[2]]) # 右子树入栈
        memory[0]= memory[temp[2]] # 右子树作为下一次的父节点
print(result)

```

4. 先整体对表达式进行分析,并对表达式进行切分。从表达式依次读取字符,每次读取一个。对于多位整数,可以用一个栈来切分,先从表达式读取一个字符进栈,然后读取表达式的字符并与栈顶元素进行比较,如果读取到的数和栈顶元素都是数字,就把栈顶元素出栈后和当前元素合并后再进栈,否则直接进栈。

如果读取到的是“-”,看当前栈顶元素如是“+”“-”“\*”“/”“(”元素的话,说明当前的“-”是负号,则需把它和下次进栈的数字合并进栈,否则直接进栈。

设置一个优先级层次函数(level),在读取字符串过程中,碰到“(”level 加一级,碰到“)”减一级。

参考程序如下:

```

class TreeNode():
    def __init__(self,data,left= None,right= None):
        self.data= data

```

```

        self.left= left
        self.right= right
    def CreateTree(expre):
        root= None
        slist= scut(expre)
        slevels= slevel(slist)
        i= 0
        for c in slist:
            if c not in [')', '(']:
                node= TreeNode(c)
                node1= slevels[i]
                if root== None:
                    root= node
                    root1= node1
                elif not root.data in ['+', '- ', '* ', '/ ', '(', ')']: # 如果根节点
                    是数字
                    node.left= root
                    root= node
                    root1= node1
                elif not node.data in ['+', '- ', '* ', '/ ', '(', ')']: # 如果当前节
                    点是数字
                    tempNode= root      # 当前节点沿右路插入最右边成为右子节点
                    while (tempNode.right != None):
                        tempNode= tempNode.right
                    tempNode.right= node
                else:
                    if node1== root1: # 先看括号里的优先级判断,对层次相同的情
                        况处理
                        if (GetPriority(node.data) <= GetPriority(root.
                            data)):
                            node.left= root
                            root= node
                            root1= node1
                        else:
                            node.left= root.right
                            root.right= node
                    elif node1< root1: # 当前层小于根的层次情况处理

```

```

        node.left= root
        root= node
        rootl= node1
    else:      # 当前层大于根的层次情况处理
        if root.right.data in ["+", "-","*","/"]:# 看是否为
层入口
        n= int(node1)- int(rootl)
        temproot= root      # 当前节点沿右路插入最右边成为右
子节点
        for k in range(n):
            temproot= temproot.right
            if not temproot.right.data in ["+", "-","*","/"]:
                node.left= temproot.right
                temproot.right= node
            else:
                if (GetPriority(node.data) < = GetPriority
(temproot.right.data)):
                    node.left= temproot.right
                    temproot.right= node
                else:
                    node.left= temproot.right.right
                    temproot.right.right= node
            else:
                node.left= root.right
                root.right= node
        i= i+ 1
    return root
def slevel(exp):      # 括号优先级设置
    level= 0
    levels= []
    for i in range(len(exp)):
        if exp[i]== '(':
            level += 1
        if exp[i]== ')':

```

```

        level -= 1
    levels.append(level)
return levels

def scut(s):      # 分割数字, 处理-号
    stack = []
    if s[0] == "-":
        stack.append(s[0] + s[1])
    else:
        stack.append(s[0])
        stack.append(s[1])
    for i in range(len(s) - 2):
        s1 = s[i + 2]
        if s1 == "(" and stack[-1] == "(":
            stack.append("1")
            stack.append("* ")
        if s1 not in ["+", "-", "*", "/", "(", ")"]:
            s2 = stack.pop()
            if s2 == "-" and stack[-1] in ["+", "-", "*", "/", "(", ")"]:
                stack.append("-" + s1)
            else:
                if s1.isdigit() and s2 not in ["+", "-", "*", "/", "(", ")"]:
                    stack.append(s2 + s1)
                else:
                    stack.append(s2)
                    stack.append(s1)
        else:
            stack.append(s1)
    return stack

def calculate(root):          # 后序遍历函数求值
    if root == None:
        return
    else:
        calculate(root.left)
        calculate(root.right)
        if root.data != "+" and root.data != "-" and root.data != "* " and root.data != "/":
            stackcal.append(int(root.data))

```

```

else:
    x= stackcal.pop()
    y= stackcal.pop()
    result= 0
    if root.data== "+":
        result= y+ x
    elif root.data== "-":
        result= y- x
    elif root.data== "*":
        result= y* x
    elif root.data== "/":
        if x== 0:
            flag.append("1")
            print("除数为零,表达式有问题")
        else:
            result= y/x
    stackcal.append(result)

def GetPriority(c):
    if c in ["+", "-"]:
        return 0
    if c in ["*", "/"]:
        return 1

if __name__ == "__main__":
    # expression= "4+ 1/(2* (3+ 2)* 2- (2+ 3))+ 2+ 3"
    expression= "- 5+ 3/((5- 2)* 3)+ 20"
    # expression= "(- 5+ 3)/ (2+ 4)+ 2* 3"
    tree= CreateTree(expression)
    flag= []
    stackcal= list()
    calculate(tree)
    if not flag:
        print("运算结果为", stackcal)

```

## 四、核心概念介绍

二叉树：

二叉树(binary tree)中每个节点的子节点最多只有两个,而且要明确左、右。左边的节点称为左子节点,右边的节点称为右子节点,当前节点则称为它们的父节点。每个子节点分别属于两棵不相交的二叉树,这两棵二叉树又分别称为以父节点为根的二叉树的左子树和右子树。从根节点到树中某节点所经路径上的边数称为该节点的层次,所有节点中层次最大的值称为二叉树的高度。图 4-1 中的根节点 A 为第 0 层,M 节点为第 3 层,此二叉树高度为 3。

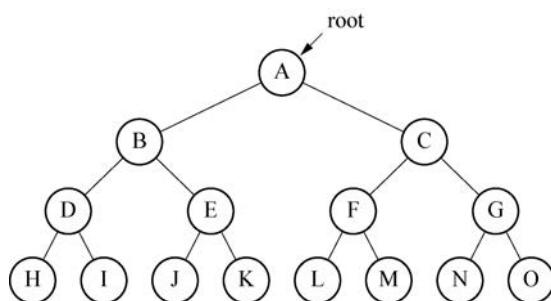


图 4-1 二叉树

有些形态的二叉树在应用中有特殊的意义。其中,如果二叉树的每个节点都有零个或两个子节点,则称这种二叉树为满二叉树,如图 4-2 所示。只有最下面两层节点的子节点数小于 2,并且最下面一层的节点都依次排列在该层最左边的位置上,这样的二叉树称为完全二叉树,如图 4-3 所示。教科书第五章中给出了完全二叉树的具体应用例子。

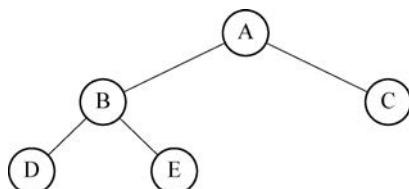


图 4-2 满二叉树

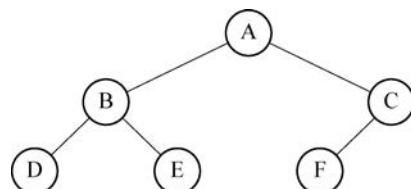


图 4-3 完全二叉树

二叉树有一些简单性质:在二叉树第  $i$  层中最多有  $2^i$  个节点;高度为  $h$  的二叉树最多有  $2^{h+1} - 1$  个节点。利用这些性质,在应用中可以很容易进行一些有效的估算,例如据此可算出一棵有 30 个数据元素的二叉树其最低高度为 4。

## 五、教学参考资源

### 1. 树

树是由  $n(n \geq 0)$  个节点组成的有限集合。当  $n > 1$  时,有一个特定的称之为根(root)

的节点,它只有直接后继,但没有直接前驱;除根以外的其他节点划分为  $m$  ( $m \geq 0$ ) 个互不相交的有限集合  $T_0, T_1, \dots, T_m$ , 每个集合又是一棵树,并且称之为根的子树(subtree),如图 4-4 所示。当  $n=0$  时,称为空树,用“ $\emptyset$ ”来表示。当  $n=1$  时,是一棵只有一个根节点的树,如图 4-5 所示。

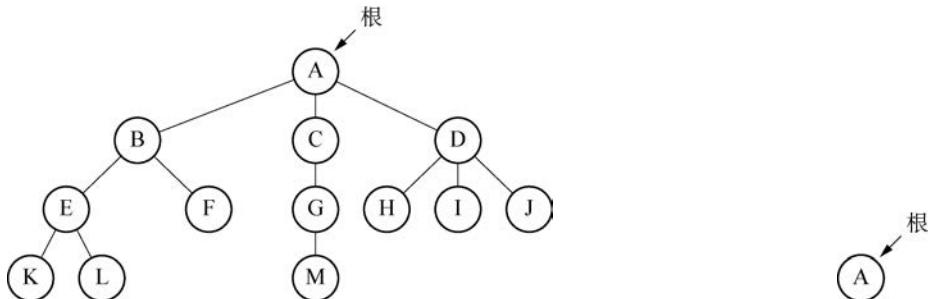


图 4-4 树

图 4-5 只有一个根节点的树

上图 4-4 树中有 13 个节点,可以看出此树由一个根节点和三个子树构成,子树仍然满足树的定义。可以用  $T = \{D, R\}$  来表示树,其中  $D$  表示数据,包括  $\{A, B, C, D, E, F, G, H, I, J, K, L, M\}$ ;  $R$  表示关系,  $R = \{\langle A, B \rangle, \langle A, C \rangle, \langle A, D \rangle, \langle B, E \rangle, \langle B, F \rangle, \langle C, G \rangle, \langle D, H \rangle, \langle D, I \rangle, \langle D, J \rangle, \langle E, K \rangle, \langle E, L \rangle, \langle G, M \rangle\}$ 。从图中也可以看出,树中的数据是一种“一对多”关系,因此树是一种非线性结构。

以图 4-4 为例来了解和树有关的几个概念,了解树的一些特征。

#### (1) 节点称谓。

可用类似家谱形式给节点取一些称谓,如每个节点的后继,被称为该节点的子节点(子女节点)(child),该节点就是子节点的双亲节点(父母节点)(father),如图 H, I, J 都是 D 的子节点。具有同一双亲的子节点互为兄弟节点(sibling),如 H, I, J 互为兄弟节点。每个节点的所有子树中的节点称为子孙节点(descendant),如 B 节点的子孙节点是 E, F, K, L。从树根节点到达某节点的路径上经过的所有节点被称作该节点的祖先节点(ancestor),如图中对于节点 M,它的祖先节点是 G, C, A。

#### (2) 节点的度(degree)与树的度。

树中某个节点的子树的个数称为节点的度,如图 4-4 中,B 节点有 2 个度,D 节点有 3 个度。树中各节点的度的最大值称为树的度,图 4-4 中树的度是 3。通常将度为  $m$  的树称为  $m$  次树。树有一个很重要的性质:树中的节点数等于所有节点的度数的总和加 1。

#### (3) 分支节点与叶子节点(leaf)。

度不为零的节点称为分支节点(非终端节点),如图 4-4 中 B, C, D 等。度为零的节点称为叶子节点(终端节点),如 K, L 等。在分支节点中,每个节点的分支数也就是该节点的度。度为 1 的节点称作单分支节点,度为 2 的节点称作双分支节点,依此类推。

#### (4) 路径与路径长度。

对于任意两个节点  $d_i$  和  $d_j$ ,如树中存在一个节点序列  $d_i, d_{i1}, d_{i2}, \dots, d_{in}, d_j$ ,序列中任

意一个节点都是其在序列中的前一个节点的后继，则称该节点序列为由  $d_i$  到  $d_j$  的一条路径。如图 4-4 中 A 到 M 的路径为(A,C,G,M)。路径长度等于路径所通过的节点数目减 1。如 A 到 M 的路径长度为 3。

#### (5) 节点的层次(level)和树的高度。

从根节点到树中某节点所经路径上的分支数称为该节点的层次，根节点为第 0 层，依次往下递增。如图 4-4 中 M 的层次为 3。所有节点的层次的最大值称为树的高度。如图 4-4 中树的高度是 3。

除了用树形来表示一棵树之外，在书写时经常用括号法来表示一棵树，如图 4-4 可以用 A(B(E(K,L),F),C(G(M)),D(H,I,J)) 来表示，A 表示树根，括号里依次是左右子树。在实际解决问题时，一般可把树转换成二叉树，二叉树结构简单，存储效率高，运算算法相对简单。

### 2. 层序遍历

对于二叉树，还有一种遍历方法叫层序遍历，即根据树的层次顺序来依次访问节点。

在层序遍历过程中，每一层都按从左到右顺序逐个访问节点，先访问到的先出队，因此可以用一个队列来实现二叉树的层序遍历。首先创建一个队列，先把 A 入队，之后出队 A，把 A 的两个左右子节点 B、C 入队。然后 B 出队，把 B 的子节点 D 再入队，接着把 C 出队，把 C 的子节点 E、F 入队。依此类推，每次出队一个节点时，入队了它的两个子节点。这样就确保了父节点优于子节点。由于每次都按先左后右顺序访问，所以左子树节点的子节点也会优于右子树节点的子节点被访问到。对教科书第 80 页图 4.1(d) 实现层序遍历的结果为：abecfgd。

——参考《大话数据结构》，程杰，清华大学出版社

## 六、教学参考案例

### ■ 参考案例

#### 二叉树及其特性

上师大附中闵行分校 董帅含

(1 课时)

##### 1. 学科核心素养

能够针对项目任务中的心算游戏，分小组讨论如何快速验证表达式的计算结果，查阅资料，寻求用计算机解决这类问题的方法。（信息意识）

通过用集合表示二叉树、分析完美二叉树高度与节点的关系等活动，感受运用计算机可以处理的方式界定问题、抽象特征、建立结构模型、合理组织数据。（计算思维）

通过查阅资料，了解如何用波兰式来表示算术表达式，选用合适的编译平台来体验二叉树的表示，有效地管理学习过程与学习资源，创造性地解决问题，从而完成学习任务，形成创新作品的能力。（数字化学习与创新）

## 2. 《课程标准》要求

结合生活实际,理解数据结构的概念,认识数据结构在解决问题过程中的重要作用。

通过列举实例,认识到抽象数据类型对数据处理的重要性,理解抽象数据类型的概念,了解二叉树的概念及其基本操作方法。

## 3. 学业要求

学生能够运用生活中的实例描述数据的内涵与外延,能够将有限制条件的、复杂生活情境中的关系进行抽象,用数据结构表达数据的逻辑关系。

## 4. 教学内容分析

二叉树是教科书第四章“常用数据结构”第一节的内容,分为 4 课时,本节课教学内容是第 1 课时。教科书中之前介绍的数据的组织形式都是一对一的线性关系,从本节课开始出现一对多的非线性关系。本节课主要介绍二叉树的形式、如何构建一棵二叉树。此部分内容作为一种新结构的起始学习内容,为后面灵活应用二叉树解决实际问题打下基础。另外,二叉树的集合表示方式、数组实现方法为后面多对多非线性关系的构建提供了一种新的思路。

## 5. 学情分析

在这节课之前,学生已经掌握了一对一的线性关系,但本节课所讲二叉树是一种一对多的非线性关系结构。二叉树是一类应用极为广泛的数据结构,但理论性较强,概念多,对学生的理解能力有很大的挑战。因此本节课通过图示、归纳总结等方法对二叉树的性质逐步分析,最终得到二叉树的性质。同时高中生已经具备了独立思考的能力,对新事物的接受能力较强,并且有一定的展示需求,因此,在课堂教学中,结合具体案例引导学生发现问题和分析问题,让他们能够自由地、充分地、广泛地进行课堂讨论,在解决问题的过程中培养学生思维能力、探索精神和创新意识。

## 6. 教学目标

- 通过生物树、家谱等类比,图解二叉树的定义。
- 类比细胞分裂,理解二叉树的分类并归纳其特性。
- 学会利用集合表示二叉树。
- 理解完美二叉树高度与节点的关系。

## 7. 教学重难点

- 教学重点:理解二叉树的定义、分类及特性,掌握二叉树的表示方法。
- 教学难点:理解完美二叉树高度与节点的关系。

## 8. 教学策略分析

本节课主要采用启发式、体验式、图示式教学。以“观察—分析—总结—实践”为主线进行学习。

- (1) 以问题为导向,引出非线性数据结构。
- (2) 针对二叉树中的一些概念和基本术语,开展讨论及课堂练习,实现教学目的。

## 9. 教学过程设计(见图 4-6 和表 4-9)

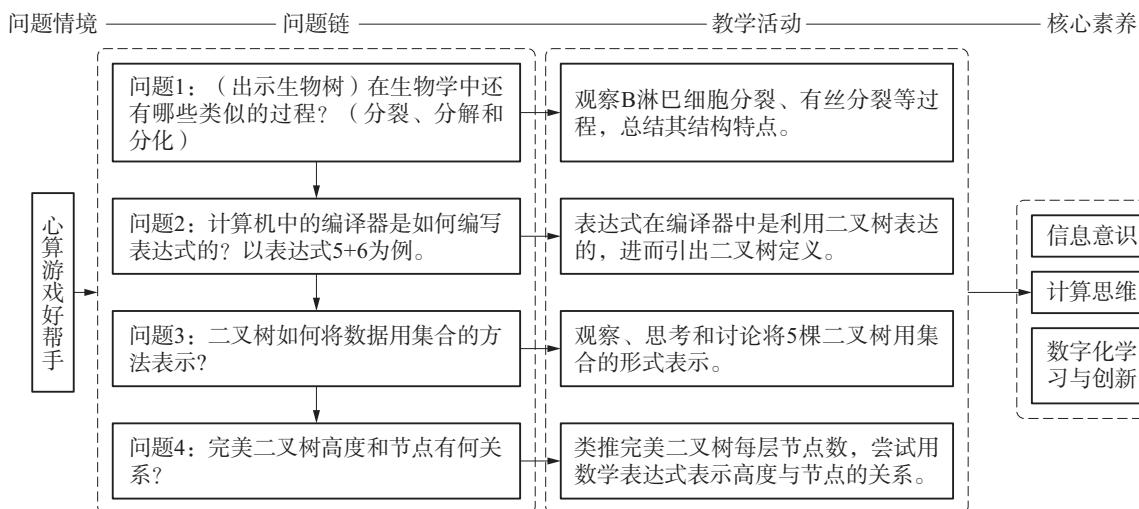
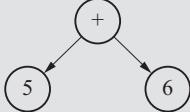


图 4-6 教学过程

表 4-9 教学过程设计表

教学环节	教师活动	学生活动	设计意图
情境导入	问题引入:出示生物树、B 淋巴细胞分裂、有丝分裂等过程图,提问图中结构有哪些特点	思考并回答问题	创设情境,引出项目主题。激发学生的学习兴趣,为新课的学习埋下伏笔
活动 1: 波兰式	为了用计算机解决心算游戏问题,介绍计算机中的编译器也会采用这种一分为二的结构来重构表达式,从而引出波兰式。计算机中的编译器利用波兰式编写,即用户输入的表达式,波兰式会将运算符前置,例如 $5 + 6$ 的波兰式可以写成 $+ 5 6$ ,将其用图形来表示如下:   请同学们将下列表达式用波兰式的方式表示,并画出图形。 示例: $5 + 2 + 8$	学生阅读学习资料,了解波兰式,思考并完成用波兰式表示表达式及绘制图形,讨论、交流结果	引出新知,理解一分为二的结构可以将表达式重新进行表示,进而引出二叉树的概念
新课讲授	二叉树的定义: 点明波兰式的表示方法本质上是二叉树结构,这是一种数据结构,二叉树可以描述多种问题的计算过程……利用该结构存储数据,可以便于查找和排序。 讲述二叉树的定义:二叉树是一个集合,可以为空;如果不为空,则含有三个元素,即“根节点”“左子树”和“右子树”	理解二叉树的定义; 通过集合的形式存储数据,理解根节点、左子树、右子树等概念	

教学环节	教师活动	学生活动	设计意图
讨论交流	在知道二叉树的定义后： (1)引导观察教科书图 4.1 中的 5 棵二叉树并提出问题：它们有什么特点？ (2)引导归纳非线性数据结构	观察并描述二叉树的特点	
活动 2： 二叉树的表示	活动内容： (1)介绍前面 2 棵二叉树的集合表示。 (2)引导学生根据集合定义写出另外 3 棵二叉树的表示	结合教师所讲内容进行思考，写出集合表示形式	掌握二叉树的表示形式
新课讲授	介绍二叉树中一些专业术语：“父节点”“子节点”“树的高度”。 分析二叉树的一些特例：满二叉树、完全二叉树和完美二叉树。 引导学生归纳三类二叉树的关系	思考，倾听。归纳各种类型二叉树的特点和关系。学习二叉树中节点、度、左子树、右子树的相关知识	认识二叉树中的一些基本概念，掌握二叉树的特性
活动 3： 探究完美二叉树	布置教科书中完美二叉树的探究活动： (1)出示细胞有丝分裂和分化过程图，判断其二叉树的类型。 (2)探究高度和节点的关系	讨论、探究高度和节点的关系，进行顺序编号和层序编号的相互推算	理解二叉树的意义，为二叉树的应用做好铺垫
课堂总结	教师回顾梳理本节课学习的内容，展示知识点：二叉树的表示、三种类型的二叉树	思考，归纳	梳理本节课的知识点，体会二叉树的含义
思考：完成项目任务 1	分析“心算游戏好帮手”实现过程中的挑战之处，例如为什么用手机上的计算器不能很好地解决问题，实现的关键点在哪里等		

## 附：“二叉树及其特性”学习单

**活动 1：**将表达式  $5+2+8$  用波兰式的方式表示，并画出图形。

**活动 2：**按照集合定义，分别写出如图 4-7、图 4-8 和图 4-9 所示 3 棵树的集合表示。

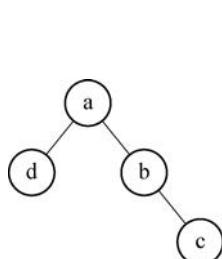


图 4-7 树 1

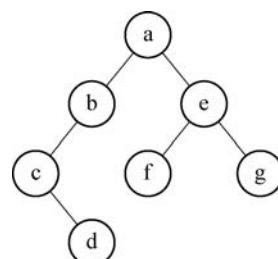


图 4-8 树 2

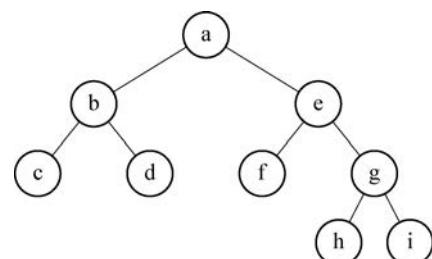


图 4-9 树 3

(1) 图 4-7 中的树用集合表示为\_\_\_\_\_。

(2) 图 4-8 中的树用集合表示为\_\_\_\_\_。

(3) 图 4-9 中的树用集合表示为\_\_\_\_\_。

**活动 3:**高度为 2 的完美二叉树如图 4-10 所示。

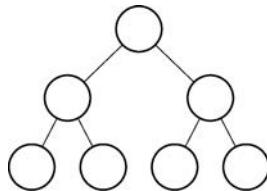


图 4-10 高度为 2 的完美二叉树

该二叉树的顺序编号和层序编号如表 4-10 所示。

表 4-10 顺序和层序编号对照表

顺序	0	1	2	3	4	5	6	...
层序	(0,0)	(1,0)	(1,1)	(2,0)	(2,1)	(2,2)	(2,3)	...

(1) 请写出顺序编号与层序编号的关系:\_\_\_\_\_。

(2) 对于某一个顺序号  $k$ , 写出对应的层序号  $(i, j)$ :\_\_\_\_\_。

(3) 如果已知层序号  $(i, j)$ , 请写出对应的顺序号  $k$ :\_\_\_\_\_。

**思考:**完成项目任务 1。

分析“心算游戏好帮手”实现过程中的挑战之处,例如为什么用手机上的计算器不能很好地解决问题,实现的关键点在哪里等。

## 第二节

## 图

### 一、教学目标与重点

#### 教学目标:

- 在体验思考和编程实践中,理解图的基本概念及其在计算机中的表示,掌握图的一种遍历方法。
- 通过案例分析、讨论思考,了解图在表达网络关系信息中的广泛应用,感悟运用图结构解决问题的高效性。
- 在项目实践中,能够运用图结构解决实际问题,发展计算思维,提高数字

化学习与创新能力。

**教学重点：**

运用图结构解决实际问题。

## 二、教学实施与评价

### 1. 教学活动建议

本节内容为选学内容,现实生活中的许多复杂问题很容易用图的形式来描述,在程序设计时,如何比较全面地将复杂的图表示清楚以便计算机进行处理显得非常重要。教学中可以列举生活中的一些现象并将其抽象成图的形式来表示。例如:有四座城市如图 4-11 所示,城市间的连线用来表示城市间有直达航班。请用图结构表示。

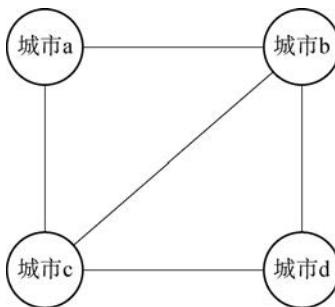


图 4-11 四座城市

图 4-11 所示的图由 4 个顶点和 5 条边构成,可以用  $G = (V, \{E\})$  来表示,将“城市 a”“城市 b”“城市 c”“城市 d”分别用 1、2、3、4 来表示,那么顶点集合就为  $V = \{1, 2, 3, 4\}$ ,边集合为  $E = \{(1, 2), (1, 3), (2, 3), (2, 4), (3, 4)\}$ 。

图结构比较抽象,从实际应用需求出发,建议结合实例讨论其结构性特征,如节点的度数、两个节点之间的距离等。在教学中可从考虑日常生活中的一个问题引入:我们在出行的时候一般会考虑使用导航软件搜一下从一个地点到另外一个地点的路线。如果把地点抽象成一个顶点,路径抽象成线,就构造了一个图。计算机遍历图后能显示出几条路程以满足你的需求(时间最短、路程最短、高速优先等)。遍历是图的基本操作,最常用的有两种方式,如表 4-11 所示。

表 4-11 两种遍历方式

方式	具体实现方法
深度优先搜索(DFS: depth first search)	以图中某个顶点 V 作为起始点,访问此顶点,然后从 V 未被访问的邻接点出发作为下一起始点,继续访问,直至图中所有和 V 有路径相通的顶点都被访问到
广度优先搜索(BFS: breadth first search)	以图中某个顶点 V 作为起始点,先把和 V 点直接相连的所有点都访问到,然后再访问和 V 连接需两步的所有顶点,依此类推,直到所有顶点都被访问到

## 2. 教学过程安排(见表 4-12)

表 4-12 教学过程设计表

课时	教学环节	教师活动	学生活动
第 1 课时 (概念和实现)	1. 导入	列举生活中图的现象	倾听、思考
	2. 图的概念	引导阅读教科书,寻找图结构数据之间的关系及定义	阅读、思考
	3. 探究活动	布置任务:本节第 1 个探究活动	讨论、交流
	4. 图的特性	讲解:节点、度数、距离	倾听、思考
	5. 探究活动	布置任务:本节第 2 个探究活动	讨论、交流
	6. 图的表示	讲解演示:用两个数组表示法描述图	倾听、思考
	7. 体验思考	布置任务:本节体验思考	思考,分享
	8. 作业练习	布置作业:作业练习第 1 题	完成方案
	9. 总结	概念和表示形式	
第 2 课时 (图的应用)	1. 导入	图在计算机中的应用	倾听、思考
		布置活动:求“最短路径”的要求	分析、编程
	2. 活动:“最短路径”	思路分析:主要步骤讲解	倾听、思考
		图解过程:建“最短路径”图	倾听、填写
		巡视,指导	编程实现
	3. 作业练习	布置作业:作业练习第 2 题	编程实现
	4. 总结	图结构的应用场景、程序设计中图结构的配合使用	

在完成各活动的过程中观察学生对于图的理解和应用。可以对学生在以下一些方面的表现进行评价:能够将有限制条件的、复杂生活情境中的关系进行抽象,用图结构表达数据的逻辑关系;会用节点集合和边集合表示图结构;能描述与图有关的概念,如节点、度数、路径、距离、连通等;掌握图的一种计算机表示方法,如用一维数组和二维数组分别表示节点和边;会用图解决现实生活中的问题,如在图结构的支持下用广度优先搜索解决最短路径问题。对于本节的项目实践,教师也可设计实践评价量规,重点考查学生在完成任务的过程中运用图结构解决问题的能力。

### 3. 思考探究提示

#### (1) 本节体验思考参考答案:

针对教科书中图 4.12(b),给出其计算机表示如表 4-13 所示。

表 4-13 图与其对应的计算机表示

图	计算机表示(抽象)																
节点	边																
	<table border="1"> <tr><td>a</td><td>b</td></tr> <tr><td>b</td><td>d</td></tr> <tr><td>c</td><td></td></tr> <tr><td>a</td><td>c</td></tr> <tr><td>c</td><td>e</td></tr> <tr><td>c</td><td>f</td></tr> <tr><td>e</td><td>f</td></tr> <tr><td>f</td><td>g</td></tr> </table>	a	b	b	d	c		a	c	c	e	c	f	e	f	f	g
a	b																
b	d																
c																	
a	c																
c	e																
c	f																
e	f																
f	g																

(2) 本节第 1 个探究活动参考答案:

略。

(3) 本节第 2 个探究活动参考答案:

生活中的图有飞机航线图、网络布线图等。路径、距离、连通如表 4-14 所示。

表 4-14 路径、距离与连通

路径	在一个图中,如果从顶点 $V_1$ 出发,沿着一些边经过顶点 $V_1, V_2, \dots, V_{n-1}$ 到达顶点 $V_n$ ,则称顶点序列( $V_1, V_2, \dots, V_{n-1}, V_n$ )为从 $V_1$ 到 $V_n$ 的一条路径
距离	在一个图中,如果从顶点 $V_1$ 到顶点 $V_n$ 存在多条路径,那么最短路径的长度称为 $V_1$ 到 $V_n$ 的距离
连通	在一个图中,如果从顶点 $V_i$ 到顶点 $V_j$ 有路径,则称 $V_i$ 和 $V_j$ 连通

#### 4. 项目实施提示

- (1) 先回顾第三章中的两种方案:方案一,按任意一条从起点到终点的路径送餐;方案二,按照最短路径进行送餐。
- (2) 比较方案一和方案二各自的特点和应用场景。
- (3) 根据项目实践中的提示,讨论还有哪些可以实现的方案。
- (4) 结合图结构,探究广度优先搜索可以实现哪些方案。
- (5) 将广度优先搜索迁移到其他场景(如导航地图)中。

### 三、作业练习与提示

#### ■ 题目描述

1. 仔细品读本节第二部分中的程序,会发现其效率很低,边集合被重复访问过多次

(n 次)。事实上,可以将基本观察点放在边集合上,只需将它扫描一次,就能得到所有节点的度数。试写出程序。

2. 在 BFS 示例程序中,只是给出了起始节点和终止节点之间的距离,并没有给出最短路径,但相关信息都记录在算法所用的数据结构中。试修改该程序(或自己写一个),不仅给出距离,也给出最短路径(注意,在广度优先搜索的层次结果中,任何一个节点只有唯一的一条由实线标识的路径往上到达起始节点,这就是最短路径)。

## ■ 作业练习参考答案

1. 逐条列举边,然后将与这条边有联系的顶点进行记录并做累加。

参考程序如下:

```
node=['a','b','c','d','e','f','g']
edge=[[['a','b'],[['a','c'],[['b','d'],[['c','e'],[['c','f'],[['e','f'],[['f','g']]]]]]]]]
n=len(node)
e=len(edge)
degree=list(0 for i in range(n))
for i in range(e):
    for j in range(n):
        if edge[i][0]==node[j] or edge[i][1]==node[j]:
            degree[j]=degree[j]+1
print(node)
print(degree)
```

2. 在遍历顶点的过程中,将这个顶点的上一个顶点记录下来。可以通过每下一层,记录层号的方式实现。把整个遍历过程记录在一个路径列表中,最后通过路径倒推,即可得出实际的最短路径。

参考程序如下:

```
import queue
a,b,c,d,e,f,g,h=range(8)          # 对节点进行从 0 到 7 编号
edge=[[a,b],[a,c],[b,f],[b,h],[c,d],[c,e],[c,h],[d,e],[e,g],[f,g],[g,h]]
G=[]
t=list()
for v in range(8):                  # 将边集合转成邻接表
    for x in edge:
        if v in x:
            t=list(set(t+x))
            t.remove(v)
            G.append(t)
            t=list()
s=1;t=4                            # 设定起点和终点
```

```

checked= set()          # 保存访问过的顶点
checked.add(s)
queue= queue.Queue()
queue.put((s, 0, -1))      # 把起始顶点和层次 0 入队
path= []
def shortest_path(path):           # 通过 path 找到最短路径
    shortestpath= []
    curNode= path[-1]
    while curNode != path[0]:
        shortestpath.append(curNode)
        curNode= path[curNode[2]]          # 根据第三个元素找到前一个点
    shortestpath.append(path[0])
    shortestpath.reverse()
    return shortestpath
while not queue.empty():
    temp= queue.get()      # 出队一个顶点和层次
    v= temp[0]
    k= temp[1]
    path.append(temp)
    if v== t:
        print("经过的距离为:", k)
        print("经过的最短路径为:")
        for i in range(k+1):
            print(shortest_path(path)[i][0], end= "-")
        break
    for w in G[v]:
        if w not in checked:
            checked.add(w)  # 添加到已访问过的集合中
            queue.put((w, k+1, len(path)-1))  # 顶点和层次入队

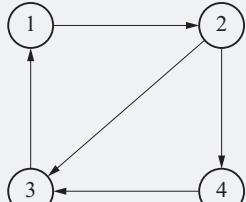
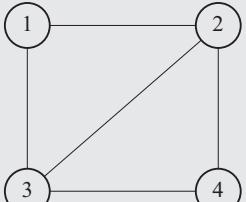
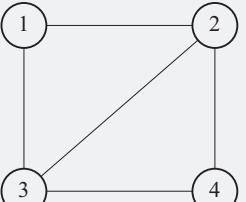
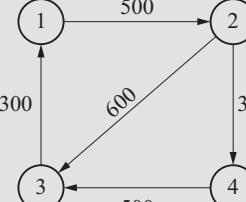
```

## 四、核心概念介绍

### 1. 图的分类

针对不同的情况,可以把图按如下方法分类:根据边是否有方向分为有向图和无向图,根据边的权值情况分为无权图和有权图,如表 4-15 所示。

表 4-15 常见图的分类

类别	有向图	无向图	无权图	有权图
形式				
含义	表示可以从 1 到 2, 但不能从 2 到 1	表示可以从 1 到 2, 也可以从 2 到 1	边无权值	边有权值

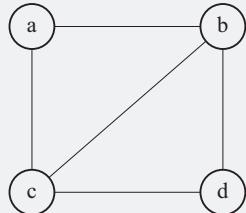
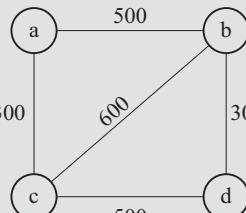
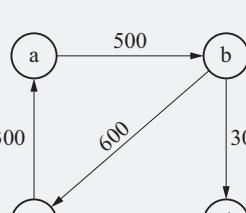
## 2. 图的存储

图的存储比较复杂,除了教科书中用两个数组来存储外,通常还有三种存储方式,分别是邻接矩阵、邻接列表、邻接加权字典,可以根据不同的需求来选择。

### (1) 邻接矩阵

图的邻接矩阵存储方式是用两个数组来表示图:用一个一维数组存储图中顶点的信息,用一个二维数组存储图中边的信息。下面分析如何用二维数组来表示图,见表 4-16。

表 4-16 图的表示形式

编号	图结构	表示形式																								
1		<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>a</td><td>b</td><td>c</td><td>d</td></tr> <tr> <td>a</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr> <td>b</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr> <td>c</td><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr> <td>d</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> </table> <p>对于无权值的边,有边存在用 1 表示,无边存在用 0 表示</p>	a	b	c	d	a	0	1	1	0	b	1	0	1	1	c	1	1	0	1	d	0	1	1	0
a	b	c	d																							
a	0	1	1	0																						
b	1	0	1	1																						
c	1	1	0	1																						
d	0	1	1	0																						
2		<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>a</td><td>b</td><td>c</td><td>d</td></tr> <tr> <td>a</td><td>0</td><td>500</td><td>300</td><td>0</td></tr> <tr> <td>b</td><td>500</td><td>0</td><td>600</td><td>300</td></tr> <tr> <td>c</td><td>300</td><td>600</td><td>0</td><td>500</td></tr> <tr> <td>d</td><td>0</td><td>300</td><td>500</td><td>0</td></tr> </table> <p>对于无向有权值的图,有边存在用权值表示。自身到自身设置为 0</p>	a	b	c	d	a	0	500	300	0	b	500	0	600	300	c	300	600	0	500	d	0	300	500	0
a	b	c	d																							
a	0	500	300	0																						
b	500	0	600	300																						
c	300	600	0	500																						
d	0	300	500	0																						
3		<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>a</td><td>b</td><td>c</td><td>d</td></tr> <tr> <td>a</td><td>0</td><td>500</td><td>0</td><td>0</td></tr> <tr> <td>b</td><td>0</td><td>0</td><td>600</td><td>300</td></tr> <tr> <td>c</td><td>300</td><td>0</td><td>0</td><td>0</td></tr> <tr> <td>d</td><td>0</td><td>0</td><td>500</td><td>0</td></tr> </table> <p>对于有向有权值的图,行代表起点,列代表终点,将有向边用权值表示</p>	a	b	c	d	a	0	500	0	0	b	0	0	600	300	c	300	0	0	0	d	0	0	500	0
a	b	c	d																							
a	0	500	0	0																						
b	0	0	600	300																						
c	300	0	0	0																						
d	0	0	500	0																						

在 Python 中,使用列表生成二维数组对上表中编号为 1 的图进行编程描述,实现代码如下:

```
n= 4  
a,b,c,d= range(4) # 对节点进行从 0 到 3 编号  
G=[[0]* n for i in range(1,5)] # 定义矩阵,通过列表推导式产生全 0 的二维  
数组  
G[a][b]= G[b][a]= 1 # 添加相应表的值,因此图为无向图所以(a,b) 和(b,  
a)都为 1  
G[a][c]= G[c][a]= 1  
G[b][c]= G[c][b]= 1  
G[d][b]= G[b][d]= 1  
G[d][c]= G[c][d]= 1  
for j in range(4): # 显示矩阵  
    print(G[j])
```

如果要得到 a 顶点的边,只需遍历 a 所在的二维数组第一行。但用邻接矩阵存储顶点较多、边较少的图时有很大的空间浪费。因为边少,所以会产生大量的 0,存储大量重复的 0,浪费空间。

### (2) 邻接列表。

邻接列表是用集合的方式存储每个顶点和其他顶点之间的关系的。对上表中编号为 1 的图进行编程描述。在代码中{b,c}表示和 a 顶点相连接的顶点是 b 和 c,其他的依此类推。实现代码如下:

```
n= 4  
a,b,c,d= range(4) # 对节点进行从 0 到 3 编号  
G=[{b,c}, # 与 a 有关系的点  
   {a,c,d}, # 与 b 有关系的点  
   {a,b,d}, # 与 c 有关系的点  
   {b,c}] # 与 d 有关系的点  
print(G[b]) # 显示 b 点的信息
```

### (3) 邻接加权字典。

邻接加权字典是用字典的方式存储每个顶点和其他顶点之间的连接关系和权值大小的。对上表中编号为 2 的图进行编程描述,实现代码如下:

```
n= 4  
a,b,c,d= range(4) # 对节点进行从 0 到 3 编号
```

```

G= [{"b":500,c:300},      # 与 a 有关系的点
      {a:500,c:600,d:300},  # 与 b 有关系的点
      {a:300,b:600,d:500},  # 与 c 有关系的点
      {b:300,c:500}]        # 与 d 有关系的点
print(G[b])             # 显示 b 点的信息

```

## 五、教学参考资源

深度优先遍历：

(1) 问题描述。

深度优先遍历方法：以图中某个顶点 V 作为起始点，访问此顶点，然后从 V 未被访问的邻接点出发作为下一起始点，继续访问，直至图中所有和 V 有路径相通的顶点都被访问到。

请对图 4-12 所示的图进行深度优先遍历。

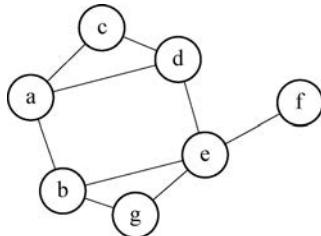


图 4-12 待遍历图

(2) 设计思路。

访问过的顶点可以看作入栈的元素，继续访问就需出栈该元素，找邻接的顶点入栈。也就是说可以将访问过的顶点依次入栈，之后出栈，直到某顶点的邻接点都访问一遍，然后，退回到该顶点的上一个顶点再访问。以上过程类似于树的先序遍历，可以用出栈和入栈的方式来实现。

(3) 过程模拟。

下面对此图进行深度优先遍历：

- ① 选一个顶点作为起始点访问，假设先访问 a。
- ② a 和 b、c、d 是邻接顶点，从 b、c、d 中任意选一点，如选择 d 访问。
- ③ 访问完 d 后，接着从 d 点开始，继续往下访问和 d 相连的顶点。
- ④ d 和 c、e、a 是邻接顶点，但 a 已访问过了，所以排除 a，从 c、e 中选一点，如选择 e 访问。
- ⑤ 访问完 e 后，继续往下访问，发现它的邻接顶点为 d、b、g、f，如选择 g 访问。
- ⑥ 访问完 g 后，g 的邻接顶点是 e、b，访问 b。

⑦ 访问完 b 后，继续往下访问，发现它的邻接顶点为 a、g、e，这些顶点都已访问过，这时就退回到上一步，到 g，g 的邻接点也都访问过，这时再回退到上一步，到 e。再找 e 点有没有未访问过的邻接顶点。找到 f，接着继续访问。依此类推。

(4) 参考程序。

```
n= 4
a,b,c,d,e,f,g= range(7)          # 对节点进行从 0 到 6 编号
change= ["a", "b", "c", "d", "e", "f", "g"]
G= [{b,d,c},                      # a 的邻接顶点
      {a,e,g},                      # b 的邻接顶点
      {a,d},                         # c 的邻接顶点
      {a,c,e},                        # d 的邻接顶点
      {b,d,f,g},                      # e 的邻接顶点
      {e},                            # f 的邻接顶点
      {b,e}                           # g 的邻接顶点
      ]
def pydfs(G,v):
    checked= set()                  # 保存访问过的顶点
    Py_stack= list()
    Py_stack.append(v)              # 把起始顶点入栈
    while Py_stack:
        u= Py_stack.pop()          # 出栈一个顶点
        if u not in checked:
            print(change[u],end= " ")
            checked.add(u)           # 添加到已访问过的集合中
            Py_stack.extend(G[u])    # 多个元素添加
if __name__== "__main__":
    pydfs(G,a)                   # 函数调用
```

(5) 运行结果。

a	d	e	g	b	f	c
---	---	---	---	---	---	---

——参考《大话数据结构》，程杰，清华大学出版社

## 六、教学参考案例

### ■ 参考案例

#### 图结构初探

上海市控江中学 覃 怡

(1课时)

##### 1. 学科核心素养

通过对论文参考文献之间关系的观察和分析,感知论文参考文献关系中数据所承载的信息;在合作解决问题的过程中,愿意与团队成员共享信息,实现信息的最大价值。(信息意识)

分析论文参考文献的关系图,提炼其中所蕴含的信息,采用计算机可以处理的方式界定问题、抽象特征、建立结构模型、合理组织数据。(计算思维)

能运用数字化学习资源与学习工具开展自主学习、协同工作、知识分享与创新创造。(数字化学习与创新)

在分析论文参考文献的关系图以及思考生活中如何用图来进行数据处理的过程中,具有积极学习的态度和理性判断的能力,负责任地使用信息。(信息社会责任)

##### 2.《课程标准》要求

结合生活实际,认识数据结构在解决问题过程中的重要作用。

通过问题解决,能对简单的数据问题进行分析,选择恰当的数据结构。

##### 3. 学业要求

学生能够运用生活中的实例描述数据的内涵与外延,能够将有限制条件的、复杂生活情境中的关系进行抽象,用数据结构表达数据的逻辑关系。

能够解释程序中数据的组织形式,描述数据的逻辑结构及其操作,评判其中数据结构运用的合理性;能够针对限定条件的实际问题进行数据抽象,运用数据结构合理组织、存储数据,选择合适的算法编程实现、解决问题。

能够分析数据与社会各领域间的关系,自觉遵守相应的伦理道德和法律法规。

##### 4. 教学内容分析

数据结构是计算机存储、组织数据的方式,是相互之间存在一种或多种特定关系的数据元素的集合。通常情况下,选择合理的数据结构,配合算法,可以使得程序的运行更快,并提升数据在计算机中存储的效率。图是教科书中第四章第二节的内容,包含了图的概念、图的计算机表示、图的典型操作、图的应用。本节课是图的第1课时,以论文参考文献的关系为情境,用四个问题串联,帮助学生认识什么是图及其特点,以及图如何在计算机中表示等。

##### 5. 学情分析

学生已经完成了必修1模块的学习,自主选择学习选择性必修1模块,已经具有较强

的逻辑思维能力，并具备了使用 Python 语言编写程序的基础。

通过对选择性必修 1 模块前面内容的学习，学生已经认识了什么是数据、什么是数据结构，掌握了线性表、队列、栈、二叉树的概念及其相关操作。

## 6. 教学目标

- 从论文参考文献的关系，感知论文参考文献关系图数据中所承载的信息，采用计算机可以处理的方式抽象特征，知道图及其特点，用图合理组织数据。
- 在计算每一篇论文被引用次数的过程中，运用数字化的资源和工具，知道度的概念，掌握图在计算机中的表示方法，建立结构模型，编写程序，调试运行。
- 在用计算机处理论文参考文献数据的过程中，愿意与同伴互助，共享信息，合作解决问题，理解图与树的异同之处。
- 在分析论文参考文献的关系图以及思考生活中如何用图来进行数据处理的过程中，能以积极学习的态度和理性判断的能力，负责任地使用信息。

## 7. 教学重难点

- 教学重点：知道图及其特点；掌握图在计算机中的表示方法。
- 教学难点：掌握图在计算机中的表示方法。

## 8. 教学策略分析

本节课选用学生学习中遇到的论文参考文献的相关关系为情境，引入了对图的学习，在学习过程中采用自主学习、小组讨论、实践体验等学习方法。

### (1) 基于问题情境，认识图及其特点。

本课从实际问题情境出发，通过对论文参考文献关系图的讨论和实现，让学生能更好地理解图的概念，并能用集合形式表示图结构。

### (2) 设计问题链，由浅入深层层破解。

本课通过由四个问题组成的问题链，引导学生认识图及其特点以及图在计算机中的表示方法。借助学习单，提供知识链接，培养学生自主学习能力。

## 9. 教学过程设计(见图 4-13 和表 4-17)

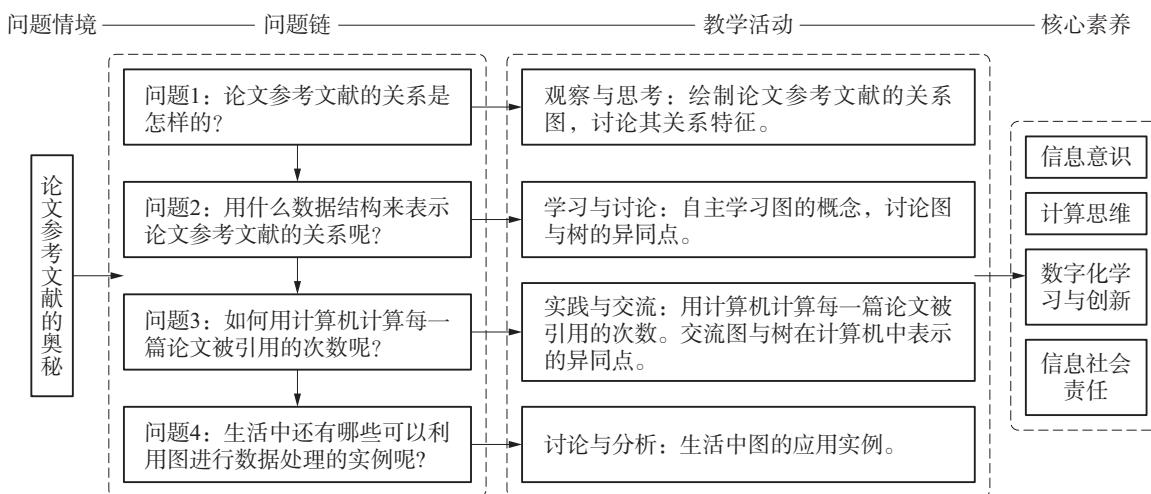


图 4-13 教学过程

表 4-17 教学过程设计表

教学环节	教师活动	学生活动	设计意图
情境导入	<p>参考文献中的奥秘 问题情境: 小申最近阅读了很多关于机器学习的论文, 发现每一篇论文都会标上作者在写作过程中引用的参考文献。在阅读多篇论文后, 小申还发现有些论文所引用的参考文献是相同的。</p> <p>问题 1: 论文参考文献的关系是怎样的? (1) 操作: 请同学们根据小申阅读的 6 篇论文的参考文献绘制关系图, 并进行交流分享。 (2) 小组交流: 观察参考文献的关系图, 思考关系图中的数据关系之间有什么特点</p> <p>图的概念、结构特点 问题 2: 用什么数据结构来表示论文参考文献的关系呢? (1) 提问: 如果要用计算机来处理这些数据, 论文参考文献关系图中的数据能否采用之前学习过的某种数据结构来表示呢? (2) 自主学习教科书内容“一、什么是图”。 提问: 图是不是线性表? 图中包含什么数据? 讨论: 图和树的异同点。 (3) 邀请小组交流, 并进行点评</p>	<p><b>【观察、思考】</b> (1) 绘制 6 篇论文的参考文献关系图, 进行组间交流分享。 (2) 小组交流参考文献关系图中数据关系的特点。(预设答案: 参考文献的关系图中一些数据之间是“连通”的, 数据的关系形成一个“网”)</p> <p><b>【学习、讨论】</b> (1) 根据参考文献关系图中数据关系的特点, 小组讨论能不能用以往的知识来解决问题。(预设答案: 不能) 小组交流不能的理由。 (2) 自主学习, 思考问题并在组内进行交流讨论, 完成学习单。(预设答案: 图不是线性表, 图中包含节点和边) (3) 展示学习结果, 分享讨论情况</p>	通过观察论文参考文献的引用关系, 体会图的结构特点。(计算思维) 感知关系图中数据所承载的信息。(信息意识)
认识图	<p>图在计算机中的表示 问题 3: 如何用计算机计算每一篇论文被引用的次数呢? 根据论文参考文献关系图, 结合自主学习, 完成学习单相关内容。 (1) 用 <math>G(V, E)</math> 来表示论文参考文献关系图, <math>V</math> 和 <math>E</math> 各表示什么? (2) 什么是图的度? (3) 自主学习教科书内容“二、图的实现”。利用 Python 语言编写程序计算每篇论文被引用的次数</p>	<p><b>【实践、交流】</b> (1) 小组讨论, <math>G(V, E)</math> 中的顶点和边分别表示论文参考文献关系中的什么数据。 (2) 自主学习“度”。 完成学习单相关内容。 (3) 自主学习教科书内容“二、图的实现”, 将程序补充完整, 并在组内交流分享</p>	通过自主学习、小组交流, 认识图的概念和特点。引导学生提炼其中所蕴含的信息, 界定问题, 并用计算机解决问题。(计算思维) 在讨论中, 愿意与团队成员共享信息, 深入认识图。(信息意识)
图的表示	<p>生活中的“图” 问题 4: 生活中还有哪些可以利用图进行数据处理的实例呢? (1) 讨论: 根据图的概念和特点, 列举生活中可以用图来进行数据处理的实例。 (2) 邀请小组交流并进行点评</p>	<p><b>【讨论、分享】</b> (1) 小组讨论利用图进行数据处理的生活实例。 在学习单中记录讨论内容。 (2) 分享小组讨论结果</p>	用计算机计算论文参考文献的引用数量, 引导学生用计算机的方法抽象特征、建立结构模型、合理组织数据。(计算思维) 能运用数字化学习资源与学习工具开展自主学习、协同工作。(数字化学习与创新)
生活中的图			通过分析生活中利用图来解决的问题, 进一步巩固对图的理解, 并培养积极学习的态度和理性判断的能力, 负责任地使用信息。(数字化学习与创新)

教学环节	教师活动	学生活动	设计意图
课后思考	(1) 利用计算机,我们还可以对论文的参考文献进行哪些数据处理呢? 请设计算法,编写程序。 (2) 思考:当我们在处理论文参考文献的相关数据时,需要注意什么? (3) 拓展学习(可选):请利用相关书籍或者网络,进一步学习图。例如:什么是连通图? 什么是有向图? 图在计算机中的表示有哪几种常用方法?		能运用数字化学习资源与学习工具进行创新创造。(数字化学习与创新) 在分析论文参考文献的关系图过程中,具有积极学习的态度和理性判断的能力,负责任地使用信息。(信息社会责任)

## 附：“图结构初探”学习单

小申最近阅读了很多关于机器学习的论文,发现每一篇论文都会标上作者在写作过程中引用的参考文献。在阅读多篇论文后,小申还发现有些论文所引用的参考文献是相同的。

### 【问题1】论文参考文献的关系是怎样的?

- (1) 根据小申阅读的6篇论文的参考文献绘制关系图,并进行交流分享。

- (2) 小组交流:观察上面绘制的关系图,思考关系图中的数据关系之间有什么特点。
- 

### 【问题2】用什么数据结构来表示论文参考文献的关系呢?

- (1) 小组交流:根据关系图中的数据特点,如果用计算机来处理这些数据,能否用已学的数据结构来进行表达呢? 请简述理由。
- 

- (2) 自主学习:请学习教科书内容“一、什么是图”,并回答以下问题。

① 图是不是线性表? \_\_\_\_\_

② 图中包含什么数据? \_\_\_\_\_

(3) 小组讨论:图和树的异同点,并将讨论结果记录下来。

\_\_\_\_\_

\_\_\_\_\_

【问题 3】如何用计算机计算每一篇论文被引用的次数呢?

(1) 根据【问题 1】中绘制的论文参考文献关系图,完成以下练习。

如果用  $G(V, E)$  表示论文参考文献关系的“图”,那么:

论文参考文献关系中的 \_\_\_\_\_ 用  $V$  来存储,论文参考文献关系中的 \_\_\_\_\_ 用  $E$  来存储,

$V =$  \_\_\_\_\_,

$E =$  \_\_\_\_\_。

(2) 度的概念: \_\_\_\_\_。请完成表 4-18 的填写。

表 4-18  $G(V, E)$  中每个顶点的度数

节点						
度数						

(3) 自主学习:请学习教科书内容“二、图的实现”,将以下程序补充完整。

计算每一篇论文被引用的次数即计算 \_\_\_\_\_。

# 根据图中的  $V$  和  $E$  计算每个顶点的度数。

$V =$  \_\_\_\_\_ #  $V$  存放图中顶点的集合

$E =$  \_\_\_\_\_ #  $E$  存放图中边的集合

$n = \text{len}(V)$  #  $n$  存放图的顶点数

$k = \text{len}(E)$  #  $k$  存放图的边数

$d = \text{list}(0 \text{ for } i \text{ in } \text{range}(n))$  #  $d$  是顶点的度数,此条语句为  $d$  赋初值

$\text{for } i \text{ in } \text{range}(n):$

$\text{for } j \text{ in } \text{range}(k):$

$\text{if } \text{_____} \text{ # 判断当前顶点有没有边}$

$d[i] = d[i] + 1$

$\text{print}(d)$  # 输出每个顶点的度数

【问题 4】生活中还有哪些可以利用图进行数据处理的实例呢?

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_。

## 一、教学目标与重点

### 教学目标:

- 通过案例分析、讨论对比,理解哈希表的概念,在运用哈希表解决实际问题的过程中,增强信息社会责任。
- 通过编程实践,实现哈希表及基于哈希表的动态查找程序。

### 教学重点:

运用哈希表解决实际问题。

## 二、教学实施与评价

### 1. 教学活动建议

本节为选学内容。教学中应注意按照哈希表是什么,为什么能快速存储,存储冲突了怎么解决,构造一个理想的哈希函数,利用哈希表解决实际问题这一逻辑顺序展开教学。哈希表是一种为了实现快速查找而设计的数据结构,通过课堂中的案例活动体会哈希表实现快速查找的原理,将其与数组的按物理地址顺序存储的方式结合起来介绍。对于哈希函数的这种映射关系,建议列举一些数学中的函数关系类比介绍。

针对教科书中的例 4.3 哈希表存储数据流这一问题,可以先让学生看书讨论,在理解了原理后,再在教师的引导下,讨论表 4.7 比较次数统计表。这样既能理解哈希表的原理又能体会到哈希表的高效性。比如输入数据流中来了 8 个数:4,3,5,1,4,2,3,1。事先在内存中构造好如表 4-19 所示的哈希表。

表 4-19 哈希表

下标	0	1	2	3	4	5	6	7	8
数组		1	2	3	4	5	0	0	0

按照哈希地址查找,比较的次数如表 4-20 所示,总共为 8 次。

表 4-20 比较次数

数据	4	3	5	1	4	2	3	1
比较次数	1	1	1	1	1	1	1	1

对于哈希表的冲突除留余数法需要学生结合实例具体操作以加深理解。对于哈希表的应用,建议在活动中对比学习哈希表的存储和读取。

## 2. 教学过程安排(见表 4-21)

表 4-21 教学过程设计表

课时	教学环节	教师活动	学生活动
第 1 课时 (概念和表示方式)	1. 导入	列举生活中快速查找数据的应用	倾听、思考
	2. 哈希表的概念	引导阅读教科书,找出哈希表数据之间的关系及定义	阅读、思考
	3. 活动:“哈希表存储数据流”	提出问题:哈希表是怎么产生的?	思考、分析
	4. 哈希表的实现	讲解演示:直接定址法和除留余数法。 布置活动:建立人口统计哈希表进行实例操作	思考、分享
	5. 总结	概念和实现方法	
第 2 课时 (应用)	1. 导入	回顾哈希表的构造	倾听、思考
	2. 体验思考	布置任务:本节体验思考	思考、分析
	3. 哈希表基本操作	讲解:哈希表基本操作和冲突解决。 布置活动:对比学习哈希表的存储和读取	倾听、思考
	4. 活动:“建立哈希表”	布置活动:“建立哈希表”的要求	分析、编程
		思路分析:主要步骤讲解	倾听、思考
		图解构建过程 巡视,指导	倾听、填写 编程实现
	5. 作业练习	布置作业:本节作业练习	编程实现
	6. 总结	哈希表的基本操作、哈希表结构的应用场景	

在完成各活动的过程中观察学生对哈希表原理的理解和对哈希表实现的掌握情况。可以对学生在以下一些方面的表现进行评价:能从数组角度解释哈希表快速查找的原理;能结合实例进行哈希表的设计;能用直接定址法构造哈希函数;会用除留余数法解决冲突问题;能在运用哈希存储和哈希查找的过程中解决实际问题。

## 3. 思考探究提示

本节体验思考参考答案:

这里我们用机内码作为区分汉字的标准,如对于汉字“土豆”,先用软件(如 WinHex)查“土豆”的机内码为“CDC1 B6B9”,之后将每个汉字的机内码转换为国标码得到“4DC1 36B9”,计算出每个汉字国标码对应的十进制数,“土”为 $4 * 16^3 + 13 * 16^2 + 12 * 16^1 + 1 * 16^0 = 19905$ ,“豆”为 $3 * 16^3 + 6 * 16^2 + 11 * 16^1 + 9 * 16^0 = 14009$ ,之后将十进制求和得出33914。然后用33914除以表长11后余数为1。1即为哈希地址。用同样的方法,计算出的哈希地址如表4-22所示。

表4-22 哈希地址表

品名	土豆	胡萝卜	卷心菜	豆荚	番茄	茄子	辣椒	玉米	洋葱
价格	1.4	2.1	0.6	2.5	4.1	2.5	4.7	4.2	1.4
国标码转换为十进制之和	33914	45197	49788	29581	32637	40868	32360	38856	34499
哈希地址	1	9	2	2	0	3	9	4	3

根据哈希地址得出相应的哈希表,如图4-14所示。

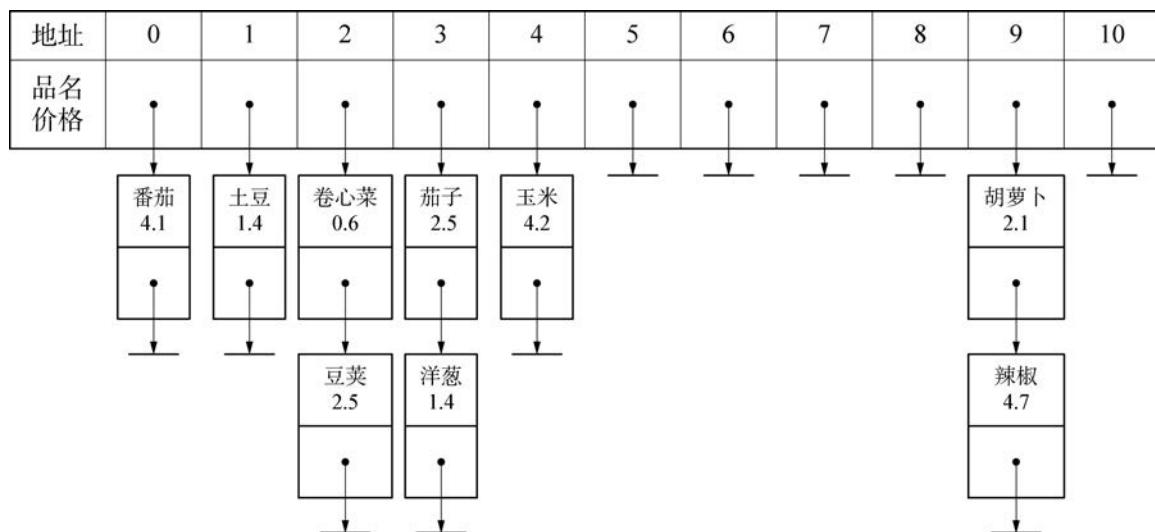


图4-14 得出的哈希表

### 三、作业练习与提示

#### ■ 题目描述

已知一组关键字序列为25, 51, 8, 22, 26, 67, 11, 16, 54, 41, 其哈希地址空间为0, …, 12, 若哈希函数定义为 $H(key) = key \% 13$ , 采用链表法处理冲突, 请画出其对应的哈希表并编程实现。

## 作业练习参考答案

```
class Node:  
    def __init__(self, key):  
        self.key = key  
        self.next = None  
  
class creat_table:  
    def __init__(self, indexbox):  
        self.indextable = [Node] * indexbox  
        for i in range(indexbox):  
            self.indextable[i] = Node(-1)  
  
    def add(self, key):          # 添加数据元素,建立哈希表子程序  
        newnode = Node(key)  
        myhash = key % 13  
        current = self.indextable[myhash]  
        if current.next == None:  
            self.indextable[myhash].next = newnode  
        else:  
            while current.next != None:  
                current = current.next  
            current.next = newnode  
  
    def get(self, key):  
        i = 0  
        myhash = key % 13  
        ptr = self.indextable[myhash].next  
        while ptr != None:  
            i = i + 1  
            if ptr.key == key:  
                return "找了" + str(i) + "次"  
            else:  
                ptr = ptr.next  
        return "未找到!"  
  
    def remove(self, key):  
        i = 0  
        myhash = key % 13  
        pre = self.indextable[myhash]  
        ptr = self.indextable[myhash].next  
        while ptr != None:
```

```

        i= i+ 1
        if ptr.key== key:
            if ptr.next== None:
                pre.next= None
            else:
                pre.next= ptr.next
            return "找了"+ str(i) + "次"
        else:
            pre= ptr
            ptr= ptr.next
    return "未找到!"

def show_table(self,sum):
    for i in range(sum):
        head= self.indextable[i].next
        print(' % 2d:\t' % i,end="")
        while head!= None:
            print('[% 2d]- ' % head.key,end="")
            head= head.next
        print()
if __name__== "__main__":      # 主程序段
    indexbox= 13              # 哈希表长度
    data=[25, 51, 8, 22, 26, 67, 11, 16, 54, 41]
    ha= creat_table(indexbox)
    for i in data:
        ha.add(i)
    print(ha.get(23))
    ha.show_table(indexbox)

```

## 四、核心概念介绍

### 1. 线性探测法

查找哈希表中离冲突索引地址最近的空闲索引地址，并且把新的键插入这个空闲地址。同样地，查找也同插入类似：从哈希函数给出的索引地址对应的位置开始查找，直到找到与键对应的值或者找到空位置。

### 2. 链表法

在哈希表的所有索引地址下面建立 n 个链表，最初默认为 n 个链表头。如果发生冲突，就把相同地址的键和值连接在链表头的后面。也就是说，每个数组下标下面挂的

不是一对 (key, value), 而是一个单链表存储着若干个 (key, value)。查找时, 先用 key 和哈希函数找到索引地址, 之后对该处的单链表进行顺序查找。链表法如图 4-15 所示。

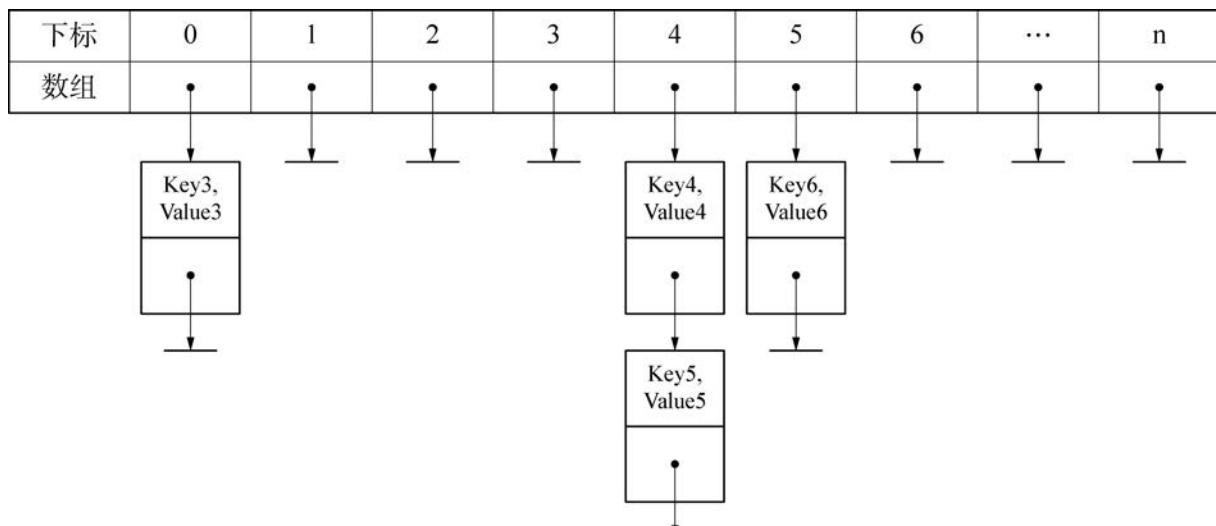


图 4-15 链表法

## 五、教学参考资源

常见的哈希函数除了直接定位法、除留余数法外还有平方取中法、数字分析法等。

### 1. 数字分析法

数字分析法根据每个数字在各个位上的出现频率, 选择均匀分布的若干位, 作为哈希地址。该方法适用于已知关键字集合, 通过观察和分析得到。

例如, 一个关键字集合, 如图 4-16 所示。第 1、2 位的数字完全相同, 不需要考虑, 4、7、8 位的数字只有个别不同, 而 3、5、6 位的数字均匀分布, 可以将 3、5、6 位的数字作为哈希地址, 或者将 3、5、6 位的数字求和后作为哈希地址。

6	0	2	5	3	6	1	9
6	0	3	5	2	4	3	0
6	0	9	1	5	5	1	9
6	0	4	5	4	2	2	9
6	0	7	5	0	0	1	9
6	0	2	5	8	1	1	9

图 4-16 数字分析法

### 2. 平方取中法

对关键字平方后, 按哈希表大小, 取中间的若干位作为哈希地址(平方后截取)。这种

方法适用于事先不知道关键字的分布且关键字的位数不是很大的情况。

例：哈希地址为 3 位，则关键字 10123 的散列地址为 475：

$$10123^2 = 102475129$$

——摘自《趣学数据结构》，陈小玉，人民邮电出版社

## 六、教学参考案例

### ■ 参考案例

#### 哈希表

上海市中原中学 张汉玉

(1 课时)

##### 1. 学科核心素养

能够根据问题需要，自觉、主动地寻求恰当的方式处理社区人口数据信息；能够敏锐察觉社区人口数据变化，分析数据中所承载的信息。（信息意识）

在问题解决过程中能够采用计算机可以处理的方式界定问题、抽象特征、建立结构模型、合理组织数据。通过判断、分析与综合各种信息资源，运用哈希表设计合理的算法，形成解决问题的方案。（计算思维）

通过评估并选用合理的数字化资源，有效地管理学习过程与学习资源，创造性地解决在哈希表操作过程中遇到的冲突问题。（数字化学习与创新）

##### 2. 《课程标准》要求

结合生活实际，认识数据结构在解决问题过程中的重要作用。

通过案例分析，理解基本数据结构的概念，并能编程实现其相关操作。

##### 3. 学业要求

学生能够运用生活中的实例描述数据的内涵与外延，能够将有限制条件的、复杂生活情境中的关系进行抽象，用数据结构表达数据的逻辑关系。

能够针对限定条件的实际问题进行数据抽象，运用数据结构合理组织、存储数据，选择合适的算法编程实现、解决问题。

##### 4. 教学内容分析

“哈希表”是教科书第四章第三节的内容，涵盖哈希表的基本概念及实现。本节创设“社区人口数据统计”的项目情境，介绍了什么是哈希表，以及运用直接定址法与除留余数法构造社区人口数据的哈希表，进而运用 Python 语言编程实现在该哈希表中的基本操作：插入数据、删除数据、查找数据。

##### 5. 学情分析

通过前面章节内容的学习，学生已了解了程序设计中的基本数据类型和基本数据结构，并能够根据现实需求选择不同的数据结构描述客观事物。同时，学生具备较强的思维逻辑能力和一定的 Python 语言编程基础，能够理解基本数据结构所涉及的基本操作并运

用 Python 语言编程实现。

#### 6. 教学目标

- 通过自主学习及填空练习,理解哈希表的基本概念,提升信息意识。
- 在构造社区人口数据哈希表的过程中,采用计算机可以处理的方式,定义哈希表的基本结构,并明确存在的弊端和遇到的冲突问题。结合实际任务需求,构造哈希表并运用 Python 语言编程实现相关基本操作,注重计算思维的养成。
- 通过数字化学习获取相应的学习资源,并能够根据实际情况,合理组织数据,运用图示化方式创造性地解决遇到的冲突问题。

#### 7. 教学重难点

- 教学重点:运用直接定址法与除留余数法构造哈希函数;编程实现哈希表的基本操作。
- 教学难点:运用链表法解决冲突问题。

#### 8. 教学策略分析

本节课立足“社区网格化管理”的生活情境,创设了“社区人口数据管理”的问题情境,并围绕该情境设计了一系列问题:如何有效管理社区数据→如何构建社区人口数据哈希表→怎么解决插入人口数据时遇到的冲突问题→如何编程实现添加人口数据、查询人口数据、删除错误人口数据。为保障各项学习活动顺利开展,设置了层层递进的学习环节:问题导入→新知学习→学以致用→能力提升→实践应用,逐步引导学生厘清基本概念、掌握如何构建哈希表,并运用 Python 语言实现哈希表的基本操作。

#### 9. 教学过程设计(见图 4-17 和表 4-23)

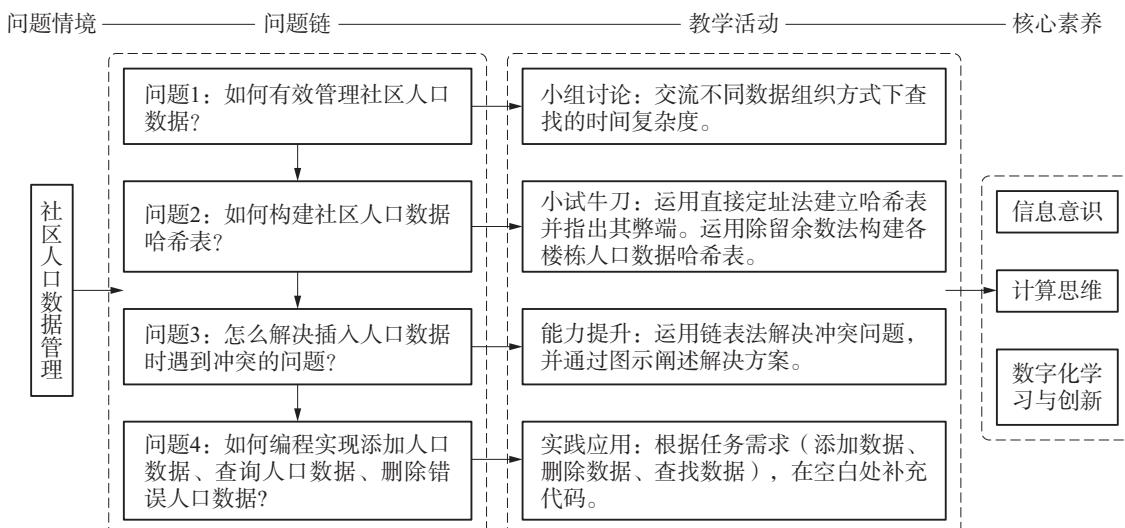


图 4-17 教学过程

表 4-23 教学过程设计表

教学环节	教师活动	学生活动	设计意图
问题导入	<p>问题 1: 如何有效管理社区人口数据?</p> <p>创设信息时代社区网格化管理的情境,阐述社区楼栋人口数据管理问题,带领同学们一起回顾先前所学习的数据组织方式及查找方式。</p> <p>提问:简述每种查找算法的时间复杂度,有没有复杂度为 <math>O(1)</math> 的?</p>	<p>小组成员为“管理社区人口数据”出谋划策,一起回顾并交流探讨每种数据组织方式查找算法的时间复杂度,并进一步寻找有没有复杂度为 <math>O(1)</math> 的解决方案</p>	回顾先前所学知识,并意识到问题所在,为新知学习做好铺垫
新知学习	<p>提供哈希表概念和哈希表示例的相关材料。</p> <p>设置填空题辅助学生提炼概念关键词</p>	<p>自主学习哈希表的基本概念和哈希表示例。</p> <p>完成学习单中的概念填空题</p>	通过实例厘清哈希表的基本概念,提升信息意识
学以致用	<p>问题 2: 如何构建社区人口数据哈希表?</p> <p>提供构造哈希函数(直接定址法与除留余数法)的学习材料。</p> <p>提供街道按照年龄统计的人口数据。</p> <p>提问:请同学们运用直接定址法建立哈希表,实现按出生年份快速查找,并思考该方法存在哪些弊端。</p> <p>提问:如何运用除留余数法构造完整的哈希表?</p>	<p>小试牛刀:</p> <p>运用直接定址法建立哈希表,实现按出生年份快速查找。</p> <p>交流讨论运用直接定址法构建哈希表的弊端。</p> <p>运用除留余数法构建某小区各楼栋人口数据哈希表</p>	通过对直接定址法与除留余数法的学习,构建人口数据哈希表,学以致用,并明确存在的弊端,注重学生计算思维的养成
能力提升	<p>问题 3: 怎么解决插入人口数据时遇到冲突的问题?</p> <p>提问:在原有哈希表(18, 75, 60, 43, 55, 90, 48)的基础上添加数据 24,此时会遇到冲突吗?那么应该如何解决?</p> <p>提供解决方法(链表法)的学习材料</p>	<p>思考如何解决运用除留余数法构建哈希表并添加新数据时遇到的问题。</p> <p>学习并尝试运用链表法解决冲突,通过图示进行阐述</p>	通过设置冲突数据,引导学生进一步思考解决方法,并运用图示形象描述解决方法
实践应用	<p>问题 4: 如何编程实现添加人口数据、查询人口数据、删除错误人口数据?</p> <p>布置实践任务:</p> <p>在创建楼栋人口数据哈希表后,根据现实情况,需要添加人口数据、查询人口数据、删除错误人口数据。</p> <p>提供程序片段,并设置相应的代码填空</p>	<p>打开程序片段,根据任务需求在空白处补充代码。</p>	在添加、删除、查找数据元素的任务驱动下,通过补充关键位置代码的方式,引导学生思考数据元素与指针之间的关系,提升计算思维
课后思考	教科书中本节作业练习		通过作业设置进一步巩固学生对使用链表法解决问题的掌握程度

## 附：“哈希表”学习单

- 新知学习

知识链接：

### 哈希表概念

哈希表(Hash table,也叫散列表),是在数据元素的关键字和数据元素的存放位置之间建立起某种对应关系,以实现根据关键字(key)直接访问数据元素存储位置,完成数据元素查找的一种数据结构。建立这种对应关系的函数称为哈希函数(Hash function),这种对应关系称为映射。利用哈希函数建立的关键字对应存储表,称为哈希表。

哈希表示例(见图 4-18)

地址 [H(key)]	0	1	2	3	4	5	6	7	8	9	10	11	12
关键字 (key)	65		67			70		33	99	48			12

图 4-18 哈希表示例

填空完成：

1. 哈希表是在数据元素的\_\_\_\_\_和数据元素的\_\_\_\_\_之间建立起某种对应关系,以实现根据\_\_\_\_\_直接访问数据元素存储位置,完成数据元素查找的一种数据结构。
2. 通过哈希函数计算一个数据元素在数组中应该存储的位置,可用数学关系式表示为:\_\_\_\_\_。

- 如何构建社区人口数据哈希表

知识链接：

### 直接定址法

以数据元素关键字 key 本身或它的线性函数作为其哈希地址,即  $H(key) = key$  或  $H(key) = a * key + b$ ; (其中 a,b 是常数)。

示例 1:

如图 4-19 所示,有一个人口统计表,记录了某地区从 1 岁到 100 岁各年龄段的人口数目,现要求建立哈希表,实现按年龄进行快速查找。

年龄	1	2	...	99	100
人数	980	800	...	20	12

↓

地址(key)	1	2	...	99	100
年龄(key)	1	2	...	99	100
人数(Value)	980	800	...	20	12

图 4-19 某人口统计表 1

小试牛刀：

请在“地址”旁的括号内填写相对应的哈希函数。

如图 4-20 所示，有一个人口统计表，记录了某地区从 1949 年到 2019 年每年出生的人口数，现要求建立哈希表，实现按出生年份进行快速查找。

The diagram illustrates the conversion of a population statistics table into a hash table. At the top is a population statistics table with columns for birth year and population count. An arrow points down to a hash table structure where the birth year is mapped to an address (index) and the population count is mapped to the value.

出生年份	1949	1950	...	2018	2019
人数	980	800	...	490	300

地址 ( )	0	1	...	59	60
出生年份 (key)	1949	1950	...	2018	2019
人数 (Value)	980	800	...	490	300

图 4-20 某人口统计表 2

交流讨论：运用这种方式构建的哈希表有哪些局限性？

知识链接：

### 除留余数法

除留余数法就是将数据除以某一个常数后，取余数作为地址，即  $H(key) = key \% m$ ，其中  $m$  为哈希表长， $\%$  为取余运算， $key$  为关键字。

思考：为什么除数为哈希表的长度？

练习：运用除留余数法构建某小区各楼栋人口数据哈希表(18, 75, 60, 43, 55, 90, 48)。

0	1	2	3	4	5	6	7	8	9	10

思考：如果在该哈希表上增加数据元素 24 会怎样？应该如何解决该情况？

怎么解决插入人口数据时遇到的冲突问题？

知识链接：

### 链表法

采用链表法解决冲突就是将哈希表设计成为数组加链表的组合形式，数组是哈希表的主体，链表则主要为解决哈希冲突而存在。此方法将所有哈希地址相同的不同关键字数据元素都链接在同一个链表中。

学以致用：

运用链表法构建某小区楼栋人口数据(18, 75, 60, 43, 55, 90, 48, 24)的哈希表，并用图示表示。

#### • 如何编程实现添加人口数据、查询人口数据、删除错误人口数据

在创建楼栋人口数据哈希表后，根据现实情况，需要添加人口数据、查询人口数据、删除错误人口数据。

分析程序片段，根据任务需求在空白处补充代码。

```
# 添加数据元素，在空白处补充代码
```

```

def add(self, key):
    newnode=Node(key)
    myhash=key%13
    current=self.indextable[myhash]
    if current.next==None:
        self.indextable[myhash].next= newnode
    else:
        while current.next!=None:
            _____
            =newnode

# 删除数据元素,在空白处补充代码
def remove(self, key):
    i=0
    mahash=key%13
    pre=self.indextable[myhash]
    ptr=slef.indextable[myhash].next
    while ptr!=None:
        i=i+1
        if ptr.key==key:
            if ptr.next==None:
                pre.next=None
            else:
                _____
            return"找了"+ str(i)+ "次"
        else:
            pre=ptr
            ptr=ptr.next
    return"未找到!"

# 查找数据元素,在空白处补充代码
def get(self, key):
    i=0
    myhash=key%13
    ptr=self.indextable[myhash].next
    while ptr!=None:
        i=i+1

```

```
if _____:  
    return "找了" + str(i) + "次"  
else:  
  
    _____  
return "未找到!"
```

作业：

已知一组关键字序列为 25, 51, 8, 22, 26, 67, 11, 16, 54, 41, 其哈希地址空间为 0, …, 12, 若哈希函数定义为  $H(key) = key \% 13$ , 采用链表法处理冲突, 请画出其对应的哈希表并编程实现。

# 数据结构应用

### 一、本章学科核心素养的渗透

本章内容包括排序和查找两节。通过思考、探究案例，分析、理解排序和查找的思想，在项目活动中掌握运用排序和查找解决实际问题的方法。掌握常用数据结构及编程实现方法后，尝试选择合适的数据结构，在采用一定的算法配合提高排序和查找效率中，培养学生的信息意识和计算思维等素养。

本章是选择性必修1 数据与数据结构模块的应用部分内容。《课程标准》中相关内容要求包括：

1.7 通过实现数据的排序和查找，体验迭代和递归的方法，理解算法与数据结构的关系。

本章的项目活动为“人脸识别系统中的信息管理”。由三个项目任务组成，在完成项目任务的过程中落实《课程标准》要求。

针对给定的任务进行需求分析，明确需要解决的关键问题。运用基本算法设计解决问题的方案，能使用编程语言或其他数字化工具实现这一方案。面对不同类型数据和功能需求时，判断其特点，并主动思考科学的数据组织形式，从而培养学生的信息意识。

在学习了几种基础数据结构和常用数据结构之后，能够正确区分问题解决中涉及的各种数据并采用适当的数据类型进行表示。在Top-K问题讨论的过程中，根据线性数据结构和树型数据结构的不同特点，找到时间复杂度更小的解决方法，尝试把利用信息技术解决问题的过程迁移到学习和生活中其他相关问题的解决过程中。在学习和讨论中认识数据结构的优势和局限性，主动利用各种学习资源和技术工具协同工作，培养数字化学习和创新能力。

通过实现不同的数据排序算法和查找算法，理解算法与数据结构的关系。本章将通过两个例子，展示在数据结构（主要是二叉树）的基本形态上，如何在新的需求下，体现出特定的含义，以支持相关的应用。在“人脸识别系统中的信息管理”项目中体会保护隐私，

以及公正合理地利用和存储数据;在数据组织和数据结构选择时,优先考虑数据的安全及稳定,从而体现信息社会责任担当。

通过本章的学习,认识算法与数据结构的协调配合,进而对实现不同效果的程序有更切实的体会,从而领会数据结构对于优化程序设计和提高程序效率的作用。在第四章二叉树相关内容的基础上,选择适合的二叉树类型,提升排序和查找效率,在一定意义上,可以说它们是二叉树结构的“高级应用”,从而在优化解决方案的过程中培养计算思维。

## 二、本章知识结构

本章遵循《课程标准》,依据学分和课时规定,紧扣学科概念体系,将内容分为两节。

第一节排序,内容包括排序概念及冒泡排序、Top-K 问题。

第二节查找,内容包括查找概念及基于线性表、二叉搜索树的查找。

## 三、本章项目活动设计思路

本章项目活动是针对第一章报名者信息管理的全面升级,不但需要收集、整理相关数据,更要合理地组织和管理数据。用适合的数据结构组织好数据,为后期更好地管理数据打下基础。随着数据流源源不断涌来,存储和替换都基于优质的排序、查找方案展开。

引言中的一个例子——数据流检测问题,在第三章线性表一节中也有过进一步学习。我们讨论过不同处理方式在效率上的差别。不过,那些讨论中都有一个隐含的假设,即存储是无限的,可以支持任意多的不同数据的存放。但现实并非如此,存储总是有限的,尤其在大数据时代,任何容量的存储器都不可能保证装得下应用中的所有数据,必须有所取舍。这就是本项目活动的出发点。

要完成一个程序,它以任意长的数据流为输入,每看到一个数据,要给出它是否曾经被看到过的判断。一个条件限制是,程序在运行过程中只能同时保存最多  $n$  个数据(例如  $n = 104$ )。这样,若数据流中不同的数据超过了  $n$ ,程序就必须做取舍:是放弃新来的数据( $x$ ),还是把先前的某个旧数据( $y$ )替换掉?若放弃  $x$ ,万一后面还有  $x$  来呢?若替换掉  $y$ ,万一后面又有  $y$  来了呢?

在实际计算机系统中通常会采用某种替换策略。“随机替换”方式就是其中一种,即在存储已经满了,又来新数据的情况下,就随机替换掉一个先前的数据,把空间让出来放新数据。当然也有基于引用计数和基于时效性的技术。

组织学生思考排序和替换策略的过程中,注重社会公平、均衡及公序良俗等人文精神的渗透,使其树立正确的价值观。

## 四、本章课时安排建议

本章教学建议用 6 课时完成,具体参见表 5-1。

表 5-1 课时安排计划表

节名	建议课时
第一节 排序	3
第二节 查找	3

## 第一节 排序

### 一、教学目标与重点

#### 教学目标:

- 在原理分析过程中,理解排序的概念及生活中排序的逻辑思想,掌握冒泡排序和选择排序的原理及方法,并能用代码实现,从而体会不同数据结构对于数据处理的支持作用。
- 通过实例分析,理解“Top-K”问题的需求,认识使用排序算法作为基础来求解“Top-K”问题的不足,感受计算机思维方法不断迭代、创新的过程。
- 在项目实践中理解“min-max heap”的概念,尤其是它与二叉树的关系(它是一种完全二叉树)。掌握建立“min-max heap”的方法,并通过它实现一种高效的“Top-K”算法。尝试在追求更优解的过程中,理解数据结构逻辑内涵及优化思想。

#### 教学重点:

- 冒泡排序和选择排序的原理及方法。
- “Top-K”问题的原理和方法。
- 运用排序解决实际问题。

### 二、教学实施与评价

#### 1. 教学活动建议

教科书中主要从排序的概念、冒泡排序的基本操作,以及解决“Top-K”问题三部分进行介绍。

排序的方法可以分为两种。

(1) 内部排序法(internal sorting):将数据都先存储在内存里,然后进行排序,例如冒

泡排序、选择排序、插入排序、快速排序、归并排序以及基数排序等。

(2) 外部排序法(external sorting):当数据太大无法全部存储在内存中时,在排序过程中使用外部存储设备,例如用磁盘或磁带机存储排序的数据,直到排序结束。此类方法我们在本书不做讨论。

本节介绍内部排序法中的冒泡排序,有兴趣的同学可以继续钻研和学习其他排序方法。

教师可以针对必修1教科书中选择排序的学习内容与冒泡排序展开对比,帮助学生理解并掌握冒泡排序的原理和操作方法。尝试使用递归思想实现冒泡排序,以便进一步理解算法与数据结构的关系。

冒泡排序参考代码如下:

```
def Py_bubbleSort(myList): # 冒泡排序
    length= len(myList) # 获取 list 的长度
    for i in range(length- 1): # 这个循环负责设置冒泡排序进行的次数
        for j in range(length- i- 1): # j 为列表下标
            if myList[j] > myList[j+ 1]:
                myList[j], myList[j+ 1]= myList[j+ 1], myList[j]
    return myList
```

如果使用递归的思想,在第一轮做完后,按同样的方法对前  $n - 1$  个数做操作即可,参考代码如下:

```
def Py_bubbleSortRecursive (myList,n): # 冒泡排序的递归实现
    if n== 0:
        return
    for i in range(n- 1):
        if myList[i]> myList[i+ 1]:
            myList[i], myList[i+ 1]= myList[i+ 1], myList[i]
    n= n- 1
    Py_bubbleSortRecursive (myList,n)
```

```
if __name__ == '__main__':
    bubble_list=[7, 4, 6, 2, 5, 8, 3, 0]
    Py_bubbleSortRecursive(bubble_list,len(bubble_list))
    print(bubble_list)
```

标准冒泡排序即使在序列已经有序的情况下依旧进行比较操作(虽然不进行交换操作),直至循环结束。改进思路:这里使用一个标志位,来标识当前序列是否已经有序。如果无序,则继续冒泡排序;如果已经有序,则退出排序算法。这样就可以很好地规避掉一些不必要的比较操作,从而提升效率。

```
def Py_bubbleSortExtend(myList): # 冒泡排序的改进
    length= len(myList) # 获取 list 的长度
```

```

isSwapped= True # 记录是否发生交换(发生为 True,未发生为 False)
for i in range(length- 1): # 这个循环负责设置冒泡排序进行的次数
    if isSwapped :
        isSwapped= False
        for j in range(length- i- 1): # j 为列表下标
            if myList[j] > myList[j+ 1]:
                myList[j], myList[j+ 1]= myList[j+ 1], myList[j]
                isSwapped= True
    else:
        break
return myList

```

运行程序发现：对随机产生的整数进行排序，此排序算法改进不大，但对已经排好序的数据，此算法改进效果明显。

理解“min-max heap”的概念，尤其是它与二叉树的关系（它是一种完全二叉树）。弄清完全二叉树在层序遍历结果中的节点与其子节点的下标关系。

掌握堆排序的原理。首先，把待排序的记录构造成大顶堆，之后通过从堆中不断选出最大元素即可完成排序。所以整个堆排序分成两个主要步骤：将初始完全二叉树调整成堆、弹出堆中最大元素并将其重新调整成堆。

## 2. 教学过程安排(见表 5-2)

表 5-2 教学过程设计表

课时	教学环节	教师活动	学生活动
第 1 课时 (排序概念及冒泡排序)	1. 导入	校跑步社团每月都要统计成员跑步次数，通过跑步次数排行榜激励社团成员坚持锻炼	思考生活中的排序问题
	2. 排序概念	引导学生认识排序： (1) 什么是排序？ (2) 排序的分类有哪些？	阅读教科书，配合上网查找资料
	3. 体验思考	引导思考：完成本节第 1 个体验思考	通过交流完成对排序的认识，并能列举一两个经典排序的例子
	4. 认识冒泡排序	布置活动：对某个数据序列进行冒泡排序操作，理解“冒泡”的精髓	小组合作用列表法尝试冒泡排序
	5. 冒泡排序的实现	讲解冒泡排序的程序实现	倾听、思考
	6. 体验递归思想	布置活动：对冒泡排序的优化进行讨论，感受递归的逻辑内涵	编程实现、分析交流
	7. 总结	排序的概念、冒泡排序原理及程序实现	

续表

课时	教学环节	教师活动	学生活动
第2课时 (Top-K问题)	1. 导入	回顾上节课内容:冒泡排序思想及其存在的不足	思考、讨论
	2. 什么是Top-K问题	引导学生认识什么是Top-K问题: (1) 什么是Top-K问题? (2) 如何实现?	自主阅读、思考分析
	3. 选择排序	讲解选择排序及其实现	倾听、思考
	4. 快速排序	布置活动:讨论快速排序的思想并用程序实现	思考分析、编程实现
	5. 总结	比较用冒泡排序、选择排序和快速排序解决Top-K问题的不同	
第3课时 (用堆排序解决Top-K问题)	1. 导入	回顾第一章,引出数据流存储和组织问题	倾听、思考
	2. 什么是堆	引导学生认识堆: (1) 什么是堆? (2) 堆的性质有哪些?	自主阅读、思考分析
	3. 体验思考	布置活动:尝试从堆的概念着手,认识堆的高度与节点数之间的关系	倾听、思考
	4. 活动:如何构造堆	布置任务:通过构造堆,进一步理解堆的概念,以及其中可以实现的递归思想	小组讨论、尝试构造堆
	5. 活动:如何用堆实现排序	布置任务:通过不断构造排序堆,完成对一组数字的有序存储	思考分析、编程实现
	6. 探究活动	布置作业:完成本节探究活动	写出逻辑关系,尝试用公式表达
	7. 总结	比较冒泡排序与堆排序的不同	

在完成项目活动的过程中观察学生对于排序思想的理解能力。可以对学生在以下一些方面的表现进行评价:能够针对限定条件的实际问题进行数据抽象;运用二叉树合理组织、存储数据,选择冒泡排序算法编程实现、解决问题;能对现实生活中数据业务问题的解决方案进行一定程度的优化分析,并能评价其合理性、完整性,分析方案优化或改进的可能性;会建立“大顶堆”和“小顶堆”;能在二叉树的支持下解决“Top-K”问题,并能进行性能评价。

### 3. 思考探究提示

#### (1) 本节第1个体验思考提示:

排序操作的应用场合如某项体育赛事中对所有运动员的成绩从高到低排列。想知道年级前10名的分数;想知道这一年网络购物开销最大的几笔;想了解最畅销的图书等,都属于Top-K问题。

#### (2) 本节第1个探究活动提示:

因为在整个数据调整的过程中,所有相对较小值都向前移动了,而越小的值越靠前,最小值直接移动到待排位置1处,就像游泳池的进水口,进水时带进大量水泡,体积大的

水泡受到的浮力大,于是向上移动的速度就快,最早到达水面。冒泡算法的思想与这个物理现象有异曲同工之妙,故取此名。

参考代码如下:

```
def BubbleSort(lst):  
    for i in range(len(lst) - 1):  
        for j in range(len(lst) - 1, i, -1):  
            if lst[j-1] > lst[j]:  
                lst[j-1], lst[j] = lst[j], lst[j-1]
```

(3) 本节第 2 个体验思考提示:

高度为  $h$  的完全二叉树( $h$  最小值为 0),它的节点数最大值为  $2^{h+1} - 1$ ,最小值为  $2^h$ 。换言之可以有  $2^h$  棵高度为  $h$  的完全二叉树。

(4) 本节第 2 个探究活动问题一提示:

如图 5-1 所示,当下标  $i$  为 0 时,其左子节点下标为 1,右子节点下标为 2;当下标  $i$  为 1 时,其左子节点下标为 3,右子节点下标为 4;当下标  $i$  为 2 时,其左子节点下标为 5,右子节点下标为 6,整理后得到的结果如表 5-3 所示。

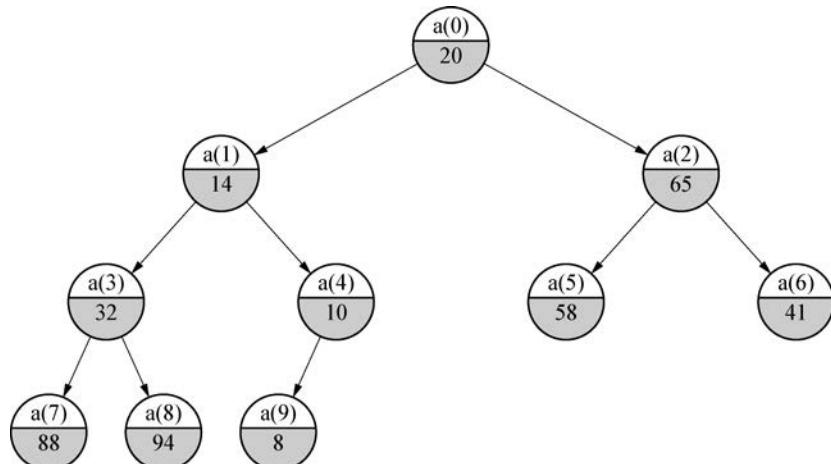


图 5-1 高度为 3 的完全二叉树

表 5-3 整理后结果

父节点下标	0	1	2	3	4	5	6	...	i
左子节点下标	1	3	5	7	9	11	13	...	$i * 2 + 1$
右子节点下标	2	4	6	8	10	12	14	...	$i * 2 + 2$

证明:用数学归纳法,当  $i=0$  时,结论显然成立,设  $i < k$  时,结论成立。

则  $i=k$  时,下标为  $k$  的节点的两个子节点应恰好在下标为  $k-1$  的节点的两个子节点的后面,又由归纳假设,下标为  $k-1$  的节点的两个子节点下标为  $2k-1$  和  $2k$ ,所以下标为  $k$  的节点的左子节点下标为  $2k+1$ ,右子节点下标为  $2k+2$ 。故  $i=k$  时,结论成立。

综上所述,结论得证。

(5) 本节第 2 个探究活动问题二提示:

如图 5-1 所示,由于  $h$  为 3 的完全二叉树有 8 种情况,整理后得到的结果如表 5-4 所示。

表 5-4 整理后结果

总节点数	8	9	10	11	12	13	14	15	n
非叶子节点数	4	4	5	5	6	6	7	7	$n//2$
叶子节点数	4	5	5	6	6	7	7	8	

证明:设出度为 0 的节点数为  $n_0$  个,出度为 1 的节点数为  $n_1$  个,出度为 2 的节点数为  $n_2$  个。

由完全二叉树的性质知  $n_1 = 0$  或  $n_1 = 1$ ,叶子节点个数为  $n_0$ ,非叶子节点个数为  $n_1 + n_2$ 。

考虑集合  $C = \{(u, v) | u \text{ 是 } v \text{ 的父节点}\}$ 。

按  $u$  计算  $C$  中元素个数得  $|C| = n_1 + 2n_2$ ;

按  $v$  计算  $C$  中元素个数得  $|C| = n_0 + n_1 + n_2 - 1$ ;

所以  $n_0 = n_2 + 1$ 。

那么  $n//2 = (n_0 + n_1 + n_2)//2 = n_2 + (1 + n_1)//2 = n_2 + n_1$ 。

故结论成立。

#### 4. 项目实施提示

本章的项目活动是人脸识别系统中的信息管理。首先引导学生讨论生活中是否还有类似的场景,使得学生有基本的感性认识。

譬如在爬虫系统中,在内存中维护着两个关于 URL 的队列,ToDo 队列和 Visited 队列,ToDo 队列存放的是爬虫从已经爬取的网页中解析出来的即将爬取的 URL,但是网页是互联的,很可能解析出来的 URL 是已经爬取过的,因此需要 Visited 队列来存放已经爬取过的 URL。当爬虫从 ToDo 队列中取出一个 URL 的时候,先和 Visited 队列中的 URL 进行比对,确认此 URL 没有被爬取过后就可以下载来分析,否则舍弃此 URL,从 ToDo 队列取出下一个 URL 继续工作。我们知道爬虫在爬取网页时,网页的量比较大,直接将所有的 URL 放入 Visited 队列很浪费空间且不现实。要想获得便于匹配的数据环境,需要对已有数据序列进行整理和优化。

尝试对生活中类似的例子加以分析,将生活中各类数据系列抽象为由简单到复杂的课堂教学实例,开展对于排序的认识和学习。

### 三、作业练习与提示

#### ■ 题目描述

- 对一组给定的数据: 35, 78, 41, 99, 23, 76, 56, 88, 21, 97, 45, 分别写出冒泡

排序和堆排序算法在处理上述数据时各轮的结果。

2. 有 6 位裁判为运动员评分,给出的分数分别为 53、50、68、51、64、62。采用直接选择排序算法对其进行排序,若完成第一轮时的结果为: 68、50、53、51、64、62,则完成第二轮时的结果是( )。

- A. 68、50、53、51、64、62
- B. 68、64、62、53、50、51
- C. 68、64、53、51、50、62
- D. 68、64、62、51、50、53

## ■ 作业练习参考答案

1. 冒泡排序的实现:

```
def BubbleSort(lst): # 冒泡排序
    for i in range(len(lst)-1):
        for j in range(len(lst)-i-1):
            if lst[j]>lst[j+1]:
                lst[j],lst[j+1]=lst[j+1],lst[j]
    print("第"+str(i+1)+"轮:",lst)
```

lst1=[35, 78, 41, 99, 23, 76, 56, 88, 21, 97, 45]

BubbleSort(lst1)

程序运行结果:

第 1 轮:[35, 41, 78, 23, 76, 56, 88, 21, 97, 45, 99]  
第 2 轮:[35, 41, 23, 76, 56, 78, 21, 88, 45, 97, 99]  
第 3 轮:[35, 23, 41, 56, 76, 21, 78, 45, 88, 97, 99]  
第 4 轮:[23, 35, 41, 56, 21, 76, 45, 78, 88, 97, 99]  
第 5 轮:[23, 35, 41, 21, 56, 45, 76, 78, 88, 97, 99]  
第 6 轮:[23, 35, 21, 41, 45, 56, 76, 78, 88, 97, 99]  
第 7 轮:[23, 21, 35, 41, 45, 56, 76, 78, 88, 97, 99]  
第 8 轮:[21, 23, 35, 41, 45, 56, 76, 78, 88, 97, 99]  
第 9 轮:[21, 23, 35, 41, 45, 56, 76, 78, 88, 97, 99]  
第 10 轮:[21, 23, 35, 41, 45, 56, 76, 78, 88, 97, 99]

堆排序的实现:

```
def siftdown(lst, i ,size):
    lchild= 2 * i + 1          # 确定 i 的左子节点的位置
    rchild= 2 * i + 2          # 确定 i 的右子节点的位置
    max= i
    if i < size // 2:           # 判断 i 不是叶子节点
        if lchild < size and lst[lchild] > lst[max]:      # 判断左子节点最大
            max= lchild                                # 记录左子节点位置
        if rchild < size and lst[rchild] > lst[max]:      # 判断右子节点最大
```

```

        max= rchild                                # 记录右子节点位置
        if max != i:                               # 判断最大的不是 i
            lst[max], lst[i]= lst[i], lst[max]
            print(" ", lst[i], "<-->", lst[max])
            siftdown(lst, max, size)               # 递归向下

def build_heap(lst):                         # 初始化堆
    size= len(lst)
    for i in range((size//2)-1, -1, -1):   # 最后一个起对非叶子节点进行向下筛选
        print("第"+ str(i+1)+ "个数"+ str(lst[i])+ "进行向下筛选")
        siftdown(lst, i, size)
        print(lst, "\n")

def chooseandrebuild(lst, i):                # 选择并重建堆
    lst[0], lst[i]= lst[i], lst[0]           # 交换堆顶与堆中最后一个元素
    print("第"+ str(len(lst)-i)+ "轮交换", lst[i], lst[0], ",然后对新二叉树的根进行向下筛选")
    siftdown(lst, 0, i)                     # 对新二叉树的根进行向下筛选
    print(lst, "\n")

def heap_sort(lst):                         # 主程序,对 lst 进行堆排序
    size= len(lst)
    print("第一步:初始化堆\n")
    build_heap(lst)                        # 初始化堆
    print("第二步:从后向前依次选择和重建堆\n")
    for i in range(size-1, -1, -1):       # 从后向前依次选择和重建堆
        chooseandrebuild(lst, i)

lst1= [35, 78, 41, 99, 23, 76, 56, 88, 21, 97, 45]
print(lst1, "\n")
heap_sort(lst1)

程序运行结果:
[35, 78, 41, 99, 23, 76, 56, 88, 21, 97, 45]
第一步:初始化堆
第 5 个数 23 进行向下筛选
97 <--> 23

```

[35, 78, 41, 99, 97, 76, 56, 88, 21, 23, 45]

第 4 个数 99 进行向下筛选

[35, 78, 41, 99, 97, 76, 56, 88, 21, 23, 45]

第 3 个数 41 进行向下筛选

76 <--> 41

[35, 78, 76, 99, 97, 41, 56, 88, 21, 23, 45]

第 2 个数 78 进行向下筛选

99 <--> 78

88 <--> 78

[35, 99, 76, 88, 97, 41, 56, 78, 21, 23, 45]

第 1 个数 35 进行向下筛选

99 <--> 35

97 <--> 35

45 <--> 35

[99, 97, 76, 88, 45, 41, 56, 78, 21, 23, 35]

第二步：从后向前依次选择和重建堆

第 1 轮交换 99 35，然后对新二叉树的根进行向下筛选

97 <--> 35

88 <--> 35

78 <--> 35

[97, 88, 76, 78, 45, 41, 56, 35, 21, 23, 99]

第 2 轮交换 97 23，然后对新二叉树的根进行向下筛选

88 <--> 23

78 <--> 23

35 <--> 23

[88, 78, 76, 35, 45, 41, 56, 23, 21, 97, 99]

第 3 轮交换 88 21，然后对新二叉树的根进行向下筛选

78 <--> 21

45 <--> 21

[78, 45, 76, 35, 21, 41, 56, 23, 88, 97, 99]

第 4 轮交换 78 23, 然后对新二叉树的根进行向下筛选

76 <--> 23

56 <--> 23

[76, 45, 56, 35, 21, 41, 23, 78, 88, 97, 99]

第 5 轮交换 76 23, 然后对新二叉树的根进行向下筛选

56 <--> 23

41 <--> 23

[56, 45, 41, 35, 21, 23, 76, 78, 88, 97, 99]

第 6 轮交换 56 23, 然后对新二叉树的根进行向下筛选

45 <--> 23

35 <--> 23

[45, 35, 41, 23, 21, 56, 76, 78, 88, 97, 99]

第 7 轮交换 45 21, 然后对新二叉树的根进行向下筛选

41 <--> 21

[41, 35, 21, 23, 45, 56, 76, 78, 88, 97, 99]

第 8 轮交换 41 23, 然后对新二叉树的根进行向下筛选

35 <--> 23

[35, 23, 21, 41, 45, 56, 76, 78, 88, 97, 99]

第 9 轮交换 35 21, 然后对新二叉树的根进行向下筛选

23 <--> 21

[23, 21, 35, 41, 45, 56, 76, 78, 88, 97, 99]

第 10 轮交换 23 21, 然后对新二叉树的根进行向下筛选

[21, 23, 35, 41, 45, 56, 76, 78, 88, 97, 99]

第 11 轮交换 21 21, 然后对新二叉树的根进行向下筛选

[21, 23, 35, 41, 45, 56, 76, 78, 88, 97, 99]

2. C。

## 四、核心概念介绍

排序: 将元素根据指定的序关系从大到小或从小到大排列。数据和序关系都可以是

任意的,排序仅仅是人类为了方便或某种需要对数据的重新整合。

**冒泡排序:**通过不断交换相邻两个元素实现将数据从小到大或从大到小的排列。冒泡排序是最简单的排序方法之一。

**完全二叉树:**一种特殊的二叉树。如果知道一棵完全二叉树的节点数,那么这棵树的结构是唯一的。因此,完全二叉树可以与列表一一对应,使得使用二叉树进行排序成为了可能。

**堆:**一种完全二叉树,对堆中的数据序关系,要求或者父节点都比子节点大,或者子节点都比父节点大。这样一个性质使得许多隐性的序关系都能通过结构得到表达,从而提高排序的效率。

**堆排序:**通过不断推出堆顶和重建堆来完成排序的方法。

## 五、教学参考资源

选择排序(selection sort)的基本思想:每一轮从  $n - i (i = 0, 1, 2, \dots, n - 2)$  个待排序的数据元素中选出最小的数据元素作为有序序列中位序为  $i$  的数据元素。对含有八个数 89, 71, 63, 78, 85, 93, 45, 97 的序列进行选择排序的过程如表 5-5 所示。

表 5-5 选择排序过程

初始值:89, 71, 63, 78, 85, 93, 45, 97	
第一轮:45, 71, 63, 78, 85, 93, 89, 97	从第 1 个开始的 8 个数中 45 最小,将 45 和第 1 个数 89 进行交换
第二轮:45, 63, 71, 78, 85, 93, 89, 97	从第 2 个开始的 7 个数中 63 最小,将 63 和第 2 个数 71 进行交换
第三轮:45, 63, 71, 78, 85, 93, 89, 97	从第 3 个开始的 6 个数中 71 最小,不用交换
第四轮:45, 63, 71, 78, 85, 93, 89, 97	从第 4 个开始的 5 个数中 78 最小,不用交换
第五轮:45, 63, 71, 78, 85, 93, 89, 97	从第 5 个开始的 4 个数中 85 最小,不用交换
第六轮:45, 63, 71, 78, 85, 89, 93, 97	从第 6 个开始的 3 个数中 89 最小,将 89 和第 6 个数 93 进行交换
第七轮:45, 63, 71, 78, 85, 89, 93, 97	从第 7 个开始的 2 个数中 93 最小,不用交换

```
def Py_selectionSort(myList): # 选择排序
    length= len(myList) # 获取 list 的长度
    for i in range(0,length- 1): # 这个循环负责设置选择排序进行的次数
        smallest= i # 默认设置最小值的位序为当前值
```

```

        for j in range(i+1, length): # j 为列表下标
            if myList[j] < myList[smallest]: # 如果找到
                smallest= j           # 则将当前位序赋给 smallest

        if i != smallest :# 如果 smallest 不等于 i,说明找到最小值,交换
            myList[i], myList[smallest]= myList[smallest], myList[i]

    return myList

```

在冒泡排序中,比较和移动的次数都很多,如果将整个待排序的记录分割成独立的两部分,其中一部分记录的关键字总比另一部分的关键字小,在小范围内进行比较和移动排序,就有可能减少处理次数,这就引出了快速排序的算法。

快速排序(quick sort)的设计思路是:假设待排序的序列为( $a_s, a_{s+1}, \dots, a_{t-1}, a_t$ ),首先任选一个数据元素  $a_i$  作为标准(称之为基准元素),一般情况下基准元素就取第一个数据元素  $a_s$ ,然后重排序列中的其他数据元素,所有关键字比它小的数据元素都排在它之前,所有关键字比它大的数据元素都排在它之后,这样就以  $a_i$  所在位序  $i$  为分界线,将序列分割成两个子序列( $a_s, a_{s+1}, \dots, a_{i-1}$ )和( $a_{i+1}, a_{i+2}, \dots, a_{t-1}, a_t$ ),这就完成了一趟排序。下一步,分别对这两个子序列操作,在每个子序列中,各选择一个基准元素,以它为界,又将该子序列再划分为两个子序列,如此划分下去,一直到各子序列中都只有一个数据元素,排序结束。

一轮快速排序的算法是:

- ① 设置两个变量  $i, j$ ,排序开始的时候: $i = 0, j = n - 1$ 。
- ② 以第一个数组元素作为关键数据,赋值给  $base$ ,即  $base = a_i$ 。
- ③ 从  $j$  开始向前搜索,即由后向前搜索( $j = j - 1$ ),找到第一个小于  $base$  的值  $a_j$ ,将  $a_j$  和  $a_i$  互换。
- ④ 从  $i$  开始向后搜索,即由前向后搜索( $i = i + 1$ ),找到第一个大于  $base$  的  $a_i$ ,将  $a_i$  和  $a_j$  互换。
- ⑤ 重复步骤③④,直到  $i = j$ 。

```

def Py_quickSort(myList,start,end):
    if start < end: # 判断 start 是否小于 end,如果为 false,直接返回
        i,j= start,end
        base= myList[i] # 设置基准元素

        while i < j:
            # 如果列表后边的数,比基准数大或相等,则前移一位直到有比基准数小的数出现
            while (i < j) and (myList[j] >= base):
                j= j- 1
            # 如找到且 i< > j 则把第 j 个元素赋值给第 i 个元素,此时表中 i,j 个元素相等

```

```

        if i != j : myList[i]=myList[j]

        # 同样的方式比较前半区
        while (i < j) and (myList[i] <= base) :
            i= i+1
            if i != j: myList[j]=myList[i]

# 做完第一轮比较之后,列表被分成了两个半区,并且 i=j,需要将这个数设置回 base
myList[i]=base

# 递归前后半区
if start< i-1 : Py_quickSort(myList, start, i-1)
if end> j+1: Py_quickSort(myList, j+1, end)

return myList

```

冒泡排序算法中,每次比较如果发现较小的元素在后面,就交换两个相邻的元素。而选择排序算法的改进在于:先不急于调换位置,从头开始逐个检查,看哪个数最小就记下该数所在的位置 P,等一轮扫描完毕,再把位置 P 的元素和第一个元素对调,这时最小的数据就换到了最前面的位置。所以,选择排序每扫描一遍序列,只需要一次真正的交换,而冒泡排序可能需要很多次交换。当然它们比较的次数是一样的。所以选择排序算法优于冒泡排序算法。编一个程序,对随机产生的一列整数进行排序,比较一下冒泡排序、选择排序、快速排序的执行效率。

程序设计思路:首先调用 random.randint(1,10000)产生 8000 个整数存入列表中,然后通过 deepcopy(Python 中的深度复制,即将被复制对象完全再复制一遍作为独立的新个体单独存在,所以改变原有被复制对象不会对已经复制出来的新对象产生影响)拷贝出 3 个列表分别用于冒泡排序、选择排序和快速排序。为了统计时间,通过调用 time.time()来记录排序起始时间和结束时间,从而算出排序所花费时间。

```

import time
import random
from copy import *

largeList=[]
for i in range(8000): # 随机产生 8000 个小于 10000 的整数
    largeList.append(random.randint(1,10000))
largeList1= deepcopy(largeList) # 深度拷贝用于冒泡排序
largeList2= deepcopy(largeList) # 深度拷贝用于选择排序
largeList3= deepcopy(largeList) # 深度拷贝用于快速排序

```

```
start= time.time()
Py_bubbleSort(largeList1)
end= time.time()
print("冒泡排序花费时间(秒): % 10.8f" % (end - start))
```

```
start= time.time()
Py_selectionSort(largeList2)
end= time.time()
print("选择排序花费时间(秒): % 10.8f" % (end - start))
```

```
start= time.time()
Py_quickSort(largeList3, 0, 7999)
end= time.time()
print("快速排序花费时间(秒): % 10.8f" % (end - start))
```

程序运行结果如下：

冒泡排序花费时间(秒)：5.16172862

选择排序花费时间(秒)：1.97527266

快速排序花费时间(秒)：0.01905584

由此看出：对随机产生的一列整数进行排序，冒泡排序不如选择排序，而快速排序远远优于另外两个排序方法。

在许多应用场合，不需要将整个数列（假设  $n$  个数）完全排序，只需要找到前  $k$  个元素就可以了。一个简单的想法是，利用排序算法，将  $n$  个数排好序，然后取前  $k$  个。这样做结果当然是对的，但效率不高，无论是选择排序还是冒泡排序，都要进行  $O(n^2)$  次比较操作。当然，如果用选择排序算法，只需要选出前  $k$  个元素就好，如果用冒泡排序算法，只需要控制运行  $k$  轮即可，这样只需执行  $O(n * k)$  次比较操作。而使用堆排序让求解 Top-K 问题的效率提高到  $O(n + k * \log_2 n)$ ，这和上面的  $O(n^2)$  或  $O(n * k)$  相比，是一个巨大的进步。

## 六、教学参考案例

### ■ 参考案例

#### 冒泡排序

上海市控江中学 覃 怡  
(1课时)

##### 1. 学科核心素养

在解决计算篮球赛成绩排名问题的过程中，合作解决问题，愿意与团队成员共享信息，实现信息的最大价值。（信息意识）

分析计算篮球赛成绩排名问题,采用计算机可以处理的方式界定问题、抽象特征、建立结构模型、合理组织数据。(计算思维)

能运用数字化学习资源与学习工具开展自主学习、协同工作、知识分享与创新创造。(数字化学习与创新)

在分析计算篮球赛成绩排名问题的过程中,思考生活中需要排序的应用场景,分析用计算机进行数据处理的过程,具有积极学习的态度和理性判断的能力,负责任地使用信息。(信息社会责任)

## 2.《课程标准》要求

结合生活实际,理解数据结构的概念,认识数据结构在解决问题过程中的重要作用。

通过实现数据的排序和查找,体验迭代和递归的方法,理解算法与数据结构的关系。

## 3.学业要求

学生能够运用生活中的实例描述数据的内涵与外延,能够将有限制条件的、复杂生活情境中的关系进行抽象,用数据结构表达数据的逻辑关系。

能够针对限定条件的实际问题进行数据抽象,运用数据结构合理组织、存储数据,选择合适的算法(如排序、查找、迭代、递归等)编程实现、解决问题。

能够分析数据与社会各领域间的关系,自觉遵守相应的伦理道德和法律法规。

## 4.教学内容分析

数据结构是计算机存储、组织数据的方式,是相互之间存在一种或多种特定关系的数据元素的集合。通常情况下,选择合理的数据结构,配合算法,可以使得程序的运行更快,并提升数据在计算机中存储的效率。

排序是教科书中第五章数据结构应用的内容。教科书中包含了排序概念、冒泡排序算法、Top-K 算法。本节课是排序的第 1 课时。

## 5.学情分析

学生已经完成了必修 1 数据与计算课程的学习,自主选择学习选择性必修 1 数据与数据结构课程,具有较强的逻辑思维能力,并具备了一定的编写程序的基础。在学习本章内容之前,已经完成了线性表、栈、队列、二叉树的学习,熟悉了一些常用数据结构的基本形态,对其在计算机程序中的基本存在形式及作用有了较深入的理解。

## 6.教学目标

- 分析生活实例,知道排序的概念,并采用计算机可以处理的方式界定问题、抽象特征、建立模型。

- 通过自主学习、小组讨论、知识分享,理解冒泡排序算法。

- 利用 Python 编写冒泡排序的程序,解决计算篮球赛成绩排名问题。

- 思考生活中需要排序的应用场景,分析用计算机进行数据处理的过程,具有积极学习的态度和理性判断的能力,负责任地使用信息。

## 7.教学重难点

- 教学重点:理解冒泡排序算法。

- 教学难点:利用 Python 编写冒泡排序的程序。

## 8. 教学策略分析

### (1) 阅读文本,理解概念,提升能力。

在学习过程中,让学生阅读文本,找出关键词。看似简单的阅读和理解,实则可培养学生的自主学习能力,有助于学生在以后学习新知的过程中,关注文本阅读和明晰关键词。

### (2) 过程描述,动手实践,加深理解。

“冒泡排序”看似是简单的相邻两数的依次比较,但是在利用计算机程序实现的过程中,存在很多细节问题。通过观看动画演示、描述排序过程、用学具模拟排序过程、填写表格等从不同角度将冒泡排序进行多次复现,加深学生对冒泡排序的理解,为后续的绘制流程图和编写程序打下扎实的基础。

## 9. 教学过程设计(见图 5-2 和表 5-6)

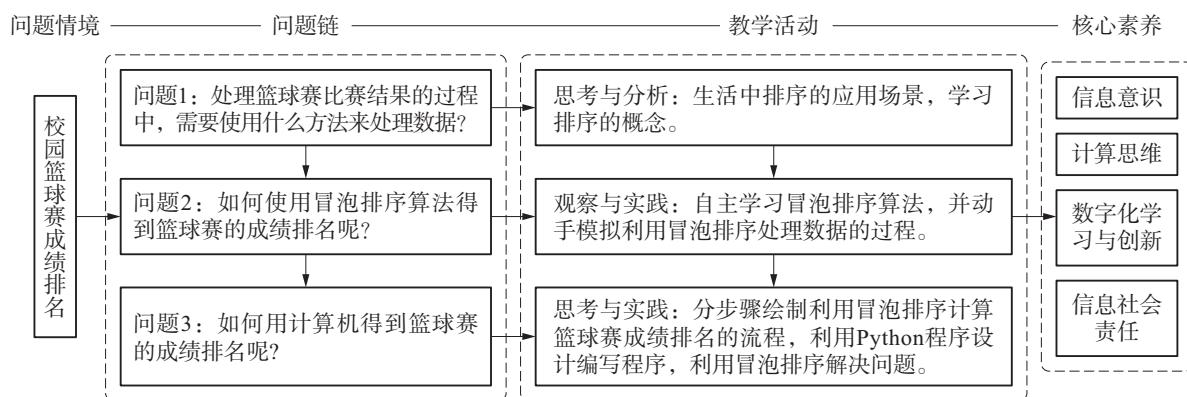


图 5-2 教学过程

表 5-6 教学过程设计表

教学环节	教师活动	学生活动	设计意图
情境导入	<p>校园篮球赛</p> <p>问题情境：每年 10 月都是学校的阳光运动季,篮球赛是其中最热门的比赛之一,报名班赛和团队赛的队伍数量众多。小申作为校学生会体育部的部长,正在思考如何在每轮比赛后更快捷地处理比赛结果,及时公布本轮比赛成绩排名。</p> <p>问题 1：处理篮球赛比赛结果的过程中,需要使用什么方法来处理数据?</p> <p>(1) 提问：如何得出篮球赛的成绩排名呢?生活中还有哪些情况与此类似呢?</p> <p>(2) 阅读教科书“一、排序的概念”,找出关键词并记录下来</p>	<p>【思考、分析】</p> <p>(1) 小组讨论,尝试用自己的语言描述“什么是排序”。</p> <p>(2) 找出教科书中排序概念中的关键词</p>	<p>从生活实例引出排序算法,感受排序在解决实际问题中的作用。初步体验用计算机可以处理的方式界定问题、抽象特征</p>

续表

教学环节	教师活动	学生活动	设计意图
认识冒泡排序	<p>冒泡排序算法 问题2:如何使用冒泡排序算法得到篮球赛的成绩排名呢?</p> <p>(1) 观察:播放演示动画,冒泡排序的过程是怎么样的? (2) 动手实践:教师提供学具(写有10支队伍篮球赛得分的卡片),请同学们利用学具演示冒泡排序过程,并完成学习单中表格的填写。 (3) 交流:请同学们用自己的语言总结冒泡排序的特点</p>	<p><b>【观察、实践】</b> (1) 观看演示动画,小组交流冒泡排序的过程。 (2) 小组合作完成,根据冒泡排序过程,利用学具模拟演示篮球赛成绩排序的完整过程,并填写学习单中的表格。 (3) 小组讨论冒泡排序的特点。 (部分)小组分享讨论的结果</p>	<p>从观看动画、小组交流、利用学具模拟过程,细化冒泡排序的过程,加深对冒泡排序的理解。</p> <p>通过填写表格,为下一个环节中如何“冒起一个数”做好铺垫。</p> <p>过程中注重合作解决问题,共享信息,实现信息的更大价值</p>
冒泡排序的实现	<p>冒泡排序的实现 问题3:如何用计算机得到篮球赛的成绩排名呢?</p> <p>(1) 当需要对10支队伍的篮球赛得分按照递减顺序进行排序时,下图中的“不断冒起一个最小数”需要进行几次? 用流程图描述多次“冒起一个最小数”。</p> <p>(2) 对10支队伍的篮球赛得分按照递减顺序进行排序,用流程图描述“冒起一个最小数”。</p> <p>(3) 根据流程图,利用Python语言编写冒泡排序的程序,并进行调试运行。</p> <pre>def BubbleSort(arr):     for i in range(1, len(arr)):         for j in range(____):             if arr[____] &gt; arr[____]:                 _____     return arr</pre>	<p><b>【思考与实践】</b> (1) 小组讨论,回答问题,绘制流程图。 (预设答案:10支队伍的篮球赛得分排序问题需要进行9轮“冒泡”)</p> <p>(2) 小组讨论,绘制流程图。</p> <p>(3) 小组讨论,将代码补充完整并进行上机实践</p>	<p>如何“冒起一个数”是本节课的难点之一,在前面的环节中通过填写表格,让学生观察到每轮比较的起点和终点。</p> <p>分析用计算机进行数据处理的过程,具有积极学习的态度和理性判断的能力,负责任地使用信息</p>

教学环节	教师活动	学生活动	设计意图
课后思考	练习: 观察冒泡排序的过程, 设计一个从左往右进行比较的程序。 思考(可选): 冒泡排序的程序能不能优化? 提示: 当某一轮没有出现交换, 说明数据已经按照特定的序排列好了。 拓展学习(可选): 排序的方法多种多样, 请利用相关书籍或者网络, 开展自主学习, 了解其他排序算法, 并比较不同排序算法之间的特点		

### 附: “冒泡排序”学习单

每年10月都是学校的阳光运动季, 篮球赛是其中最热门的比赛之一, 报名班赛和团队赛的队伍数量众多。小申作为校学生会体育部的部长, 正在思考如何在每轮比赛后更快捷地处理比赛结果, 及时公布本轮比赛成绩排名。

【问题1】处理篮球赛比赛结果的过程中, 需要使用什么方法来处理数据?

1. 在处理篮球赛比赛结果的过程中, 使用\_\_\_\_\_。

生活中还有哪些类似的情况, 如:\_\_\_\_\_。

2. 阅读教科书“一、排序的概念”, 找出关键词并记录下来。

【问题2】如何使用冒泡排序算法得到篮球赛的成绩排名呢?

1. 观看动画, 体验冒泡排序的过程。你可以适当做些笔记:

请将老师发给你的学具中的数据记录在表5-7中。

表5-7 待排序数据

序号									
数值									

2. 在模拟的过程中, 按照冒泡排序的思想方法, 填写表5-8。(说明: 表格的最后两行分别填写比较的最后两轮)

表5-8 排序比较过程记录表

第几轮	“从右往左比较相邻数”的第一个数	“从右往左比较相邻数”的最后一个数	共比较次数
第1轮			
第2轮			
第3轮			
.....	.....	.....	.....
第_____轮			
第_____轮			

### 【问题3】如何用计算机得到篮球赛的成绩排名呢？

<p>(1) 当需要对10支队伍的篮球赛得分按照递减顺序进行排序时,下图中的“不断冒起一个最小数”需要进行_____次</p>	用流程图描述多次“冒起一个最小数”:
(2) 对10支队伍的篮球赛得分按照递减顺序进行排序,用流程图描述“冒起一个最小数”	
(3) 根据流程图,将程序补充完整,并进行上机调试	<pre>def BubbleSort(arr):     for i in range(1, len(arr)):         for j in range(______):             if arr[_____] &gt; arr[_____]:                 _____      return arr</pre>

## 第二节      查      找

### 一、教学目标与重点

#### 教学目标:

- 通过案例分析,理解查找的概念及生活中排序的逻辑思想,在探究活动中掌握顺序查找和对分查找的原理及方法,并能用代码实现,感受排序对于查找的作用。
- 在项目实践中,理解基于二叉树的查找效率与树高成正比,因而寻求“最矮的”二叉树很有意义。理解当节点数为n,最矮二叉树的高度约为 $\log_2 n$ ,体会数据结构类型对于数据处理的支持作用。

- 在原理分析中,理解“平衡二叉树”的概念。通过实现一种平衡二叉树(例如 AVL 树),完成支持  $\log_2 n$  量级的查找,感受计算思维在追求更优解过程中的思想内涵。

#### 教学重点:

- 掌握基于顺序表查找的两种方法。
- 掌握二叉搜索树的原理及方法。

## 二、教学实施与评价

### 1. 教学活动建议

教科书中主要对查找的概念、顺序查找的基本操作以及二叉搜索树三部分进行了介绍。

查找表是由同一类型的数据元素构成的序列。查找就是根据给定的某个值,在查找表中确定一个其关键字等于给定值的数据元素的操作。

教科书中实现了顺序存储的线性表的顺序查找算法,不妨请学生自行设计一个用单链表作存储结构的顺序查找算法。

对分查找的条件:线性表必须采用顺序结构(否则效率很低);表中数据元素按关键字有序排列。因为有了前面排序中应用递归思想的基础,在查找中可以进一步让学生思考如何使用递归算法实现对分查找。

下面是使用递归思想实现查找的参考:

假设 key 为给定值,序列 a 存放了 n 个已按升序排序的数据元素(假设关键字即为数据元素的值),查找范围是 [low,high]。

(1) 确定该区间的中间位置 mid。

(2) 将给定值 key 与 a[mid] 进行比较,有如下三种情况:

① a[mid] = key,则返回此位置 mid。

② a[mid] > key,则在新的范围 [low,mid - 1] 中进行递归调用查找。

③ a[mid] < key,则在新的范围 [mid + 1,high] 中进行递归调用查找。

```
def Py_binarySearchByRecurse(list, low, high, key): # 递归的二分查找
    if low <= high:                      # 查找区间存在一个及以上元素
        mid= (low + high) // 2 # 求中间位置
        if list[mid]== key:
            return mid          # 如果找到,则返回所在位序
        elif list[mid] > key: # 在 list[low..mid- 1] 中递归查找
            return Py_binarySearchByRecurse(list, low, mid- 1, key)
        else:                  # 在 list[mid+ 1..high] 中递归查找
```

```

        return Py_binarySearchByRecurse(list, mid + 1, high, key)
    else :
        return - 1 # 否则返回- 1

```

二叉树在查找问题的高效求解上也扮演重要角色,一般地称为“二叉搜索树”(binary search tree, BST)。二叉搜索树没有完全二叉树那样的结构性要求,但对节点中的数据关系有特别的要求——父节点大于左子节点,小于右子节点(注意这种要求的递归性)。实现二叉搜索树最关键的操作是数据项的插入和删除。

掌握二叉搜索树的查找数据、插入数据、删除数据的实现方法,并编程实现。

理解当节点数为 n,最矮二叉树的高度约为  $\log_2 n$ 。理解平衡二叉树的概念,通过实现一种平衡二叉树(例如 AVL 树),实现  $O(\log_2 n)$  效率的查找。

## 2. 教学过程安排(见表 5-9)

表 5-9 教学过程设计表

课时	教学环节	教师活动	学生活动
第 1 课时 (查找概念及基于顺序表的查找)	1. 导入	列举生活中的查找问题,引发学生对查找方法的思考	思考如何解决生活中的查找问题
	2. 查找的概念	引导学生认识什么是查找	阅读教科书,回答问题
	3. 顺序查找	布置活动:理解顺序查找原理并编写程序实现	阅读教科书,程序实现
	4. 对分查找	布置活动:分析对某个数据序列进行对分查找的过程,理解对分查找的精髓	倾听、思考
	5. 对分查找的基本操作	布置活动:编写程序实现对分查找	编程实现
	6. 体验思考	完成本节第一个体验思考	描述对分查找过程
	7. 总结	比较顺序查找与对分查找的异同	
第 2 课时 (基于二叉搜索树的查找)	1. 导入	回顾查找概念	交流
	2. 二叉搜索树的概念	引导学生认识什么是二叉搜索树。 (1) 什么是二叉搜索树? (2) 二叉搜索树有什么特性?	自我阅读、思考分析、讨论
	3. 体验思考	布置活动:构造一棵简单的二叉搜索树,并说明其特点	通过交流讨论完成构造
	4. 编程实现二叉搜索树	布置活动:编写程序实现二叉搜索树	思考分析,编程实现
	5. 用二叉搜索树实现查找	布置活动:编写程序实现基于二叉搜索树的查找	思考并编程实现
	6. 总结	二叉搜索树的概念、特性及构造和查找方法	
第 3 课时 (项目综合实践)	1. 导入	回顾第一章,引出数据流检测问题	倾听、思考
	2. 活动:用递归实现查找	布置活动:用递归思想完成程序,实现对分查找	编程实现、对比分析

课时	教学环节	教师活动	学生活动
	3. 活动:认识几种查找方法的效率	布置任务:争取有较高执行效率,即判定一个编码是否新来所用的比较次数尽量少	思考分析、编程实现
	4. 完成项目报告	布置活动:以小组为单位完善项目报告	整理、归纳
	5. 总结	比较不同的数据结构对于查找效率的影响	

本章的项目评价在完成各项目任务的过程中进行,着重观察学生对于查找及递归思想的理解。教师也可设计实践评价量规,重点考查学生在本节项目活动中运用查找算法解决问题的能力,如表 5-10 所示。

表 5-10 实践评价量规参考示例

	评价内容	是否达成	反思与建议
项目主题	有限存储下的数据流检测问题		
项目分析	(1) 使用什么样的数据结构来进行有限存储; (2) 数据流中不同的数据超过了设定,替换算法的选择		
方案设计	(1) 明确数据分析的目标,分解任务; (2) 选择合适的数据结构,实现数据组织和存储; (3) 选择合适的排序方法并寻找是否有匹配(查找)的数据		
探究实践	(1) 检查结果的正确性和实现效率; (2) 能用 Python 语言编程实现系统; (3) 能用恰当的方法实现算法的优化		
成果交流	(1) 分享各自的作品,讨论数据处理方式和执行流程是否合理、正确; (2) 讨论各自的程序,并修改完善		

### 3. 思考探究提示

(1) 本节第 1 个体验思考提示:

192、224、251。

(2) 本节第 1 个探究活动提示:

```
def Py_binarySearchByRecurse(list, low, high, key): # 递归的二分查找
    if low <= high:          # 查找区间存在一个及以上元素
        mid= (low + high) // 2 # 求中间位置
        if list[mid]== key:
            return mid         # 如果找到,则返回所在位序
        elif list[mid] > key:   # 在 list[low..mid- 1]中递归查找
            return Py_binarySearchByRecurse(list, low, mid- 1, key)
        else:                  # 在 list[mid+ 1..high]中递归查找
```

```
    return Py_binarySearchByRecurse(list, mid + 1, high, key)
else :
    return -1 # 否则返回-1
```

(3) 本节第 2 个体验思考提示:

答案参见图 5-3。

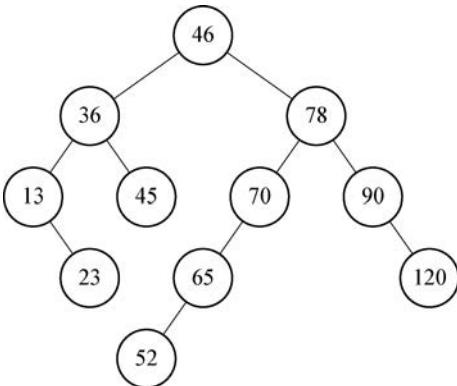


图 5-3 第 2 个体验思考答案

#### 4. 项目实施提示

按照本章的项目要求分组讨论实现方案。综合考量已经学习过的几种数据结构,教师可以引导学生重点讨论基于线性表、二叉树和哈希表的实施办法。

首先组织 n 个元素模拟数据流,基于上一节的学习,可对数据进行更科学的组织(排序),讨论选择合适的替换算法,将各部分任务进行拆解,学生以小组为单位完成方案的制订,包括数据存储(选择合适的数据结构)、整理(排序或按一定策略存放),随着数据不断增加,如何更科学地管理这些数据(替换)……

将总的项目模块化,降低难度,分块进行算法设计并编写程序实现。在项目规划过程中可以以发现问题(数据流检测问题)、分析问题(设计合理的实现方案)到解决问题(选用合适的数据结构并程序实现)为切入视角,将每部分设计成小的项目活动,通过完成一个个项目活动,体会数据结构的要义。

### 三、作业练习与提示

#### ■ 题目描述

1. 判断以下说法的对错:

- ( ) 顺序查找时,被查找的数据必须有序;
- ( ) 对分查找时,被查找的数据不一定有序;
- ( ) 顺序查找总能找到要查找的关键字;
- ( ) 一般情况下,对分查找的效率较高。

2. 设计一个用单链表作存储结构的顺序查找算法。

3. 现有一个包含 7 个元素的序列 A, 元素依次为 68、149、273、321、475、591、666。若采用对分查找法在该数组中查找数据 666, 需要查找几次?

4. 为数列 10、9、8、4、11、5、3、2、6、7 按照给出的顺序建立一棵二叉搜索树, 然后试图查找 1、7、11、2 等数字, 并给出这些数字的查找次数。

### ■ 作业练习参考答案

1.  $\times \times \times \checkmark$ 。

2. 用单链表作存储结构的顺序查找算法:

```
def search(self, item):  
    cur = self._head  
    while cur != None:  
        if cur.item == item:  
            return True  
        cur = cur.next  
    return False
```

3. 需要 3 次。

68、149、273、321、475、591、666

1          2          3

4. 二叉搜索树如图 5-4 所示。

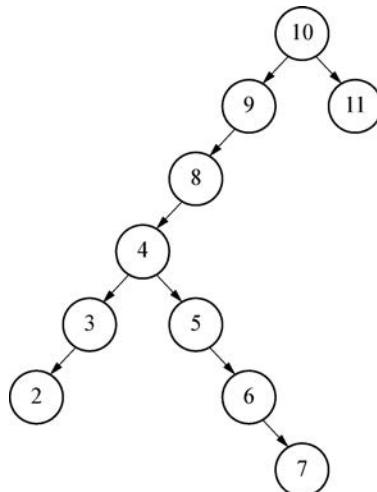


图 5-4 二叉搜索树

查找 1 时, 依次比较数据为 10, 9, 8, 4, 3, 2。没有找到, 查找次数为 6。

查找 7 时, 依次比较数据为 10, 9, 8, 4, 5, 6, 7。找到, 查找次数为 7。

查找 11 时, 依次比较数据为 10, 11。找到, 查找次数为 2。

查找 2 时, 依次比较数据为 10, 9, 8, 4, 3, 2。找到, 查找次数为 6。

参考程序如下:

```
class TreeNode:
```

```

def __init__(self, data, left= None, right= None):
    self.data= data
    self.left= left
    self.right= right

class BSTree:
    def __init__(self):
        self.root= None

    def search(self, key):
        cur= self.root
        time= 0
        while cur:
            time += 1
            if key== cur.data:
                return [cur, time]
            elif key > cur.data:
                cur= cur.right
            else:
                cur= cur.left
        return [None, time]

    def insert(self, key):
        new= TreeNode(key)
        if not self.root:
            self.root= new
            return
        cur= self.root
        cpar= None
        while cur:
            if cur.data== key:
                return
            elif cur.data < key:
                cpar= cur
                cur= cur.right
            else:
                cpar= cur

```

```

        cur= cur.left
    if cpar.data > key:
        cpar.left= new
    else:
        cpar.right= new

def delete(self, key):
    cur, cpar= self.root, None
    while cur and key != cur.data:
        cpar= cur
        if key < cur.data:
            cur= cur.left
        else:
            cur= cur.right
        if not cur:
            return
    if cur.left is None:
        if cpar is None:
            self.root= cur.right
        elif cur== cpar.left:
            cpar.left= cur.right
        elif cur== cpar.right:
            cpar.right= cur.right
        else:
            mostright= cur.left
            while mostright.right:
                mostright= mostright.right
            mostright.right= cur.right
            if cpar is None:
                self.root= cur.left
            elif cpar.left== cur:
                cpar.left= cur.left
            else:
                cpar.right= cur.left

Py_tree= BSTree()
data= [10, 9, 8, 4, 11, 5, 3, 2, 6, 7]

```

```

for i in data:
    Py_tree.insert(i)

f1= Py_tree.search(1)
f7= Py_tree.search(7)
f11= Py_tree.search(11)
f2= Py_tree.search(2)
print('找 1 用了 % d 次 '% f1[1])
print('找 7 用了 % d 次 '% f7[1])
print('找 11 用了 % d 次 '% f11[1])
print('找 2 用了 % d 次 '% f2[1])

```

## 四、核心概念介绍

**查找:**给定一个需要的元素,通过计算机实现一个完整的战略,使得在一些数据中能找到这个元素,或者提示不存在这样的元素。

**顺序查找:**从头到尾依次判断该元素是否为我们需要的元素。优点是易于理解而且不会错判漏判,缺点是时间复杂度高。

**对分查找:**对分查找的一个必要条件是数据是按特定的序关系排列的。通过每次将待查找的数据一分为二快速定位数据。优点是速度快,缺点是对数据的排列有严格的要求。

**二叉搜索树:**一棵存储数据的二叉树,要求父节点比左子节点及其子孙大,比右子节点及其子孙小。优点是查找简单,缺点是插入和删除的操作复杂。

## 五、教学参考资源

### 插值查找法:

插值查找法(interpolation search)又叫作插补查找法,是二分查找法的改进版。它是按照数据位置的分布,利用公式预测数据所在的位置,再以二分法的方式渐渐逼近。使用插值法是假设数据平均分布在数组中,而每一项数据的差距相当接近或有一定的距离比例。插值法的公式为:

$$\text{Mid} = \text{low} + ((\text{key} - \text{data}[\text{low}]) / (\text{data}[\text{high}] - \text{data}[\text{low}])) * (\text{high} - \text{low})$$

其中 key 是要查找的键,data[high]、data[low]是剩余待查找记录中的最大值和最小值,假设数据项数为 n,其插值查找法的步骤如下:

步骤 1:将记录从小到大的顺序给予 1,2,3,⋯,n 的编号。

步骤 2:令 low = 1,high = n。

步骤 3:当 low < high 时,重复执行步骤 4 和步骤 5。

步骤 4:令 Mid = low + ((key - data[low]) / (data[high] - data[low])) \* (high - low)。

步骤5:若  $\text{key} < \text{key}_{\text{Mid}}$  且  $\text{high} \neq \text{Mid} - 1$ , 则令  $\text{high} = \text{Mid} - 1$ 。

步骤6:若  $\text{key} = \text{key}_{\text{Mid}}$ , 表示成功查找到键值的位置。

步骤7:若  $\text{key} > \text{key}_{\text{Mid}}$  且  $\text{low} \neq \text{Mid} + 1$ , 则令  $\text{low} = \text{Mid} + 1$ 。

### 插值查找法的分析:

(1) 一般而言,插值查找法优于顺序查找法,数据的分布越平均,查找速度越快,甚至可能第一次就找到数据。此法的时间复杂度取决于数据分布的情况,平均而言优于  $O(\log_2 n)$ 。

(2) 使用插值查找法数据需先经过排序。

——参考《图解数据结构——使用 Python》,吴灿铭,清华大学出版社

## 六、教学参考案例

### ■ 参考案例

#### 查找算法初步研究

上海理工大学附属中学 张烨琼

(1课时)

##### 1. 学科核心素养

能够根据查找问题的需要,观察问题中的数据;能够敏锐认识到顺序查找和对分查找数据的异同点,选择合适的抽象数据类型;在小组讨论、分享的过程中,愿意与团队成员共享信息,实现信息的最大价值。(信息意识)

在编程活动中能够采用计算机可以处理的方式界定问题、抽象特征、建立循环嵌套分支结构的模型、采用列表的数据类型。通过判断、分析与综合各种信息资源,运用合理的顺序查找、对分查找和递归的算法形成解决问题的方案。(计算思维)

能够选择专业的网站获取相关的学习资源,愿意自主选择合适的平台完成编程活动,通过对数字资源和平台的综合利用创造性地完成项目任务。(数字化学习与创新)

##### 2.《课程标准》要求

通过列举实例,认识到抽象数据类型对数据处理的重要性,理解抽象数据类型的概念,了解二叉树的概念及其基本操作方法。

通过实现数据的排序和查找,体验迭代和递归的方法,理解算法和数据结构的关系。

##### 3. 学业要求

学生能够运用生活中的实例描述数据的内涵与外延,能够将有限制条件的、复杂生活情境中的关系进行抽象,用数据结构表达数据的逻辑关系。

能够针对限定条件的实际问题进行数据抽象,运用数据结构合理组织、存储数据,选择合适的算法编程实现、解决问题。

能够从数据结构的视角审视程序,解释程序中数据的组织形式,评判其中数据结构运用的合理性。

能够分析数据与社会各领域间的关系,自觉遵守相应的伦理道德和法律法规。

#### 4. 教学内容分析

数据结构是计算机存储、组织数据的方式,是相互之间存在一种或多种特定关系的数据元素的集合。通常情况下,选择合理的数据结构,配合算法,可以使得程序的运行更快,并提升数据在计算机中存储的效率。查找是教科书中第五章第二节的内容。教科书在之前的章节里已经介绍过一些常见的数据结构,包括数组、链表和二叉树,学生也完成了相关的程序应用实例。本节将通过对两个查找算法示例的研究分析,展示数据结构对于算法设计的意义,以及使用不同的算法产生的效果差异。

#### 5. 学情分析

学生已经完成了数据与计算课程的学习,了解过顺序查找和对分查找的基本思想;自主选择学习数据与数据结构的学生,一般具有较强的逻辑思维能力,并具备了一定的使用Python语言编写程序的基础。

通过数据结构前面的内容学习,已经理解线性表和二叉树的概念,掌握了数组、链表、栈、队列和二叉树的相关操作,并且能够使用递推和迭代的思想解决问题。

#### 6. 教学目标

- 通过体验生活中的查找问题,理解顺序查找和对分查找的基本思想,总结两种算法对于数据结构的不同要求,认识到抽象数据类型对数据处理的重要性。
- 经历程序代码编写实践活动,掌握循环嵌套分支的结构组合和递归思想在具体问题中的应用,体验 Python 中列表的使用对算法设计的作用,理解算法和数据结构的关系。
- 通过对比顺序查找和对分查找的效率及优缺点,厘清两种算法的适用情况,感受算法设计的魅力,培养学生的计算思维,提升其信息技术学科的学习动力。

#### 7. 教学重难点

- **教学重点:**理解顺序查找和对分查找对于数据结构的要求;认识两种算法的效率及优缺点。
- **教学难点:**实现顺序查找和对分查找的程序设计。

#### 8. 教学策略分析

本节课主要采用师生互动、实践体验、小组讨论、分享交流的方式进行教学。整节课以生活情境引入,通过问题链,引导学生一步步深入思考查找算法的思想、条件、效率和优缺点,再实现具体问题的解决。

(1) 从实际问题到算法思想,从算法思想到程序实现,从程序实现到不同算法的效率对比。

本节课通过三组有难度的任务,明晰两种查找算法的基本思想、适用条件,让学生体验算法实现的过程,学会从算法效率的角度分析算法的优缺点。

(2) 设计问题链,提供学习材料,由浅入深层层破解。

本节课通过多组问题链,引导学生由浅入深地掌握相关算法的思想、使用条件及实现程序编写。借助学习单,提供学习支架,培养学生自主学习能力。

#### 9. 教学过程设计(见图 5-5 和表 5-11)

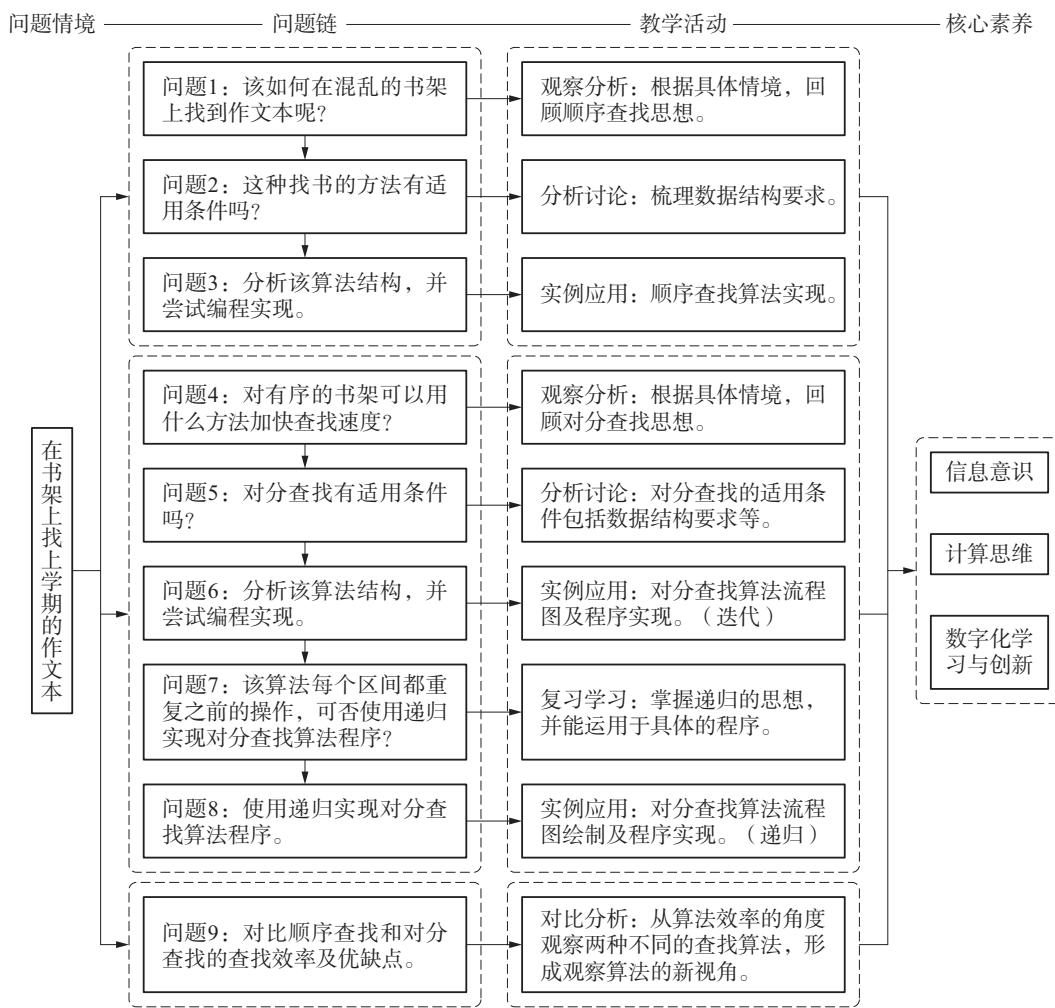


图 5-5 教学过程

表 5-11 教学过程设计表

教学环节	教师活动	学生活动	设计意图
情境导入 一	<p>情境描述：老师要求同学们回家把上学期的作文本找出来。（说明查找的概念）</p> <p>小明的书架：（混乱）</p>  <p>问题1：请问小明该如何在混乱的书架上找到作文本呢？（可参考学习单回答）</p> <p>问题2：这种找书的方法有适用条件吗？（可参考学习单回答）</p> <p>追问：如果小明在找作文本的时候，他的妈妈正在打扫书架，妈妈一会儿把书拿下</p>	<p><b>【讨论,分析】</b></p> <p>(1) 帮助学生回顾之前学习过的顺序查找思想，整理顺序查找实现的具体步骤。</p> <p>(2) 学生可能根据之前的学习经历回答：查找范围有限。这个答案在数据与计算模块学习的时候是对的，但是在学习过数据结构后，就有些不准确、不完整了。教师可以追问。</p> <p>参考答案：</p> <p>(1) 查找对象最好是静态的数据集；</p> <p>(2) 适用的数据结构为线性表，包括数组、链表等</p>	<p>通过对实际问题的观察与分析，整理顺序查找的思想以及适用条件，明确顺序查找主要用于静态数据集，数据结构首选线性表。</p> <p>在该问题的分析中，培养学生对实际生活中数据结构特点的观察，以及体会不同的</p>

教学环节	教师活动	学生活动	设计意图
	来,一会儿又把其他书放上去。你觉得小明用这个方法找得到他的作文本吗?那么你能否更加准确地概括出使用这种查找方法的条件?		数据结构对于算法设计的影响,加深学生的信息意识
顺序查找实现	<p>顺序查找的程序实现</p> <p>问题3:请同学们观察顺序查找的基本思路,分析该算法可以用怎样的结构组合来实现,并尝试完成程序填空。(程序见学习单)</p> <p>展示学生代码,评价完成情况</p>	<p><b>【回答、上机编程】</b> 该程序结构为循环嵌套分支结构,学生较为熟悉。 程序代码参考答案:</p> <pre>def sequentialSearch(list, key):     pos = 0     while pos &lt; len(list):         if list[pos] == key:             return pos         else:             pos += 1     return -1</pre>	<p>通过回忆循环结构嵌套分支结构的算法模型,完成程序填空。</p> <p>在实现的过程中能够结合所学知识,借助学习单等数字化学习资源,选择合适的编程平台,完成任务</p>
情境导入二	<p>小红的书架(分类、有序;按时间排序)</p> <p>问题4:小红还需要从头开始一本一本找吗?如果不需要,小红可以用什么方法加快查找速度?(可参考学习单回答)</p> <p>问题5:对分查找有适用条件吗?(可参考学习单回答)</p>	<p><b>【讨论,分析】</b></p> <p>(1) 帮助学生回顾之前学习过的对分查找思想,整理对分查找实现的具体步骤。  (2) 可类比顺序查找的分析结果。  参考答案:</p> <p>(1) 查找对象是静态数据集,数据结构为线性表;  (2) 数据必须按关键字排序</p>	<p>通过对实际问题的观察与分析,整理对分查找的思想;类比顺序查找的限制条件,明确对分查找的前提与顺序查找相比还要求查找数据有序,实现对分查找问题的抽象与建模,培养学生的计算思维能力</p>
对分查找实现	<p>对分查找的程序实现</p> <p>播放对分查找的动画,观察mid变量的变化。</p> <p>问题6:请同学们结合对分查找的基本思路,尝试绘制对分算法的流程图。在学习单上完成。</p> <p>请同学们根据上述对分查找的流程图,尝试完成程序填空。(程序见学习单)</p> <p>展示学生代码,评价完成情况。</p> <p>问题7:每个区间都重复以上操作,可否使用递归实现对分查找算法?(可参考学习单上有关递归的说明)</p> <p>问题8:可以的话请同学们尝试完成程序填空。(程序见学习单)</p> <p>展示学生代码,评价完成情况</p>	<p><b>【观察、绘制流程图、交流】</b></p> <p>(1) 注意mid变量在程序中运用了迭代的思想,进行值的替换。  流程图参考答案如下:</p> <p><b>【编程、交流】</b></p> <p>(2) 程序代码参考答案:</p>	<p>通过回忆循环结构嵌套多分支结构的算法模型,完成程序填空。</p> <p>分析体会算法思想,在教师指导下,尝试使用递归解决问题;通过编程实践体验问题解决的全部过程,理解算法和数据结构的关系</p>

教学环节	教师活动	学生活动	设计意图												
		<pre>def binarySearch(list, key)     low= 0     high= len(list)- 1     while (low&lt;= high):         mid= (high+ low)//2         if key== list[mid]:             return mid         elif key&gt; list[mid]:             low= mid+ 1         else:             high= mid- 1     return -1</pre> <p>(3) 通过学习单,自主复习递归的思想,完成程序填空</p>													
对比分析	<p>算法效率对比及总结 问题 9: 对比顺序查找和对分查找的查找效率及优缺点。</p> <table border="1"> <thead> <tr> <th>查找算法</th> <th>时间复杂度</th> <th>优点</th> <th>缺点</th> </tr> </thead> <tbody> <tr> <td>顺序查找</td> <td>O(n)</td> <td>易于理解且不会错判漏判</td> <td>时间复杂度高</td> </tr> <tr> <td>对分查找</td> <td>O(log<sub>2</sub>n)</td> <td>速度快</td> <td>对数据的排列有严格要求</td> </tr> </tbody> </table>	查找算法	时间复杂度	优点	缺点	顺序查找	O(n)	易于理解且不会错判漏判	时间复杂度高	对分查找	O(log <sub>2</sub> n)	速度快	对数据的排列有严格要求	<p>【讨论、回答】 (1) 顺序查找每一项的匹配次数求和除以查找对象的个数 n,就是查找平均关键字的匹配次数 <math>\frac{n}{2}</math>;对分查找的单个数据查找的最大次数为 <math>\log_2 n</math>,n 为查找对象个数。 (2) 顺序查找的优点是易于理解而且不会错判漏判;对分查找的优点是速度快,在数据较多时尤为明显。 (3) 顺序查找的缺点是时间复杂度高,在数据较多时尤为明显;对分查找的缺点是对数据的排列有严格的要求</p>	通过分析、对比、讨论,总结顺序查找和对分查找的效率和优缺点,能够根据实际情况,选择合适的算法
查找算法	时间复杂度	优点	缺点												
顺序查找	O(n)	易于理解且不会错判漏判	时间复杂度高												
对分查找	O(log <sub>2</sub> n)	速度快	对数据的排列有严格要求												

## 附：“查找算法初步研究”学习单

### 一、顺序查找

#### 【基本思路】

- 从表中的一端第一个数据开始,用给定的值与表里的各个数据元素的关键字\_\_\_\_\_。
- 如果找到所需的数据元素,则查找\_\_\_\_\_。
- 如果整个表都查遍了,仍未找到所需的数据元素,则查找\_\_\_\_\_。

#### 【适用条件】

- \_\_\_\_\_
- \_\_\_\_\_

### 二、顺序查找程序实现

```
def sequentialSearch(list, key):
    pos= 0
```

```

while pos< _____:
    if _____:
        return pos
    else:
        _____
return -1

List=[31,3,58,12,81,18,25,69,25,73,94,87]
k= int(input('输入需要查找的数字:'))
x= sequentialSearch(List,k)
print('数字 %d 在数列中排在第 %d 位' % (k,x))

```

### 三、对分查找

#### 【基本思路】

- 先取表\_\_\_\_\_位置的数据元素的值和给定值(key)比较,如果\_\_\_\_\_则查找成功。
- 如果给定值比该数据元素的值大,则所查找的数据元素在表的\_\_\_\_\_。
- 如果给定值比该数据元素的值小,则所查找的数据元素在表的\_\_\_\_\_。
- 如此反复进行,直到查找成功或确定查找不到为止。

#### 【适用条件】

- 查找表必须是用顺序存储结构的\_\_\_\_\_表。
- 表中的数据元素按关键字\_\_\_\_\_排列。

### 四、对分查找程序实现

- 完成流程图的绘制(见图 5-6)

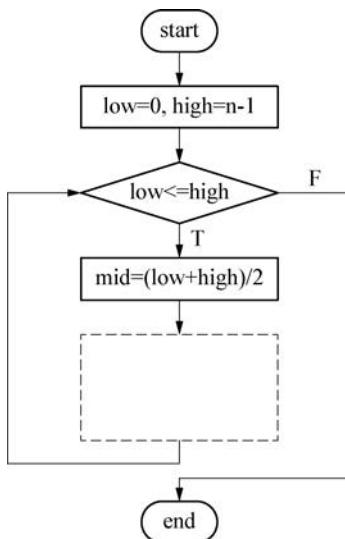


图 5-6 对分查找程序流程图

## 2. 程序填空

```
def binarySearch(list, key)
    low= 0
    high= _____
    while (low< = high):
        mid= _____
        if key== list[mid]:
            return mid
        elif key> list[mid]:
            low= _____
        else:
            high= _____
    return - 1
```

**知识回顾——递归:**程序调用自身的编程技巧称为递归(recursion)。一个过程或函数在其定义或说明中有直接或间接调用自身的一种方法。一般来说,递归需要有边界条件、递归前进段和递归返回段。当边界条件不满足时,递归前进;当边界条件满足时,递归返回。

举例:计算阶乘运算  $n! = 1 * 2 * 3 * \dots * n$ 。

递推法:

```
n= int(input('输入项数 n:'))
s= 1
i= 1
for i in range(1,n+ 1):
    s= s* i
print(s)
```

递归法:

```
def jc(n):
    if n== 1:
        return 1
    elif n> 1:
        return n* jc(n- 1)
print(jc(3))
```

## 3. 程序填空(对分查找的递归实现)

```
def binarySearch2(list, low, high, key):
    if low< = high:
        mid= (low+ high)//2
        if key== list[mid]:
            _____
        elif key> list[mid]:
            _____
        else:
            _____
```

```
else:  
    return -1
```

## 五、算法效率对比及总结(见表 5-12)

表 5-12 算法效率对比及总结

查找算法	时间复杂度	优点	缺点
顺序查找			
对分查找			

经上海市中小学教材审查委员会审查  
准予使用 淮用号 II-GJ-2022020

责任编辑：丁倩



绿色印刷产品

ISBN 978-7-5760-2951-2

A standard EAN-13 barcode representing the ISBN number.

9 787576 029512 >

定价：30.00 元