### 3.2 数据与结构

本节内容按2课时设计。

**第一课时**

**【**教学重点**】**简单数据类型与线性数据结构。

**【**教学难点**】**线性数据结构。

**【教学过程】**

**一、引入**

学生预习，阅读第56、57页“任务一 探究网购订单处理”之“活动1 了解订单数据”，填写第57页的表3.2.1。教师检查，并评讲填写情况，引出数据类型。

表3.2.1 网购中的订单数据

|  |  |  |
| --- | --- | --- |
| 网站名称 | 订单中的数据 | Python中对应的数据类型 |
| 某电商网站 | 商品名称 | 字符串 |
| 单价 | 浮点型 |
| 数量 | 整型 |
| 高中生学籍网 | 姓名 | 字符串 |
| 出生日期 | 日期型 |
| 是否团员 | 布尔型（逻辑型） |

**二、数据类型**

**1.认识Python简单数据类型**

在Python语言中，简单数据类型有整数（int）、浮点数（float）、字符串（str）、布尔（bool）等数据类型。

【教师示范操作】

>>> type(8) #type()函数返回数据的类型

<class 'int'> #返回'int'类型

>>> type(3.14)

<class 'float'> #返回'float'类型

>>> type('Thank you!')

<class 'str'> #返回'str'类型

>>> type(True)

<class 'bool'> #返回'bool'类型

**2.了解Python复合数据类型**

在Python语言中，复合数据类型有元祖（tuple）、集合（set）、列表(list)、字典(dict)等。

**①元祖** 例如，某用户预订了商品编号为“ID0010230”、单价为15.68元、数量为36，可将这3个不同类型的简单数据组织成一个复合数据类型——元祖。记作：

BookInfo0=("ID0010230",15.68,36)

另一用户预订了商品编号为“ID2315937”、单价为20元、数量为2，可记作：

BookInfo1=("ID2315937",20,2)

>>> BookInfo0=("ID0010230",15.68,36)

>>> type(BookInfo0)

<class 'tuple'> #返回元祖类型

>>> BookInfo1=("ID2315937",20,2)

>>> BookInfo1[1]

20 #返回元祖BookInfo1中索引为1的项的值

计算订单额：

>>> BookInfo0[1]\*BookInfo0[2]+BookInfo1[1]\*BookInfo1[2]

604.48 #返回计算结果

**②集合** 今天的订单汇总，可以定义为一个集合(集合里的项称为元素，彼此之间没有顺序)：

BookSet={BookInfo0,BookInfo1}

>>> BookSet={BookInfo0,BookInfo1}

>>> type(BookSet)

<class 'set'> #返回集合类型

>>> BookSet

{('ID2315937', 20, 2), ('ID0010230', 15.68, 36)} #返回集合的值

计算订单额：

>>> t=0

>>> for i in BookSet:

t+=i[1]\*i[2]

(将本行光标后退至行首，再按回车)

>>> t

604.48

**③列表** 今天的订单汇总，也可以按订单产生的新后顺序组成一个列表(列表里的项是有顺序编号的)：

BookList=[BookInfo0, BookInfo1]

>>> BookList=[BookInfo0, BookInfo1]

>>> type(BookList)

<class 'list'> #返回列表类型

>>> BookList[0]

('ID0010230', 15.68, 36)

>>> BookList[1]

('ID0010359', 0.68, 5)

>>> BookList[0][1]\*BookList[0][2]+BookList[1][1]\*BookList[1][2]

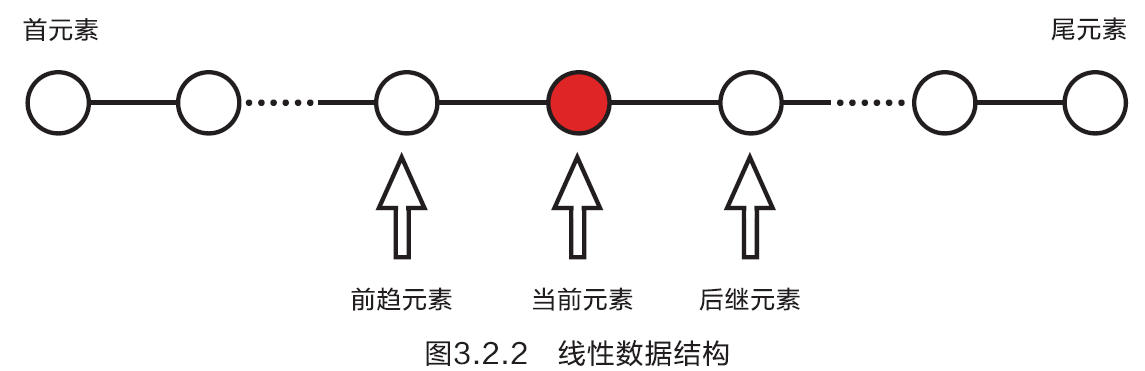
604.48 #返回计算结果

**3.编制订单数据处理程序**

（1）给出流程图，完成填空，运行程序，体会“队列”操作。

（2）程序代码（**详见配套资源“订单.py”**）。

**4.线性数据结构**



以上面的处理订单程序为例，在列表listque中：首元素索引号为0，向后走，索引号递增，这是正向编号。还有一种索引编号方式，尾元素编号为-1，向首元素方向依次递减，这是反向编号。

**（1）队列** 队列是先进先出（FIFO）表，它的数据元素只能在一端依次添加（进队），在另一端依次删除（出队）。典型的例子，如超市里排队付款的队伍。

比如在上述代码中，listque是列表类型的数据，存放了一组字符串类型的数据，表示订单编号。我们可以通过对应的方法对列表进行操作：

pop(0)方法可以删除列表的首元素(相当于队列的“出队”操作)，append方法可以在列表尾部添加一个数据元素（相当于队列的“入队”操作）。

**（2）栈** 栈是先进后出（FILO）表。它的数据元素只能在同一端（称为栈顶）进行操作，添加（进栈），删除（出栈）。典型的例子，如单车道死胡同停车，后进去的车先倒出来，先进去的车最后才能倒出来。

pop()方法可以删除列表的尾元素(相当于栈的“出栈”操作)，append方法可以在列表尾部添加一个数据元素（相当于栈的“入栈”操作）。

**三、小结**

Python数据类型包括：

1.简单数据类型

①整数（int）

②浮点数（float）

③字符串（str）

④布尔（bool）

2.复合数据类型

①元祖（tuple）

②集合（set）

③列表(list)

④字典(dict)

3.数据结构

①集合结构

②队列、栈（线性结构）

③树结构

④图结构

**四、练习**

试着模仿订单数据处理程序，编写单车道死胡同停车管理程序（栈的应用）。

参考程序：（**详见配套资源“单车道死胡同停车管理程序.py”**）。

**第二课时**

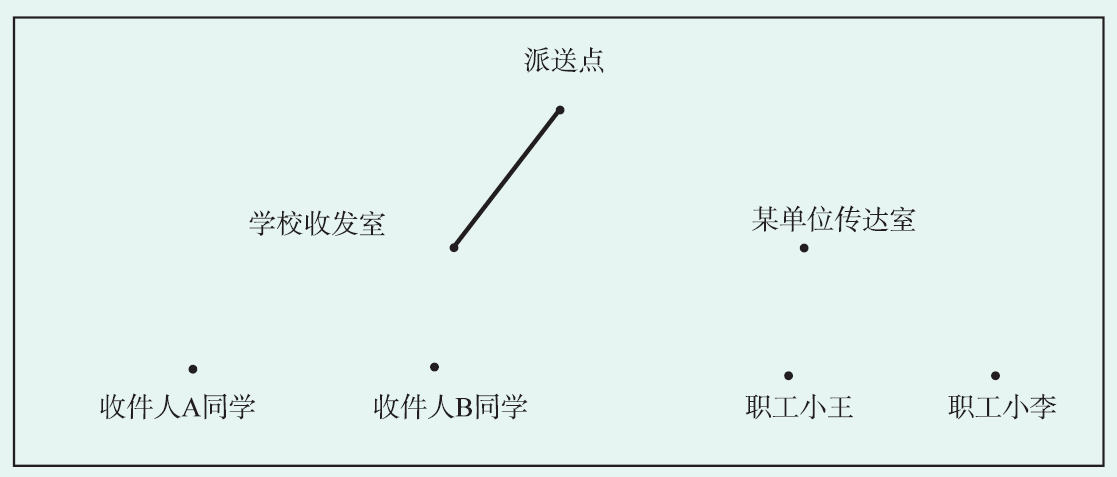
**【**教学重点**】**数据结构中的树结构和图结构。

**【**教学难点**】**数据结构中的树结构和图结构。

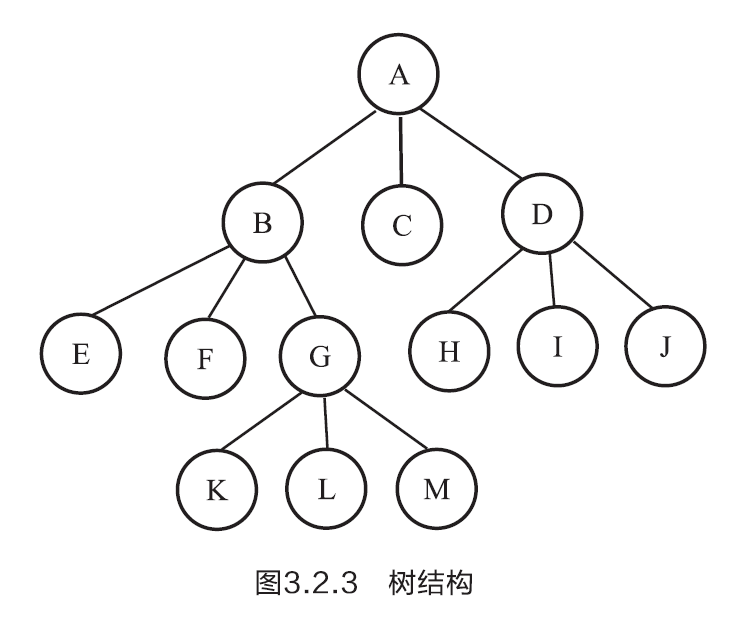
**【教学过程】**

**一、引入**

学生预习，阅读第59、60页“任务二 探究快递配送过程”之“活动1 了解快递派送线路”，完成第60页的连点成树（见下图）。教师检查，并评讲填写情况，引出树结构。



**二、树结构**



树的递归定义：

树是由n（n≥0）个节点组成的有限集合。若n = 0，则称为空树。任何一个非空树均满足以下两个条件：

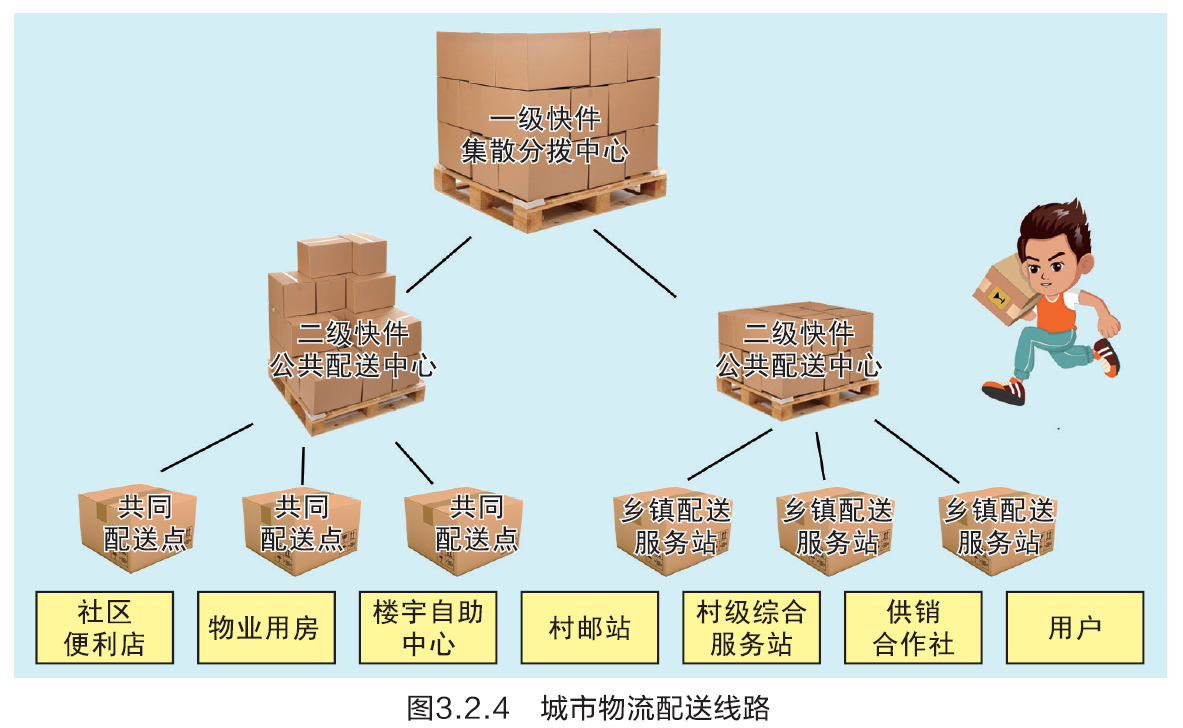
（1）仅有一个称为根的节点。

（2）当n>0时，其余节点可分为m（m≥0）个互不相交的有限集合，其中每个集合又是一棵树，并称为根的子树。

子概念：节点，根（节点），叶子节点，父节点，子节点；空树，子树。

【说一说】社会、工作、生活中的树形结构实例：

**①快递公司物流配送体系**



**②磁盘文件存储结构**

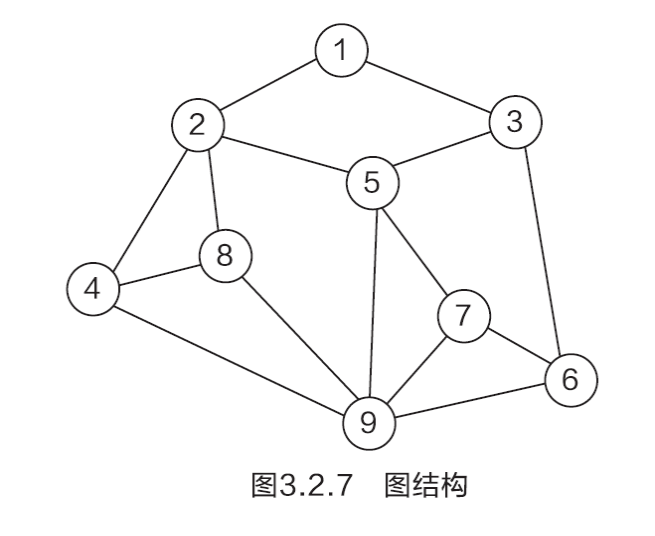
**③注册表**

**④树结构之行政区划**（**详见配套资源“行政区划树.py”和“行政区划.txt”**）

**三、图结构**

图结构是由一组节点（称为顶点）和一组节点间的连线（称为边或弧）构成的一种数据结构。图结构中的每个顶点都可以与其他顶点有边相连，图结构中数据元素之间是多对多的关系。

子概念：顶点，边（弧）。

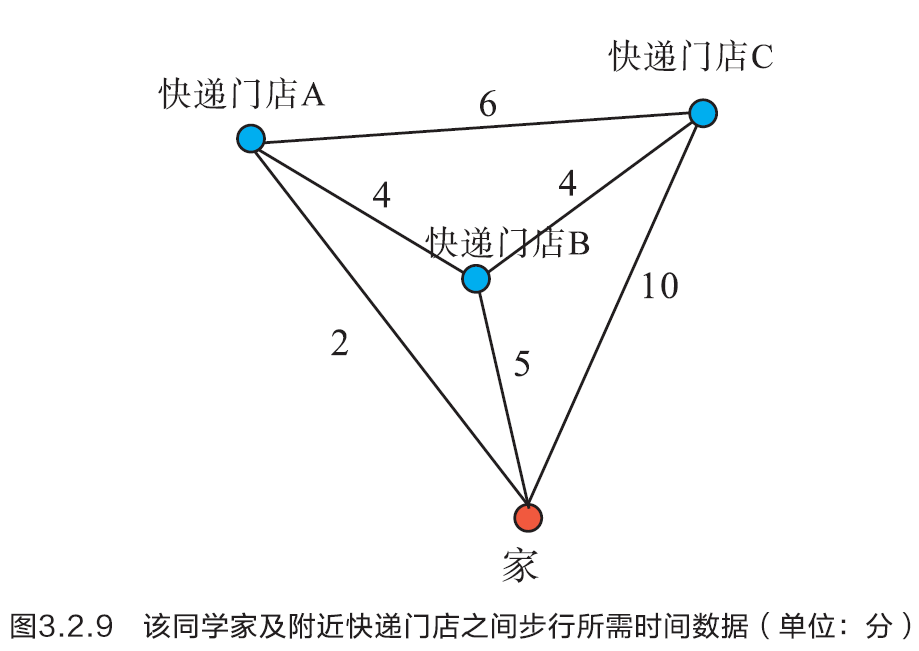


【说一说】社会、工作、生活中的图结构实例：

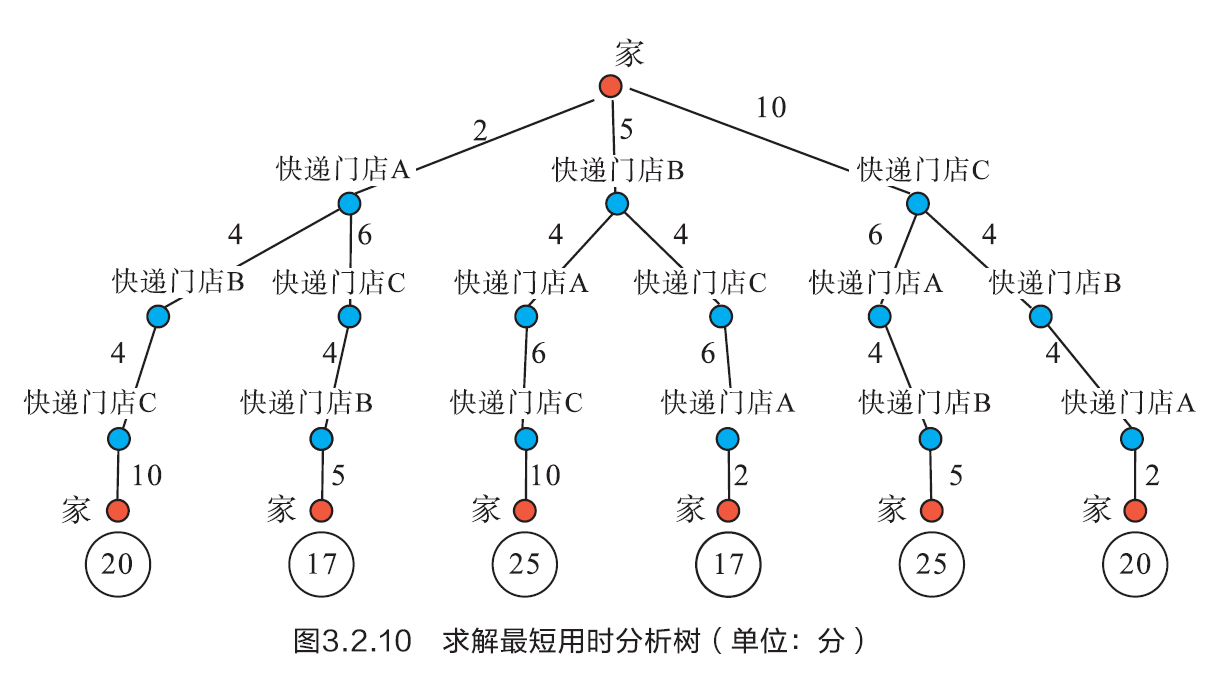
城市交通图（铁路网，公路网，航空网）

电话网、互联网

【活动3】规划取快递最快路线



【朴素算法】穷举遍历，依次列出所有可能走法如图3.2.10。将图中每个节点进行编号，编号互不相同：如作为根节点的“家”编号为“X”,其3个子节点（快递门店A，快递门店B，快递门店C）分别编号为“A” “B” “C”，…，详见下图。



【算法演示1】求解最短时间（基于图3.2.10的分析树）

（**详见配套资源“最短时间分析树.py”和“最短时间分析树.txt”**）

【算法演示2】求解最短时间（直接对图3.2.9进行深度优先遍历）

（**详见配套资源“取快递最短用时线路.py”**）

**四、小结**

树结构和图结构，是两种比较难的数据结构。

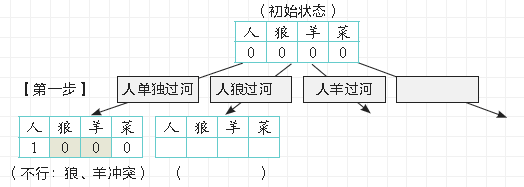
领会树结构和图结构的本质特征，会用树结构、图结构对工作学习生活中的具体问题进行抽象和分析，解决一些简单问题。

本课中的Python程序，难度较大，可作为较高要求下的研习之用。

**五、练习**

人、狼、羊、菜过河问题：有一个人带着一只狼、一只羊和一捆白菜，来到一条河边，河边只有一条小船，人每次过河最多只能带一样，如果人不在现场，狼就要吃羊，羊就要吃菜。他应该怎样安排过河呢？请完成下面的“树”结构分析图，帮他找到可行的过河方案。

提示：可约定对象在左岸用0表示，在右岸用1表示。



要求：能以纸笔方式画出分析树得出结论即可。（解答略）

（**详见配套资源“人狼羊菜过河游戏(动画版).exe”和“人狼羊菜过河游戏.py”**）