第一篇

引　入　篇

第1章

算法概述

1.1 用计算机求解问题与算法

与“计算机”和“解决问题”有关的一个概念是问题求解(Problem Solving),问题求解是个大课题,它涉及归约、推断、决策、规划、常识推理、定理证明和相关过程等核心概念。人工智能是这个课题下的一个分支,人工智能的第一个大成就是发展了能够求解难题的下棋(如国际象棋)程序。在下棋程序中应用的某些技术,发展成为搜索和问题归约这样的人工智能基本算法。今天的计算机程序能够下锦标赛水平的各种方盘棋,如五子棋和国际象棋等。有些软件甚至还能够用自动总结的经验来改善软件自身性能。由此可以理解“问题求解”的重点是要制造智能计算机模拟人的智能去进行问题求解,属于尖端科技。而一般计算机面对现实问题是无能为力的,需要人类对问题抽象化、形式化后才能去机械地执行,读者学习算法设计的重点就是把人类找到的求解问题的方法、步骤,以过程化、形式化、机械化的形式表示出来,以便让计算机执行(当然人工智能软件系统也离不开“算法设计”这个最基本的软件设计环节)。因此,本书就把学习的目标定为“用计算机求解问题”。

1.1.1 用计算机求解问题的步骤

人类在解决一个问题时,根据不同的经验、不同的环境会采用不同的方法,用计算机解决现实中的问题,同样也有很多不同的方法,但解决问题的基本步骤是相同的。下面给出用计算机求解问题的一般步骤。

1. 问题分析

准确、完整地理解和描述问题是解决问题的第一步。要做到这一点,必须注意以下一些问题:在未经加工的原始表达中,所用的术语是否都有准确的定义?题目提供了哪些信息?这些信息有什么用?题目要求得到什么结果?题目中做了哪些假定?是否有潜在的信息?判定求解结果所需要的中间结果有哪些?等等。针对每个具体的问题,必须认真审查问题描述,理解问题的真实要求。

2. 数学模型建立

用计算机解决实际问题必须有合适的数学模型。对一个实际问题建立数学模型，可以考虑这样两个基本问题：最适合于此问题的数学模型是什么？是否有已经解决了的类似问题可以借鉴？

如果上述第二个问题的答复是肯定的，那么通过类似的问题的分析、比较和联想，可加速问题的解决。但上述第一个问题是更重要的。如何选择恰当的数学工具来表达已知的和要求的量，受多种因素影响：设计人员是否有足够的数学知识水平，已知的数学模型是否表达方便，计算是否简单，所要进行的操作种类的多少与功能的强弱等。同一问题可以用不同的数学工具建立不同的模型，因此要对不同的模型进行分析、比较，从中选出最有效的模型。然后根据选定的数学模型，对问题进行重新描述。

此时，应考虑下列一些问题：模型能否清楚地表示与问题有关的所有重要信息？模型中是否存在与所期望的结果相关的数学量？能否正确反映输入、输出的关系？用计算机实现该模型是否有困难？如能取得满意的回答，那么该数学模型可作为候选模型。

3. 算法设计与选择

算法设计是指设计求解某一特定类型问题的一系列步骤，并且这些步骤是可以通过计算机的基本操作来实现的。算法设计要同时结合数据结构的设计，简单地说，数据结构的设计就是选取存储方式，如确定问题中的信息是用数组存储还是用普通变量存储等（或数据结构课程中介绍的更多存储方式）。因为不同的数据结构的设计将导致算法的差异很大。算法的设计与模型的选择更是密切相关的，但同一模型仍然可以有不同的算法，而且它们的有效性可能有相当大的差距。选择方法和模型建立大致相同，首先考虑学过的方法是否可以借鉴，最适合于此问题的算法是什么。

4. 算法表示

对于复杂的问题，确定算法后可以通过图形准确表示算法。算法的表示方式很多，如算法流程图、盒图、PAD 图和伪码（类似于程序设计语言）等。本书对简单的算法不进行图形表示。

5. 算法分析

算法分析有两个目的：首先是为了对算法的某些特定输入，估算该算法所需的内存空间和运行时间；其次是为了建立衡量算法优劣的标准，用以比较同一类问题的不同算法。通常将时间和空间的增长率作为衡量的标准。

6. 算法实现

求解某一特定类型问题的算法设计完成，并证明其正确性之后，就要根据算法编制计算机算法来实现它。在编制算法之前，还要选取存储类型，用来表达所用模型的各个方面。因此，根据选用的程序设计语言，要解决下列一些问题：有哪些变量，它们是什么类型？需要多少数组，规模有多大？用什么结构来组织数据？需要哪些子算法？等等。

算法的实现方式，对运算速度和所需内存容量都有很大影响。

7. 程序调试

算法测试的实质是对算法应完成任务的实验证实，同时确定算法的使用范围。测试方法一般有两种：一种是对算法的各个分支（每条路径）进行测试（即白盒测试）；另一种是检验对给定的输入是否有指定输出（即黑盒测试）。

如何选择算法测试中的输入（即选择测试用例），还没有一般的标准。通常采用的方法是，对输入数据做有代表性的采样，使之对被测试算法的各个语句、分支和路径尽可能都检查到。对输入集中的边界点也要进行测试。经测试验证是正确的算法，在较大程度上可以相信它的正确性。

8. 结果整理文档编制

编制文档的目的是让人理解所编写的算法。首先要把代码编写清楚。代码本身就是文档。同时还要采用注释的方式，另外还包括算法的流程图，自顶向下各研制阶段的有关记录，算法的正确性证明（或论述），算法测试过程、结果，对输入输出的要求及格式的详细描述等。

在这些步骤中，算法设计是解决问题的核心，本书在例题讲解时侧重的是第 1～5 步。

1.1.2　算法及其要素和特性

1. 算法的定义

算法是指在解决问题时，按照某种机械步骤一定可以得到问题结果（有解时给出问题的解，无解时给出无解的结论）的处理过程。当面临某个问题时，需要找到用计算机解决这个问题的方法和步骤，算法就是解决这个问题的方法和步骤的描述。

所谓机械步骤，是指算法中有待执行的运算和操作必须是相当基本的。换言之，它们都是能够精确地被计算机运行的算法，执行者（计算机）甚至不需要掌握算法的含义，即可根据该算法的每一步骤要求，进行操作并最终得出正确的结果。

“算法”这个词其实并不是一个陌生的词，因为从小学大家就开始接触算法了。例如，做四则运算，必须按照一定的算法步骤一步一步地做。“先运算括号内再运算括号外，先乘除后加减”可以说是四则运算的算法。以后学习的指数运算、矩阵运算和其他代数运算的运算规则都是一种算法。

就本课程而言，算法就是计算机解决问题的过程。在这个过程中，无论是形成解题思路还是编写算法，都是在实施某种算法。前者是推理实现的算法，后者是操作实现的算法。

2. 算法的三要素

算法由操作、控制结构和数据结构三要素组成。

（1）操作

算法实现平台尽管有许多种类，它们的函数库、类库也有较大差异，但必须具备的最基

本的操作功能是相同的。这些操作包括以下几个方面。

- 算术运算：加、减、乘、除。
- 关系比较：大于、小于、等于、不等于。
- 逻辑运算：与、或、非。
- 数据传送：输入与输出(计算)、赋值(计算)。

(2) 控制结构

一个算法功能的实现不仅取决于所选用的操作，还取决于各操作之间的执行顺序，即控制结构。算法的控制结构给出了算法的框架，决定了各操作的执行次序。算法的控制结构包括以下几种。

- 顺序结构：各操作是依次执行的。
- 选择结构：由条件是否成立来决定选择执行。
- 循环结构：有些操作要重复执行，直到满足某个条件时才结束，这种控制结构也称为重复或迭代结构。

模块间的调用也是一种控制结构，特别地，模块自身的直接或间接调用是递归结构，是一种功能很强的控制重复的结构。在 3.1 节将重点介绍利用循环、递归机制设计算法中重复操作的要点。

(3) 数据结构

算法操作的对象是数据，数据间的逻辑关系、数据的存储方式及处理方式就是数据的数据结构。它与算法设计是紧密相关的，在 3.2 节和第 6 章中将通过例题进行介绍。

3. 算法的基本性质

算法就是把人类找到的求解问题的方法，经过过程化、形式化后，用上述三要素表示出来。在算法的表示中要满足以下性质：

- 目的性。算法有明确的目的，算法能完成赋予它的功能。
- 分步性。算法为完成其复杂的功能，由一系列计算机可执行的步骤组成。
- 有序性。算法的步骤是有序的，不可随意改变算法步骤的执行顺序。
- 有限性。算法是有限的指令序列，算法所包含的步骤是有限的。
- 操作性。有意义的算法总是对某些对象进行操作，使其改变状态，完成其功能。

4. 算法的地位

算法是计算机学科中最具有方法论性质的核心概念，也被誉为计算机学科的灵魂。

数学大师吴文俊指出："我国传统数学在从问题出发以解决问题为主旨的发展过程中，建立了以构造性与机械化为其特色的算法体系，这与西方数学以欧几里得《几何原本》为代表的所谓公理化演绎体系正好遥遥相对。……肇始于我国的这种机械化体系，在经过明代以来几百年的相对消沉后，由于计算机的出现，已越来越为数学家所认识与重视，势将重新登上历史舞台。"吴文俊创立的几何定理的机器证明方法(世称吴方法)，用现代的算法理论，焕发了中国古代数学的算法传统，享有很高的国际声誉，也受到国家的高度关注。他因此于 2001 年获得了第一届国家最高科学技术奖。

5. 算法的基本特征

并不是所有问题都有解决的方法，也不是所有人类解决问题的方法都能设计出相应的算法。算法必须满足以下 5 个重要特性：

(1) 有穷性

一个算法在执行有穷步之后必须结束，也就是说，一个算法所包含的计算步骤是有限的。即算法中的每个步骤都能在有限时间内完成。

(2) 确定性

对于每种情况下所应执行的操作，在算法中都有确切的规定，使算法的执行者或阅读者都能明确其含义及如何执行。并且在任何条件下，算法都只有一条执行路径。

(3) 可行性

算法中描述的操作都可以通过已经实现的基本操作运算有限次实现。

(4) 算法有零个或多个输入

输入作为算法加工对象的数据，通常体现为算法中的一组变量。有些输入量需要在算法执行过程中输入，而有的算法表面上可以没有输入，实际上输入已被嵌入算法之中。

(5) 算法有一个或多个输出

它是一组与输入有确定关系的量值，是算法进行信息加工后得到的结果。

1.1.3　算法设计及基本方法

算法设计作为用计算机解决问题的一个步骤，其任务是对各类具体问题设计出良好的算法。算法设计作为一门课程，是研究设计算法的规律和方法。

在设计算法时，应当严格考虑算法的以下质量指标：

(1) 正确性(Correctness)

首先，算法对于一切合法的输入数据都能得出满足要求的结果；其次，对于精心选择的、典型的、苛刻的几组输入数据，算法也能够得出满足要求的结果。

(2) 可读性(Readability)

算法主要是为了方便人的阅读与交流，其次才是让计算机执行。因此算法应该易于人的理解。另一方面，晦涩难读的算法易于隐藏较多错误而难以调试。有些算法设计者总是把自己设计的算法写的只有自己才能看懂，这样的算法反而没有太大的实用价值。

(3) 健壮性(Rubustness)

当输入的数据非法时，算法应当恰当地做出反映或进行相应处理，而不是产生莫名其妙的输出结果。这就需要我们充分地考虑到可能的异常情况(Unexpected Exceptions)，并且处理出错的方法不应该是简单地中断算法的执行，而应是返回一个表示错误或错误性质的值，以便在更高的抽象层次上进行处理。

(4) 高效率与低存储量需求

通常，效率指的是算法执行时间；存储量指的是算法执行过程中所需的最大存储空间。两者都与问题的规模有关。这一点在第 2 章中详细介绍。

针对不同的问题，算法设计的方法、策略很多，学习和掌握它们是本书的主要任务，这里

先介绍几个算法设计的基本模型。

1. 结构化方法

算法的质量首先取决于它的结构。算法设计和建筑设计极为相似，一座建筑物的整体质量首先取决于它的钢筋混凝土结构是否牢固，然后才是它的外装修质量。同样，一个算法的质量优劣，首先取决于它的结构，其次才是它的速度、界面等其他特性；如果一个程序中的所有模块都只使用顺序、选择和循环 3 种基本结构，那么不管这个程序中包含多少个模块，它仍然具有清晰的结构。

结构化方法总的指导思想是自顶向下、逐步求精。它的基本原则是功能的分解与模块化。

所谓“自顶向下”，是将现实世界的问题经抽象转化为逻辑空间或求解空间的问题；是将复杂且规模较大的问题划分为较小问题，找出问题的关键和重点，然后抽象、概括地描述问题。

所谓“逐步求精”，是将复杂问题经抽象化处理变为相对比较简单的问题。经若干步精化处理，最后细化到用“3 种基本结构”及基本操作去描述算法。

所谓“模块化”，是指把一个大的程序按照一定的原则划分为若干个相对独立但又相关的小模块（函数）的方法。

结构算法设计技术具有以下优越性：

① 自顶向下逐步求精的方法符合人类解决复杂问题的普遍规律，因此可以显著提高软件开发工程的成功率和生产率。

② 用先全局后局部、先整体后细节、先抽象后具体的逐步求精过程开发出的算法有清晰的层次结构，因此容易阅读和理解。

2. 面向对象方法

所谓对象是包含数据和对数据操作代码的实体，或者说是在传统的数据结构中加入一些被称为成员函数的过程，因而赋予对象以动作。在面向对象算法设计中，对象具有与现实世界的某种对应关系，人们正是利用这种关系对问题进行分解。如图 1-1 所示。

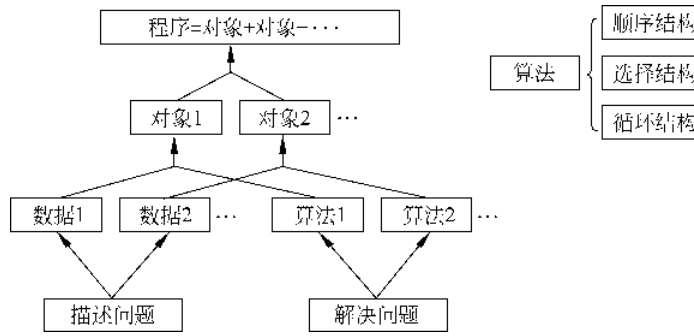

图 1-1　面向对象程序设计方法

面向对象算法设计方法的过程包括以下步骤：

① 在给定的问题域中抽象地识别出类和对象。

② 识别这些类和对象的语义。

③ 识别这些类和对象之间的关系。

④ 实现类和对象。

面向对象方法引入了对象、消息、类、继承、封装、抽象、多态性等机制和概念。用面向对象方法进行算法设计时，以问题域中的对象为基础，将具有类似性质的对象抽象成类，并利用继承机制，仅对差异进行算法设计。它的特征主要包括以下几个方面：

- 抽象化。将各种独立的操作分解成为可以用命名区分的单元。
- 封装性。不同的操作具有不同的作用范围。
- 多态性。对于不同数据类型的相似操作使用相同的命名。
- 继承性。类可以被继承，从而实现不同层次的对象。

抽象化是面向对象的一个重要特征，但是并不是它所独有的特征。重用是面向对象的一个重要优点。最大特点是能够大幅度地提高软件项目的成功率，减少日后的维护费用，提高软件的可移植性和可靠性。

3. 本书采用的设计方法

综合以上方法，设计都是将大而复杂的问题，分解为人脑可以思维的小问题。对象是一个封装体，是数据和操作的封装，面向对象的设计主要包括两个方面：一是确定对象及其中的数据，二是设计实现对象的行为。后者与结构化算法设计虽不完全相同，需要考虑封闭性、隐藏性、继承性等，但基本的算法设计方法和技巧是相同的。根据多年的教学工作经验，本书采用结构化设计方法，这样可以使读者专心学习设计解决问题的算法。

结构化算法设计的具体细节介绍如下。

(1) 自顶向下设计——从抽象到具体

抽象包括算法抽象和数据抽象。算法抽象是指算法的寻求(或开发)采用逐步求精、逐层分解的方法；数据抽象是指在算法抽象的过程中逐步完善数据结构和引入新的数据及确定关于数据的操作。

模块算法的设计采用逐步求精设计方法，即先设计出一个抽象算法，这是一个在抽象数据上实施一系列抽象操作的算法，由粗略的控制结构和抽象的计算步骤组成。抽象操作只指明“做什么”，对这些抽象操作的细化就是想方设法回答它“如何做”。

具体步骤如下：

① 把一个较大的算法概要地划分为若干个子任务(模块)，每一个子任务(模块)完成一个功能。

② 每一个模块再继续划分为更小的子模块。

③ 直到用 3 种控制结构和具体操作表示算法。

注：运用这种编程方法，考虑问题必须先进行整体分析，避免边写边想。

(2) 模块划分的基本要求——简单性、独立性和完整性

① 模块的功能在逻辑上尽可能地单一化、明确化。

② 模块之间的联系及相互影响尽可能地小；尽量避免传递控制信号，而仅限于传递处理对象。即应当尽量避免逻辑耦合，而仅限于数据耦合。

③ 模块的规模应当足够小，以便使模块本身的调试易于进行。

(3) 模块间的接口问题

分解好的模块不可能是完全独立的，它们之间一定有信息传递。一般有以下几种信息传递方式。

① 按名共享：全局变量。

② 子模块返回调用模块信息：子模块名。

③ 调用模块传递给子模块信息：值参数传递。

④ 调用模块与子模块互相传递信息：变量参数传递(C 语言没有此种传递方式，PASCAL、C++ 等程序设计语言提供此类参数)。

⑤ 按地址共享变量：地址参数传递(参数为指针变量名、数组名、变量地址)。

(4) 算法细节设计的基本方法——从具体到抽象

设计算法“如何做”的细节是比较容易出错的，比如循环变量的初始值或终值，数组的下标与循环变量间的关系等算法细节的确定。最基本的方法是通过“枚举”一些真实数据，从具体的实例中，抽象“归纳”出算法的这些细节。

(5) 算法的正确性证明

关于设计出的算法的正确性证明，本书不做深入讨论。因为有些算法的正确性是比较直观的，无需证明；还有一些算法的正确性比较难以证明，即使有方法可以证明，所涉及的数学理论和推导证明过程也很复杂。现实中常采用的办法是，设计算法时力争严谨，考虑周全，在可能的情况下，要分析论证算法设计的合理性，最后在算法实现后，靠大量的测试，以保证算法的质量。

1.1.4 从算法到实现

有些读者认为，程序设计的难点就是确定算法，有了正确的算法就很容易编写出正确的程序了。其实不然，在用程序设计语言实现算法的过程中还有许多要注意的问题，特别是由于不同程序设计语言的差异造成的问题。并没有也不可能给出一个具体的方法，保证由正确的算法一定得到正确的程序，这里只能给出一些注意事项。

这部分知识不属于本书的必要内容。在此进行简述是为了让读者在学习算法设计过程中上机实验时，不要走太多的弯路。

1. 数据类型的选择

算法设计中要先设计数据结构，即如何存储数据，如用数组还是用普通变量存储信息等，但并不深入地考虑数据可能的范围，也就不确定数据存储的类型。

要注意，若用 PASCAL 语言实现算法，数据溢出时，系统会给出提示信息(对应的输出为“*”)；而用 C 语言实现算法，算法会输出截取高位后的数据，让人难以理解错误所在。所以对于用后者实现算法，更要注意数据类型的选择。

数据类型的选择主要取决于要解决问题中数据可能的范围、参与的运算等实际情况。