

普通高等院校“十一五”规划教材

算法 与数据结构

SUANFA YU SHUJU JIEGOU

张永 李睿 年福忠 编著



教学课件索取: jfxing@ndip.cn



国防工业出版社
National Defense Industry Press

TP311.12
548
1:

普通高等院校“十一五”规划教材

算法与数据结构

张永 李睿 年福忠 编著

国防工业出版社

·北京·

内 容 简 介

本书分为基本概念、简单数据结构(线性表、栈、队列)、复杂数据结构(树、图)和算法与数据结构应用(排序、查找、算法设计基础)四部分,详细介绍了常用数据结构和算法的基本概念及其不同的实现方法,对各种数据结构,讨论了在不同存储结构上实现线性和非线性结构的不同运算,并对算法设计的方法和技巧进行了介绍。

本书概念清晰,逻辑严密,重点突出,将抽象的描述与具体的实现结合,便于教学,也使初学者容易掌握其重点内容,有利于自学。本书的算法描述和实现采用类 C 和 C 语言。

本书可以作为计算机科学与技术、信息与计算科学和相关专业的本科或大专教材。

图书在版编目(CIP)数据

算法与数据结构/张永,李睿,年福忠编著. —北京:
国防工业出版社,2008.8
普通高等院校“十一五”规划教材
ISBN 978-7-118-05852-9

I. 算... II. ①张... ②李... ③年... III. ①算法
分析—高等学校—教材②数据结构—高等学校—教材
IV. TP301.6 TP311.12

中国版本图书馆 CIP 数据核字(2008)第 101959 号

※

国防工业出版社 出版发行

(北京市海淀区紫竹院南路 23 号 邮政编码 100044)

北京诚信伟业印刷有限公司印刷

新华书店经售

*

开本 787×1092 1/16 印张 19 字数 438 千字

2008 年 8 月第 1 版第 1 次印刷 印数 1—4000 册 定价 30.00 元

(本书如有印装错误,我社负责调换)

国防书店: (010)68428422

发行邮购: (010)68414474

发行传真: (010)68411535

发行业务: (010)68472764

前 言

在计算机及相关专业众多的基础课程里面,算法与数据结构是基础而又十分重要的课程。当人们用计算机来解决实际问题时,就要涉及数据的表示及数据的处理,而数据表示及数据处理正是数据结构课程的主要研究对象,通过这两方面众多内容的学习,将为后续课程,特别是未来的软件开发和基础性研究工作打下坚实的基础,同时也提供数据结构和算法描述、实现等必要的技能训练。因此,算法与数据结构课程在计算机及其相关专业中具有举足轻重的地位。

本课程的教学任务围绕以下两方面进行:在基础方面,要求学生掌握常用数据结构的基本概念及其不同的实现方法;在技能方面,对各种数据结构,通过系统学习能够在不同存储结构上实现其不同的运算,并对算法设计的方式和技巧有所掌握。

由于算法与数据结构涉及的内容众多,在教学过程中如何精选授课内容,在有限的课堂教学中达到既定目标,长期以来,都是个难点。但如下内容应是本课程必须包括的:

1. 数据、数据元素和数据项的概念及其相互间的关系;数据结构的逻辑结构、存储结构的关系与区别以及作用在数据结构上的运算及其实现;算法的基本概念、算法的描述方法及算法评价。

2. 线性表的定义及其运算;顺序表和链表的定义、组织形式、结构特征和类型说明以及在这两种表上实现的插入、删除和查找的算法;循环链表、双链表的结构特点和在其上施加的插入、删除等操作。

3. 栈和队列的定义、特征及在其上所定义的基本运算,在两种存储结构上对栈和队列所施加的基本运算的实现。

4. 树的定义、性质及其存储方法;二叉树的二叉链表存储方式、结点结构和类型定义;二叉树的遍历算法;树与二叉树间的相互转换;哈夫曼树的构造方法及应用。

5. 图的基本概念及术语;图的两种存储结构,邻接矩阵和邻接表的表示方法;图的遍历,深度优先搜索遍历和广度优先搜索遍历算法;最小生成树的构造;拓扑排序、关键路径以及最短路径算法。

6. 排序的基本思想和基本概念;插入排序(直接插入排序、折半插入排序、希尔排序)、交换排序(如冒泡排序、快速排序)、选择排序(简单选择排序、堆排序)、归并排序(如2—路归并排序)、基数排序五种类型排序的基本思想、步骤及算法。

7. 查找的基本思想和基本概念;在顺序表、有序表、散列表等上的查找方法和算法;二叉排序树、平衡二叉树以及 B-树的概念和有关算法;散列表的构造。

本书由张永、李睿和年福忠编著,其中由张永统筹并编写了第 1 章、第 2 章、第 3 章、第 4 章、第 5 章、第 8 章和第 12 章,李睿编写了第 6 章、第 7 章和第 9 章,年福忠编写了第 10 章、第 11 章和第 13 章。

本书是编著者在长期从事计算机专业教学工作的基础上,结合已有的教学实践经验及参考了众多的相关教科书后编写而成的。由于考虑到学习本书的对象主要是大学二年级并大部分只学习过 C 语言,故本书的算法描述和实现采用类 C 和 C 语言。

本书可以作为计算机科学与技术、信息与计算科学和相关专业的本科或大专教材。

限于编者水平,书中不妥或错误之处敬请读者指正和赐教。

编 著 者

2008-6-20

目 录

第一部分 基本概念

第1章 数据结构基础	1
1.1 问题求解分析	1
1.2 数据结构	3
1.3 数据结构的分类	5
1.4 数据的四种基本存储方法	5
1.5 数据结构三方面的关系	6
习题	6
第2章 算法及算法分析基础	7
2.1 算法的基本概念	7
2.2 算法的描述	8
2.3 算法分析方法	9
2.4 程序语言的基本语句与基本结构	13
2.5 数组与结构	15
2.6 抽象数据类型的表示与定义	17
习题	18

第二部分 简单数据结构

第3章 线性表	20
3.1 线性表的定义	20
3.2 线性表的运算	21
3.3 线性表的顺序存储结构及实现	22
3.3.1 线性表的顺序存储结构	22
3.3.2 顺序表的实现	23
3.4 线性表的链式存储结构及实现	28
3.4.1 单链表	28
3.4.2 循环链表	37
3.4.3 双向链表	41
3.4.4 静态链表	44
3.4.5 顺序表和链表的比较	47
3.5 线性表的应用	48
习题	51

第4章 栈和队列	53
4.1 栈	53
4.1.1 问题的提出	53
4.1.2 定义及其操作	53
4.1.3 栈的存储结构及实现	54
4.1.4 栈的应用举例:表达式求值	59
4.2 队列	62
4.2.1 问题的提出	62
4.2.2 队列的定义及操作	62
4.2.3 队列的存储结构及实现	63
4.2.4 队列的应用举例	69
习题	78
第5章 矩阵和广义表	81
5.1 矩阵的存储	81
5.2 特殊矩阵	81
5.3 稀疏矩阵	84
5.4 广义表	87
习题	89

第三部分 复杂数据结构

第6章 二叉树和树	90
6.1 二叉树的定义和性质	91
6.1.1 二叉树的定义及相关术语	91
6.1.2 特殊二叉树	93
6.1.3 二叉树的性质	93
6.2 二叉树的存储结构	95
6.2.1 二叉树的顺序存储表示	95
6.2.2 二叉树的链式存储表示	97
6.3 二叉树的遍历	99
6.3.1 问题的提出	99
6.3.2 二叉树的遍历算法	100
6.3.3 二叉树遍历的非递归实现	102
6.3.4 遍历算法的应用	104
6.4 二叉树的线索化	109
6.4.1 线索二叉树的定义	109
6.4.2 线索二叉树的结构	109
6.4.3 二叉树的线索化算法	110
6.4.4 线索二叉树基本操作的实现	112
6.5 二叉树的应用——哈夫曼树	115

6.5.1	问题的引入	115
6.5.2	哈夫曼树的基本概念及其构造	116
6.5.3	哈夫曼树的应用——哈夫曼编码	117
6.5.4	哈夫曼树的构造及其编译码算法	120
6.6	树	124
6.6.1	树的定义及相关术语	124
6.6.2	树的存储结构	125
6.6.3	树、森林与二叉树的转换	129
6.6.4	树及森林的遍历	130
	习题	131
第7章	图	133
7.1	定义及相关术语	133
7.1.1	图的定义	134
7.1.2	相关术语	134
7.2	图的存储结构	136
7.2.1	图的顺序存储——邻接矩阵	136
7.2.2	图的链式存储——邻接表	138
7.3	图的遍历	141
7.3.1	深度优先遍历	142
7.3.2	广度优先遍历	143
7.3.3	遍历的应用	144
7.4	图的应用	146
7.4.1	最小生成树	146
7.4.2	最短路径	151
7.4.3	拓扑排序	158
7.4.4	关键路径	162
	习题	167
第8章	散列结构	169
8.1	散列表的概念	169
8.2	散列函数的构造方法	170
8.3	处理冲突的方法	171
8.3.1	开放定址法	171
8.3.2	拉链法	174
8.4	散列表的查找及其分析	175
	习题	178
第9章	集合结构	179
9.1	集合概述	180
9.2	集合的存储结构	181
9.2.1	集合的位向量实现	181

9.2.2	集合的表实现	183
9.2.3	集合的散列实现	188
9.3	并查集	191
9.3.1	等价类	191
9.3.2	并查集及运算	192

第四部分 算法与数据结构应用

第10章	排序	197
10.1	引言	197
10.2	插入排序	198
10.3	交换排序	202
10.4	归并排序	206
10.5	选择排序	210
10.6	基数排序	216
	习题	220
第11章	查找	222
11.1	基本概念	222
11.2	线性表查找	223
11.2.1	顺序查找	223
11.2.2	折半查找	224
11.2.3	分块查找	226
11.3	树表查找	228
11.3.1	二叉树查找	228
11.3.2	平衡二叉树(AVL树)	233
11.3.3	B-树	236
	习题	243
第12章	文件	245
12.1	文件的基本概念	245
12.2	顺序文件	248
12.3	索引文件	249
12.4	ISAM文件和VSAM文件	252
12.5	散列文件	257
12.6	多重表文件	259
	习题	261
第13章	算法设计技术与应用	263
13.1	背包问题	263
13.2	货郎担问题	268
13.3	学生成绩管理系统	272
参考文献		296

第一部分 基本概念

第1章 数据结构基础

算法与数据结构在计算机科学与技术中,尤其是在计算机软件设计中有着举足轻重的重要作用。在几乎所有的计算机软件系统中,如操作系统、数据库系统、编译系统、计算机网络技术、软件工程等都要用到算法与数据结构的知识。算法与数据结构已经成为计算机科学与技术专业和其他与软件有关专业的重要的专业基础课程。本章将介绍有关算法与数据结构的基础性知识。

1.1 问题求解分析

大家知道,要让计算机处理现实问题,首先要解决两个问题。

(1) 现实问题的计算机化表示问题。也就是将现实问题表示成计算机可以识别、存储的问题。这里涉及将现实问题逻辑表示和物理表示的问题。

(2) 现实问题的处理过程程序化问题。也就是将问题的处理过程按一定先后顺序进行有效编排以便让计算机有效处理的问题。

前者就是选用合适的数据结构将计算机要处理的对象有效表示、存储的问题;后者就是选择针对具体问题的合适算法以便使计算机能够进行程序化处理的问题。这里就涉及数据结构和算法知识。

下面先通过一个实例来说明数据结构与算法在计算机问题求解中的重要性。

例 1-1 魔方求解问题。

一个魔方 (magic square) 就是一个由 1 到 n^2 的整数构成的 $n \times n$ 的矩阵,其中每行、每列以及两个主对角线上的数字之和都相等。

当 $n=3$ 时,魔方如图 1-1 所示。

当 n 为奇数时, Coxeter 给出了产生魔方的具体过程。

(1) 把 1 放入第一行最中间的方格中。

(2) 向左上方移动,并按照数字的递增顺序,把数字填入空方格。

(3) 如果移出了魔方,即超越了魔方边界,则进入魔方对边的对应方格中。

(4) 如果一个方格已被填入数字,则向下继续填写方格。

6	1	8
7	5	3
2	9	4

图 1-1 $n=3$ 时的魔方

(5) 依据上述方法，直到全部填写完毕。

要让计算机求解上述问题，首先选择一个数据结构表示魔方，这里选择一个行、列各为 n 的数组来表示魔方。

数据结构形式化结果为（以类 C 语言描述）

```
int square[MAX_SIZE][MAX_SIZE]; /*定义一个二维数组来表示魔方*/
```

其次，将上述产生魔方的过程算法化，用简洁的描述方式描述产生魔方的过程，即算法描述（这里假定 size 为奇数，行、列为 size）：

(1) `square[0][(size-1)/2]=1;` /*把 1 放入第一行最中间的方格中 */

(2) `for (count=2; count<=size*size; count++){` /*依次放入其后的数字*/

`/*i=0;j=(size-1)/2; */`

`row=(i-1<0)? (size-1): (i-1); /*i=0;j=(size-1)/2; 上方*/`

`column=(j-1<0)? (size-1): (j-1); /*左方*/`

`if (square[row][column]) /*下方*/`

`i=(++i) % size;`

`else { /* 如果方格已被填入数字*/`

`i=row;`

`j=(j-1<0)?(size-1) : --j;`

`}`

`square[i][j]=count;`

`}`

用 C 语言实现如下：

```
#include <stdio.h>
```

```
#define MAX_SIZE 15 /*魔方行列最大为 15 */
```

```
void main(void)
```

```
{
```

```
    static int square[MAX_SIZE][MAX_SIZE];
```

```
    int i, j, row, column; /* 索引*/
```

```
    int count; /* 计数*/
```

```
    int size; /* 魔方行列数*/
```

```
    printf("Enter the size of the square: ");
```

```
    scanf("%d",&size);
```

```
    /* 输入错误检查*/
```

```
    if (size<1 || size>MAX_SIZE+1){
```

```
        printf("Error! Size is out of range\n");
```

```
        exit(1);
```

```
    }
```

```
    for (i=1; i<size; i++)
```

```
        for(j=0; j<size; j++)
```

```
            square[i][j]=0;
```

```

square[0][(size-1)/2]=1; /* 第一行中间 */
/*i 和 j 代表当前位置 */
i=0;
j=(size-1)/2;
for (count=2; count<=size*size; count++){
    row=(i-1<0)? (size-1): (i-1); /* 上 */
    column=(j-1<0)? (size-1): (j-1); /* 左 */
    if (square[row][column]) /* 下 */
        i=(++i) % size;
    else { /* 位置有数据 */
        i=row;
        j=(j-1<0)?(size-1) : --j;
    }
    square[i][j]=count;
}
/* 魔方输出 */
printf("Magic square of size %d:\n\n",size);
for ((i=1; i<size; i++) {
    for(j=0; j<size; j++)
        printf("%5d", square[i][j]);
    printf("\n");
}
printf("\n\n");
}

```

从上面这个实际问题的求解过程可以看出，计算机在解决问题的时候，首先是将现有的问题有效表示，即选择合适的数据结构来表示；而后将问题的求解过程形式化、程序化，即建立算法。这样，计算机才可以有效识别、存储并解决相关问题。

1.2 数据结构

数据结构与计算机环境下的问题求解有密切关系。那么，什么是数据结构呢？先介绍一些本书中要用到的概念，而后引入数据结构的定义。

1. 数据 (Data)

数据是信息的载体。它能够被计算机识别、存储和加工处理，是计算机程序加工的“原料”。

随着计算机应用领域的扩大，数据的范畴包括：整数、实数、字符串、图形、图像和声音等。

2. 数据元素 (Data Element)

数据元素是数据的基本单位。数据元素也称为元素、结点、记录。

一个数据元素可以由若干个数据项（也可称为字段、域、属性）组成。
数据项是具有独立含义的最小标识单位。

3. 数据对象 (data object)

数据对象是性质相同的数据元素的集合，是数据的一个子集。

4. 数据结构 (Data Structure)

数据结构是相互之间存在一种或多种特定关系的数据元素的集合。特别关注的是数据之间的相互关系，即数据的组织形式。

数据结构的形式定义为

$$\text{Data_Structure} = (D, S)$$

其中：D 是数据元素的集合；S 是 D 上关系的有限集合。

从上面定义可以知道，数据结构一般包括以下三方面内容。

(1) 数据元素之间的逻辑关系，也称数据的逻辑结构 (Logical Structure)。数据的逻辑结构是从逻辑关系上描述数据，与数据的存储无关，是独立于计算机的。数据的逻辑结构可以看作是从具体问题抽象出来的数学模型。

(2) 数据元素及其关系在计算机存储器内的表示，称为数据的存储结构 (Storage Structure)。数据的存储结构是逻辑结构用计算机语言的实现（亦称为映像），它依赖于计算机语言。对机器语言而言，存储结构是具体的。一般地，只在高级语言的层次上讨论存储结构。

(3) 数据的运算，即对数据施加的操作。数据的运算定义在数据的逻辑结构上，每种逻辑结构都有一个运算的集合。最常用的检索、插入、删除、更新、排序等运算实际上只是在抽象的数据上所施加的一系列抽象的操作。

所谓抽象的操作，是指只知道这些操作是“做什么”，而无须考虑“如何做”。只有确定了存储结构之后，才考虑如何具体实现这些运算。

下面举例来说明有关数据结构的概念。

例 1-2 学生成绩表，见表 1-1。

表 1-1 学生成绩表

学 号	姓 名	数学分析	普通物理	高等代数	平均成绩
10001	丁一	86	76	78	80
10002	刘二	79	83	84	82
10003	张三	92	88	78	86
10004	李四	87	79	86	84
10005	王五	90	92	79	87
⋮	⋮	⋮	⋮	⋮	⋮

分析上表涉及的数据的逻辑结构、存储结构和相关运算。

1) 逻辑结构

表中的每一行是一个记录（数据元素），它由学号、姓名、各科成绩及平均成绩等数据项组成。

表中记录之间的逻辑关系是：对表中任意一记录，与它相邻且在它前面的记录（亦称为直接前趋，Immediate Predecessor）最多只有一个；与表中任一记录相邻且在其后的记录（亦称为直接后继，Immediate Successor）也最多只有一个。表中只有第一个记录没有直接前趋，故称为开始记录；也只有最后一个记录没有直接后继，故称之为终端记录。例如，表中“刘二”所在记录的直接前趋记录和直接后继记录分别是“丁一”和“张三”所在的记录，上述记录间的关系构成了这张学生成绩表的逻辑结构。

2) 存储结构

该表的存储结构是指用计算机语言如何表示记录之间的这种关系，即表中的记录是顺序邻接地存储在一片连续的单元之中，还是用指针将这些结点链接在一起？

3) 数据的运算

在上面的学生成绩表中，可能要经常查看某一学生的成绩；当学生退学时要删除相应的记录；进来新学生时要增加记录。究竟如何进行查找、删除、插入，这就是数据的运算问题。

搞清楚了上述三个问题，也就弄清了学生成绩表这个数据结构。

1.3 数据结构的分类

从系统构成层次角度来看，可将数据结构分为逻辑结构和物理结构。但经常也将数据的逻辑结构简称为数据结构。数据的逻辑结构有两大类。

1. 线性结构

线性结构的逻辑特征是：若结构是非空集，则有且仅有一个开始元素和一个终结元素，并且所有数据元素都最多只有一个直接前趋和一个直接后继。

线性表是一种典型的线性结构。栈、队列、串等都是线性结构。

2. 非线性结构

非线性结构的逻辑特征是：一个数据元素可能有多个直接前趋和直接后继。数组、广义表、树和图等数据结构都是非线性结构。

1.4 数据的四种基本存储方法

数据的存储结构可用以下四种基本存储方法得到。

1. 顺序存储方法

该方法把逻辑上相邻的结点存储在物理位置上相邻的存储单元里，结点间的逻辑关系由存储单元的邻接关系来体现。

由此得到的存储表示称为顺序存储结构（Sequential Storage Structure），通常借助程序语言的数组描述。

该方法主要应用于线性的数据结构，非线性的数据结构也可通过某种线性化的方法实现顺序存储。

2. 链式存储方法

该方法不要求逻辑上相邻的结点在物理位置上亦相邻，结点间的逻辑关系由附加的

指针字段表示。由此得到的存储表示称为链式存储结构 (Linked Storage Structure), 通常借助于程序语言的指针类型描述。

3. 索引存储方法

该方法通常在存储结点信息的同时, 还建立附加的索引表。

索引表由若干索引项组成。若每个结点在索引表中都有一个索引项, 则该索引表称之为稠密索引 (Dense Index)。若一组结点在索引表中只对应一个索引项, 则该索引表称为稀疏索引 (Sparse Index)。索引项的一般形式为

(关键字、地址)

关键字是能唯一标识一个结点的那些数据项。稠密索引中索引项的地址指示结点所在的存储位置; 稀疏索引中索引项的地址指示一组结点的起始存储位置。

4. 散列存储方法

该方法的基本思想是: 根据结点的关键字直接计算出该结点的存储地址。

四种基本存储方法, 既可单独使用, 也可组合起来对数据结构进行存储映像。

同一逻辑结构采用不同的存储方法, 可以得到不同的存储结构。选择何种存储结构来表示相应的逻辑结构, 视具体要求而定, 主要考虑运算方便及算法的时空要求。

1.5 数据结构三方面的关系

数据的逻辑结构、数据的存储结构及数据的运算这三方面是一个整体。孤立地去理解一个方面, 而不注意它们之间的联系是不可取的。

存储结构是数据结构不可缺少的一个方面: 同一逻辑结构的不同存储结构可冠以不同的数据结构名称来标识。

例 1-3 线性表是一种逻辑结构, 若采用顺序方法的存储表示, 可称其为顺序表; 若采用链式存储方法, 则可称其为链表; 若采用散列存储方法, 则可称为散列表。

数据的运算也是数据结构不可分割的一个方面。在给定了数据的逻辑结构和存储结构之后, 按定义的运算集合及其运算的性质不同, 也可能导致完全不同的数据结构。

例 1-4 若对线性表上的插入、删除运算限制在表的一端进行, 则该线性表称之为栈; 若对插入限制在表的一端进行, 而删除限制在表的另一端进行, 则该线性表称之为队列。

更进一步, 若线性表采用顺序表或链表作为存储结构, 则对插入和删除运算做了上述限制之后, 可分别得到顺序栈或链栈, 顺序队列或链队列。

习 题

1. 数据结构与数据类型有什么不同?
2. 简述数据结构与存储结构的关系。
3. 数据的存储结构有哪几种?
4. 简述数据结构的构成, 并举一个数据结构的例子, 指出其逻辑结构、存储结构及运算的内容。

第2章 算法及算法分析基础

数据结构与算法之间存在本质联系,在某一类型数据结构上,总要涉及其上施加的运算,而只有通过对定义运算的研究,才能清楚理解数据结构的定义和作用;在涉及运算时,总要联系到该算法处理的对象和结果的数据。

2.1 算法的基本概念

在数据结构与算法这门课程中,将遇到大量的算法问题,因为算法联系着数据在计算过程中的组织方式,为了描述实现某种操作,常常需要设计算法。因而,算法是研究数据结构的重要途径,也是计算机求解问题的核心。

1. 算法(Algorithm)定义

算法是规则的有限集合,是为解决特定问题而规定的一系列有限操作序列。

2. 算法的特性

- (1) 有限性。有限步骤之内正常结束,不能形成无穷循环。
 - (2) 确定性。算法中的每一个步骤必须有确定含义,无二义性。
 - (3) 输入。有多个或0个输入。
 - (4) 输出。至少有一个或多个输出。
 - (5) 可行性。原则上能精确进行,操作可通过已实现基本运算执行有限次而完成。
- 在算法的五大特性中,最基本的是有限性、确定性、可行性。

3. 算法设计的要求

当用算法来解决某问题时,算法设计要达到的目标:正确、可读、健壮、高效、低耗。通常作为一个好的算法,一般应该具有以下几个基本特征。

1) 算法的正确性

算法正确性是指算法应该满足具体问题求解的需要。

从程序角度来看,其中“正确性”的含义大体上可以分为四个层次。

- (1) 所设计的程序没有语法错误。
- (2) 所设计的程序对于几组输入数据能够得出满足要求的结果。
- (3) 所设计的程序对于精心选择的典型、苛刻而带有刁难性的几组输入数据能够得到满足要求的结果。
- (4) 程序对于一切合法的输入数据都能产生满足要求的结果。

对于这四层含义,其中达到第(4)层含义下的正确是极为困难的。一般情况下,以第(3)层含义的正确作为衡量一个程序是否正确的标准。

例2-1 要求 n 个数的最大值问题,给出示意算法如下:


```

max:=0;
for(i=1; i<= n; i++)
{
    scanf("%f", &x);
    if (x>max) max=x;
}

```

求最大值的算法无语法错误；虽当输入 n 个数全为正数时，结果也对，但对输入 n 个数全为负数时，求到的最大值为 0，显然这个结果不对，由这个简单的例子可以说明算法正确性的内涵。

2) 可读性

一个好的算法首先应该便于人们理解和相互交流，其次才是机器可执行。可读性好的算法有助于人对算法的理解，并且难懂的算法易于隐藏错误且难于调试和修改。

3) 健壮性

作为一个好的算法，当输入的数据非法时，也能适当地做出正确反应或进行相应的处理，而不会产生一些莫名其妙的输出结果。

4) 高执行速度和低存储量

算法的效率通常是指算法的执行时间。对于一个具体问题的解决通常可以有多个算法，对于执行时间短的算法其效率就高。所谓的存储量需求是算法在执行过程中所需要的最大存储空间，这两者都与问题的规模有关。

2.2 算法的描述

著名的计算机科学家 N.沃思(Niklaus Wirth)给出了一个著名的公式：算法+数据结构=程序，说明数据结构和算法是程序的两大要素，二者相辅相成，缺一不可。

1. 算法、语言、程序的关系

先分析数据结构中算法、语言、程序的关系。

(1) 算法：描述了数据对象的元素之间的关系(包括数据逻辑关系、存储关系描述)。

(2) 描述算法的工具：自然语言、框图或高级程序设计语言。

算法的描述可用自然语言、框图或高级程序设计语言，自然语言简单但易于产生二义。框图直观但不擅长表达数据组织结构，而其中以高级程序语言较为准确但又过于计较细节。

(3) 程序是算法在计算机中的实现(与所用计算机及所用语言有关)。

2. 设计实现算法过程步骤

(1) 找出与求解有关的数据元素之间的关系(建立结构关系)。

(2) 确定在某一数据对象上所施加运算。

(3) 考虑数据元素的存储表示。

(4) 选择描述算法的语言。

(5) 设计实现求解的算法，并用程序语言加以描述。

3. 类描述算法的语言选择

高级语言描述算法，具有严格准确的优点，但用于描述算法，也有语言细节过多的