

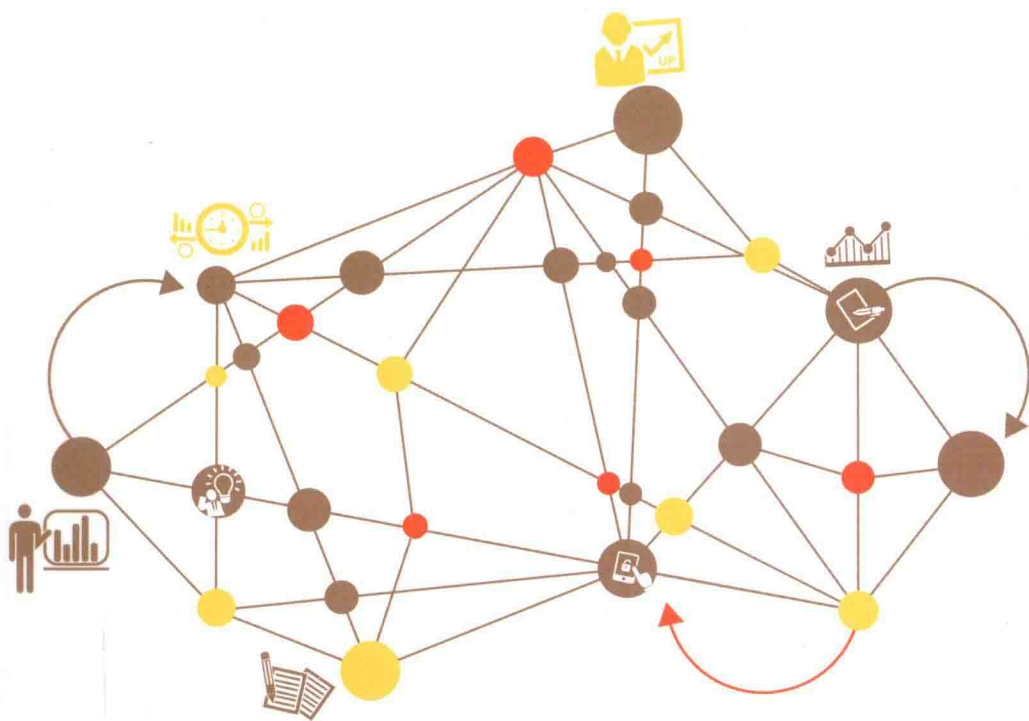
需要整本电子书，联系我QQ: [2667271557](#);
此处是样章，取的完整版的前面几页，和最后
面几页；完整版是带书签的，样章没带书签；
另外需要其他书，也可以找我。

非常白话并且很好理解的算法介绍和实现思路
温故知新，带你轻松走进互联网企业的大门

Broadview®
www.broadview.com.cn

轻松学 算法

互联网算法面试宝典 / 赵焱 编著

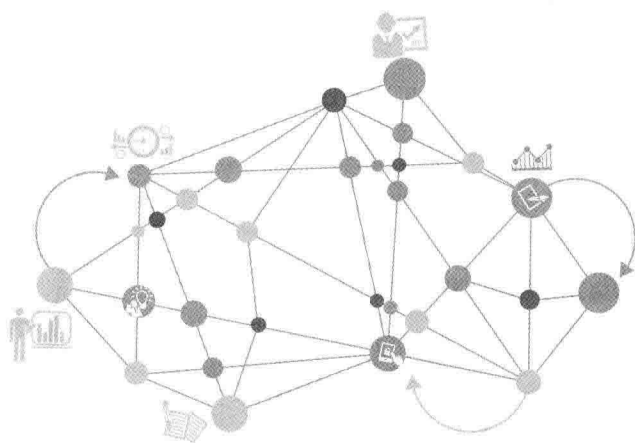


中国工信出版集团

电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

轻松学算法

互联网算法面试宝典 / 赵焯 编著



电子工业出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书共分为 12 个章节, 首先介绍了一些基础的数据结构, 以及常用的排序算法和查找算法; 其次介绍了两个稍微复杂一些的数据结构——树和图, 还介绍了每种数据结构和算法的适用场景, 之后是一些在工作与面试中的实际应用, 以字符串、数组、查找等为例介绍了一些常见的互联网面试题及分析思路, 便于读者了解这些思路, 顺利地通过互联网公司的面试; 最后介绍了一些常见的算法思想, 便于读者对今后遇到的算法问题更轻易地想出解决方案。

本书的讲解轻松有趣, 易于读者把烦琐、枯燥的算法学习变为有趣、愉快的学习, 把被动学习变为主动学习。本书也介绍了一些会在工作面试中用到的算法。对于一些正在学习算法的人来说, 本书绝对是可以帮助你轻松掌握算法的辅助资料; 对于已经了解算法的人来说, 可以从本书中了解到这些算法是如何在实际工作中使用的。

本书适合即将毕业的学生、初入职场的工程师及想补充基础算法知识的人学习, 也适合作为一本互联网公司面试的参考书, 更是一本不可多得的便于读者时常补习算法知识的收藏宝典。

未经许可, 不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有, 侵权必究。

图书在版编目 (CIP) 数据

轻松学算法: 互联网算法面试宝典 / 赵焯编著. —北京: 电子工业出版社, 2016.7

ISBN 978-7-121-29194-4

I. ①轻… II. ①赵… III. ①算法语言 IV. ①TP312

中国版本图书馆 CIP 数据核字 (2016) 第 143464 号

策划编辑: 董 英

责任编辑: 徐津平

印 刷: 三河市双峰印刷装订有限公司

装 订: 三河市双峰印刷装订有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 787×980 1/16 印张: 25.5 字数: 500 千字

版 次: 2016 年 7 月第 1 版

印 次: 2016 年 7 月第 1 次印刷

印 数: 3000 册 定价: 69.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888, 88258888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式: 010-51260888-819 faq@phei.com.cn。

前言

互联网越来越热门了，相信每个人都或多或少地在体验如今的互联网带给我们的各种便利。借助之前非常火热的电商，我们可以足不出户地购买衣服和日常用品；现在，就算是一个超级宅的人，也可以不出门便可完成订餐、在线交水电费、通过各种到家服务进行保洁和按摩、通过应用将在超市买的東西轻松配送到家，等等。在线办公在一些行业中也开始流行，视频会议更是可以轻松地实现异地交流。之前我们还需要见面签约，现在越来越多地在进行在线签约。

不得不感谢互联网带给我们的各种便利，但这背后是由很多产品、运营和技术人员的努力在支撑的。

我最初所在的公司属于传统行业，但也是一家服务与互联网公司。相信任何行业的技术员工都想进入互联网公司，这其中的好处有太多：技术的挑战、大用户量、大数据、高并发，这些都是我们所渴望的。

经常听到身边有很多人在抱怨算法不好学、学会了记不住、记住了不知道怎么用等，所以我决定写本书，结合自己的经验讲解一些算法的实际应用及适用场景，希望通过本书帮助更多的朋友进入互联网公司。

其实很多人怯场时无非担心的是自己的算法太差、技术太烂、别人会瞧不起，等等。本书可以帮助读者解决一些基础、常见的算法问题，当然，在技术上仍需自己努力，若再有一点运气，则一定可以找到理想的公司。不要害怕，很多时候就算没有面试成功，也应

该总结一下，等过段时间后便能感悟到自己的成长。

算法有很多，而且不停地有新的算法出现。本书将介绍其中一些比较基础且常用的算法，当然，会先简单介绍几个基础的数据结构作为学习算法的铺垫；之后会介绍一些在工作中可能用到的算法；最后会介绍一些新兴的算法，以拓展读者的思路。

我会尽量以轻松、愉快的方式介绍每一个算法，由浅入深地介绍如何在工作中使这些算法变成我们的代码，让我们在开发应用时更高效。

目 录

第 1 章 数组、集合和散列表	1
1.1 要用就要提前想好的数据结构——数组	2
1.1.1 什么是数组	2
1.1.2 数组的存储结构	3
1.1.3 数组的特点	6
1.1.4 数组的适用场景	7
1.2 升级版数组——集合	8
1.2.1 什么是集合	8
1.2.2 集合的实现	8
1.2.3 集合的特点	13
1.2.4 集合的适用场景	13
1.2.5 数组与变长数组的性能	14
1.3 数组的其他应用——散列表	14
1.3.1 什么是散列表	15
1.3.2 对散列表函数产生冲突的解决办法	16
1.3.3 散列表的存储结构	17
1.3.4 散列表的特点	18
1.3.5 散列表的适用场景	20
1.3.6 散列表的性能分析	21

1.4	小结	28
第 2 章	栈、队列、链表	29
2.1	汉诺塔游戏——栈	30
2.1.1	什么是汉诺塔	30
2.1.2	什么是栈	31
2.1.3	栈的存储结构	31
2.1.4	栈的特点	36
2.1.5	栈的适用场景	36
2.2	火爆的奶茶店——队列	37
2.2.1	什么是队列	37
2.2.2	队列的存储结构	38
2.2.3	队列的特点	43
2.2.4	队列的适用场景	44
2.3	用栈实现队列	45
2.3.1	用两个栈实现队列	46
2.3.2	两个队列实现栈	50
2.4	链表	53
2.4.1	什么是链表	54
2.4.2	链表的存储结构	54
2.4.3	链表的操作	55
2.4.4	链表的特点	66
2.4.5	链表的适用场景	66
2.4.6	链表的性能分析	67
2.4.7	面试举例：如何反转链表	68
2.5	链表其实也可以用数组模拟	69
2.5.1	静态链表	70
2.5.2	静态链表的实现	70
2.5.3	静态链表的特点	80
2.6	再谈汉诺塔	81
2.6.1	汉诺塔的移动原理	81
2.6.2	汉诺塔的递归实现	82

第 3 章 排序算法	84
3.1 算法基础	85
3.1.1 时间复杂度	85
3.1.2 空间复杂度	88
3.1.3 稳定性	88
3.2 快而简单的排序——桶排序	89
3.2.1 举个例子	89
3.2.2 什么是桶排序	90
3.2.3 桶排序的实现	90
3.2.4 桶排序的性能及特点	92
3.2.5 桶排序的适用场景	93
3.3 咕嘟咕嘟的冒泡排序	94
3.3.1 什么是冒泡排序	94
3.3.2 冒泡排序的原理	94
3.3.3 冒泡排序的实现	96
3.3.4 冒泡排序的特点及性能	99
3.3.5 冒泡排序的适用场景	99
3.3.6 冒泡排序的改进方案	100
3.4 最常用的快速排序	100
3.4.1 什么是快速排序	101
3.4.2 快速排序的原理	101
3.4.3 快速排序的实现	105
3.4.4 快速排序的特点及性能	107
3.4.5 快速排序的适用场景	108
3.4.6 快速排序的优化	108
3.5 简单的插入排序	109
3.5.1 什么是插入排序	110
3.5.2 插入排序的原理	110
3.5.3 插入排序的实现	112
3.5.4 插入排序的特点及性能	114
3.5.5 插入排序的适用场景	115

3.6	直接插入的改进——希尔排序	115
3.6.1	什么是希尔排序	116
3.6.2	希尔排序的原理	116
3.6.3	希尔排序的实现	118
3.6.4	希尔排序的特点及性能	120
3.6.5	希尔排序的适用场景	121
3.7	简单选择排序	121
3.7.1	什么是选择排序	122
3.7.2	简单选择排序的原理	122
3.7.3	简单选择排序的实现	123
3.7.4	选择排序的特点及性能	125
3.7.5	简单选择排序的优化	125
3.7.6	选择排序的适用场景	126
3.8	小结	126
第 4 章	搜索，没那么难	128
4.1	最先想到的——顺序查找	129
4.1.1	最先想到的	129
4.1.2	顺序查找的原理与实现	129
4.1.3	顺序查找的特点及性能分析	131
4.1.4	顺序查找的适用场景	132
4.2	能不能少查点——二分查找	133
4.2.1	某些特殊情况的查找需求	133
4.2.2	二分查找的原理及实现	133
4.2.3	二分查找的优化	137
4.2.4	二分查找的特点及性能分析	138
4.2.5	二分查找的适用场景	139
4.2.6	我是来还债的——之前欠的二分插入排序	139
4.3	行列递增的矩阵查找——二分查找思维拓展	141
4.3.1	一道题	142
4.3.2	几个解法	142
4.3.3	其他拓展	153

4.4	分块查找	154
4.4.1	每次插入元素都要有序吗	154
4.4.2	什么是分块查找	155
4.4.3	分块查找的原理及实现	155
4.4.4	分块查找的特点与性能分析	159
4.4.5	分块查找的适用场景	160
4.5	查找算法小结	161
4.6	搜索引擎与倒排索引	162
4.6.1	什么是搜索引擎	162
4.6.2	倒排索引	162
4.6.3	索引实例	163
4.6.4	倒排索引的关键字提取	164
4.6.5	商业索引的其他拓展	164
第 5 章	树	166
5.1	树的定义及存储结构	167
5.1.1	什么是树	167
5.1.2	其他相关术语	168
5.1.3	都有哪些树	170
5.1.4	树的存储结构及实现	170
5.2	二叉树	173
5.2.1	什么是二叉树	173
5.2.2	还有两种特殊的二叉树	173
5.2.3	二叉树的实现	175
5.2.4	二叉树的遍历	185
5.2.5	完全二叉树	187
5.3	二叉树的查找算法	188
5.3.1	二叉查找树	188
5.3.2	平衡二叉树	198
5.4	B-树、B+树	202
5.4.1	什么是 B-树	203
5.4.2	什么是 B+树	204

5.4.3	B-树、B+树的特点及性能分析	205
5.5	在 MySQL 数据库中是如何应用 B+树的	206
5.6	哈夫曼树	209
5.6.1	几个术语	209
5.6.2	什么是哈夫曼树	209
5.6.3	哈夫曼树的构造	211
5.6.4	哈夫曼编码及解码	213
5.7	堆	215
5.7.1	什么是堆	215
5.7.2	堆的基本操作	216
5.7.3	堆的性能分析	221
5.7.4	堆排序	222
5.8	红黑树	224
5.8.1	什么是红黑树	224
5.8.2	红黑树的特点与优势	224
第 6 章	图	226
6.1	图的定义及相关术语	227
6.1.1	什么是图	227
6.1.2	图的分类	227
6.1.3	图的相关术语	228
6.2	图的表示与存储方式	229
6.2.1	邻接矩阵	229
6.2.2	邻接表	234
6.3	更多的图	237
6.3.1	连通图	238
6.3.2	强连通图	238
6.4	深度优先遍历与广度优先遍历	238
6.4.1	深度优先遍历	239
6.4.2	广度优先遍历	248
6.4.3	两种遍历方法的对比	253

6.5 最短路径	254
6.5.1 带权图	254
6.5.2 Dijkstra 算法	255
6.5.3 Floyd 算法	269
第 7 章 字符串	272
7.1 字符及字符串简介	273
7.1.1 什么是字符	273
7.1.2 什么是字符串	273
7.2 字符的全排列	275
7.2.1 问题描述及分析	275
7.2.2 最先想到的	275
7.2.3 利用字典序排列	278
7.3 反转字符串	283
7.3.1 问题描述及分析	283
7.3.2 最先想到的	283
7.3.3 对换反转法	285
7.3.4 拓展——旋转字符串	287
7.4 判断回文	288
7.4.1 问题描述及分析	288
7.4.2 对半判断	289
7.5 寻找最大的回文子串	290
7.5.1 问题描述及分析	290
7.5.2 遍历实现	291
7.6 将字符串转换为数字	293
7.6.1 问题描述及分析	293
7.6.2 解决	293
7.7 判断字符串包含的问题	297
7.7.1 问题描述及分析	297
7.7.2 非常简单的解决思路	297
7.7.3 利用排序进行优化	299
7.7.4 投机取巧的素数方案	302

7.7.5	用散列表进行实现	304
7.7.6	用位运算进行实现	305
第 8 章	数组还有好多玩法	308
8.1	从数组中找出其和为指定值的两个数	309
8.1.1	问题描述及分析	309
8.1.2	最简单的办法	309
8.1.3	一切为了速度	310
8.1.4	排序总是好的选择	311
8.1.5	还能更好	313
8.2	找出连加值最大的子数组	315
8.2.1	问题描述及分析	315
8.2.2	暴力穷举法	316
8.2.3	动态规划法	319
8.2.4	问题拓展	321
8.3	数组正负值排序	323
8.3.1	问题描述及分析	323
8.3.2	最直观的解法	324
8.3.3	借鉴简单插入排序	325
8.3.4	借鉴快速排序	327
8.3.5	拓展	329
8.4	将数组随机打乱顺序	329
8.4.1	问题描述及分析	329
8.4.2	随便糊弄一下	330
8.4.3	插入排序的思想又来了	331
8.5	数组赋值	333
8.5.1	问题描述及分析	333
8.5.2	分解计算	333
8.6	寻找旋转数组的拐点	335
8.6.1	问题描述及分析	335
8.6.2	最简单的方法	336
8.6.3	利用“有序”	337

8.7 荷兰国旗问题	339
8.7.1 问题描述及分析	339
8.7.2 排序法	340
8.7.3 快速排序带给人的灵感	340
第 9 章 查找又来了	344
9.1 出现次数超过一半的数字	345
9.1.1 问题描述及分析	345
9.1.2 排序法	345
9.1.3 散列表	347
9.1.4 删除法	349
9.1.5 更优的解法	349
9.2 寻找缺少的数字	352
9.2.1 问题描述及分析	352
9.2.2 借助快速排序	352
9.2.3 借助散列表实现	354
9.2.4 投机取巧法	355
9.2.5 思路拓展	357
9.3 在 10 亿个数中找出最大的 1 万个数	357
9.3.1 问题描述及分析	357
9.3.2 拍脑袋想问题	358
9.3.3 借助快速排序	358
9.3.4 不想都放入内存	358
9.3.5 传说中的大顶堆	359
9.3.6 拓展——找出数组中第 k 大的数	359
第 10 章 更多	363
10.1 不使用额外的空间交换两个数	364
10.1.1 问题描述	364
10.1.2 分析问题	364
10.1.3 解决问题	364

10.2	拿乒乓球的问题	365
10.2.1	问题描述	365
10.2.2	分析问题	365
10.2.3	解决问题	365
第 11 章	实现一些集合类	367
11.1	栈 (Stack) 的实现	368
11.1.1	实现前的思考	368
11.1.2	实现栈	368
11.1.3	参考 JDK 的实现	372
11.2	变长数组 (ArrayList) 的实现	372
11.2.1	实现前的思考	372
11.2.2	实现变长数组	373
11.2.3	参考 JDK 的实现	380
11.3	散列表 (HashMap) 的实现	381
11.3.1	实现前的思考	381
11.3.2	实现散列表	381
11.3.3	参考 JDK 的实现	389
第 12 章	方向	390
12.1	算法的一些常用思想	391
12.1.1	分治法	391
12.1.2	动态规划	391
12.1.3	贪心算法	391
12.1.4	回溯法	392
12.1.5	分支限界法	392
12.2	新兴算法	392
12.2.1	加密算法	392
12.2.2	商业算法	393
12.3	其他算法	393
12.3.1	基数估计算法	393
12.3.2	蚁群算法	394

数组、集合和散列表

我们将在本章中学习最基础、最简单的数据结构及其延伸。

数组，作为数据结构中最基础的一个存储方式，是我们学习一切数据结构、算法的基石。大部分数据结构都可以用数组来实现。本章会介绍数组的概念、存储结构、特点及适用场景。

集合这个结构其实并不存在于我们在学校学习的数据结构知识中，而是在一些高级语言中出现的，算是升级版的数组。本章会介绍集合的特点、实现及它的适用场景。

散列表，又叫哈希表（Hash Table），其实在很多高级语言里是在数组的基础上实现的，当然散列表也有其他实现形式。本章会详细介绍散列表的基本概念、实现方式，并结合实现方式介绍其对应的存储结构、特点及适用场景。

1.1 要用就要提前想好的数据结构——数组

要用就要提前想好？为什么？这其实是由数组的一个特点决定的，那就是对于数组这个数据结构，在它之前必须提前想好它的长度；有了长度，才能知道该为这个存储结构开辟多少空间；而在决定了长度之后，不管我们最后往里面填充的数据够不够长，没有用到的空间也就都浪费了；如果我们想往这个数组中放入的数据超过了提前设定好的长度，那么是不可行的，因为空间只有这么大。

1.1.1 什么是数组

数组（Array），就是把有限个数据类型一样的元素按顺序放在一起，用一个变量命名，然后通过编号可以按顺序访问指定位置的元素的一个有序集合。

其实简单来说，就是为了方便而把这些元素放在一起。我们通过编号去获取每个元素，这个编号叫作下标或者索引（Index），一般的语言是从 0 开始的。

我们常说的数组一般指一维数组，当然还有多维数组，虽然多维数组并不常用。

多维的实现其实是数组的某些元素本身也是一个数组，这里以一个标准的二维数组为例进行介绍。其实，二维数组相当于每个元素的长度都一样的一个一维数组（也就是我们常说的数组）。可以想象一下矩阵（若不了解矩阵，则请阅读 1.1.2 节中标准二维数组的存储结构示例），其实它和数学中的矩阵类似。

在很多弱语言中，并不要求每个元素的长度都一样，可以某些元素是数组（长度可以不一样），某些元素不是数组，甚至每个元素的数据类型都不同。这里讲的二维数组指的是标准的二维数组。

注：弱类型语言也叫作弱类型定义语言，简称弱语言。弱语言一般对语言的标准没有特别的要求。比如在 JavaScript 中用 var 声明变量，不会指定该变量是哪种类型。如果想更多地了解弱语言，则请参考 JavaScript，该语言主要用于前端开发。强语言对编写规则比较

有要求，所以不容易出现问题，很多开发工具都会帮助检查其基本规则。

1.1.2 数组的存储结构

在了解了什么是数组之后，我们来看下数组的存储结构。

1. 数组的存储结构

首先我们来看下一维数组的存储结构，如图 1-1 所示。

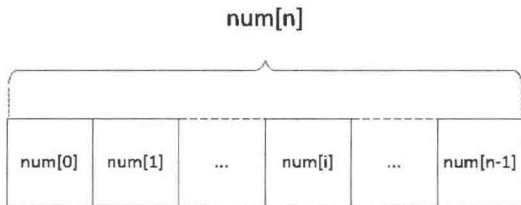


图 1-1 数组的存储结构

其实，我们先要确定一个值，也就是数组的长度；然后，系统会根据我们声明的数据类型开辟一些空间（当然，每种数据类型需要开辟的空间也不一样）。这时，这些空间就归这个变量所有了。一般在编程语言的实现中，这些空间会默认对我们声明的数据类型赋值，比如整型值是 0，布尔值是 false，等等。

所以有以下几种情况。

（1）只声明了指定长度的空间，没有初始化值（以整型为例，所有值都会默认为 0，如图 1-2 所示）。

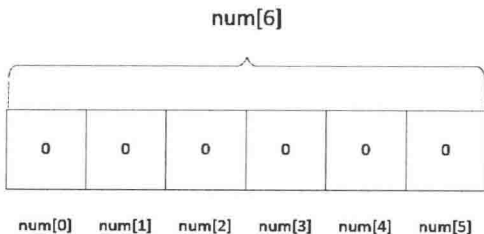


图 1-2 在未初始化值时会默认初始化为 0

(2) 声明了指定长度的空间，初始化了部分值（以整型为例，未初始化的值都会默认为 0，如图 1-3 所示）。

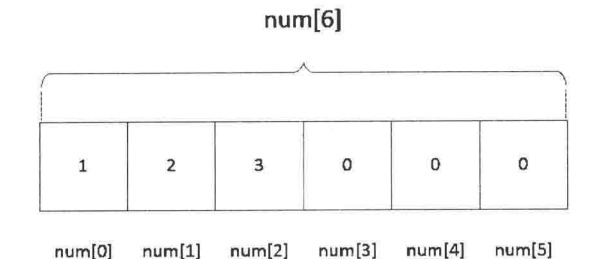


图 1-3 只初始化了前三个值，其余值自动初始化为 0

(3) 声明了指定长度的空间，初始化了全部的值，如图 1-4 所示。

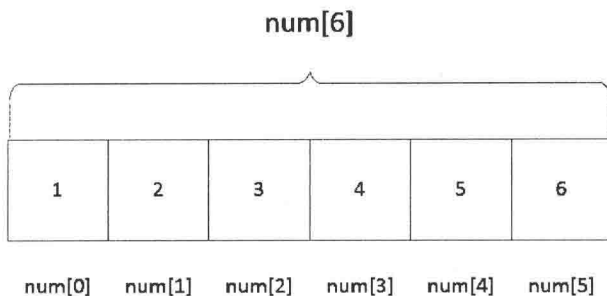


图 1-4 初始化了全部的值

2. 数组在编程语言中的初始化及操作

在多数语言中，数组的声明都是非常简单的，一般有下面几种声明方式（以 Java 语言、整型为例，其他语言、数据类型差异不大）。

```
int[] num1 = new int[10];  
int[] num2 = {1, 2, 5};  
int[] num3 = new int[3];  
num3[0] = 1;  
num3[1] = 2;  
num3[2] = 5;
```

数组指定位置的元素的值，是通过下标获取的，下标在大部分语言中是从 0 开始的。

```
int[] num = {1, 2, 5};
int a = num[0]; // a的值为1
int b = num[1]; // b的值为2
```

为数组赋值，和获取元素的值类似，可以直接赋值。

```
int[] num = {1, 2, 5};
num[1] = 10; // 现在num的数组元素分别为1, 10, 5
```

数组常用的另一种方式是按顺序访问每一个值，一般通过编程语言中的循环语句实现（比如 for 循环）。

```
int[] num = {1, 2, 5};
for (int i = 0; i < num.length; i++) {
    System.out.println(num[i]);
}
```

上面展示了循环打印数组的值的代码，其中 `num.length` 可以获取数组的长度。细心的同学可以发现这个 `length` 后面没有括号，是的，这个 `length` 是数组的内置属性（以Java为例，有些语言会同时提供两种或更多的获取数组长度的方式）。

下面我们看下二维数组的存储结构，如图 1-5 所示。

num[3][6]	num[0]	num[0][0]	num[0][1]	num[0][2]	num[0][3]	num[0][4]	num[0][5]
	num[1]	7	8	9	10	11	12
	num[2]	13	14	15	16	17	18

图 1-5 二维数组的存储结构

二维数组的初始化方式实际上和一维数组没有太大的区别，只不过我们需要提前确定第 1 维和第 2 维的长度。

在图 1-5 中，第 1 维的长度为 3，每维的元素又是一个长度为 6 的数组。

3. 多维数组在编程语言中的初始化及操作

由于多维数组与一维数组的初始化及访问区别不大，下面集中进行列举。

```
int[][] num = new int[3][3];  
int[][] num2 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};  
int a = num2[1][0]; // a 的值是 4  
int b = num2[1].length; // b 的值为 3，即第二维的长度
```

二维数组的访问方式其实和一维数组和多维数组的访问方式没什么区别，我们只需要理解为数组的元素也是个数组。

1.1.3 数组的特点

因为本身存储的方式，数组有如下特点。

1. 定长

数组的长度是固定的，这是数组最重要的一个特点。

只要我们在声明时确定了数组的长度，在赋值时就算没有给数组的所有元素赋值，未赋值的元素也是有初始默认值的；而如果我们在赋值时发现数组的长度不够用，这时也没有什么好办法，因为数组的长度无法改变。要是想继续存放数据，就只能重新声明一个数组了。

2. 按顺序访问

我们在访问一个数组中的某个元素时，必须从第 1 个元素开始按顺序访问，直到访问到指定位置的元素。

这里想说明一点，虽然我们在开发时可以直接通过下标访问指定位置的元素，但是实际上计算机在处理时也是按顺序访问的。

1.1.4 数组的适用场景

数组其实是一个非常简单的数据结构，用起来也比较简单。但是，数组是所有数据结构的基础，我们必须掌握好数组，才能更好地学习其他数据结构和算法。

数组适合在什么时候用，其实根据数组的特点我们就可以想到，由于数组的长度一般是固定的，所以在不会出现变化的业务上比较适合使用数组。

注：本书所举例的适用场景只是可以并且多数会这样做，并不是说这种使用方式是最优的或者可不替代的。实际上有些团队可能会有一些更好的实现方式。

1. 技能快捷键

不知道大家对 RPG（ARPG）等类似的游戏了解多少，在这类游戏中会有一排快捷键技能格，比如 F1~F9 这样 9 个快捷键技能格，我们每个人可以把自己惯用的技能拖动到这些技能格上，这样就可以直接通过技能快捷键操控技能了。一般在这种设计中，一个游戏的快捷键格子会有固定的个数。于是，我们在程序里就可以通过数组来存储每个人的技能快捷键对应的技能（当然，肯定会通过一定的映射存到数据库之类的磁盘上）。

2. 优酷 8+1

先声明，我没有参与制作优酷 8+1，这里只是举例。优酷 8+1 是什么？打开优酷首页，会看到最上面的 1 个大图、8 个小图，这就是优酷 8+1 了。

这里我们可以声明一个长度为 9 的数组，里面的每个元素是一个对象（对象是高级语言中的一个名词，包含了一系列的变量和方法），这些对象至少应该包含图片地址（用于展示图片）和 URL 地址（用于在单击后跳转）。

到这里，我们应该已经很清晰地认识到了数组的劣势，那就是在用之前必须提前确定数组的长度，而后不管我们的技能是否需要增加快捷键位，或者优酷首页从 8+1 变成了 11+1，都会导致对程序进行一定的改动。这时我们就该认识一下数组的升级版——集合了。

1.2 升级版数组——集合

数组的致命缺点就是长度固定，如果我们一开始不确定长度，该怎么办呢？这时就需要集合了，其实集合也是基于数组实现的。在许多高级语言中，集合是对数组的一个拓展，我们在向里面放数据时，想放多少就可以放多少，不用在意集合到底能放多少（当然得内存够用才行）。

1.2.1 什么是集合

集合的概念有些宽泛。本节讲的集合主要是可变长度的列表（也叫作动态数组）。

下面这些都是集合。

- ◎ 列表：一般是有序的集合，特点就是有顺序，比如链表、队列、栈等，我们会在第2章学习这些集合。
- ◎ 集：一般是无序的集合，特点就是没有顺序并且数据不能重复，多数语言是使用散列表实现的，支持对集进行添加、删除、查找包含等操作。
- ◎ 多重集：一般是无序的集合，特点是没有顺序，但是数据可以有重复的值，支持对集进行添加、删除、查找包含、查找一个元素在集中的个数等操作。多重集一般可以通过排序转换为列表。
- ◎ 关联数组：其实多数语言是使用散列表实现的，就是可以通过键（Key）获取到值（Value）。同样是没有顺序的。
- ◎ 树、图：同样是集合，我们会在后面进行详细了解。

1.2.2 集合的实现

本节说的集合在数据结构书中本来是不会有的，但我还是决定介绍一下，这不管是对

我们拓展思路还是了解更多的内容，都是有帮助的。

这里以 Java 中的 ArrayList 为例，它是一个数组的列表，其实就是数组的拓展，或者说是可变长度的数组。

上面一直提到，这个 ArrayList 是基于数组实现的，这如何理解呢？其实就是在 ArrayList 里有个属性，这个属性是一个数组。另外，还会有个属性记录我们放了多少数据，这样我们再向其中放数据时，就会知道该向这个内部数组的哪个位置放数据了，但是这个数组也会有长度限制，若超过了这个限制该怎么办呢？当超过这个限制时，其内部会创建一个具有更长的长度的数组，然后把旧数组的数据复制到新数组里面，这样就可以继续往里面放数据了。

在外部，我们感觉不到这个 ArrayList 是有长度限制的，它在自己内部都处理好了。

下面我们通过图 1-6 来形象地理解一下这个流程吧。

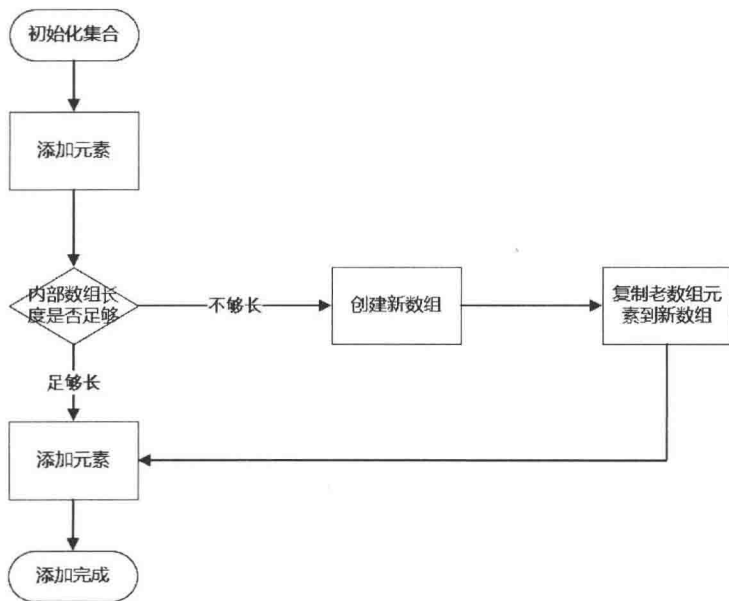


图 1-6 ArrayList 插入数据的流程

注：在面向对象的编程语言中一般把成员变量称为属性。

下面我们先来简单地用代码实现这个变长数组。在第 10 章中,我们会进一步实现 ArrayList 这个集合。

```
package me.irfen.algorithm.ch01;

import java.util.Arrays;

public class ArrayList {

    private static final int INITIAL_SIZE = 10;

    private int size = 0;

    private int[] array;

    public ArrayList() {
        array = new int[INITIAL_SIZE];
    }

    public ArrayList(int initial) {
        if (initial <= 0) {
            initial = INITIAL_SIZE;
        }
        array = new int[initial];
    }

    /**
     * 添加元素
     * @param num
     */
    public void add(int num) {
        if (size == array.length) {
            array = Arrays.copyOf(array, size * 2);
```

12.1 算法的一些常用思想

之前我们介绍了很多算法，并介绍了很多问题的解决方案，其中的很多内容用到了同样的算法思想，下面会进行简单介绍。

12.1.1 分治法

在计算机科学中，分治法是一种很重要的算法。其字面上的意思是“分而治之”，就是把一个复杂的问题分成两个或更多的相同或相似的子问题，再把子问题分成更小的子问题……直到最后子问题可以被简单地直接求解，原问题的解即子问题的解的合并。

快速排序和二分搜索使用的就是这种思想。

12.1.2 动态规划

动态规划的基本思想与分治法类似，也是将待求解的问题分解为若干个子问题（阶段），按顺序求解子问题，前一个子问题的解为后一个子问题的求解提供了有用的信息。在求解任意子问题时，列出各种可能的局部解，通过决策保留那些有可能达到最优的局部解，丢弃其他局部解。依次解决各子问题，最后一个子问题就是初始问题的解。

与分治法最大的差别是：适合用动态规划法求解的问题经分解后得到的子问题往往不是互相独立的（即下一个子阶段的求解是建立在上一个子阶段的解的基础上进行进一步的求解）。

我们在寻找最短路径的时候就用到了这种思想。

12.1.3 贪心算法

贪心算法是指在对问题求解时，总是做出在当前看来最好的选择。也就是说，不从整

体最优方面加以考虑，而是仅考虑在某种意义上局部最优的解。

12.1.4 回溯法

回溯法类似于枚举的搜索尝试过程，主要是在搜索尝试的过程中寻找问题的解，当发现已不满足求解条件时，就“回溯”返回，尝试别的路径。

回溯法是一种选优搜索法，按选优条件向前搜索，以达到目标。当搜索到某一步时，若发现原先的选择并不优或达不到目标，就退回一步重新选择。这种走不通就退回再走的方法叫作回溯法，而满足回溯条件的某个状态的点叫作回溯点。

许多复杂的规模较大的问题都可以使用回溯法，回溯法有“通用解题方法”的美称。

12.1.5 分支限界法

分支限界法类似于回溯法，也是一种在问题的解空间树 T 上搜索问题解的算法。但在一般情况下，分支限界法与回溯法的求解目标不同。回溯法的求解目标是找出 T 中满足约束条件的所有解；而分支限界法的求解目标则是找出满足约束条件的一个解，或者在满足约束条件的解中找出使某一目标的函数值达到极大或极小的解，即在某种意义下的最优解。

12.2 新兴算法

12.2.1 加密算法

加密算法其实不算是新兴算法，却是最近人们一直在讨论的内容。在 MD5 加密被碰撞破解之后，人们开始考虑各种各样的加密实现了。

这里给大家介绍几种常用的加密算法。

从理论上说，摘要算法也属于加密算法，MD5、SHA 系列的算法都是摘要算法，是不可逆的，也就是无法解密的。

而 RSA、3DES 之类的可逆算法，是可以解密的。

加密算法又分为对称加密算法和非对称加密算法。对称加密算法使用同一密钥进行加解密；而非对称加密算法使用不同的密钥进行加解密。

当然，除了我们应该熟知的这些常用的加密算法，还有很多很少见的算法，比如椭圆曲线算法等。

12.2.2 商业算法

其实，近年来有很多新兴算法，因为人们在不断地探索更好的东西，去改变或者适应更好的业务。

很多算法是为了具体的产品而设计出来的。

比如图像识别算法，为了保证互联网用户上传的内容是健康的，我们需要一定的算法去识别图片中有没有不好的内容，比如用户在新浪微博、微信朋友圈上传的图片等，这靠人工审核是完全不可能做到的。

又比如推荐算法，若在淘宝上买东西，就总能看到向我们推荐的相关商品，这是如何推荐和排序的？这就是算法的作用。淘宝的商品那么多，如何能够达到秒级推荐？当然，没有一个良好的算法设计是不可能做到的。

12.3 其他算法

12.3.1 基数估计算法

给定一个数据集，求解数据集的基数（Cardinality，也译作“势”，表示一个数据集中

不同数据项的数量)是非常普遍的需求。许多业务需求最终可以归结为基数求解,例如网站访问分析中的UV(访客数,指在一段时间内访问网站的不同用户的数量)。由于数据集基数是不可聚集指标(两个数据集的总基数无法通过各自的基数进行简单计算),因此如果要得到 N 个数据集任意组合的基数,则需要 $2N$ 次数据集的重计算,这是一个复杂度非常高的计算过程。当数据量较小时,可以采取bitmap“按位或”方法获得较高的计算速度;而当数据量很大时,一般会采取概率算法对基数进行估计。

假如我们有一个巨大的含有重复数据项的数据集,无法将其全部放到内存中进行处理,若想知道其中有多少不同的元素,由于数据集没有排好序,则对如此大的一个数据集进行排序和计数几乎是不可行的,这时该怎么办呢?

这就是一个典型的应用了,当然,这一切没有那么简单。

我们发现,在很多商业应用中,在某些情况下是允许我们对大量数据集的计算结果有偏差或者错误的,但错误的范围是多少,性能好多少,是验证这个算法好坏的标准。

12.3.2 蚁群算法

每个蚂蚁会在没有事先告诉它们食物在什么地方的前提下开始寻找食物。当一只蚂蚁找到食物后,它会向环境释放一种挥发性分泌物pheromone(叫作信息素,该物质随着时间的推移会逐渐挥发、消失,信息素浓度的大小表征路径的远近),吸引其他蚂蚁过来,这样越来越多的蚂蚁会找到食物。有些蚂蚁并没有像其他蚂蚁一样总重复同样的路,它们会另辟蹊径,如果另开辟的道路比原来的道路更短,那么渐渐地,更多的蚂蚁会被吸引到这条较短的路上来。最后,经过一段时间的运行,可能会出现一条最短的路径被大多数蚂蚁使用。

这就是蚁群算法(Ant Colony Optimization, ACO),又称蚂蚁算法,是一种用来在图中寻找优化路径的几率型算法。

淘宝也有一个类似的算法,它将人们进行了兴趣分类,认为某类人有着相同的兴趣,把各类有相同兴趣的人喜欢的东西集合(这个分类一定是非常细致的),这样就可以方便地为人们推荐更好的商品了。

本书主要介绍了一些数据结构的基础知识及面试中的常见问题,在看到本书前,我一直感觉大学的数据结构教材比较枯燥,本书将排序、查找、图论、树等重新进行了阐述,不再照本宣科。书中结合作者的工作经验对大量的案例进行了分析,并对算法进行了剖析,具有良好的学习性。在本书中,作者将基础知识融会贯通到工作项目中,对于初学者及应聘者都有很好的指导意义。

优酷直播互动高级技术总监 白云龙

数据结构、算法是软件开发的基础,在大数据时代的海量数据、高并发、高吞吐的需求场景下显得尤为重要。本书会结合日常工作中的应用场景来介绍各种基础数据结构和算法,帮助读者更好地理解其中的精髓。例如,结合Java集合框架来介绍线性数据结构、集合、散列表,结合MySQL索引来介绍B+树等。深入理解数据结构与算法,对日常的软件开发工作有深刻的意义。

优酷高级技术经理 戴润

数据结构与算法是每个编程人员都需要掌握的基础知识。如果你是一名计算机专业的学生,那么数据结构与算法是你必学、必考的内容;如果你是一名程序员,则不论是面试还是工作,你都会遇到与数据结构、算法相关的问题。而学习过数据结构与算法的人,可能会觉得其中的内容太多、范围太广,在实际应用(包括考试与面试)中又难以抓住重点。

本书向你介绍了常用的数据结构与算法,结合在工作与面试中常遇到的题目,既有理论基础,又不脱离应用,本书还为有兴趣学习更高级的算法的读者提供了指引。如果你对数据结构与算法感兴趣,而又担心其内容深奥、不易理解,则不如先看看本书。

步步高教育电子高级系统架构师 彭伟

算法是程序的灵魂,数据结构是算法的核心。作者灵巧生动地诉说了算法的原理及所涉及的数据结构的精髓。从目前主流技术的发展来看,算法对处于各个阶段的程序员来说十分重要。移动互联网时代已到达顶峰,内容变成了当前的主流趋势。对于优质的内容,传统的人工运营方式已经不能满足人们的需求,机器学习、数据挖掘会成为内容运营的中坚力量。在数据挖掘和机器学习的过程中,理解和运用基础算法是取得成功的关键。本书很适合对算法感兴趣但又不想钻研晦涩算法书籍的朋友阅读。

猎豹移动数据分析师 刘德顺

上架建议:计算机/算法

ISBN 978-7-121-29194-4



定价: 69.00元



博文视点Broadview



@博文视点Broadview



策划编辑:董英
责任编辑:徐津平
封面设计:侯士卿