数据结构与算法之美

- 第1章 复杂度分析 1
  - 1.1 复杂度分析（上）：如何分析/统计算法的执行效率和资源消耗 2
    - 1.1.1 复杂度分析 2
    - 1.1.2 大O复杂度表示法 4
    - 1.1.3 时间复杂度分析方法 6
    - 1.1.4 几种常见时间复杂度量级 6
    - 1.1.5 空间复杂度分析方法 7
    - 1.1.6 内容小结 7
    - 1.1.7 思考题 8
  - 1.2 最好时间复杂度/最坏时间复杂度/平均时间复杂度/均摊时间复杂度 8
    - 1.2.1 平均时间复杂度 9
    - 1.2.2 均摊时间复杂度 10
    - 1.2.3 内容小结 11
    - 1.2.4 思考题 12
  - 1.3 复杂度分析（下）：递归时间、循环、字符、空间复杂度时间复杂度 8
- 第2章 数组、链表、栈和队列 13
  - 2.1 数组（上）：为什么数组下标一般从0开始呢？ 14
    - 2.1.1 数组的定义 14
    - 2.1.2 寻址公式和随机访问特性 15
    - 2.1.3 低效的插入和删除操作 15
    - 2.1.4 警惕数组的边界问题 16
    - 2.1.5 容器能否完全替代数组？ 17
    - 2.1.6 解答关于数组的题 18
    - 2.1.7 内容小结 18
    - 2.1.8 思考题 19
  - 2.2 C/C++中数组的实现方式 19
    - 2.2.1 C/C++中数组的实现方式 19
    - 2.2.2 Java中数组的实现方式 20
    - 2.2.3 Javascript中数组的实现方式 21
    - 2.2.4 内容小结 21
    - 2.2.5 思考题 21
  - 2.3 链表（上）：为什么数据的下标一般从0开始？ 22
    - 2.3.1 链表的定义和特点 22
    - 2.3.2 链表的实现 23
    - 2.3.3 链表的优点和缺点 24
    - 2.3.4 链表与数组的性能对比 26
    - 2.3.5 解答关于字符题 28
    - 2.3.6 内容小结 29
    - 2.3.7 思考题 30
  - 2.4 链表（中）：如何写出一个正确的链表程序？ 30
    - 2.4.1 指针/引用的定义及其语义 30
    - 2.4.2 指针/引用在链表中的作用 30
    - 2.4.3 指针/引用的“常见操作” 31
    - 2.4.4 指针/引用：如何写出一个正确的链表程序？ 32
    - 2.4.5 指针/引用：哨兵简化编程 34
    - 2.4.6 指针/引用：边界条件、异常处理 34
    - 2.4.7 内容小结 34
    - 2.4.8 思考题 35
  - 2.5 链表（下）：如何判断一个字符串是否是回文？ 35
    - 2.5.1 回文的定义 35
    - 2.5.2 如何判断字符串是回文？ 35
    - 2.5.3 常见的字符串操作 36
    - 2.5.4 哨兵在字符串中的应用 37
    - 2.5.5 哨兵在链表中的应用 37
    - 2.5.6 解答关于字符题 38
    - 2.5.7 内容小结 39
    - 2.5.8 思考题 40
  - 2.6 栈：如何实现浏览器的前进和后退功能？ 40
    - 2.6.1 栈的定义 40
    - 2.6.2 栈的特点和操作 41
    - 2.6.3 栈的应用 41
    - 2.6.4 栈的实现 42
    - 2.6.5 解答关于字符题 44
    - 2.6.6 内容小结 45
    - 2.6.7 思考题 45
  - 2.7 队：为什么说排队是一种很有效率的做法？ 46
    - 2.7.1 什么是排队？ 47
    - 2.7.2 排队需要满足的一个条件 48
    - 2.7.3 如何编写排队程序？ 48
    - 2.7.4 编写排队程序的实现要点 49
    - 2.7.5 警惕排队程序的实现陷阱 49
    - 2.7.6 警惕排队程序的实现重复工作问题 50
    - 2.7.7 通过排队函数写出合格的排队程序 52
    - 2.7.8 解答关于字符题 52
    - 2.7.9 内容小结 53
    - 2.7.10 思考题 53
  - 2.8 循环队列：如何用一个数组来实现一个高效的队列？ 54
    - 2.8.1 什为什么要引入循环队列？ 54
    - 2.8.2 循环队列的实现 55
    - 2.8.3 循环队列的应用 56
    - 2.8.4 解答关于字符题 56
    - 2.8.5 内容小结 57
    - 2.8.6 思考题 57
  - 2.9 Java语言中的数组 57
    - 2.9.1 Java语言中的数组 57
    - 2.9.2 Java语言中的数组的特点 58
    - 2.9.3 Java语言中的数组的应用 58
    - 2.9.4 解答关于字符题 59
    - 2.9.5 内容小结 59
    - 2.9.6 思考题 59
  - 2.10 Linux内核中的链表 59
    - 2.10.1 Linux内核中的链表 59
    - 2.10.2 Linux内核中的链表的特点 60
    - 2.10.3 Linux内核中的链表的应用 60
    - 2.10.4 解答关于字符题 61
    - 2.10.5 内容小结 61
    - 2.10.6 思考题 62
- 第3章 递归、排序、二分查找 63
  - 3.1 递归：如何用10行代码找到“最终推荐人”？ 67
    - 3.1.1 递归的概念 67
    - 3.1.2 递归的三个条件 68
    - 3.1.3 如何编写递归代码 68
    - 3.1.4 编写递归代码的实践步骤 69
    - 3.1.5 警惕递归代码实现要点的陷阱 69
    - 3.1.6 警惕递归代码时间复杂度的陷阱 70
    - 3.1.7 通过递归函数写出合格的递归代码 72
    - 3.1.8 解答关于字符题 72
    - 3.1.9 内容小结 73
    - 3.1.10 思考题 73
  - 3.2 递归树：如何用递归树来求解递归公式时间复杂度？ 73
    - 3.2.1 什为什么要引入递归树？ 74
    - 3.2.2 递归树的实现 74
    - 3.2.3 递归树的应用 75
    - 3.2.4 解答关于字符题 75
    - 3.2.5 内容小结 76
    - 3.2.6 思考题 76
  - 3.3 冒泡排序：为什么冒泡排序是一个稳定的排序？ 76
    - 3.3.1 冒泡排序 76
    - 3.3.2 冒泡排序的实现 77
    - 3.3.3 冒泡排序的应用 77
    - 3.3.4 解答关于字符题 78
    - 3.3.5 内容小结 78
    - 3.3.6 思考题 78
  - 3.4 插入排序：为什么插入排序比冒泡排序更受欢迎？ 78
    - 3.4.1 插入排序的实现 79
    - 3.4.2 插入排序的应用 79
    - 3.4.3 解答关于字符题 80
    - 3.4.4 内容小结 80
    - 3.4.5 思考题 80
  - 3.5 选择排序：如何用递归来实现选择排序？ 81
    - 3.5.1 选择排序 81
    - 3.5.2 选择排序的实现 81
    - 3.5.3 选择排序的应用 82
    - 3.5.4 解答关于字符题 82
    - 3.5.5 内容小结 82
    - 3.5.6 思考题 82
  - 3.6 插入排序、冒泡排序、选择排序的时间复杂度是多少？ 83
    - 3.6.1 插入排序的时间复杂度 83
    - 3.6.2 冒泡排序的时间复杂度 83
    - 3.6.3 选择排序的时间复杂度 84
    - 3.6.4 解答关于字符题 84
    - 3.6.5 内容小结 84
    - 3.6.6 思考题 84
  - 3.7 如何用两个栈实现一个高效的队列？ 85
    - 3.7.1 两个栈实现一个队列的思路 85
    - 3.7.2 用两个栈实现一个队列的实现 86
    - 3.7.3 解答关于字符题 86
    - 3.7.4 内容小结 86
    - 3.7.5 思考题 86
  - 3.8 如何用递归和非递归两种方式来实现二分查找？ 87
    - 3.8.1 递归实现二分查找的实现 87
    - 3.8.2 递归实现二分查找的应用 88
    - 3.8.3 非递归实现二分查找的实现 88
    - 3.8.4 非递归实现二分查找的应用 89
    - 3.8.5 解答关于字符题 89
    - 3.8.6 内容小结 89
    - 3.8.7 思考题 89
  - 3.9 如何快速地在一个字符串中查找一个子串？ 90
    - 3.9.1 字符串匹配问题 90
    - 3.9.2 字符串匹配问题的实现 90
    - 3.9.3 字符串匹配问题的应用 91
    - 3.9.4 解答关于字符题 91
    - 3.9.5 内容小结 91
    - 3.9.6 思考题 91
  - 3.10 二分查找的变体：如何快速定位IP对应的城市？ 92
    - 3.10.1 IP对应城市的问题 92
    - 3.10.2 IP对应城市的实现 93
    - 3.10.3 IP对应城市的应用 93
    - 3.10.4 解答关于字符题 93
    - 3.10.5 内容小结 93
    - 3.10.6 思考题 93
- 第4章 哈希表、散列表和哈希算法 94
  - 4.1 哈希表（上）：Map和Set的数据结构是怎么实现的？ 98
    - 4.1.1 哈希表 98
    - 4.1.2 哈希函数 99
    - 4.1.3 哈希冲突 99
    - 4.1.4 解答关于字符题 100
    - 4.1.5 内容小结 100
    - 4.1.6 思考题 100
  - 4.2 哈希表（中）：如何打造一个工业级的哈希表？ 101
    - 4.2.1 哈希冲突的定义 101
    - 4.2.2 哈希冲突的实现 102
    - 4.2.3 哈希冲突的应用 102
    - 4.2.4 解答关于字符题 103
    - 4.2.5 内容小结 103
    - 4.2.6 思考题 103
  - 4.3 哈希表（下）：如何利用哈希表优化LRU缓存淘汰算法？ 104
    - 4.3.1 LRU缓存淘汰算法 104
    - 4.3.2 Java LinkedHashMap 104
    - 4.3.3 内容小结 105
    - 4.3.4 思考题 105
  - 4.4 散列表：基于散列表的最大优势？ 106
    - 4.4.1 基于散列表的最大优势 106
    - 4.4.2 基于位图的散列表 107
    - 4.4.3 基于布隆过滤器的最大优势 108
    - 4.4.4 内容小结 109
    - 4.4.5 思考题 109
  - 4.5 散列表冲突问题：如何高效地解决冲突？ 110
    - 4.5.1 散列表冲突的实现 110
    - 4.5.2 开放寻址法 111
    - 4.5.3 开放寻址法的应用 111
    - 4.5.4 解答关于字符题 112
    - 4.5.5 内容小结 112
    - 4.5.6 思考题 112
  - 4.6 散列表：如何设计一个高性能、低延迟的高效散列表？ 113
    - 4.6.1 散列表的选择 118
    - 4.6.2 散列函数的设计原则 119
    - 4.6.3 装载因子和动态扩容 120
    - 4.6.4 冲突解决的高效方法 121
    - 4.6.5 内容小结 122
    - 4.6.6 思考题 122
- 第5章 树 123
  - 5.1 树：为什么说数组和链表是最简单的树？ 127
    - 5.1.1 树的定义 128
    - 5.1.2 二叉树的定义 128
    - 5.1.3 二叉树的存储 129
    - 5.1.4 二叉树的遍历 129
    - 5.1.5 解答关于字符题 132
    - 5.1.6 内容小结 132
    - 5.1.7 思考题 132
  - 5.2 二叉搜索树：为什么说二叉搜索树是一种很有效率的数据结构？ 132
    - 5.2.1 二叉搜索树的定义及操作 133
    - 5.2.2 二叉搜索树的实现 135
    - 5.2.3 二叉搜索树的时间复杂度分析 135
    - 5.2.4 二叉搜索树的应用 137
    - 5.2.5 解答关于字符题 137
    - 5.2.6 内容小结 137
    - 5.2.7 思考题 138
  - 5.3 二叉搜索树：如何高效地实现一个搜索引擎的关键词提示？ 138
    - 5.3.1 Trie树的定义 139
    - 5.3.2 Trie树的实现 139
    - 5.3.3 Trie树的应用 140
    - 5.3.4 解答关于字符题 140
    - 5.3.5 内容小结 140
    - 5.3.6 思考题 140
  - 5.4 二叉搜索树：如何实现一个高性能、稳定的动态数据结构？ 141
    - 5.4.1 二叉搜索树的实现 142
    - 5.4.2 二叉搜索树的应用 142
    - 5.4.3 解答关于字符题 143
    - 5.4.4 内容小结 143
    - 5.4.5 思考题 143
  - 5.5 B-树：MySQL是如何高效索引数据的？ 143
    - 5.5.1 B-树的定义 144
    - 5.5.2 B-树的实现 145
    - 5.5.3 B-树的应用 146
    - 5.5.4 解答关于字符题 146
    - 5.5.5 内容小结 146
    - 5.5.6 思考题 146
  - 5.6 B+树：为什么说数据库索引选择B+树？ 147
    - 5.6.1 B+树的定义 147
    - 5.6.2 B+树的实现 148
    - 5.6.3 B+树的应用 148
    - 5.6.4 解答关于字符题 149
    - 5.6.5 内容小结 149
    - 5.6.6 思考题 149
  - 5.7 递归：递归是一种很高效的编程技巧吗？ 149
    - 5.7.1 递归的概念 150
    - 5.7.2 递归的应用 150
    - 5.7.3 递归的实现 150
    - 5.7.4 解答关于字符题 151
    - 5.7.5 内容小结 151
    - 5.7.6 思考题 151
  - 5.8 递归：递归的时间复杂度怎么分析？ 151
    - 5.8.1 递归时间复杂度的分析 152
    - 5.8.2 递归时间复杂度的应用 152
    - 5.8.3 递归时间复杂度的实现 152
    - 5.8.4 解答关于字符题 153
    - 5.8.5 内容小结 153
    - 5.8.6 思考题 153
  - 5.9 递归：递归的时间复杂度怎么分析？ 153
    - 5.9.1 递归的时间复杂度的分析 154
    - 5.9.2 递归的时间复杂度的应用 154
    - 5.9.3 递归的时间复杂度的实现 154
    - 5.9.4 解答关于字符题 155
    - 5.9.5 内容小结 155
    - 5.9.6 思考题 155
  - 5.10 递归：递归的时间复杂度怎么分析？ 155
    - 5.10.1 递归的时间复杂度的分析 156
    - 5.10.2 递归的时间复杂度的应用 156
    - 5.10.3 递归的时间复杂度的实现 156
    - 5.10.4 解答关于字符题 157
    - 5.10.5 内容小结 157
    - 5.10.6 思考题 157
- 第6章 堆 158
  - 6.1 堆：为什么说堆是一种不错的数据结构？ 162
    - 6.1.1 堆的定义 162
    - 6.1.2 如何实现一个堆？ 163
    - 6.1.3 如何实现一个高效的堆？ 163
    - 6.1.4 解答关于字符题 164
    - 6.1.5 内容小结 164
    - 6.1.6 思考题 165
  - 6.2 堆排序：如何高效地实现一个排序算法？ 165
    - 6.2.1 什么是堆排序？ 165
    - 6.2.2 堆排序的实现 166
    - 6.2.3 堆排序的应用 166
    - 6.2.4 解答关于字符题 167
    - 6.2.5 内容小结 167
    - 6.2.6 思考题 167
  - 6.3 堆的应用：如何快速求解一个数据集合的第K大元素？ 168
    - 6.3.1 什么是优先级队列？ 168
    - 6.3.2 优先级队列的实现 169
    - 6.3.3 解答关于字符题 169
    - 6.3.4 内容小结 169
    - 6.3.5 思考题 169
  - 6.4 堆的应用：如何快速求解一个数据集合的中位数？ 170
    - 6.4.1 求中位数问题 170
    - 6.4.2 求中位数问题的实现 171
    - 6.4.3 求中位数问题的应用 171
    - 6.4.4 解答关于字符题 172
    - 6.4.5 内容小结 172
    - 6.4.6 思考题 172
  - 6.5 堆的应用：如何实现一个高性能、稳定的定时任务？ 173
    - 6.5.1 定时任务的实现 173
    - 6.5.2 定时任务的应用 174
    - 6.5.3 解答关于字符题 174
    - 6.5.4 内容小结 174
    - 6.5.5 思考题 174
  - 6.6 堆的应用：如何实现一个高性能、稳定的流量控制？ 174
    - 6.6.1 流量控制的概念 175
    - 6.6.2 流量控制的实现 175
    - 6.6.3 解答关于字符题 176
    - 6.6.4 内容小结 176
    - 6.6.5 思考题 176
- 第7章 排序、外排序、线性时间复杂度 177
  - 7.1 排序：Redis中的有序集合是怎麼如何实现的？ 178
    - 7.1.1 排序的场景 178
    - 7.1.2 冒泡、插入、选择三种时间复杂度分析 179
    - 7.1.3 排序基于稳定性的考虑 180
    - 7.1.4 高级排序算法的思路 181
    - 7.1.5 排序算法的实现要点 182
    - 7.1.6 解答关于字符题 182
    - 7.1.7 内容小结 182
    - 7.1.8 思考题 183
  - 7.2 外排序：如何对一个大小为10G的文件进行排序？ 183
    - 7.2.1 外排序的应用 183
    - 7.2.2 基于单机实现的思路 184
    - 7.2.3 基于多机实现的思路 186
    - 7.2.4 内容小结 188
    - 7.2.5 思考题 188
  - 7.3 线性时间复杂度排序中的计数排序和基数排序 188
    - 7.3.1 计数排序的实现 189
    - 7.3.2 计数排序的应用 189
    - 7.3.3 基数排序的实现 190
    - 7.3.4 基数排序的应用 190
    - 7.3.5 解答关于字符题 191
    - 7.3.6 内容小结 191
    - 7.3.7 思考题 191
  - 7.4 “时间换空间”：如何利用有限内存来排序？ 192
    - 7.4.1 “时间换空间”的实现 192
    - 7.4.2 “时间换空间”的应用 193
    - 7.4.3 解答关于字符题 193
    - 7.4.4 内容小结 193
    - 7.4.5 思考题 193
- 第8章 字符串匹配算法 179
  - 8.1 朴素匹配：如何用10行代码实现一个简单的字符串匹配算法？ 180
    - 8.1.1 朴素匹配的思路与实现 180
    - 8.1.2 朴素匹配的复杂度分析 181
    - 8.1.3 解答关于字符题 181
    - 8.1.4 内容小结 182
    - 8.1.5 思考题 182
  - 8.2 BM算法：如何用一个数组来解决坏字符问题？ 182
    - 8.2.1 BM算法的核心思想 183
    - 8.2.2 BM算法的思路分析 184
    - 8.2.3 BM算法的实现 184
    - 8.2.4 解答关于字符题 185
    - 8.2.5 内容小结 185
    - 8.2.6 思考题 185
  - 8.3 BM算法：如何实现更为高效的好后缀规则？ 186
    - 8.3.1 好后缀规则的实现 186
    - 8.3.2 好后缀规则的应用 187
    - 8.3.3 解答关于字符题 187
    - 8.3.4 内容小结 187
    - 8.3.5 思考题 188
  - 8.4 KMP算法：如何理解KMP算法的核心思想？ 188
    - 8.4.1 KMP算法的核心思想 189
    - 8.4.2 KMP算法的思路分析 189
    - 8.4.3 KMP算法的实现 190
    - 8.4.4 解答关于字符题 190
    - 8.4.5 内容小结 190
    - 8.4.6 思考题 190
  - 8.5 Trie树：如何实现敏感词过滤功能？ 190
    - 8.5.1 Trie树的定义 191
    - 8.5.2 Trie树的实现思路 191
    - 8.5.3 Trie树的应用分析 192
    - 8.5.4 解答关于字符题 192
    - 8.5.5 内容小结 192
    - 8.5.6 思考题 192
  - 8.6 AC自动机：如何将单模式串匹配扩展到多模式串？ 193
    - 8.6.1 AC自动机的应用场景 193
    - 8.6.2 AC自动机的思路分析 194
    - 8.6.3 解答关于字符题 194
    - 8.6.4 内容小结 194
    - 8.6.5 思考题 195
- 第9章 图 196
  - 9.1 图的定义 197
    - 9.1.1 图的定义 197
    - 9.1.2 图的表示 198
    - 9.1.3 图的相关概念 198
    - 9.1.4 解答关于字符题 199
    - 9.1.5 内容小结 199
    - 9.1.6 思考题 199
  - 9.2 深度优先搜索和广度优先搜索：如何找出社交网络中的三度好友关系？ 199
    - 9.2.1 什么是深度优先搜索？ 200
    - 9.2.2 广度优先搜索 200
    - 9.2.3 深度优先搜索 200
    - 9.2.4 广度优先搜索 200
    - 9.2.5 解答关于字符题 201
    - 9.2.6 内容小结 201
    - 9.2.7 思考题 201
  - 9.3 拓扑排序：如何确定代码源文件的编译依赖关系？ 202
    - 9.3.1 什么是拓扑排序？ 202
    - 9.3.2 拓扑排序的实现 203
    - 9.3.3 拓扑排序的应用 203
    - 9.3.4 解答关于字符题 203
    - 9.3.5 内容小结 204
    - 9.3.6 思考题 204
  - 9.4 最短路径算法：如何利用地图软件计算最优出行路线？ 204
    - 9.4.1 什么是单源最短路径？ 205
    - 9.4.2 Dijkstra算法的思路分析 205
    - 9.4.3 Dijkstra算法的实现 206
    - 9.4.4 解答关于字符题 206
    - 9.4.5 内容小结 206
    - 9.4.6 思考题 207
  - 9.5 多源最短路径：如何利用Floyd算法解决多对多问题？ 207
    - 9.5.1 什么是多对多问题？ 208
    - 9.5.2 Floyd算法的思路分析 208
    - 9.5.3 利用Floyd算法求解所有顶点 209
    - 9.5.4 内容小结 209
    - 9.5.5 思考题 209
  - 9.6 欧拉回路：如何用一笔画画出一个图？ 209
    - 9.6.1 什么是欧拉回路？ 210
    - 9.6.2 一笔画问题的解决思路 210
    - 9.6.3 解答关于字符题 211
    - 9.6.4 内容小结 211
    - 9.6.5 思考题 211
  - 9.7 最小生成树：如何搭设光缆网络可以发挥作用？ 212
    - 9.7.1 什么是最小生成树？ 212
    - 9.7.2 Prim算法的思路分析 212
    - 9.7.3 Prim算法的实现 213
    - 9.7.4 解答关于字符题 213
    - 9.7.5 内容小结 213
    - 9.7.6 思考题 213
  - 9.8 最大流：如何求解一个图中的最大流？ 213
    - 9.8.1 什为什么要引入最大流？ 214
    - 9.8.2 Ford-Fulkerson方法 214
    - 9.8.3 Edmonds-Karp算法 215
    - 9.8.4 最大二分匹配 216
    - 9.8.5 解答关于字符题 216
    - 9.8.6 内容小结 216
    - 9.8.7 思考题 216
- 第10章 贪心、分治、回溯和动态规划 251
  - 10.1 贪心算法：如何利用贪心算法实现最大最短时间？ 252
    - 10.1.1 如何理解贪心算法？ 252
    - 10.1.2 贪心算法的应用分析 253
    - 10.1.3 解答关于字符题 253
    - 10.1.4 内容小结 253
    - 10.1.5 思考题 253
  - 10.2 分治算法：这一定大规模计算框架MapReduce的分治思想？ 253
    - 10.2.1 分治算法的应用分析 257
    - 10.2.2 分治算法的实现思路 257
    - 10.2.3 分治算法在大数据计算中的应用 259
    - 10.2.4 解答关于字符题 259
    - 10.2.5 内容小结 260
    - 10.2.6 思考题 260
  - 10.3 回溯算法：从电影《蝴蝶效应》中学习回溯算法的核心思想？ 260
    - 10.3.1 如何理解回溯算法？ 261
    - 10.3.2 八皇后问题 261
    - 10.3.3 0-1背包问题 262
    - 10.3.4 正则表达式匹配问题 263
    - 10.3.5 内容小结 264
    - 10.3.6 思考题 264
  - 10.4 初识动态规划：如何巧妙解决“双11”期间的资源调度？ 264
    - 10.4.1 动态规划的“前世今生” 265
    - 10.4.2 利用动态规划解决0-1背包问题 265
    - 10.4.3 0-1背包问题的思路分析 266
    - 10.4.4 解答关于字符题 270
    - 10.4.5 内容小结 270
    - 10.4.6 思考题 271
  - 10.5 动态规划理论：如何巧妙解决“最长公共子串”问题？ 271
    - 10.5.1 “一个模型对二个特征”的应用 271
    - 10.5.2 “一个模型对二个特征”的实现思路 272
    - 10.5.3 动态规划的两种解题思路 274
    - 10.5.4 a种硬币凑整问题的分析 277
    - 10.5.5 内容小结 277
    - 10.5.6 思考题 278
  - 10.6 动态规划实战：如何实现搜索引擎中的拼写纠错功能？ 278
    - 10.6.1 如何通过编辑距离来实现拼写纠错 279
    - 10.6.2 如何通过编辑距离来实现搜索引擎中的拼写纠错？ 279
    - 10.6.3 解答关于字符题 280
    - 10.6.4 内容小结 280
    - 10.6.5 思考题 281
- 第11章 数据结构和算法实战 282
  - 11.1 实战：剖析Redis底层数据类型对应的数据结构 283
    - 11.1.1 Redis数据类型介绍 283
    - 11.1.2 字符（SDS） 283
    - 11.1.3 动态（Dict） 286
    - 11.1.4 集合（set） 288
    - 11.1.5 有序集合（sorted set） 287
    - 11.1.6 数据结构的内存占用情况 287
    - 11.1.7 总结回答问题 288
    - 11.1.8 思考题 288
  - 11.2 实战：如何理解不同数据结构最底层的实现方式？ 288
    - 11.2.1 数组 289
    - 11.2.2 动态数组 290
    - 11.2.3 哈希 290
    - 11.2.4 栈 291
    - 11.2.5 队列 292
    - 11.2.6 哈希表 292
    - 11.2.7 动态哈希表 293
    - 11.2.8 渐进式哈希 293
    - 11.2.9 跳表 294
    - 11.2.10 思考题 294
  - 11.3 实战：如何理解散列表与红黑树在现代语言中的应用？ 295
    - 11.3.1 散列表的应用 296
    - 11.3.2 如何设计一个高效的散列表？ 296
    - 11.3.3 如何实现一个高性能、稳定的动态数据结构？ 297
    - 11.3.4 如何实现一个高性能、稳定的动态数据结构？ 297
    - 11.3.5 总结回答问题 298
    - 11.3.6 思考题 298
  - 11.4 实战：如何理解不同数据结构最底层的实现方式？ 298
    - 11.4.1 数组 299
    - 11.4.2 动态数组 299
    - 11.4.3 哈希 299
    - 11.4.4 栈 300
    - 11.4.5 队列 300
    - 11.4.6 哈希表 301
    - 11.4.7 动态哈希表 301
    - 11.4.8 渐进式哈希 301
    - 11.4.9 跳表 302
    - 11.4.10 思考题 302
  - 11.5 实战：如何理解散列表与红黑树在现代语言中的应用？ 302
    - 11.5.1 散列表的应用 303
    - 11.5.2 如何设计一个高效的散列表？ 303
    - 11.5.3 如何实现一个高性能、稳定的动态数据结构？ 304
    - 11.5.4 如何实现一个高性能、稳定的动态数据结构？ 304
    - 11.5.5 总结回答问题 305
    - 11.5.6 思考题 305
  - 11.6 实战：如何理解不同数据结构最底层的实现方式？ 305
    - 11.6.1 数组 306
    - 11.6.2 动态数组 306
    - 11.6.3 哈希 306
    - 11.6.4 栈 307
    - 11.6.5 队列 307
    - 11.6.6 哈希表 308
    - 11.6.7 动态哈希表 308
    - 11.6.8 渐进式哈希 308
    - 11.6.9 跳表 309
    - 11.6.10 思考题 309
  - 11.7 实战：如何设计实现数据结构算法实现时比较用？ 309
    - 11.7.1 总结回答问题 310
    - 11.7.2 思考题 310
- 附录A 思考题答案 304