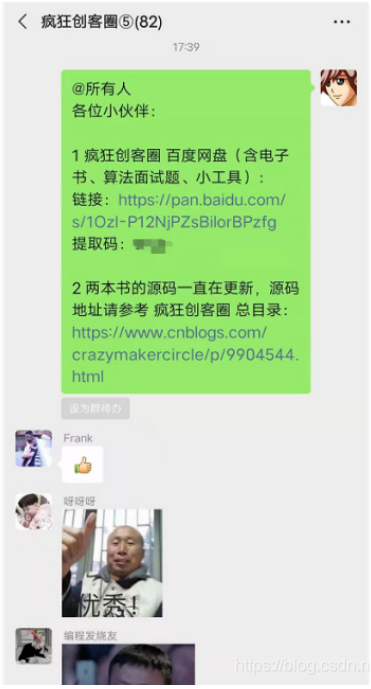
史上最全 Java 面试题：算法篇（入群获取资料）



刷算法的秘籍

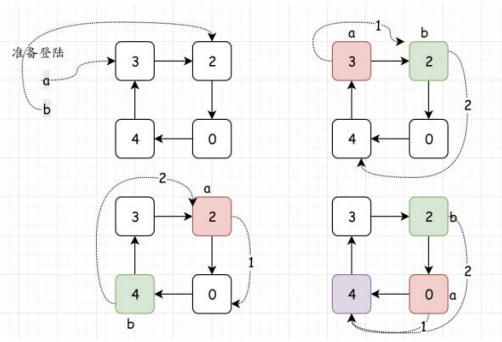
如何刷算法，这块有秘籍， 具体找40岁老架构尼恩，微信交流就可以了

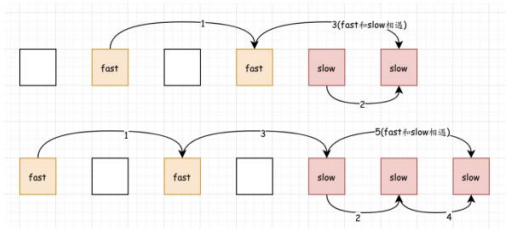
算法刷题宝典

刷题任务的题目，是根据题目的类型来汇总的，总结了八个类别，每个类别下面也总结了5个左右的题型，帮助大家分门别类的突破，所以刷起来相对会更有重点和针对性。如果从头到尾的刷，每周按顺序刷42题，很容易让自己坚持不下来，也会觉得很枯燥。所以在制定计划的时候可以让这个计划变得更“有趣"和针对性，让它看起来更容易实现一点，才会更容易坚持。

- 数组系列
  - 两个数组的交集(350)
  - 最长公共前缀(14)
  - 买卖股票的最佳时机(122)
  - 旋转数组(189)
  - 原地删除(27)
  - 加一(66)
  - 两数之和(1)
- 链表系列
  - 删除链表倒数第N个节点(19)
  - 合并两个有序链表(21)
  - 环形链表(21)
- 动态规划系列
  - 爬楼梯(70)
  - 最大子序和(53)
  - 最长上升子序列(300)
  - 三角形最小路径和(120)
  - 最小路径和(64)
  - 打家劫舍(198)
- 字符串系列
  - 反转字符串(301)
  - 字符串中的第一个唯一字符(387)
- 二叉树系列
  - 最大深度与DFS(104)
  - 层次遍历与BFS(102)
  - BST与其验证(98)
  - BST 的查找(700)
  - BST 的删除(450)
- 滑动窗口系列
  - 滑动窗口最大值（239）
  - 无重复字符的最长子串（3）
  - 找到字符串中所有字母异位词（438）
- 博弈论系列
  - 囚徒困境
  - 辛普森悖论
  - 红眼睛和蓝眼睛
  - 海盗分金币
  - 排序类题目
  - 按奇偶排序数组（905）
- 位运算系列
  - 使用位运算求和
  - 2的幂(231)
  - 返回一个数二进制中1的个数(191)
  - 只出现一次的数字(136)
  - 只出现一次的数字Ⅱ(137)
  - 缺失数字(268)
- 二分法系列
  - 爱吃香蕉的珂珂（875）
  - x的平方根（69）
  - 第一个错误的版本（287）
- 其他题目
  - 螺旋矩阵（54）
  - 只有两个键的键盘（650）
  - 24点游戏（679）
  - 飞机座位分配概率（1227）
  - 水分子的产生
  - 救生艇（881）
  - 救生艇（881）
  - 灯泡开关（319）
  - 三门问题
  - 猜数字游戏（299）
  - LRU缓存机制（146）
  - 最小的k个数
  - 不同路径
  - 不同路径 - 障碍物
  - 伪装特卡洛
  - 盛最多水的容器
  - 扑克牌中的顺子容器
  - 整数拆分（343）
  - 移动石子直到连续（1033）
  - Nim 游戏（292）

部分内容展示：





所以我们的快指针的步长可以设置为 2。

在上一篇中，我们通过题目"**最长上升子序列**"以及"**最大子序和**"，学习了DP（动态规划）在线性关系中的分析方法。这种分析方法，也在运筹学中被称为"线性动态规划"，具体指的是"目标函数为特定变量的线性函数，约束是这些变量的线性不等式或等式，目的是求目标函数的最大值或最小值"。这点大家作为了解即可，不需要死记，更不要生搬硬套！

在本节中，我们将继续分析一道略微区别于之前的题型，希望可以由此题与之前的题目进行对比论证，进而顺利求解！

01、题目分析

第120题：三角形最小路径和

给定一个三角形，找出自顶向下的最小路径和。每一步只能移动到下一行中相邻的结点上。

例如，给定三角形：

```
[
     [2],
    [3,4],
   [6,5,7],
  [4,1,8,3]
]
```

则自顶向下的最小路径和为 11（即，2 + 3 + 5 + 1 = 11）。

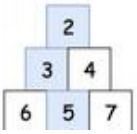
这道题有一定难度哦！如果没有思路请回顾上一篇的学习内容！

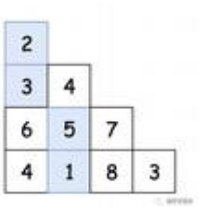
不建议直接看题解！

02、题目图解

首先我们分析题目，要找的是**三角形最小路径和**。这是个啥意思呢？假设我们有一个三角形：[[2],[3,4],[6,5,7],

[4,1,8,3]]



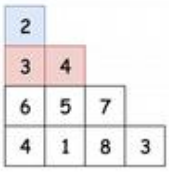


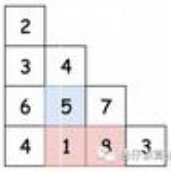
这样相当于我们将整个三角形进行了拉伸，这时候，我们根据题目中给出的条件：每一步只能移动到下一行中相邻

的结点上。其实也就等同于，**每一步我们只能往下移动一格或者右下移动一格。**将其转化成代码，假如2所在的元

素位置为(0,0)，那我们往下移动就只能移动到(1,0)或者(1,1)的位置上。假如5所在的位置为(2,1)，同样也只能移动

到(3,1)和(3,2)的位置上。如下图所示：





题目明确了之后，现在我们开始进行分析，题目很明显是**一个找最优解的问题，并且可以从子问题的最优解进**

行构建。所以我们通过动态规划进行求解。首先，我们定义状态：

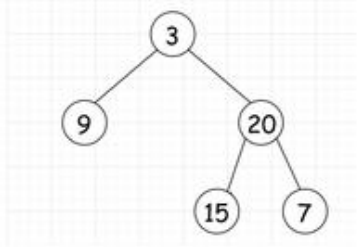
dp[i][j]：表示包含第i行j列元素的最小路径和

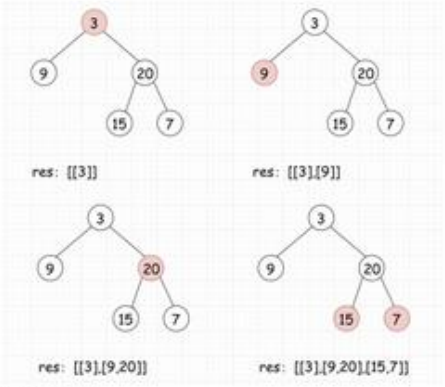
我们很容易想到可以自顶向下进行分析，并且，无论最后的路径是哪一条，它一定要经过最顶上的元素，即(0,0)。所以我们需要对 dp[0][0] 进行初始化。

03、递归求解

同样，我们先考虑本题的递归解法。想到递归，我们一般先想到DFS，我们可以对该二叉树进行**先序遍历（根左右的顺序）**，同时，记录节点所在的层次level，并且对每一层都定义一个数组，然后将访问到的节点值放入对应层的数组中。

假设给定二叉树为[3,9,20,null,null,15,7]，图解如下：





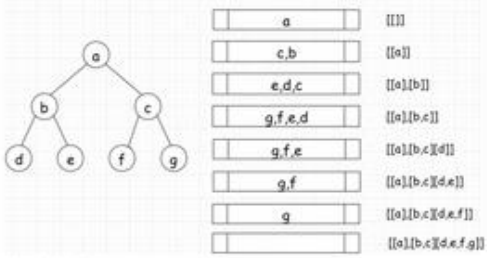
根据以上分析，代码如下：

```
func dfs(root *TreeNode, level int, res [][]int) [][]int {
    if root == nil {
        return res
    }
    if len(res) == level {
        res = append(res, []int{root.Val})
    } else {
        res[level] = append(res[level], root.Val)
    }
    res = dfs(root.Left, level+1, res)
    res = dfs(root.Right, level+1, res)
    return res
}
```

04、BFS求解

上面的解法，其实相当于是用DFS的方法实现了二叉树的BFS。那我们能不能直接使用BFS的方式进行解题呢？当然，我们可以使用Queue的数据结构。我们将root节点初始化进队列，通过**消耗尾部，插入头部**的方式来完成BFS。

具体步骤如下图：



根据以上分析，代码如下：

```
func levelOrder(root *TreeNode) [][]int {
    var result [][]int
    if root == nil {
        return result
    }
    // 定义一个切片队列
    queue := list.New()
```

（小浩os：我觉得自己画的这个图对于双端队列的解法还是介绍的比较清晰的，大家对好看一下哦，这样我的努力也就没有白费啦。）

根据分析，得出代码：

```go
func maxSlidingWindow(nums []int, k int) []int {
    if len(nums) == 0 {
        return []int{}
    }
    //用切片模拟一个双端队列
    queue := []int{}
    result := []int{}
    for i := range nums {
        for i > 0 && (len(queue) > 0) && nums[i] > queue[len(queue)-1] {
            //将比当前元素小的元素弹出
            queue = queue[:len(queue)-1]
        }
        //将当前元素放入queue中
        queue = append(queue, nums[i])
        if i >= k && nums[i-k] == queue[0] {
            //维护队列，保证其头元素为当前窗口最大值
            queue = queue[1:]
        }
        if i >= k-1 {
            //放入结果数组
            result = append(result, queue[0])
        }
    }
    return result
}
```

执行用时：20 ms，在所有 Go 提交中击败了 98.41 的用户

内存消耗：6.3 MB，在所有 Go 提交中击败了 88.18% 的用户

Perfect题目完成！看看一下子超越百分之99的用户，是不是感觉很爽呢

无重复字符的最长子串（3）

在上一节中，我们使用双端队列完成了滑动窗口的一道较为困难的题目，以此展示了什么是滑动窗口。在本节中我们将继续深入分析，探索滑动窗口题型一些具有模式性的解法。

01、滑动窗口介绍

对于大部分滑动窗口类型的题目，一般是**考察字符串的匹配**，比较标准的题目，会给出一个**模式串m**，以及一个**目标串A**，然后提出问题，找到**A中符合对m一些限定规则的子串**或者**对A一些限定规则的结果**，部件**再将搜索出的子串完成题目中要求的组合或者其他**。

比如：给定一个字符串 s 和一个非空字符串 p，找到 s 中所有是 p 的字母异位词的子串，返回这些子串的起始索引。

又或者：给你一个字符串 S、一个字符串 T，请在字符串 S 里面找出：包含 T 所有字母的最小子串。

再如：给定一个字符串 s 和一些长度相同的单词 words。找出 s 中恰好可以由 words 中所有单词串联形成的子串的起始位置。

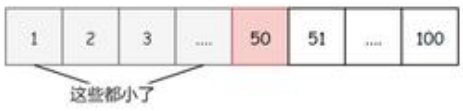
都是属于这一类的标准题型，而对于这一类题目，我们常用的解题思路，**是去维护一个可变长度的滑动窗口**。无论是使用**双指针**，还是使用**双端队列**，又或者用**游标**等其他奇技淫巧。目的都是一样的。

Now，我们通过一道题目来进行具体学习吧

02、题目分析

在最简单的形式中，二分查找对具有指定左索引和右索引的**连续序列**进行操作。我们也称之为**查找空间**。二分查找维护查找空间的左，右和中间指示符，并比较查找目标：如果条件不满足或值不相等，则清除目标不可能存在的那一半，并在剩下的一半上继续查找，直到成功为止。

举例说明：比如你需要找1-100中的一个数字，你的目标是**用最少的次数**猜到这个数字。你每次猜测后，我会说大了或者小了。而你只需要每次猜测中间的数字，就可以将余下的数字排除一半。



不管我心里想的数字如何，你在7次之内都能猜到，这就是一个典型的二分查找。每次筛选掉一半数据，所以我们也称之为**折半查找**。一般而言，对于包含n个元素的列表，用二分查找最多需要log2n步。

当然，一般题目不太可能给你一个如此规成的题型，让你上手就可以使用二分，所以我们需要思考，如何来构造一个成功的二分查找。大部分的二分查找，基本都由以下三步组成：

- 预处理过程（大部分场景就是对未排序的集合进行排序）
- 二分查找过程（找到合适的循环条件，每一次将查找空间一分为二）
- 后处理过程（在剩余的空间中，找到合适的目标值）

了解了二分查找的过程，我们对二分查找进行**一般实现**（这里给出一个java版本，比较正宗的代码，没

```go
        int mid = low + (high - low) / 2;
        if (des == array[mid]) {
            return mid;
        } else if (des < array[mid]) {
            high = mid - 1;
        } else {
            low = mid + 1;
        }
    }
    return -1;
}
```

注意：上面的代码，mid 使用 low + (high - low)/2 的目的，是防止 high low 溢出内存。

为什么说是一般实现？

1、**根据边界的不同（开闭区间调整），有时需要弹性调整low与high的值，以及循环的终止条件。**

2、根据元素是否有重复值，以及是否需要找到重复值区间，有时需要对原算法进行改造。

那上面我们说了，一般二分查找的过程分为：预处理 - 二分查找 - 后处理。上面的代码，就没有后处理的过程。因为在每一步中，你都检查了元素，如果到达末尾，也已经知道没有找到元素。

总结一下一般实现的几个条件：

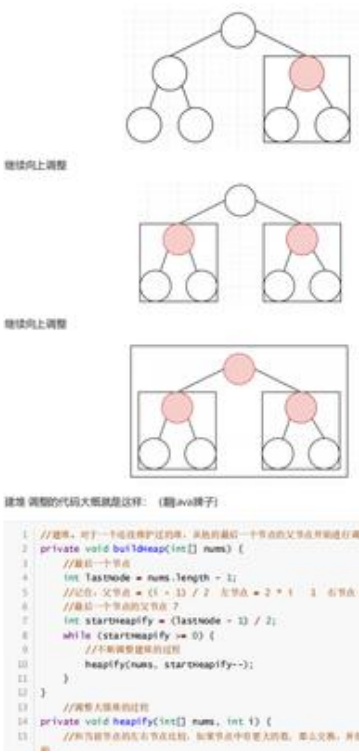
- **初始条件**：left = 0, right = length-1
- **终止**：left > right
- **向左查找**：right = mid-1
- **向右查找**：left = mid+1

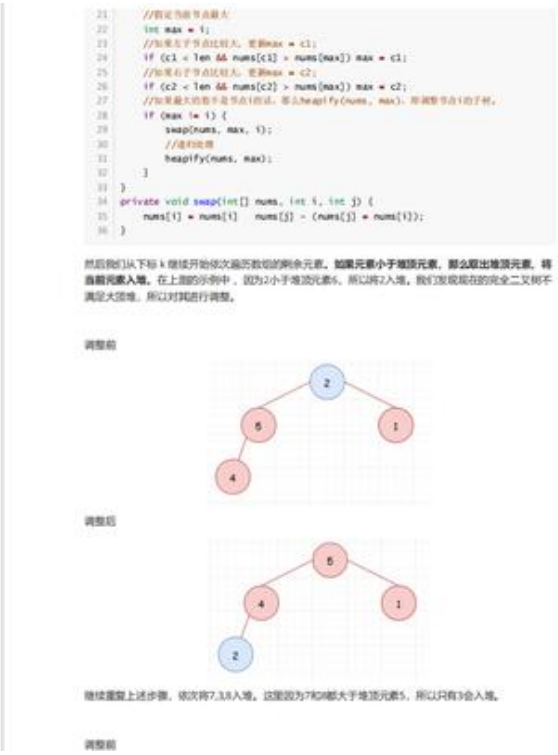
请大家记住这个模板原则，在后面的系列中，我们将介绍二分查找其他的模板类型。

03、题目分析

简单复习了二分查找，我们来看本题。

注意，绝大部分**「在连续递减区间中搜索目标值」**的问题，都可以转化为二分查找问题，并且，二分查找的题目，基本逃不出三种：找特定值，找大于特定值的元素（上界），找小于特定值的元素（下

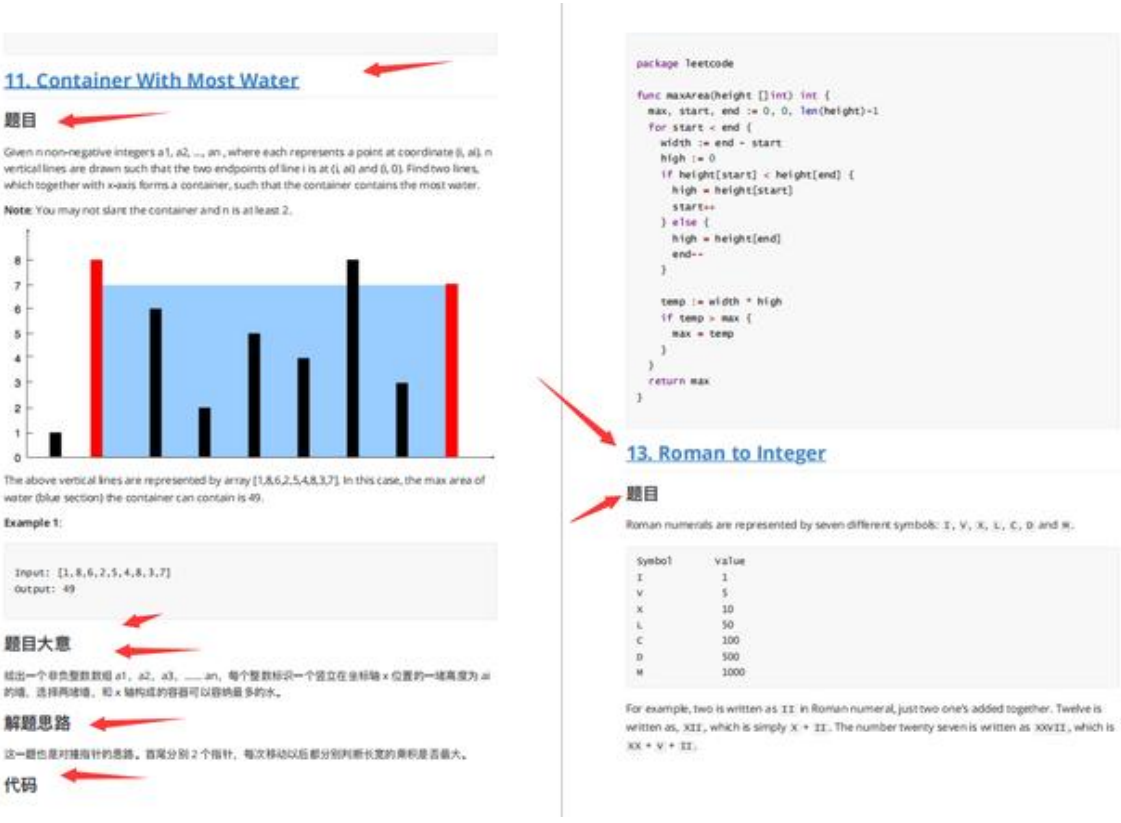


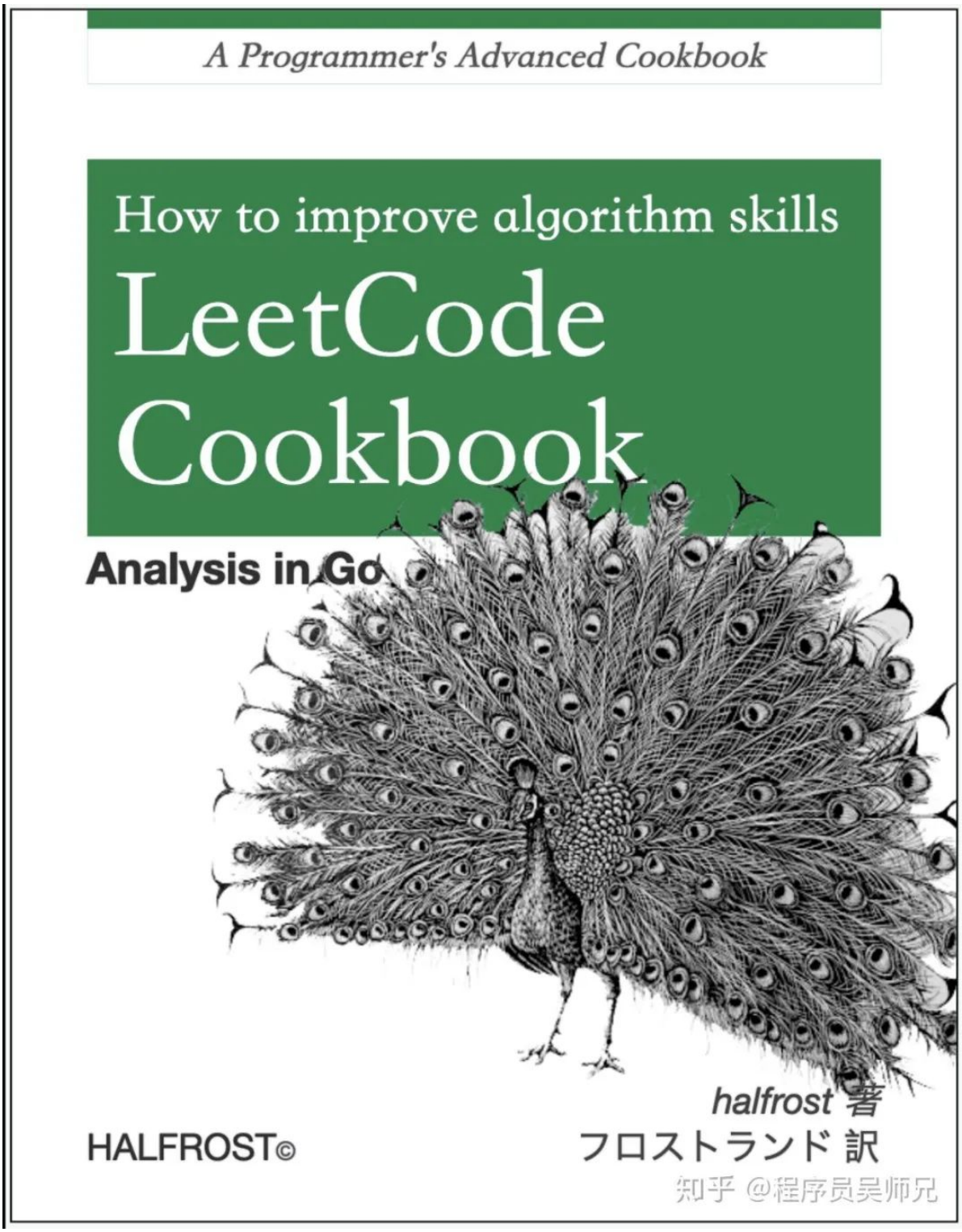


LeetCode（520 道题）

除此之外，这里再跟大家推荐一本前不久火爆 GitHub 的 LeetCode 中文刷题手册，**这本小册里面共包含刷 LeetCode 后整理的 520 道题，每道题均附有详细题解过程。自发布以后，受到技术圈内广大开发者的赞赏，建议大家收藏阅读。**目录如下：

部分目录展示：





力扣 Cookbook是@halfrost（中文名：**霜神**）去年刷的 **力扣**整理出的 520 题，每道题都写了解题思路，并且每题都 runtime beats 100% 了。

至于为什么要求每题都 runtime beats 100%?

霜神是这样回复的：优化到 beats 100% 才算是把这题做出感觉了。有好几道 Hard 题，可以用暴力解法 AC 了，但只 beats 了 5%，这题就如同没做一样；

而且面试中如果给了暴力的答案，面试官也不会满意，通过自己的思考给出更优解，面试官也会更满意一些。

所以如果你把这些题解都摸透，相信在面试环节你可以从容的回答“还有没有更优解”这个问题。

现在就把这本电子书分享给大家，希望能帮助大家克服刷题的恐惧，顺利拿到大厂offer。

Linux 归纳笔记

此外这里还有一份华为大牛总结的 Linux 归纳笔记，一并分享给大家。

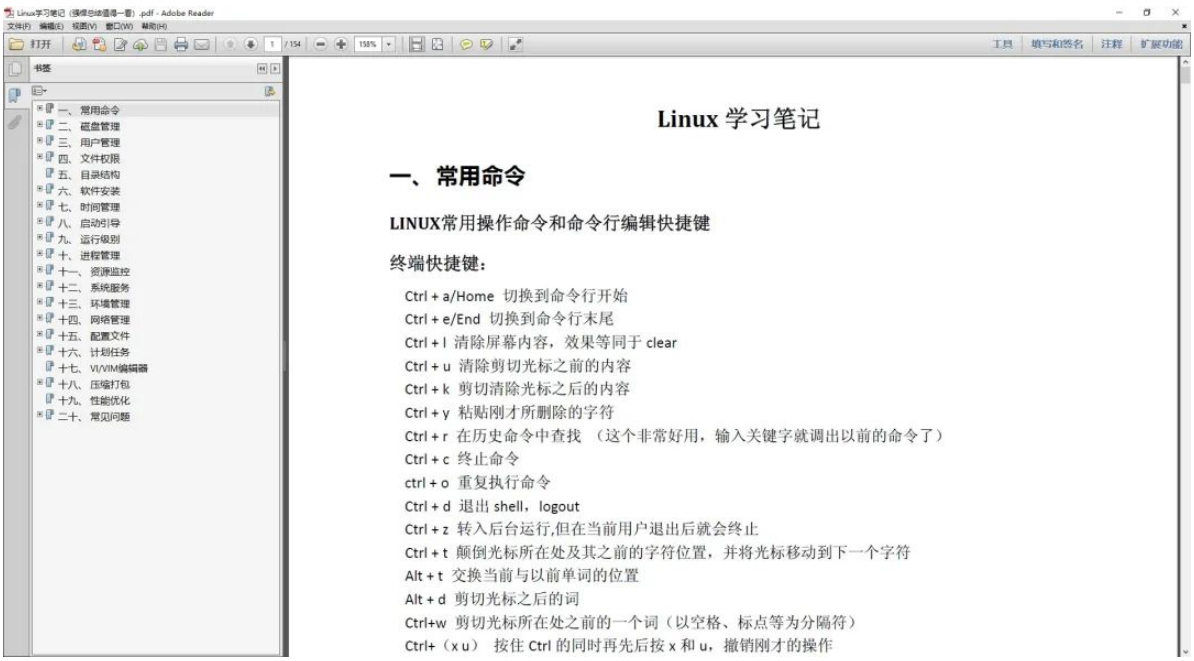
这份资料非常全面且详细，从 **Linux 常用命令**到 **Linux 常用操作**，再到**网络管理**、**性能优化**，几乎覆盖了 Linux 基础学习的方方面面，非常适合初学者入门！

资料也按目录进行编排，每一章下面都有更具体的内容：

非常详细的目录

每章都有更细分的内容

而且，这份资料不是扫描版的，里面的文字都可以直接复制，非常便于我们学习：



获取方式

通过疯狂创客圈网盘，统一获取，进群后请参见群公告， 加尼恩微信即可，具体的微信二维码，请参见

[博客园总入口]

另外附带n本电子书。