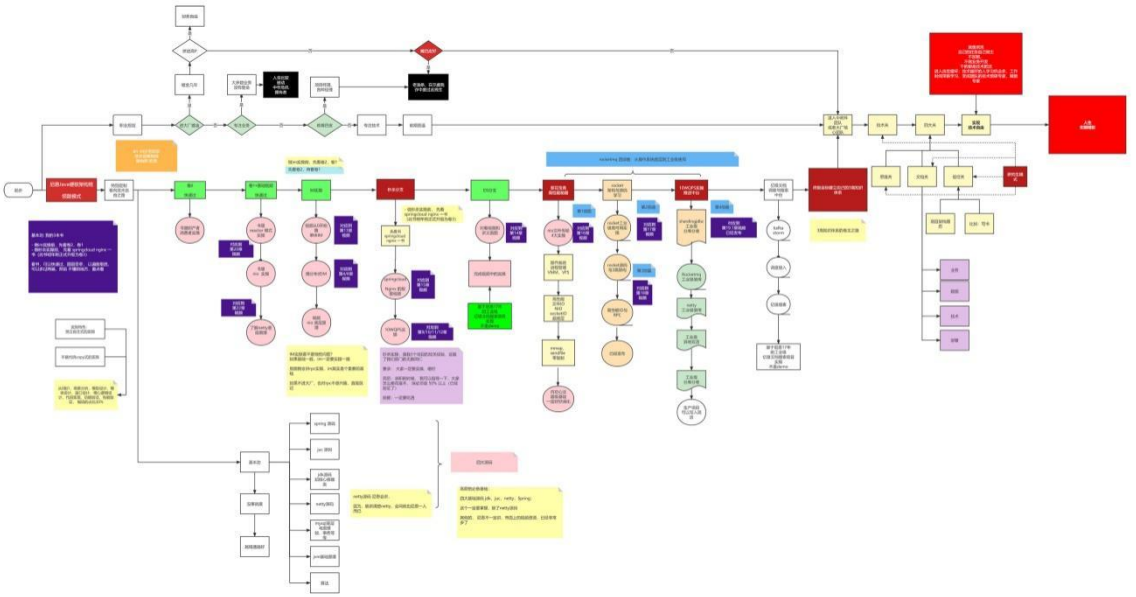
牛逼的职业发展之路

40 岁老架构尼恩用一张图揭秘：Java 工程师的高端职业发展路径，走向食物链顶端的之路

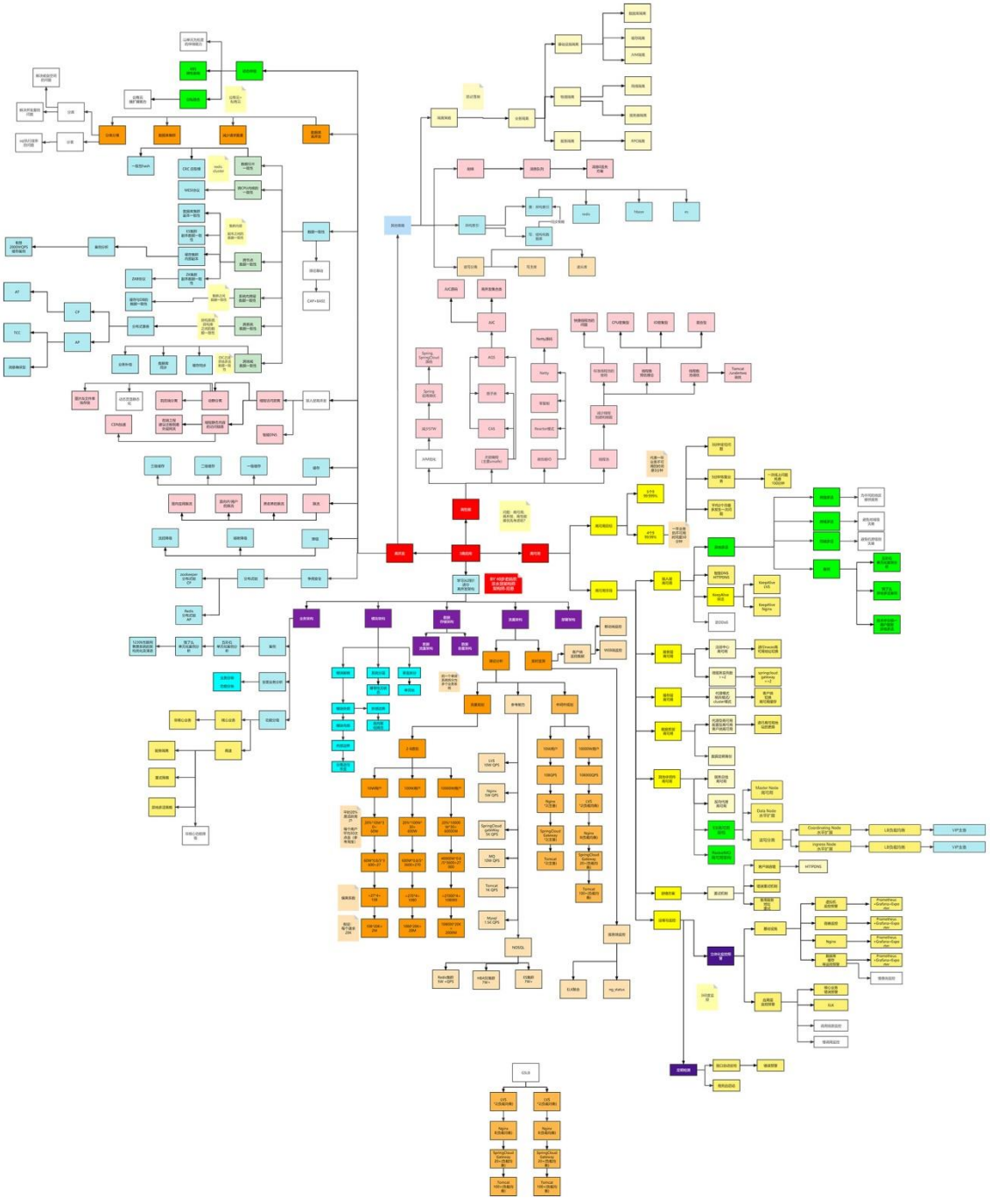
链接：https://www.processon.com/view/link/618a2b62e0b34d73f7eb3cd7



史上最全：价值10W的架构师知识图谱

此图梳理于尼恩的多个 3 高生产项目：多个亿级人民币的大型 SAAS 平台和智慧城市项目

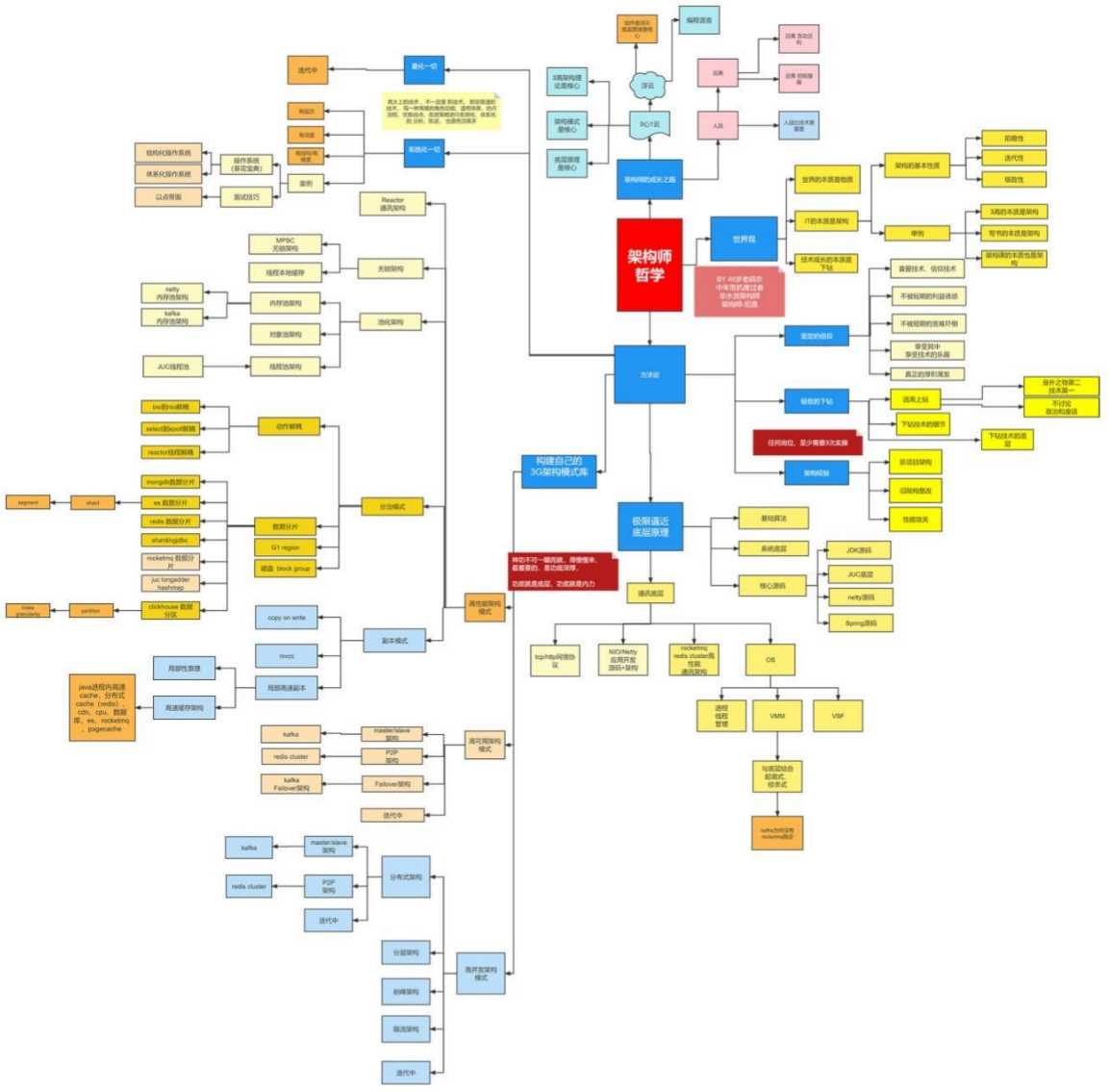
链接：https://www.processon.com/view/link/60fb94216376897l9d246739



牛逼的架构师哲学

40 岁老架构师尼恩对自己的 20 年的开发、架构经验总结

链接：https://www.processon.com/view/link/616f801963768961e9d9aec8



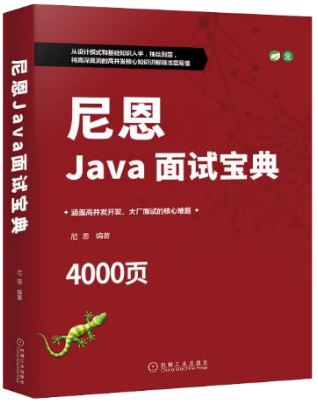
牛逼的3高架构知识宇宙

尼恩 3 高架构知识宇宙，帮助大家穿透 3 高架构，走向技术自由，远离中年危机

链接：https://www.processon.com/view/link/635097d2e0b34d40be778ab4

尼恩Java面试宝典

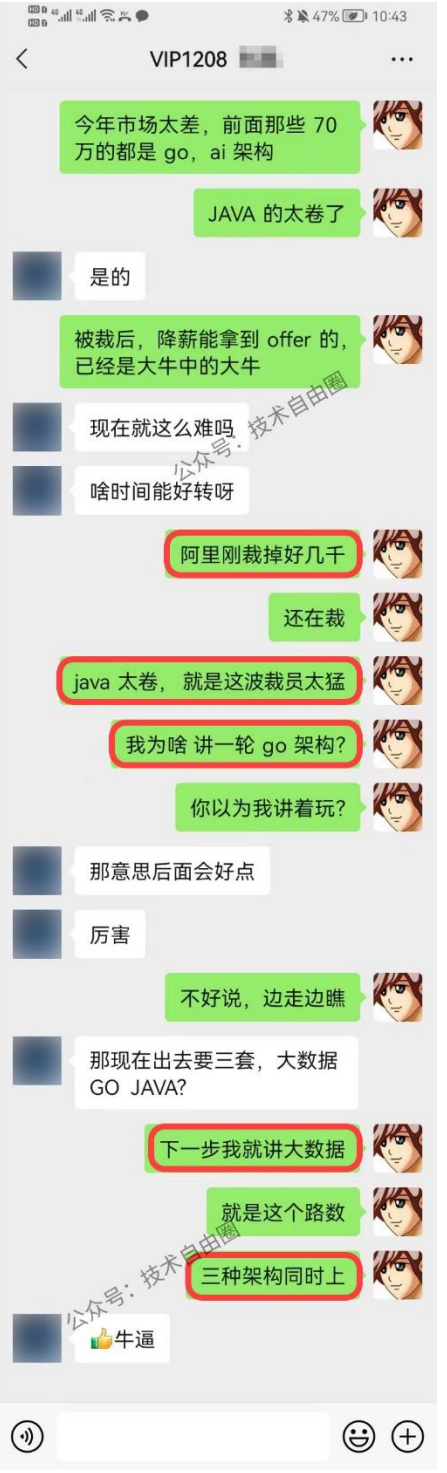
40 个专题（卷王专供+ 史上最全 + 2023 面试必备）
详情：https://www.cnblogs.com/crazymakercircle/p/13917138.html



名称

专题01：JVM面试题（卷王专供 + 史上最全 + 2022面试必备）-V81-from-尼恩Java面试宝典.pdf
专题02：Java算法面试题（卷王专供 + 史上最全 + 2022面试必备）-V80-from-Java面试红宝书.pdf
专题03：Java基础面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书 -release.pdf
专题04：架构设计面试题（卷王专供+ 史上最全 + 2023面试必备）-V86-from-尼恩Java面试宝典.pdf
专题05：Spring面试题_专题06：SpringMVC_专题07：Tomcat面试题（卷王专供+ 史上最全 + 2023面试必备）-V3-from-尼恩面试宝典-release.pdf
专题08：SpringBoot面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题09：网络协议面试题（卷王专供+ 史上最全 + 2023面试必备）-V46-from-尼恩Java面试宝典-release.pdf
专题10：TCPIP协议（卷王专供+ 史上最全 + 2022面试必备）-V57-from-Java面试红宝书.pdf
专题11：JUC并发包与容器类（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题12：设计模式面试题（卷王专供+ 史上最全 + 2022面试必备）-V84-from-Java面试红宝书.pdf
专题13：死锁面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题14：Redis 面试题 （卷王专供+ 史上最全 + 2022面试必备）-V65-from-Java面试红宝书.pdf
专题15：分布式锁 面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题16：Zookeeper 面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题17：分布式事务面试题（卷王专供 + 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题18：一致性协议 （卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题19：Zab协议（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题20：Paxos 协议（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题21：raft 协议（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题22：Linux面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题23：Mysql 面试题（卷王专供+ 史上最全 + 2023面试必备）-V82-from-尼恩Java面试宝典.pdf
专题24：SpringCloud 面试题（卷王专供+ 史上最全 + 2023面试必备）-V12-from-Java面试红宝书-release.pdf
专题25：Netty 面试题 （卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题26：消息队列面试题：RabbitMQ、Kafka、RocketMQ（卷王专供+ 史上最全 + 2023面试必备）-V10-from-Java面试红宝书-release.pdf
专题27：内存泄漏 内存溢出（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题28：JVM 内存溢出 实战（卷王专供+ 史上最全 + 2023面试必备）-V17-from-Java面试红宝书-release.pdf
专题29：多线程面试题（卷王专供+ 史上最全 + 2023面试必备）-V66-from-Java面试红宝书.pdf
专题30：HR面试题：过五关斩六将后，小心阴沟翻船！（史上最全、避坑宝典）-V2-from-Java面试红宝书-release.pdf
专题31：Hash连环炮面试题（卷王专供+ 史上最全 + 2022面试必备）-V68-from-Java面试红宝书.pdf
专题32：大厂面试的基本流程和面试准备（阿里、腾讯、网易、京东、头条.....）-V2-from-Java面试红宝书-release.pdf
专题33：BST、AVL、RBT红黑树、三大核心数据结构（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题34：Elasticsearch面试题（卷王专供+ 史上最全 + 2023面试必备）-V3-from-Java面试红宝书-release.pdf
专题35：Mybatis面试题（卷王专供+ 史上最全 + 2023面试必备）-V3-from-尼恩Java面试宝典-release.pdf
专题36：Dubbo面试题（卷王专供+ 史上最全 + 2023面试必备）-V21-from-尼恩Java面试宝典-release.pdf
专题37：Docker面试题（卷王专供+ 史上最全 + 2023面试必备）-V47-from-尼恩Java面试宝典.pdf
专题38：K8S面试题（卷王专供+ 史上最全 + 2023面试必备）-V59-from-尼恩Java面试宝典.pdf
专题39：Nginx面试题（卷王专供+ 史上最全 + 2023面试必备）-V27-from-尼恩Java面试宝典-release.pdf
专题40：操作系统面试题（卷王专供+ 史上最全 + 2023面试必备）-V28-from-尼恩Java面试宝典-release.pdf
专题41：大厂面试真题（卷王专供+ 史上最全 + 2023面试必备）-V84-from-尼恩Java面试宝典.pdf

未来职业，如何突围：三栖架构师



史上最全 Java 面试题：算法篇（入群获取资料）

刷算法的秘籍

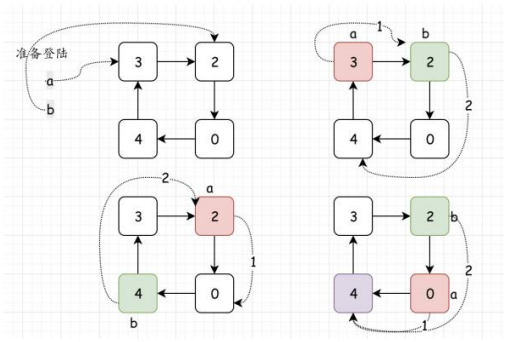
如何刷算法，这块有秘籍， 具体找40岁老架构尼恩，微信交流就可以了

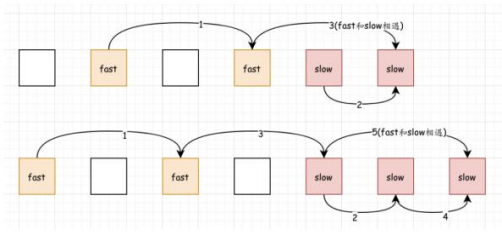
文后有尼恩微信

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。

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）在线性类事中的分析方法。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）。

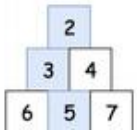
这道题有一定难度哦！如果没有思路请回顾上一篇的学习内容！

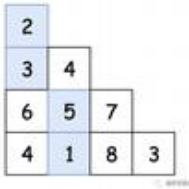
不建议直接看题解！

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]]



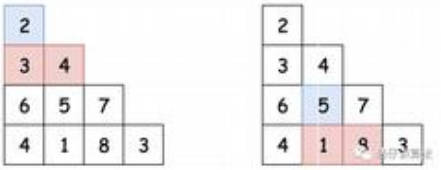


这样相当于我们将整个三角形进行了拉伸，这时候，我们根据题目中给出的条件：每一步只能移动到下一行中相邻

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：

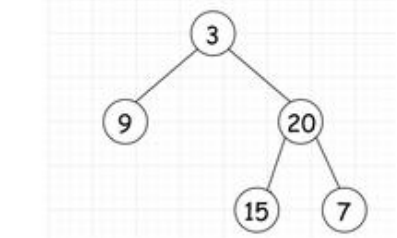
dp[i][j]：表示包含第i行j列元素的最小路径和

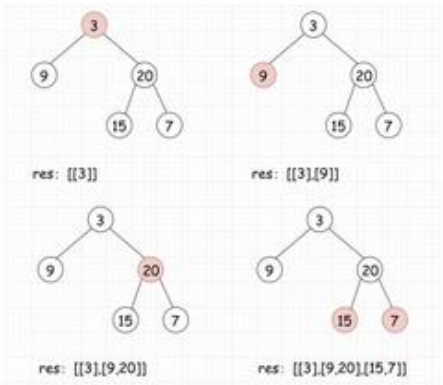
我们很容易想到可以自顶向下进行分析，并且，无论最后的路径是哪一条，它一定要经过最顶上的元素，即(0,0)。所以我们需要对 dp[0][0] 进行初始化。

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：





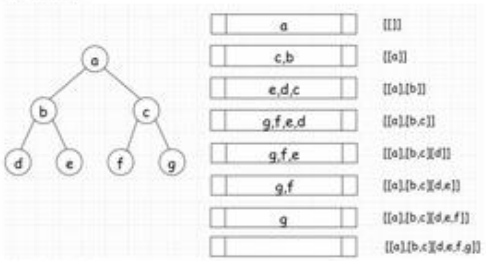
根据以上分析，代码如下：

```
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呢。）

根据分析，得出代码：

```go
func maxSlidingWindow(nums []int, k int) []int {
    if len(nums) == 0 {
        return []int{}
    }
    //将值为模数一个双端队列
    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天
            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，都不是感觉很爽呢

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**。**即作者将搜索出的子串完成题意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。

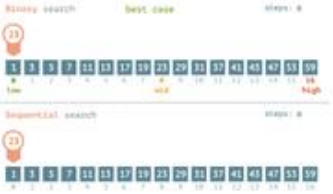
再如：给定一个字符串 s 和一些长度相同的单词 words。找出 s 中恰好可以由 words 中所有单词串联形成的子串的起始位置。

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。

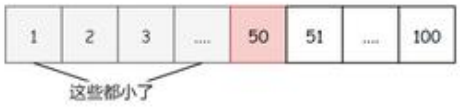
Now，我们通过一道题目来进行具体学习吧

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。每次筛选掉一半数量，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，一般题目不大可能给你一个如此现成的题型，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（开闭区间问题），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：

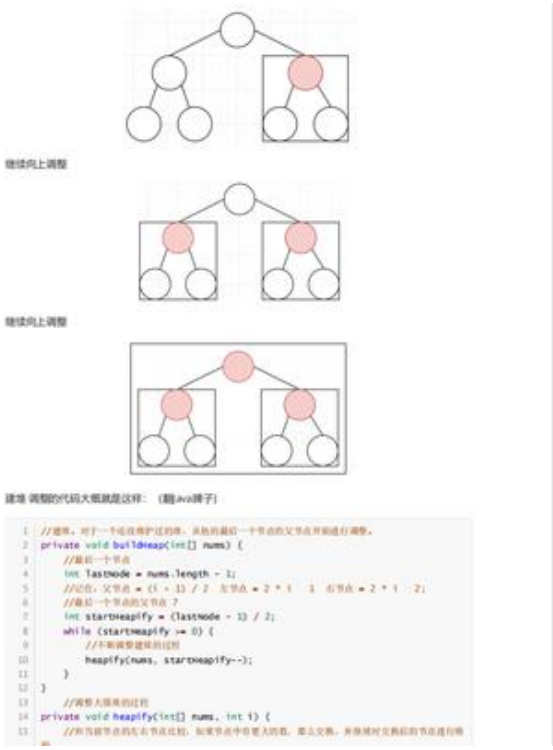
- **初始条件：left = 0, right = length-1**
- **终止：left > right**
- **向左查找：right = mid-1**
- **向右查找：left = mid +1**

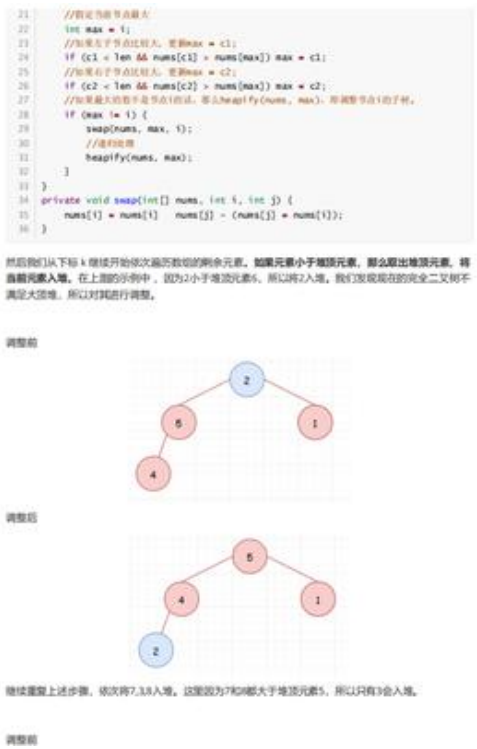
请大家记住这个模板原则，在后面的系列中，我们将介绍二分查找其他的模板类型。

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**「在递增或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：

11. Container With Most Water

题目

Given n non-negative integers a1, a2, ..., an , where each represents a point at coordinate (i, ai). n vertical lines are drawn such that the two endpoints of line i is at (i, ai) and (i, 0). Find two lines, which together with x-axis forms a container, such that the container contains the most water.

Note: You may not slant the container and n is at least 2.

The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

Example 1:

```
Input: [1,8,6,2,5,4,8,3,7]
Output: 49
```

题目大意

给出一个非负整数数组 a1，a2，a3，……an，每个整数标识一个竖立在坐标轴 x 位置的一堵高度为 ai 的墙，选择两堵墙，和 x 轴构成的容器可以容纳最多的水。

解题思路

这一题也是对撞指针的思路。首尾分别 2 个指针，每次移动以后都分别判断长宽的乘积是否最大。

代码

```go
package leetcode

func maxArea(height []int) int {
  max, start, end := 0, 0, len(height)-1
  for start < end {
    width := end - start
    high := 0
    if height[start] < height[end] {
      high = height[start]
      start++
    } else {
      high = height[end]
      end--
    }

    temp := width * high
    if temp > max {
      max = temp
    }
  }
  return max
}
```

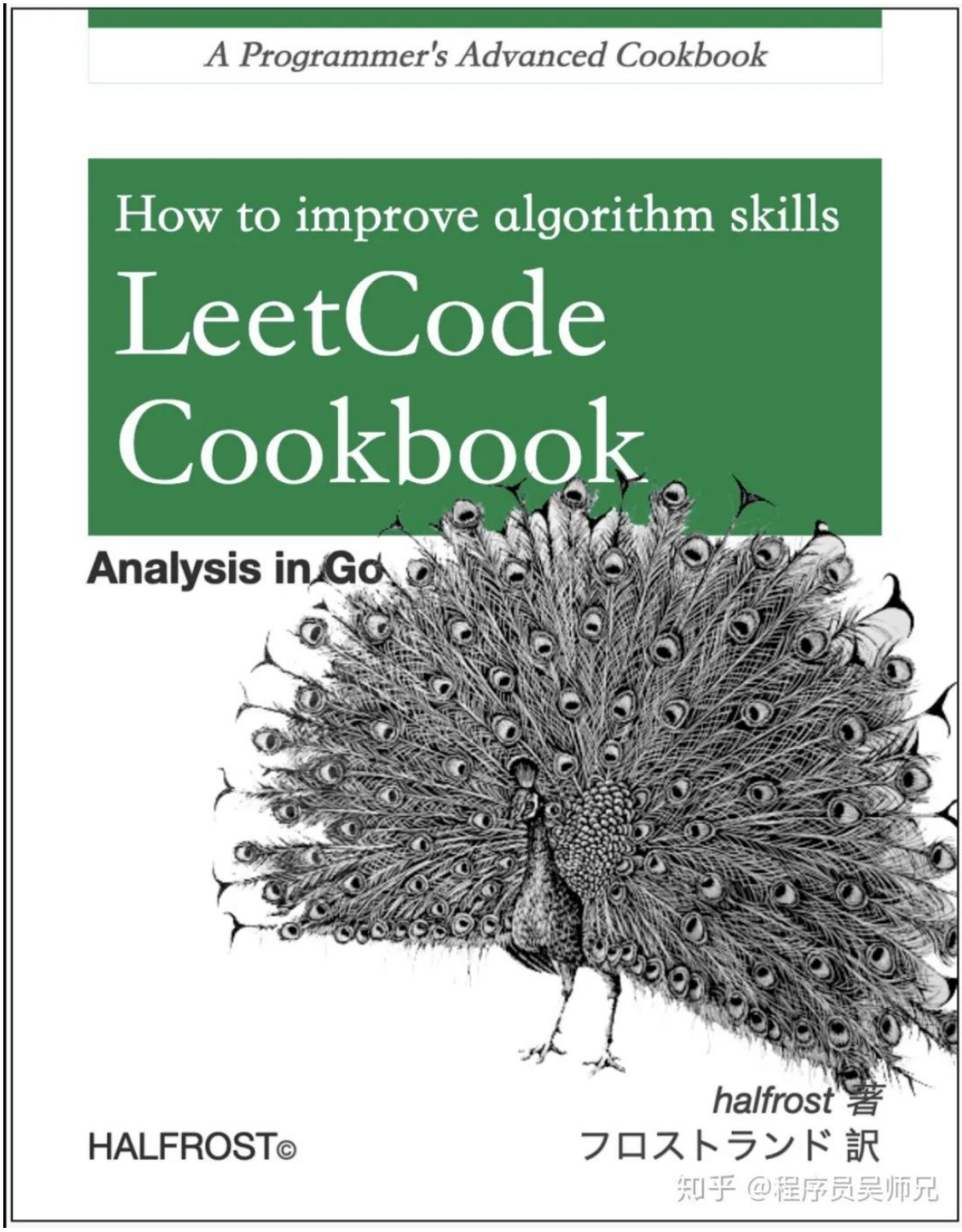
13. Roman to Integer

题目

Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

```
Symbol       Value
I             1
V             5
X             10
L             50
C             100
D             500
M             1000
```

For example, two is written as `II` in Roman numeral, just two one's added together. Twelve is written as, `XII`, which is simply `X + II`. The number twenty seven is written as `XXVII`, which is `XX + V + II`.



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。

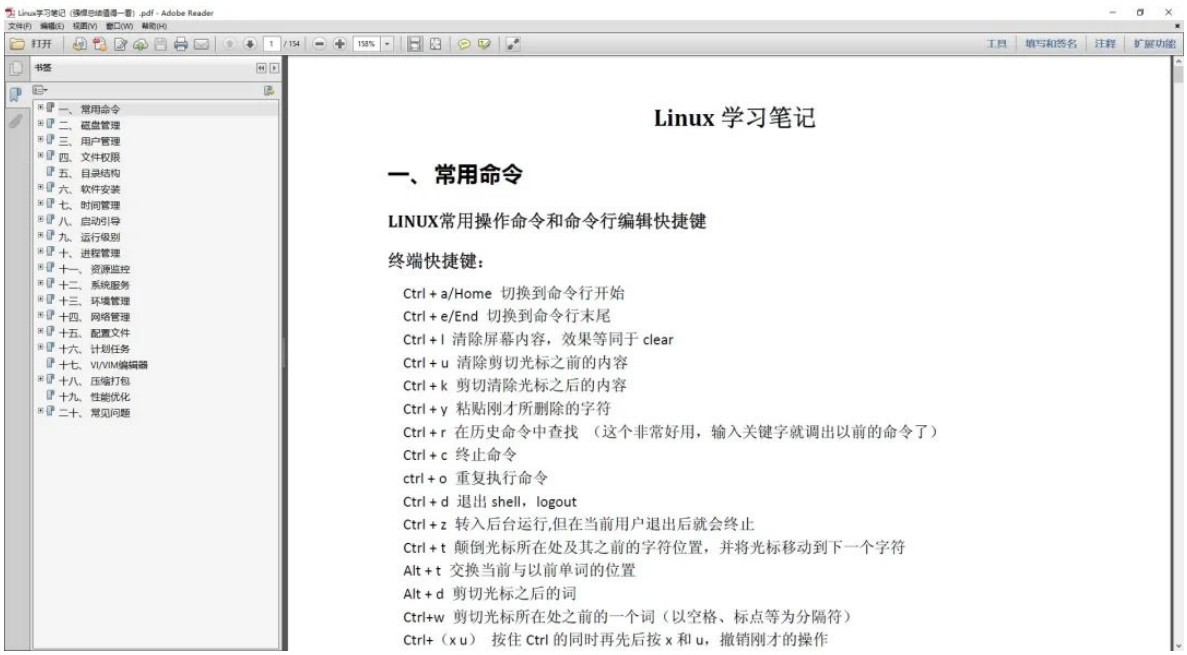
这份资料非常全面且详细，从 **Linux 常用命令**到 **Linux 常用操作**，再到**网络管理**、**性能优化**，几乎覆盖了 Linux 基础学习的方方面面，非常适合初学者入门！

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

找尼恩获取

一些来自互联网的算法题

题目一：有10 亿个 url，每个 url 大小小于 56B，要求去重，内存只给你4G

思路：

1.首先将给定的url调用hash方法计算出对应的hash的value，在10亿的url中相同url必然有着相同的value。

2.将文件的hash table 放到第value%n台机器上。

3.value/n是机器上hash table的值。

将文件分布在多个机器上，这样要处理网路延时。假设有n台机器。

> 首先hash文件得到hash value v
>
> 将文件的hash table 放到第v%n 台机器上。
>
> v/n是机器上hash table的值。

分析：

将文件的url进行hash，得到值value，相同的url的文件具有相同的value，所以会被分配到同一台机器v%n上。在同一台机器上的重复的url文件具有相同的value/n值，如果出现了冲突，不同的url在同一台机器上也可能有相同的value/n值。在每个机器上将value/n值作为key，url值作为value构成hash表进行去重。最后将内存中去重后的hash表中的value值即url写入磁盘。合并磁盘中的各部分url文件，完成去重。

56byte;

4G =*41024=4096kb=4096*1024 byte;

题目二：给定a、b两个文件，各存放50亿个url，每个url各占64字节，内存限制是4G，让你找出a、b文件共同的url?

假如每个url大小为10bytes，那么可以估计每个文件的大小为50G×64=320G，远远大于内存限制的4G，所以不可能将其完全加载到内存中处理，可以采用分治的思想来解决。

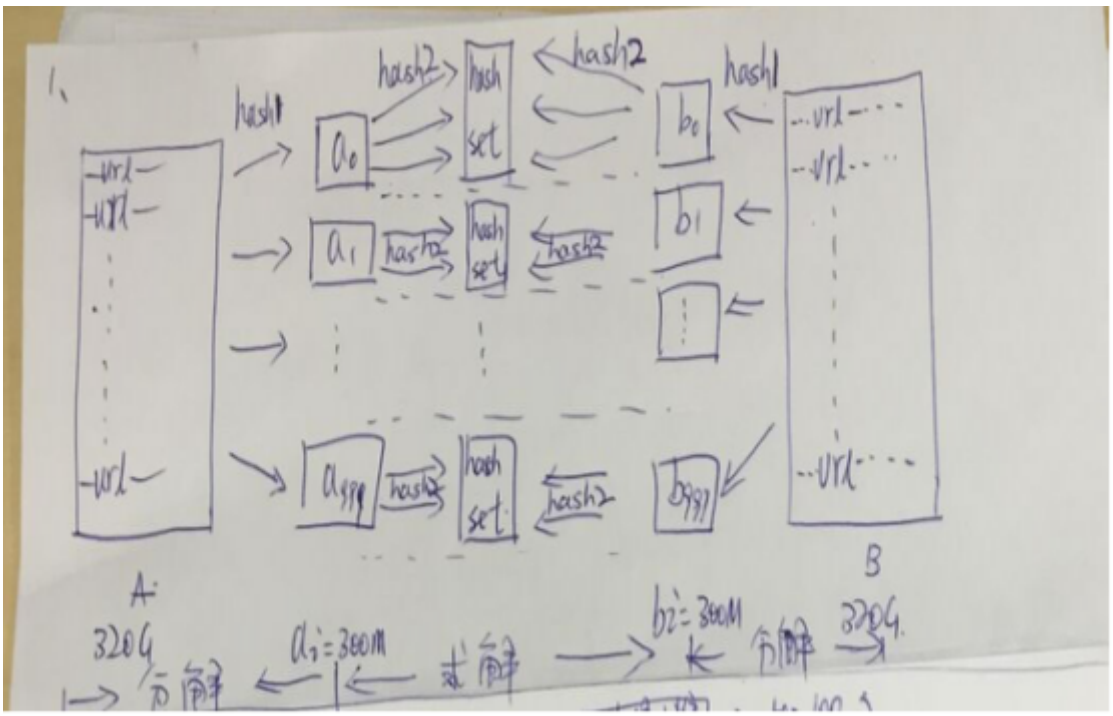
Step1：遍历文件a，对每个url求取hash(url)%1000，然后根据所取得的值将url分别存储到1000个小文件(记为a0,a1,...,a999，每个小文件约300M);

Step2:遍历文件b，采取和a相同的方式将url分别存储到1000个小文件(记为b0,b1,...,b999);

巧妙之处：这样处理后，所有可能相同的url都被保存在对应的小文件(a0vsb0,a1vsb1,...,a999vsb999)中，不对应的小文件不可能有相同的url。然后我们只要求出这个1000对小文件中相同的url即可。

Step3：求每对小文件ai和bi中相同的url时，可以把ai的url存储到hash_set/hash_map中。然后遍历bi的每个url，看其是否在刚才构建的hash_set中，如果是，那么就是共同的url，存到文件里面就可以了。

草图如下(左边分解A，右边分解B，中间求解相同url)：



题目三：有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过16字节，内存限制大小是1M，要求返回频数最高的100个词。

Step1：顺序读文件中，对于每个词x，取hash(x)%5000，然后按照该值存到5000个小文件(记为f0,f1,...,f4999)中，这样每个文件大概是200k左右，如果其中的有的文件超过了1M大小，还可以按照类似的方法继续往下分，直到分解得到的小文件的大小都不超过1M;

Step2：对每个小文件，统计每个文件中出现的词以及相应的频率(可以采用trie树/hash_map等)，并取出出现频率最大的100个词(可以用含100个结点的最小堆)，并把100词及相应的频率存入文件，这样又得到了5000个文件;

Step3：把这5000个文件进行归并(类似与归并排序);

草图如下(分割大问题，求解小问题，归并)：

题目四：现有海量日志数据保存在一个超级大的文件中，该文件无法直接读入内存，要求从中提取某天出访问百度次数最多的那个IP。

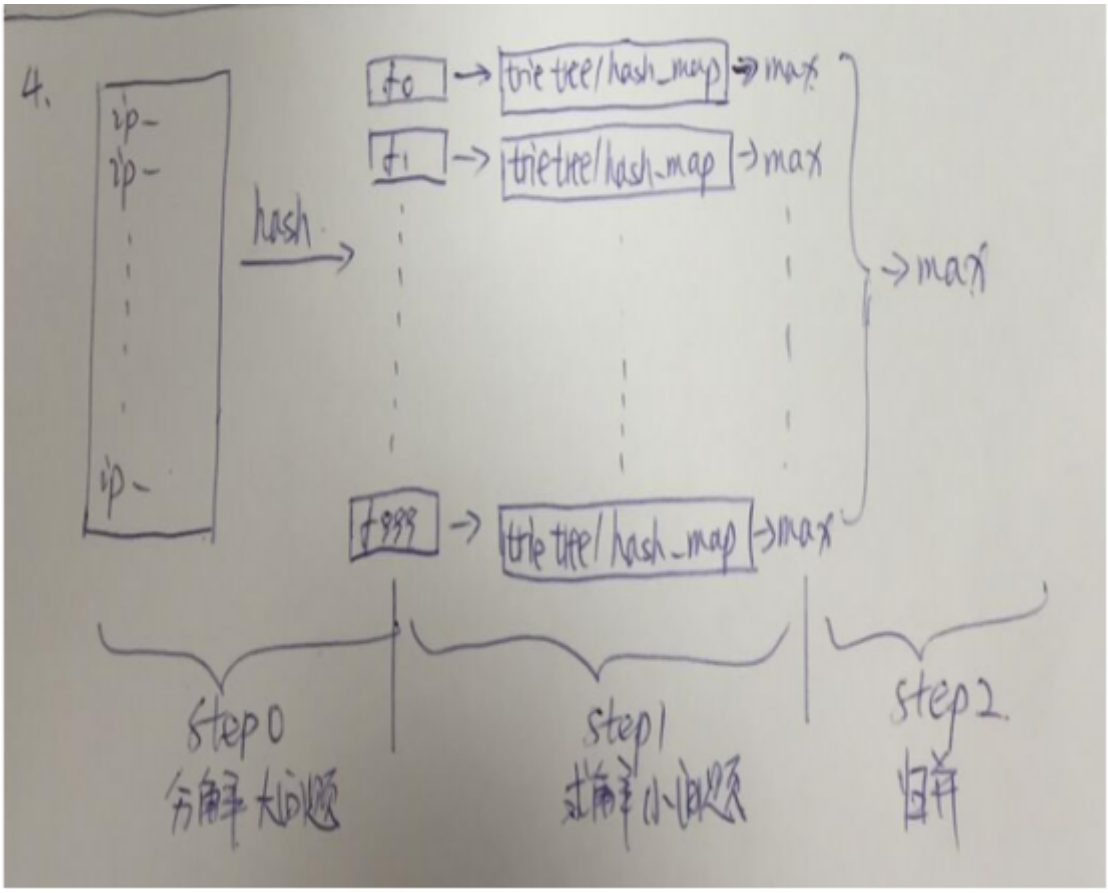
Step1：从这一天的日志数据中把访问百度的IP取出来，逐个写入到一个大文件中;

Step2：注意到IP是32位的，最多有2^32个IP。同样可以采用映射的方法，比如模1000，把整个大文件映射为1000个小文件;

Step3：找出每个小文中出现频率最大的IP(可以采用hash_map进行频率统计，然后再找出频率最大的几个)及相应的频率;

Step4：在这1000个最大的IP中，找出那个频率最大的IP，即为所求。

草图如下：



腾讯二面：40亿QQ号, 1G内存，怎么去重?

说在前面

在40岁老架构师 尼恩的**读者交流群**(50+)中，最近有小伙伴拿到了一线互联网企业如腾讯、美团、阿里、拼多多、极兔、有赞、希音的面试资格，遇到一几个很重要的面试题：

- 40亿QQ号如何设计算法去重，相同的QQ号码仅保留一个，内存限制为1个G?
- 40亿个QQ号，限制1G内存，如何去重?

与之类似的、其他小伙伴遇到过的问题还有：

- 60亿个URL，限制1G内存，如何去重?
- 文件中有40亿个QQ号码，请设计算法对QQ号码去重，相同的QQ号码仅保留一个，内存限制1G.
- 等等等等.....

这里尼恩给大家做一下系统化、体系化的梳理，使得大家可以充分展示一下大家雄厚的 “技术肌肉”，**让面试官爱到 “不能自已、口水直流”。**

也一并把这个题目以及参考答案，收入咱们的 《尼恩Java面试宝典》V70版本，供后面的小伙伴参考，提升大家的 3高 架构、设计、开发水平。

> 最新《尼恩 架构笔记》《尼恩高并发三部曲 》、《尼恩Java面试宝典》 的PDF文件，请通过公众号（技术自由圈）获取。

问题场景分析

分析一下QQ号码的数量：

腾讯的QQ号都是4字节正整数 32个bit位，所以QQ号码的个数是43亿左右，理论值2^32-1个，

又因为是无符号的，翻倍了一下，所以43亿左右。

回顾一下问题：40亿QQ号如何设计算法去重，相同的QQ号码仅保留一个，内存限制为1个G?

问题的本质： 这个就是一个海量数据去重的问题， 但是有一个受限条件， 内存限制为1个G。

解决方案有很多，但是主流的方案有两种：

- 方案1： 使用BitMap进行海量数据去重
- 方案2： 使用布隆过滤器进行海量数据去重

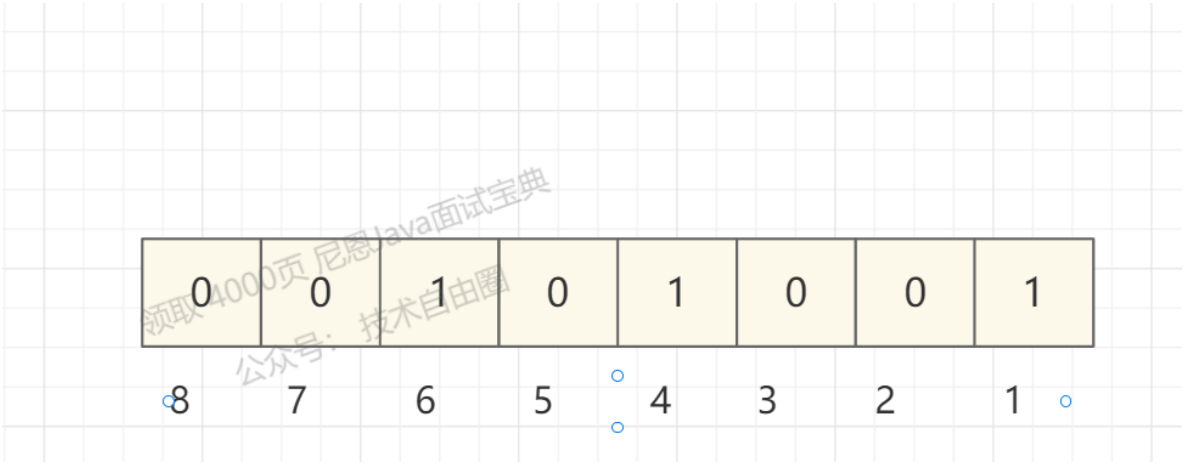
方式1： 使用BitMap进行海量数据去重

首先看看，什么是BitMap？BitMap的使用场景

什么是BitMap？有什么用?

所谓位图（BitMap）其实就是一个bit数组，即每一个位置都是一个bit，其中的取值可以是0或者1

位图（BitMap）思想：就是用一个bit来标记元素，bit是计算机中最小的单位，也就是我们常说的计算机中的0和1，这种就是用一个位来表示的。



像上面的这个位图，可以用来记录三个数：1，4，6。 为啥呢？ 第1位、第4位、第6位 三个位置为 1。

如果不用位图的话，我们想要记录1，4，6 这三个整型的话，怎么办?

就需要用三个unsigned int，已知每个unsigned int占4个字节，那么就是`3*4 = 12`个字节，一个字节有8 bit，那么就是`12*8 = 96`个bit。

结论是：位图最大的好处就是节省空间。这里节省了 12倍。

如何使用BitMap进行40亿个QQ号去重?

回到问题：40亿个QQ号，限制1G内存，如何去重?

前面分析过：一个qq号码，就是一个unsigned int。

40亿个QQ号，就是40亿个 unsigned int，一个 unsigned int占用4个字节。

假如，40亿个 unsigned int直接用内存存储的话，需要多少内存呢?

简单计算一下：

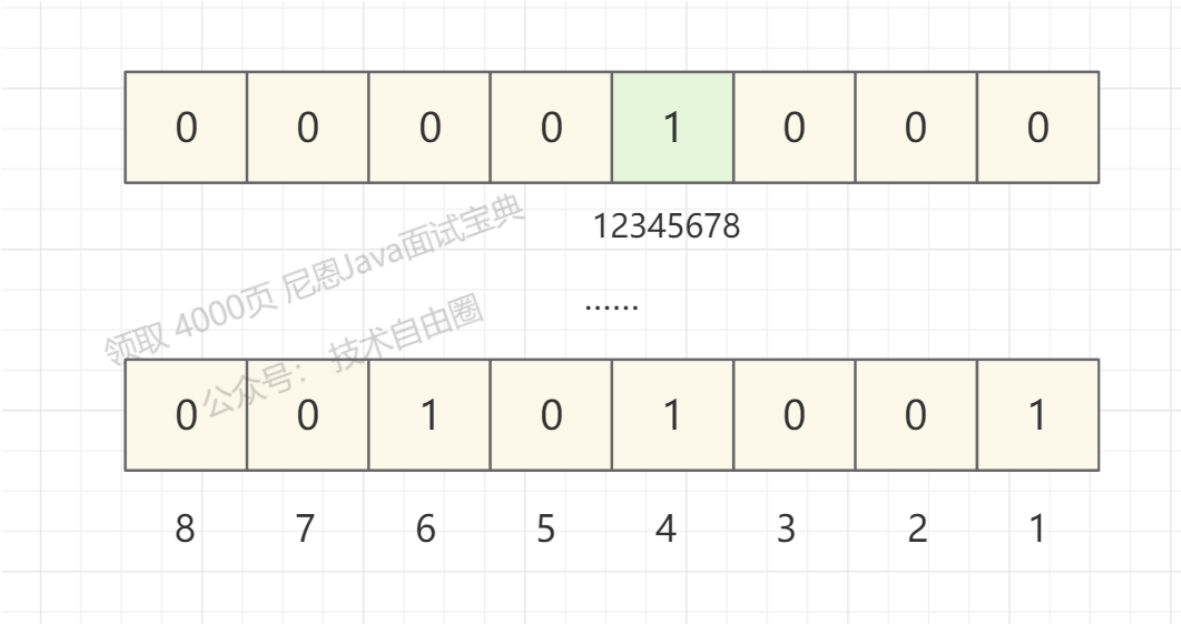
```
4000000000*4 /1024/1024/1024 = 14.9G
```

所以，如果直接把 40亿个QQ号放入内存，需要15个G，1G的空间也是不够用的。

那么，怎么办呢?

qq号是数字，刚好可以使用bitmap。

比如要把一个QQ号"12345678"放到Bitmap中，就需要找到第12345678这个位置，然后把他设置成1就可以了。



这样，把40亿个数字都放到Bitmap之后，位置上是1的表示存在，不为1的表示不存在。

相同的QQ号只需要设置一次1就可以了，那么，最终就把所有是1的数字遍历出来就行了。

使用位图的话，一个数字只需要占用1个bit，那么40亿个数字也就是：

```
4000000000 * 1 /8/1024/1024 = 476M
```

相比于之前的14.9G来说，大大的节省了很多空间。

大约节省了 30倍的空间。

BitMap位图的优势和不足

位图（BitMap），基本思想就是用一个bit来标记元素，bit是计算机中最小的单位，也就是我们常说的计算机中的0和1，这种就是用一个位来表示的。

BitMap位图的主要优势在于它可以非常高效地进行集合运算。具体来说，如果我们需要对一个集合进行多次交集、并集、差集等操作，使用 BitMap 可以将这些操作的时间复杂度降低到 O(1) 级别，而传统的集合实现则需要 O(n) 的时间复杂度，其中 n 是集合的大小。

此外，BitMap 还可以节省存储空间。对于一个只包含 0 和 1 的集合，我们可以使用一个比特位来表示一个元素是否在集合中，这样可以将集合的存储空间降低到原来的 1/8 左右。

所以，位图最大的好处就是节省空间。

位图有很多种用途，特别适合用在去重、排序等场景中，著名的布隆过滤器就是基于位图实现的。

但是位图也有着一定的限制，那就是他只能表示0和1，无法存储其他的数字。

所以BitMap只适合这种能表示ture or false的场景。

其次，BitMap 只适用于值域比较小的集合，因为如果值域过大，BitMap 的存储空间也会过大，这时候使用布隆过滤器可能更为合适。

最后，BitMap 不支持删除操作，因为删除一个元素需要将对应的比特位设置为 0，这可能会影响到其他元素的状态。

方式2： 使用布隆过滤器进行海量数据去重

如果 值域过大，BitMap 的存储空间也会过大，这个时候，需要使用布隆过滤器，进一步进行空间的压缩。

什么是布隆过滤器，实现原理是什么?

布隆过滤器是一种数据结构，用于快速检索一个元素是否可能存在于一个集合(bit 数组)中。

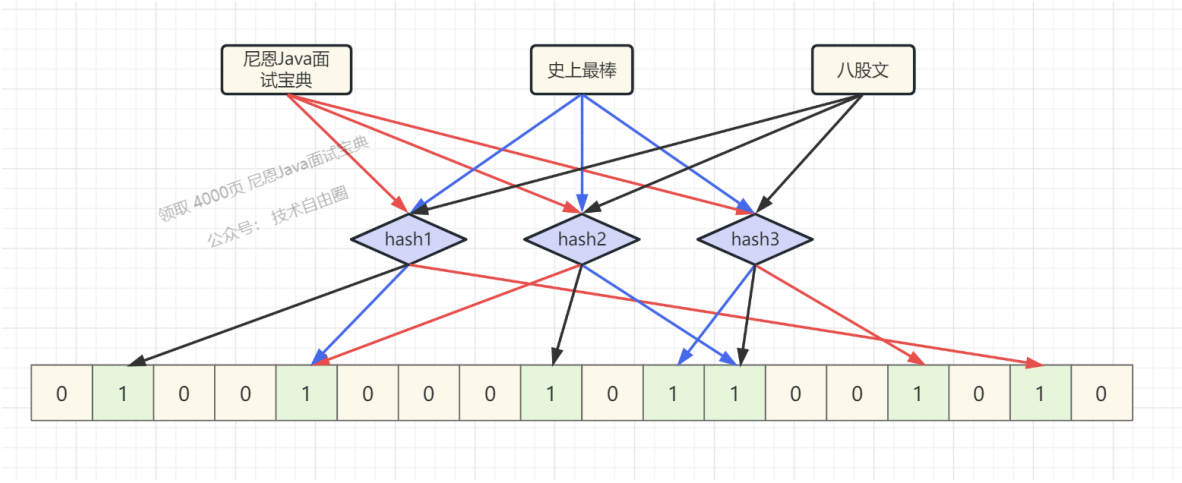
它的基本原理是利用多个哈希函数，将一个元素映射成多个位，然后将这些位设置为 1。

本质上：布隆过滤器内部包含一个bit数组和多个哈希函数，每个哈希函数都会生成一个index 索引值。

由两个部分组成：

- 一个bit数组， 存储数据
- 多个哈希函数， 计算key的 index 索引

如下图所示，里边有三个key： 尼恩Java面试宝典、史上最棒、八股文



问题：如何做 exist（key）这种存在性的判定呢？

答案：当查询一个元素时，如果这些位都被设置为 1，则认为元素可能存在于集合中，否则肯定不存在

比如说：

```
exist（"尼恩Java面试宝典"） 的结果为 true
```

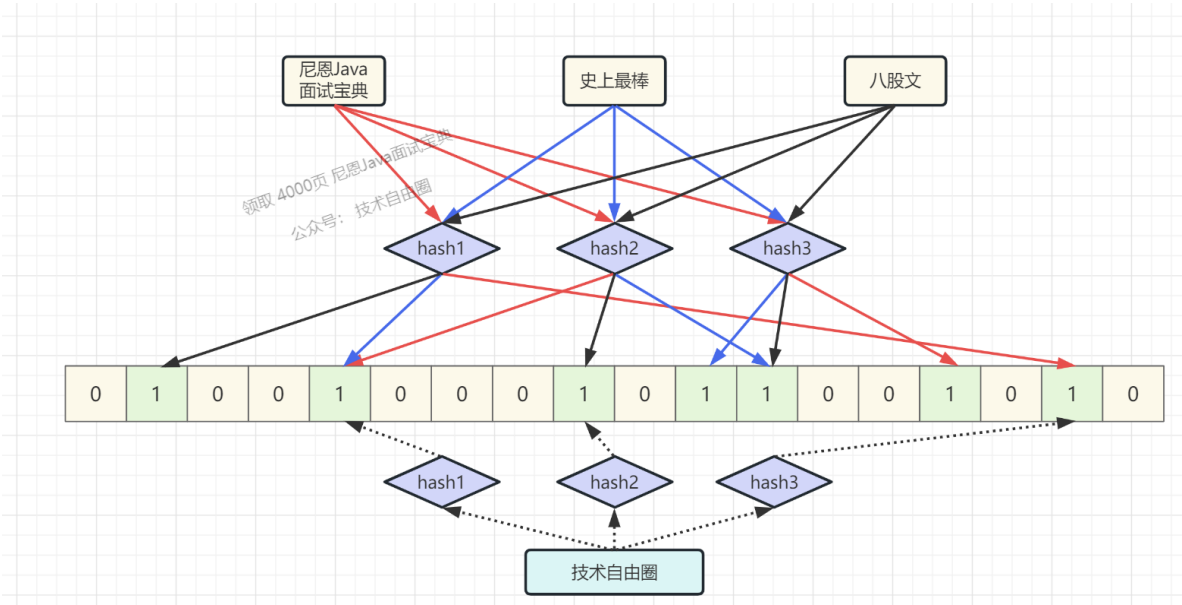
但是： 布隆过滤器可以准确的判断一个元素是否一定不存在。注意，是判断一定不存在。

为啥呢？ 因为哈希冲突的存在。

什么是哈希冲突

什么是 哈希冲突？ 哈希冲突是指两个或多个不同的key 键值被映射到了同一个哈希值。

下面有个例子：



比如说，来了一个新的key "技术自由圈"，现在要判是否存在？

```
exist（"技术自由圈"） 的结果为 true
```

结果是存在的。

为啥呢?

hash1（"技术自由圈"）=1，为啥呢？ 这个之前被 hash1（"史上最棒"）， hash2（"尼恩Java面试宝典"）设置过了，设置两次1.

hash2（"技术自由圈"）=1，为啥呢？ 这个之前被 hash2（"史上最棒"）设置过了1.

hash3（"技术自由圈"）=1，为啥呢？ 这个之前被 hash1（"尼恩Java面试宝典"）设置过了1.

由于， hash1（"技术自由圈"）=1 、 hash2（"技术自由圈"）=1、 hash3（"技术自由圈"）=1，所以，exist（"技术自由圈"） 的结果为 true。

可以，key "技术自由圈" 之前真的没有设置过，是不存在的。

结论是：由于hash冲突，布隆过滤器没办法判断一个元素一定存在。只能判断可能存在。或者判不存在。

如何降低存在性误判的概率

想要降低这种存在性误判的概率，主要的办法就是降低哈希冲突的概率及引入更多的哈希算法。

布隆过滤器的工作过程

下面是布隆过滤器的工作过程：

1、初始化布隆过滤器

在初始化布隆过滤器时，需要指定集合的大小和误判率。

2、添加元素到布隆过滤器

要将一个元素添加到布隆过滤器中，首先需要将该元素通过多个哈希函数生成多个索引值，然后将这些索引值对应的位设置为 1。如果这些索引值已经被设置为 1，则不需要再次设置。

3、查询元素是否存在于布隆过滤器中

要查询一个元素是否存在于布隆过滤器中，需要将该元素通过多个哈希函数生成多个索引值，并判断这些索引值对应的位是否都被设置为 1。如果这些位都被设置为 1，则认为元素可能存在于集合中，否则肯定不存在。

布隆过滤器的主要优点是可以快速判断一个元素是否属于某个集合，并且可以在空间和时间上实现较高的效率。

但是，它也存在一些缺点，例如：

（1）.布隆过滤器在判断元素是否存在时，有一定的误判率。、

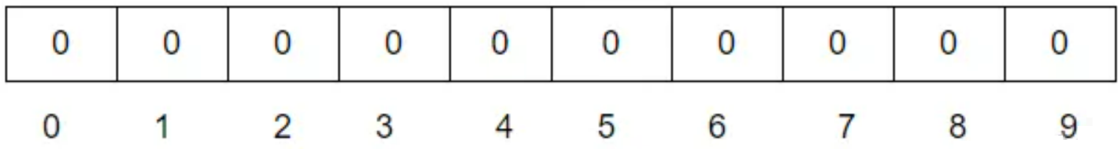
（2）.布隆过滤器删除元素比较困难，因为删除一个元素需要将其对应的多个位设置为 0，但这些位可能被其他元素共享。

布隆过滤器举例

1.布隆过滤器初始状态

布隆过滤器 也是用一个二进制数组进行数据存储。

一开始，二进制数组里是没有值的

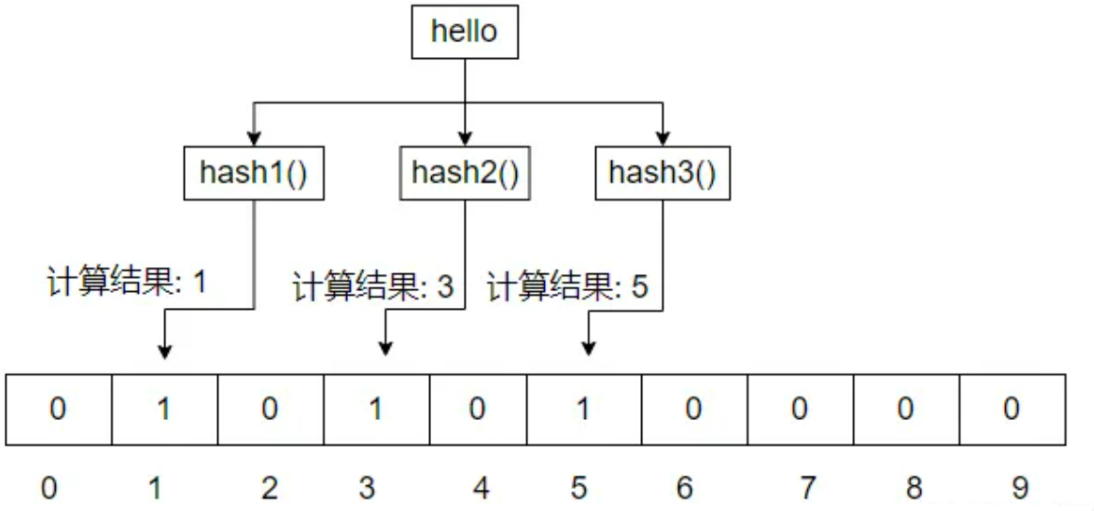


2.存储操作

假设，存储一个数据数据hello

首先，对数据hello经过三次hash运算，分别得到三个值（假设1，3，5）。

然后，在对应的二进制数组里，将下标为1，3，5的值置为1。



3.查询操作

对于数据hello。对数据hello经过三次hash运算，分别得到三个值（假设1，3，5）。

在二进制数组里，将下标为1，3，5的值取出来，如果都为1，则表示该数据已经存在。

4.删除操作

布隆过滤器在使用的时候，不建议进行删除操作。

布隆过滤器里边的部分bit位，完全可能被复用。

假设两个 key： hello、world，如果hash2(hello)结果为3，hash2(world)结果也为3，那么如果删除了hello的hash2(hello)值，就意味着world的hash2(world)值也会被其删除。造成数据的误删。

5.误判率

假设保存两个值，hello和world。hello对应的三个 hash 计算后的index为1，3，5，world三个 hash 对应的index（也就是hash计算后的值）也为1，3，5， 那么 exist(world) = true ，就是一种误判

布隆过滤器应用场景

布隆过滤器因为他的效率非常高，所以被广泛的使用，比较典型的场景有以下几个：

1、网页爬虫：爬虫程序可以使用布隆过滤器来过滤掉已经爬取过的网页，避免重复爬取和浪费资源。

2、缓存系统：缓存系统可以使用布隆过滤器来判断一个查询是否可能存在于缓存中，从而减少查询缓存的次数，提高查询效率。布隆过滤器也经常用来解决缓存穿透的问题。

3、分布式系统：在分布式系统中，可以使用布隆过滤器来判断一个元素是否存在于分布式缓存中，避免在所有节点上进行查询，减少网络负载。

4、垃圾邮件过滤：布隆过滤器可以用于判断一个邮件地址是否在垃圾邮件列表中，从而过滤掉垃圾邮件。

5、黑名单过滤：布隆过滤器可以用于判断一个IP地址或手机号码是否在黑名单中，从而阻止恶意请求。

如何实现实现布隆过滤器

Java中可以使用第三方库来实现布隆过滤器，常见的有Google Guava库和Apache Commons库以及Redis。

Guava版本的布隆过滤器:

尼恩特别说明： 这个版本的布隆过滤器，尼恩指导简历的时候，指导小伙伴用过。

Guava 20.0版本已经引入了布隆过滤器(BloomFilter)的实现。你可以使用以下步骤来使用Guava的布隆过滤器：

1. 引入Guava依赖：

```
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>20.0</version>
</dependency>
```

2. 创建布隆过滤器：

```
int expectedInsertions = 1000000;
double fpp = 0.01;
BloomFilter<String> bloomFilter =
BloomFilter.create(Funnels.stringFunnel(Charset.defaultCharset()),
expectedInsertions, fpp);
```

其中，`expectedInsertions`表示预期插入的元素数量，`fpp`表示误判率(false positive probability)，`Funnels.stringFunnel(Charset.defaultCharset())`表示元素类型为String。

3. 添加元素：

```
bloomFilter.put("hello");
bloomFilter.put("world");
```

4. 判断元素是否存在：

```
bloomFilter.mightContain("hello"); // true
bloomFilter.mightContain("world"); // true
bloomFilter.mightContain("test"); // false
```

注意，布隆过滤器判断元素是否存在，有一定的误判率。如果`mightContain`返回`false`，则可以确定该元素一定不存在；如果`mightContain`返回`true`，则该元素可能存在，需要进一步验证。

5. 序列化和反序列化：

```
// 序列化
FileOutputStream fos = new FileOutputStream("bloom_filter.bin");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(bloomFilter);
oos.close();

// 反序列化
FileInputStream fis = new FileInputStream("bloom_filter.bin");
ObjectInputStream ois = new ObjectInputStream(fis);
BloomFilter<String> bloomFilter2 = (BloomFilter<String>) ois.readObject();
ois.close();

```

注意，序列化和反序列化的过程中，需要将`BloomFilter`类实现`Serializable`接口。

Redisson版本的布隆过滤器

Redisson 是一个基于 Redis 的 Java 客户端，提供了丰富的分布式对象和服务，其中包括布隆过滤器。Redisson 的布隆过滤器实现了标准的布隆过滤器算法，并提供了一些额外的功能，如自动扩容和持久化等。

使用 Redisson 的布隆过滤器非常简单，只需要创建一个 RedissonClient 对象，然后通过该对象获取一个 RBloomFilter 对象即可。

RBloomFilter 提供了一系列的方法，包括添加元素、判断元素是否存在、清空过滤器等。

以下是一个简单的使用 Redisson 布隆过滤器的示例代码：

```
// 创建 Redisson 客户端
Config config = new Config();
config.useSingleServer().setAddress("redis://127.0.0.1:6379");
RedissonClient redisson = Redisson.create(config);

// 获取布隆过滤器对象
RBloomFilter<String> bloomFilter = redisson.getBloomFilter("bloom-filter");

// 初始化布隆过滤器，设置预计元素数量和误判率
bloomFilter.tryInit(10000, 0.03);

// 添加元素
bloomFilter.add("hello");
bloomFilter.add("world");

// 判断元素是否存在
System.out.println(bloomFilter.contains("hello"));
System.out.println(bloomFilter.contains("redis"));

// 清空过滤器
bloomFilter.delete();

```

需要注意的是，Redisson 的布隆过滤器并不支持动态修改预计元素数量和误判率，因此在初始化时需要仔细考虑这两个参数的取值。

Jedis版本的布隆过滤器

如果没有用Redisson， Jedis也可以使用布隆过滤器，参考代码如下：

```
Jedis jedis = new Jedis("localhost");
jedis.bfCreate("myfilter", 100, 0.01);
jedis.bfAdd("myfilter", "Lynn");
jedis.bfAdd("myfilter", "666");
jedis.bfAdd("myfilter", "八股文");
System.out.println(jedis.bfExists("myfilter", "Lynn"));
System.out.println(jedis.bfExists("myfilter", "张三"));jedis.close();

```

由于布隆过滤器存在一定的误判率，因此不能完全替代传统的数据结构，应该根据具体应用场景进行选择。

海量数据去重场景：布隆过滤器和位图如何选择

布隆过滤器和位图都是常用的数据结构，但它们的应用场景和实现方式不同。

布隆过滤器是一种概率型数据结构，用于判断一个元素是否存在于一个集合中。但有一定的误判概率。因此，Bloom Filter不适合那些“零错误”的应用场合。而在能容忍低错误率的应用场合下，Bloom Filter通过极少的错误换取了存储空间的极大节省。

位图是一种简单的数据结构，用于表示一个二进制序列。它通过一个比特位数组来表示一个集合，其中每个比特位表示一个元素是否存在于集合中。当需要判断一个元素是否存在于集合中时，只需要检查对应的比特位是否为 1 或 0 即可。

相比之下，布隆过滤器的空间效率更高，但存在一定的误判概率；而位图的空间效率较低，但不存在误判。因此，在实际应用中，需要根据具体的场景选择合适的数据结构。

说在最后

海量数据去重的的方案，是非常常见的面试题。

以上2大方案，如果大家能对答如流，如数家珍，基本上 面试官会被你 震惊到、吸引到。

最终，**让面试官爱到 “不能自已、口水直流”。** offer， 也就来了。

学习过程中，如果有啥问题，大家可以来 找 40岁老架构师尼恩交流。

参考文献：

清华大学出版社 《尼恩 Java 高并发核心编程 卷2 加强版》

4000页《尼恩 Java面试宝典》中 专题29 多线程 面试专题

[1]. https://www.infoq.cn/article/1afyz3b6hnhprrg12833

[2].https://www.iamle.com/archives/2900.html

[3].https://blog.51cto.com/lianghecai/4755693

[4].https://qinyuanpei.github.io/posts/1333693167/

[5].https://github.com/alibaba/canal/wiki/ClientAdapter

面试题：海量数据去重、Top-k、BitMap问题整理

精选原创

wx641137f3e6d322023-04-03 14:38:20博主文章分类：Java©著作权

文章标签mavenjavaintellij-idea海量数据数据**文章分类Python后端开发阅读数411****

问题引入

首先直接进入正题，40亿QQ号如何设计算法去重，相同的QQ号码仅保留一个，内存限制为1个G。（腾讯的QQ号都是4字节正整数，所以QQ号码的个数是43亿左右，理论值2^32-1个，又因为是无符号的，翻倍了一下，所以43亿左右）

- 方法1：排序

这估计也是最多人能够想到的解决方法，那就是排序，重复的QQ肯定会挨在一起，然后保留第一个，去重就行了。排序后的去重比较简单就不在这里赘述。

但是这么做的问题显然很大，时间复杂大太高了，效率低下。

- 方法2：hsahmap
- hashmap的意思：
- 如果使用hashmap，好处是由于hashmap的去重性质，就可以做到去重。看似上个问题的效率问题解决了，但是又遇到了新的问题，1g的内存太小，解决不了。
- 方法3：文件切割

对于海量数据问题，可以使用文件切割方式，避免内存过大。但是还是需要使用文件间的归并排序、桶排序、堆排序等等，但是这么多文件操作，40亿的数据量还是比较大，那么有没有更好的方法。

- 方法4：bitmap位图操作

可以对hashmap进行优化，采用bitmap这种数据结构，可以顺利地同时解决时间问题和空间问题。

在很多实际项目中，bitmap经常用到。

bitmap就是一种位图。（位图的每一位都只有0和1两个状态）

上面是一个unsigned char类型的位图，一共有八位，那么取值范围就是0-255，就可以标识0-7数字都在，就比如说bitmap【6】=1，那么就可以理解为，QQ号为6的号码是存在的。

那么可以理解为实际上是利用他的下标来表示某个号码是否存在的。

一个unsigned int类型数据是4个字节，那么就是可以表示0-31这 32个数字是否存在。
那么两个unsigned int就是8个字节，那么8x8bit就是0-63个数字表示是否存在。

显然，容易推到，只需要申请512MB的空间大小就可以标识所有QQ号码是否存在，那么就可以完成题目要求了。 形成一个bitmap，例如 bitmap【118677520】= 1（ture）就代表我这个QQ号码是存在的。

然后从小到大遍历所有的正整数（4个字节），当bitmap【】=1的时候，就可以表明是存在的了。

这里再回顾下最开始的问题：40亿QQ号如何设计算法去重，相同的QQ号码仅保留一个，内存限制为1个G。

现在进行问题拓展：
1、文件中有40亿个互不相同的QQ号码，请设计算法对QQ号码进行排序，内存限制1G.
很显然还是上面的思路，直接用bitmap，然后输出，输出后的正整数序列就是排序后的序列了。

2、文件中有40亿个互不相同的QQ号码，求这些QQ号码的中位数，内存限制1G.
还是用bitmap排序比较合适。

3、文件中有40亿个互不相同的QQ号码，求这些QQ号码的top-K，内存限制1G.
可以使用小顶堆或者文件切割，但是用bitmap会更好一些。

经过上面的问题，会发现这类问题有趣多了，明显是有套路通解的。

海量数据题总结

1.Hash算法处理海量数据部分

【题目1】(安卓越 2012) 给定a、b两个文件，各存放50亿个url，每个url各占64字节，内存限制是4G，让你找出a、b文件共同的url?
【题目2】海量日志数据，提取出某日访问百度次数最多的那个IP。
【题目3】有10个文件，每个文件1G，每个文件的每一行存放的都是用户的query，每个文件的query都可能重复。要求你按照query的频度排序。
【题目4】有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过16字节，内存限制大小是1M。返回频数最高的100个词。

2.Top-K海量数据部分

【题目1】(360公司 2012) 100万条记录的文本文件，取出重复数最多的前10条。
【题目2】(360公司 2012) 100亿条记录的文本文件，取出重复数最多的前10条。
【题目3】(腾讯公司 2011) 服务器内存1G，有一个2G的文件，里面每行存着一个QQ号（5-10位数），怎么最快找出出现过最多次的QQ号。
【题目4】(腾讯公司 2015 牛客网) 搜索引擎的日志要记录所有查询串,有一千万条查询,不重复的不超过

三百万，要统计最热门的10条查询串，内存<1G，字符串长 0-255。
(1) 主要解决思路；(2) 算法及其复杂度分析。

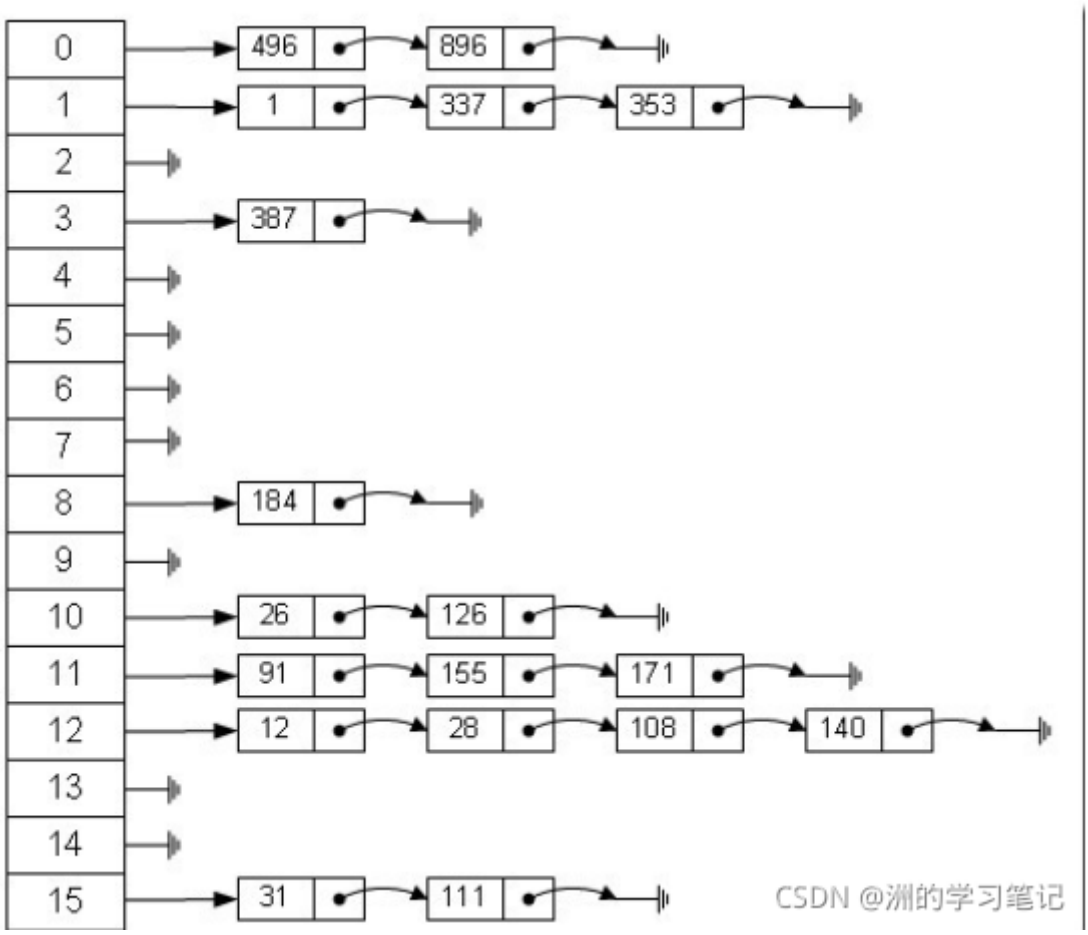
3.bit海量数据部分

【题目1】(腾讯公司)给40亿个不重复的unsigned int的整数，没排过序的，然后再给一个数，如何快速判断这个数是否在那40亿个数当中?
【题目2】(July整理) 在2.5亿个整数中找出不重复的整数，注，内存不足以容纳这2.5亿个整数。
【题目3】40亿QQ号如何设计算法去重，相同的QQ号码仅保留一个，内存限制为1个G。

Hash算法处理海量数据

hash：任意长度的输入通过散列算法，变换成固定长度的输出，该输出就是散列值。这种转换是一种压缩映射，其实hash就是找到一种数据内容和数据存放地址之间的映射关系。

hash有几种常用的hash函数：
直接取余法、乘法取余法、平方取中法。



数组的特点是：寻址容易，插入和删除困难；而链表的特点是：寻址困难，插入和删除容易。那么我们能不能综合两者的特性，做出一种寻址容易，插入删除也容易的数据结构？答案是肯定的，这就是哈希表，哈希表有多种不同的实现方法，我接下来解释的是最常用的一种方法——拉链法，我们可以理解为“链表的数组”，如上图所示。

适用范围：快速查找，删除的基本数据结构，通常需要总数据量可以放入内存。

那么hash算法在海量数据中的应用过程可以分这么几步：

1、分而治之思想、hash思想。
也就是说采用hash取模进行等价映射，将巨大的文件进行等价分割，注意，如果采用hash取模，不论是什么hash函数，符合一定规律也就是同一种规律或同类数据都是会被分割到同一个小文件中的，变成若干个小文件的操作。这个方法对于数据量很大的时候，内存有限制的时候是十分有效的。

2、利用haspmap在内存中进行统计。
通过Hash映射将大文件分割为小文件后，就可以采用HashMap这样的存储结构来对小文件中的关注项进行频率统计。具体的做法是将要进行统计的Item作为HashMap的key，此Item出现的次数作为value。

3、对存储在HashMap中的数据根据出现的次数进行排序。
排序我们可以采用堆排序、快速排序、归并排序等方法。

现在可以看看具体的解法了。

题目1：给定a、b两个文件，各存放50亿个url，每个url各占64字节，内存限制是4G，让你找出a、b文件共同的url?

思路: 还是老一套，先Hash映射降低数据规模，然后统计排序。
具体做法：
(1) 分析现有数据的规模
按照每个url64字节来算，每个文件有50亿个url，那么每个文件大小为5G*64=320G（按照1000换算10亿字节=1GB）。320G远远超出内存限定的4G，所以不能将其全部加载到内存中来进行处理，需要采用分而治之的方法进行处理。

(2) Hash映射分割文件
逐行读取文件a，采用hash函数：Hash(url)%1000将url分割到1000个小文件中，文件即为f1_1,f1_2,f1_3,...,f1_1000。那么理想情况下每个小文件的大小大约为300M左右。再以相同的方法对大文件b进行相同的操作再得到1000个小文件，记为： f2_1,f2_2,f2_3,...,f2_1000。

经过一番折腾后我们将大文件进行了分割并且将相同url都分割到了这2组小文件中下标相同的两个文件中,其实我们可以将这2组文件看成一个整体：

f1_1& f2_1， f1_2& ,f2_2, f1_3& f2_3,..., f1_1000& f2_1000

那么我们就可以将问题转化成为求这1000对小文件中相同的url就可以了。接下来，求每对小文件中的相同url，首先将每对对小文件中较小的那个的url放到HashSet结构中，然后遍历对应这对小文件中的另一个文件，看其是否存才刚刚构建的HashSet中，如果存在说明是一样的url，将这url直接存到结果文件就ok了。如果存在大文件接着hash划分即可。

【题目2】 海量日志数据，提取出某日访问百度次数最多的那个IP

思路：当看到这样的业务场景，我们脑子里应该立马会想到这些海量网关日志数据量有多大？这些IP有多少中组合情况，最大情况下占多少存储空间？解决这样的问题前我们最重要的先要知道数据的规模，这样才能从大体上制定解决方案。所以现在假设这些这些**网关日志量有3T**。下面大体按照我们上面的步骤来对解决此场景进行分析：

（1）首先，从这些海量数据中过滤出指定一天访问百度的用户IP，并逐个写到一个大文件中。

（2）**采用“分而治之”的思想用Hash映射将大文件进行分割降低数据规模。**

按照IP地址的Hash(IP)%1024值，把海量IP日志分别存储到1024个小文件中，其中Hash函数得出值为分割后小文件的编号。

（3）逐个读小文件，对于每一个小文件**构建一个IP为key，出现次数为value的HashMap。**

对于怎么利用HashMap记录IP出现的次数这个比较简单，因为我们可以通过程序读小文件将IP放到HashMap中key的之后可以先判断此IP是否已经存在如果不存在直接放进去，其出现次数记录为1，如果此IP已经存储则过得其对应的value值也就是出现的次数然后加1就ok。最后，按照IP出现的次数采用排序算法对HashMap中的数据进行**排序**， 同时记录当前出现次数最多的那个IP地址。

（4）走到这步，我们可以得到1024个小文件中出现次数最多的IP了，再采用常规的排序算法找出总体上出现次数最多的IP就ok了。

这个我们需要特别地明确知道一下几点内容：

第一：我们通过Hash函数:**Hash(IP)%1024**将大文件映射分割为了1024个小文件，那么这1024个小文件的大小是否均匀？另外，我们采用HashMap来进行IP频率的统计，内存消耗是否合适？

首先是第一个问题，被分割的小文件的大小的均匀程度是取决于我们使用怎么样的Hash函数 ，对本场景而言就是：Hash(IP)%1024。设计良好的Hash函数可以减少冲突，使数据均匀的分割到1024个小文件中。但是尽管数据映射到了另外一些不同的位置，但数据还是原来的数据，只是代替和表示这些原始数据的形式发生了变化而已。

另外，看看第二个问题：用HashMap统计IP出现频率的内存使用情况。

要想知道HashMap在统计IP出现的频率，那么我们必须对IP组合的情况有所了解。32Bit的IP最多可以有**2^32**种的组合方式，也就是说去所有IP最多占**4G存储空间**。在此场景中，我们已经根据IP的hash值将大文件分割出了1024个小文件，也就是说这4G的IP已经被分散到了1024个文件中。那么在Hash函数设计合理最perfect的情况下针对每个小文件的HashMap占的内存大小最多为4G/1024+存储IP对应的次数所占的空间，所以内存绝对够用。

第二：Hash取模是一种**等价映射**，换句话说通过映射分割之后相同的元素只会分到同一个小文件中去的。就本场景而言，相同的IP通过Hash函数后只会被分割到这1024个小文件中的其中一个文件。

Top-K问题解决

【题目2】(360公司 2012) 100亿条记录的文本文件，取出重复数最多的前10条。

刚才是100万的数据，你的计算机可以单批正常处理，现在有100亿的数据，假设由于你的计算机内存、cpu限制，无法单批处理 ...

处理方法主要参考July中关于Top-K海量数据的介绍，方法如下：

1.100万的直接用hash存储**key(值),value(次数)**。同时建一个10个元素的数组，一个整数记录数据中最小次数，循环一次没有出现过的插入hash表中，value记1，如已存则value加1，value时大于数组中的最小次数则进行替换，改写整数值。

PS：假设一条记录64字节，100万应该为64MB（10亿字节=1GB，1000换算时），内存此时肯定够用；而100亿条此时需要640GB内存，显然分治实现。

2.100亿类似100万时的处理方法，对**数据进行切片**，可以都切为100万的记录，对100万最前10，不同在于这前10也存入hash，如果key相同则合并value，显然100亿的数据分割完后的处理结果也要再进行类似的处理，hash表不能过长，原理其实也就是map和reduce。

简单总结步骤如下：

(1) 分而治之

(2) HashMap<key,value>=<字符串,次数>

(3) 数据合并类似MapReduce

(4) 排序输出TopK，可以采用高效的堆排、快排 、归并排序等

bitmap处理海量数据

【题目1】(腾讯公司)给40亿个不重复的unsigned int的整数，没排过序的，然后再给一个数，如何快速判断这个数是否在那40亿个数当中?

方案一：申请512M内存，一个bit位代表一个unsigned int值，读40亿个数，设置相应bit位，读入要查询位，查看bit是否为1，若为1表示存在否则表示不存在。

方案二：这个问题在《编程珠玑》里有很好的描述，大家可以参考下面的思路，探讨一下：

又因为2^32为40亿多，所以给定一个数可能在，也可能不在其中；这里我们把40亿个数中的每一个用32位的二进制来表示。假设这40亿个数开始放在一个文件中，然后将这40亿个数分成两类：

1.最高位为0

2.最高位为1

并将这两类分别写入到两个文件中，其中一个文件中数的个数<=20亿，而另一个>=20亿（这相当于折半了）；与要查找的数的最高位比较并接着进入相应的文件再查找。再然后把这个文件为又分成两类:

1.次最高位为0

2.次最高位为1

并将这两类分别写入到两个文件中，其中一个文件中数的个数<=10亿，而另一个>=10亿（相当于折半）；与要查找的数的次最高位比较并接着进入相应的文件再查找。

.....

以此类推，就可以找到了,而且时间复杂度为O(logn)，方案2完。

【题目2】(July整理) 在2.5亿个整数中找出不重复的整数，注，内存不足以容纳这2.5亿个整数。

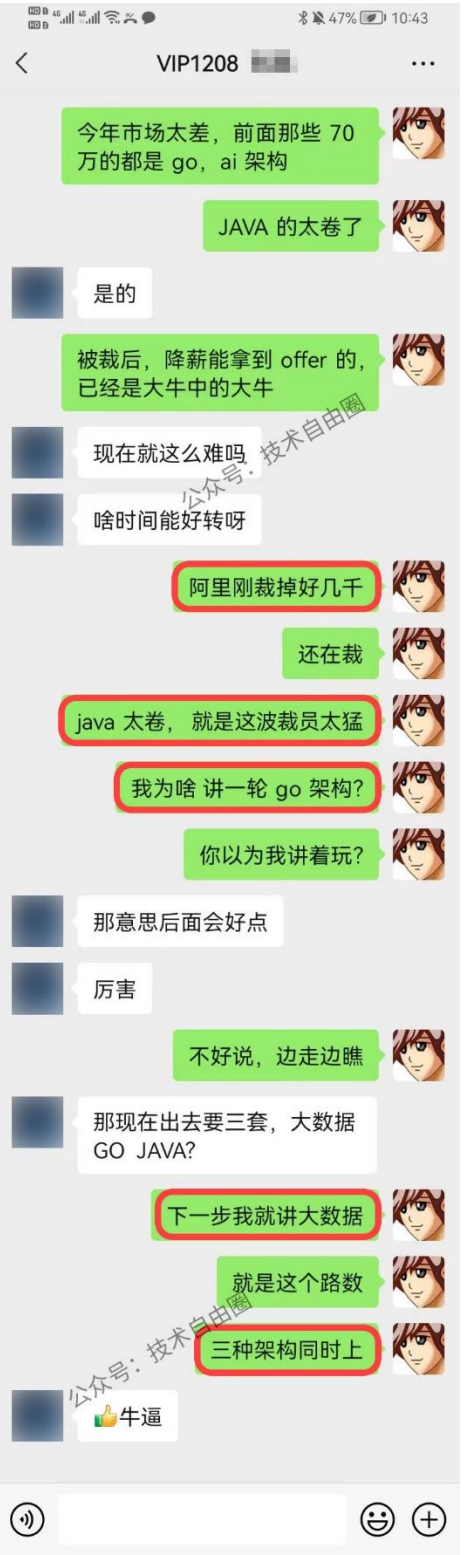
方案一：采用2-Bitmap（每个数分配2bit，00表示不存在，01表示出现一次，10表示多次，11无意义）进行，共需内存2^32 * 2 bit=1 GB内存，还可以接受。然后扫描这2.5亿个整数，查看Bitmap中相对应位，如果是00变01，01变10，10保持不变。所描完事后，查看bitmap，把对应位是01的整数输出即可。

方案二：也可采用与海量日志中找IP次数最多的类似方法，进行划分小文件的方法。然后在小文件中找出不重复的整数，并排序。然后再进行归并，注意去除重复的元素。

这类题的方法涉及到很多知识，包括：

Bloom Filter、Hash、Bit-Map、堆(Heap)、分而治之、数据库索引、倒排索引、外排序、Trie树(字典树)、MapReduce

未来职业，如何突围：三栖架构师



成功案例：2年翻3倍，35岁卷王成功转型为架构师

详情：http://topcoder.cloud/forum.php?mod=forumdisplay&fid=43&page=1

最新　最后发表　热门　精华

- 成功案例：[1057号卷王] 3年小伙拿到外企offer，薪酬涨了200%
  卷王1号　超级版主　前天 17:41
- 成功案例：[645号卷王] 4年经验卷王逆袭，被毕业后，反涨24W
  卷王1号　超级版主　2022-9-21
- 成功案例：[878号卷王] 小伙8年经验，年薪60W
  卷王1号　超级版主　2022-8-13
- 年薪70W案例：通过尼恩的指导，小伙伴年薪从40W涨到70W
  卷王1号　超级版主　2022-2-11
- 成功案例：[493号卷王] 5年小伙拿满意offer，就业寒冬季逆涨30%
  卷王1号　超级版主　前天 17:43
- 成功案例：[250号卷王] 就业极寒时代，收offer 涨25%
  卷王1号　超级版主　前天 17:38
- 成功案例：[612号卷王] 就业极寒时代，从外包到自研
  卷王1号　超级版主　前天 17:15
- 成功案例：[913号卷王] 热烈祝贺6年经验卷王，年薪40W
  卷王1号　超级版主　2022-9-21
- 成功案例：[959号卷王] 4年经验卷王，喜获百度、Boss直聘等N个优质offer，最高涨100%
  卷王1号　超级版主　2022-9-21
- 成功案例：[529号卷王] 5年经验卷王喜收2大offer，最高涨5K
  卷王1号　超级版主　2022-9-21
- 成功案例：[811号卷王] 热烈祝贺7年经验卷王，薪酬涨30%
  卷王1号　超级版主　2022-9-21
- 成功案例：[287号卷王] 不惧大寒潮，卷王逆市收4 offer，涨30%，可喜可贺
  卷王1号　超级版主　2022-5-30
- 成功案例：[1002号卷王] 5月份"被毕业"，改简历后，斩获顶级央企Offer，涨薪7000+
  卷王1号　超级版主　2022-7-5

成功案例：[7号卷王] 热烈祝贺小伙伴涨薪120%

卷王1号 超级版主 2022-8-13

成功案例：[134号卷王] 大三小伙卷1年，斩获顶级央企Offer，成功逆袭

卷王1号 超级版主 2022-7-6

成功案例：[1008号卷王] 5年经验卷王收42W offer，月涨8000，可喜可贺

卷王1号 超级版主 2022-5-30

成功案例：[453号卷王] 非全日制 6年卷王喜提3 offer，年薪30W，可喜可贺

卷王1号 超级版主 2022-5-21

成功案例：[924号卷王] 6年卷王喜提4 offer，最高涨薪9000，可喜可贺

卷王1号 超级版主 2022-5-21

成功案例：[15号卷王] 4年卷王入职 微软，涨薪50%，可喜可贺

卷王1号 超级版主 2022-5-12

成功案例：[527号卷王] 4年卷王喜提2 offer，涨薪50%，可喜可贺

卷王1号 超级版主 2022-5-13

成功案例：[788号卷王] 3年卷王喜提优质Offer，涨薪60%

卷王1号 超级版主 2022-5-11

成功案例：热烈祝贺：非全日制卷王，喜提2个心仪offer，面3家过2家

卷王1号 超级版主 2022-4-21

成功案例：[693号卷王] 二线城市6年卷王喜提4大优质Offer，含央企offer，最高薪酬35W

卷王1号 超级版主 2022-4-16

成功案例：[85号卷王] 双非2本小伙，春招大捷，喜提9个offer，最高薪酬近30万

卷王1号 超级版主 2022-4-14

成功案例：[741号卷王] 卷王逆袭！6年小伙从很少面试机会到搞定35K*14薪Offer

卷王1号 超级版主 2022-4-12

成功案例：[642号卷王] 热烈祝贺，6年卷王喜提优质国企offer

卷王1号 超级版主 2022-4-7

成功案例：[796号卷王] 热烈祝贺，36岁卷王喜提52万优质offer

卷王1号 超级版主 2022-3-25

成功案例：[15号卷王] 小伙卷1年，涨薪9K+，喜收ebay等多个优质offer

卷王1号 超级版主 2022-3-24

成功案例：[821号卷王] 小伙狠卷3个月，喜提10多个offer

卷王1号 超级版主 2022-3-21

成功案例：[736号卷王] 3年半经验收22k offer，但是小伙志存高远，冲击25k+

卷王1号 超级版主 2022-3-20

成功案例：热烈祝贺1群小卷王offer拿到手软，甚至拒了阿里offer

卷王1号 超级版主 2022-3-16

简历案例：简历一改，腾讯的邀请就来了！热烈祝贺，小伙收到一大堆面试邀请

卷王1号 超级版主 2022-3-10

成功案例：祝贺我圈两大超级卷王，一个过了阿里HR面，一个过了阿里2面

卷王1号 超级版主 2022-3-10

成功案例：小伙伴php转Java，卷1.5年Java，涨薪50%，喜收多个优质offer

卷王1号 超级版主 2022-3-10

成功案例：4年小伙狠卷半年，拿到 移动、京东 两大顶级offer

尼恩 超级版主 2022-3-5

成功案例：[267号卷王] 助力3年经验卷王，拿到蜂巢的17k x 14薪的offer

卷王1号 超级版主 2022-2-27

成功案例：[143号卷王] 二本院校00后卷神，毕业没到一年跳到字节，年薪45W

卷王1号 超级版主 2022-2-27

成功案例：[494号卷王] 尼恩分布式事务助力卷王拿到 中信银行offer

卷王1号 超级版主 2022-2-27

成功案例：[76号卷王] 2线城市卷王，狠卷1.5年，喜收22K offer

卷王1号 超级版主 2022-2-27

成功案例：[429号卷王] 小伙伴在社群卷5个月，涨8k+

卷王1号 超级版主 2022-2-27

成功案例：[154号卷王] 双非学校毕业卷王，连拿 京东到家&滴滴 两个大厂Offer

卷王1号 超级版主 2022-2-27

成功案例：[232号卷王] 涨薪10K，继续卷向食物链顶端

卷王1号 超级版主 2022-2-27

成功案例：狠卷1年技术，喜收 腾讯、阿里、微软三大Offer，最高年薪56W

卷王1号 超级版主 2022-2-27

成功案例：[449号卷王] 应届毕业卷王喜收 滴滴offer, 年薪33W

卷王1号 超级版主 2022-2-27

成功案例：[551号卷王] 小伙伴学完后，成功进入大厂，并且推荐自己的朋友加VIP学习

卷王1号 超级版主 2022-2-10

成功案例：[214号卷王] 助力2年经验卷王，成功拿到17K月薪

卷王1号 超级版主 2022-2-10

成功案例：[92号卷王] 课程实操助力社群小伙伴喜收 喜马拉雅Offer

卷王1号 超级版主 2022-2-10

成功案例：社群卷王小伙伴成功过了滴滴三面 获滴滴Offer

卷王1号 超级版主 2022-2-10

[612号卷王]滴滴小伙伴，蹲点考察半年，觉得靠谱后加入 疯狂创客圈

卷王1号 超级版主 2022-2-10

成功案例：[732号卷王] 尼恩助力3年经验卷王收获 京东offer, 年薪35W

卷王1号 超级版主 2022-2-27

成功案例：[558号卷王] 2年经验卷王，喜收 网易和阿里子公司两个优质offer

卷王1号 超级版主 2022-2-27

成功案例：[569号卷王] 双非应届生卷王, 喜收字节跳动实习offer

卷王1号 超级版主 2022-2-25

成功案例：[420号卷王] 狠卷1年，卷王涨薪80%，涨薪12000元!

卷王1号 超级版主 2022-2-25

成功案例：[76号卷王] 通过尼恩1年半的指导，专科学历小伙伴从0.8K涨到22K

卷王1号 超级版主 2022-2-10

硬核推荐：尼恩Java硬核架构班

详情：https://www.cnblogs.com/crazymakercircle/p/9904544.html

架构班（社群 VIP）的起源：

最初的视频，主要是给读者加餐。很多的读者，需要一些高质量的实操、理论视频，所以，我就围绕书，和底层，做了几个实操、理论视频，然后效果还不错，后面就做成迭代模式了。

架构班（社群 VIP）的功能：

提供高质量实操项目整刀真枪的架构指导、快速提升大家的:

- 开发水平
- 设计水平
- 架构水平

弥补业务中 CRUD 开发短板，帮助大家尽早脱离具备 3 高能力，掌握：

- 高性能
- 高并发
- 高可用

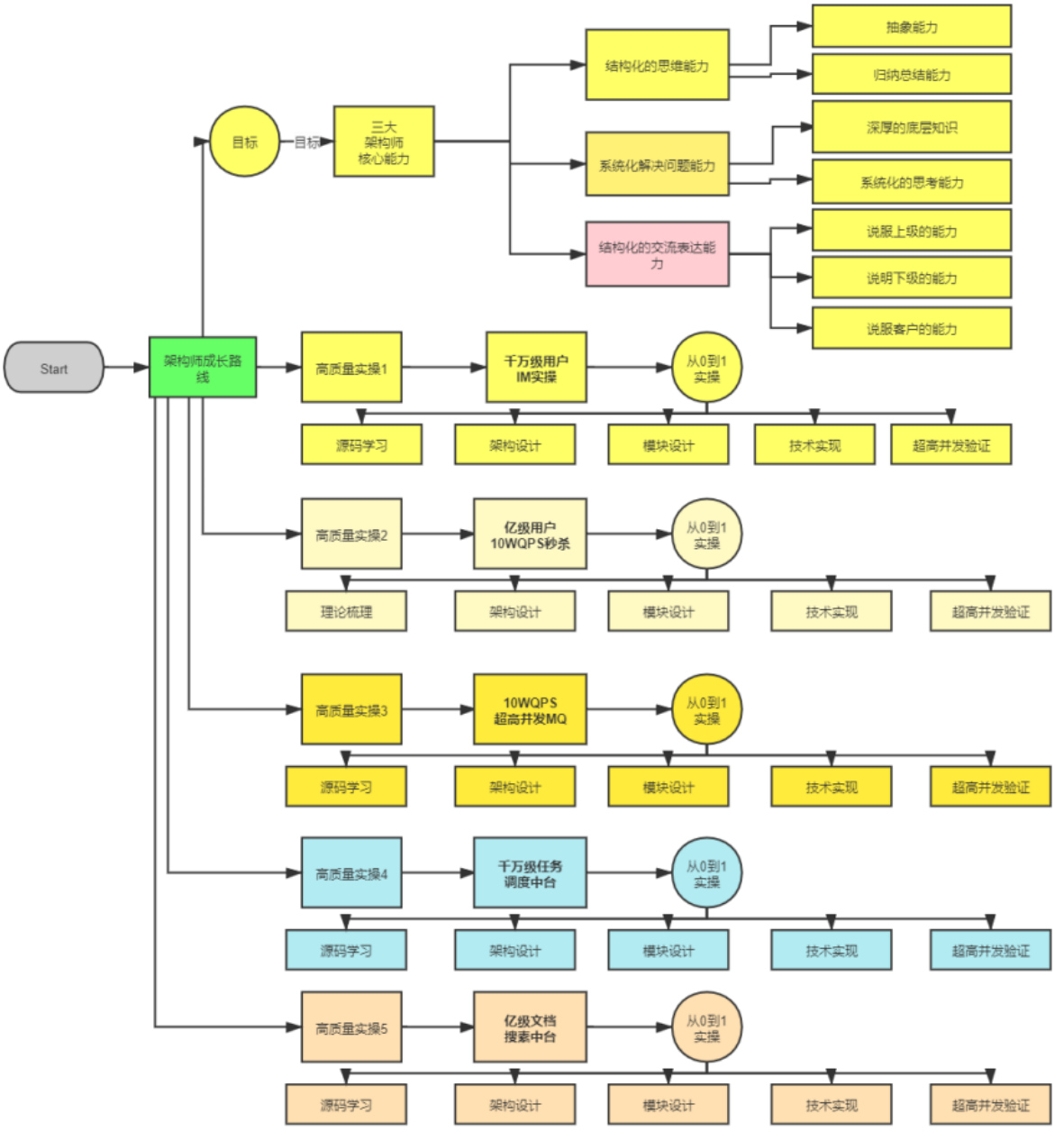
作为一个高质量的架构师成长、人脉社群，把所有的卷王聚焦起来，一起卷：

- 卷高并发实操
- 卷底层原理
- 卷架构理论、架构哲学
- 最终成为顶级架构师，实现人生理想，走向人生巅峰

架构班（社群 VIP）的目的：

- 高质量的实操，大大提升简历的含金量，吸引力，增强面试的召唤率
- 为大家提供九阳真经、葵花宝典，快速提升水平
- 进大厂、拿高薪
- 一路陪伴，提供助学视频和指导，辅导大家成为架构师
- 自学为主，和其他卷王一起，卷高并发实操，卷底层原理、卷大厂面试题，争取狠卷 3 月成高手，狠卷 3 年成为顶级架构师

N 个超高并发实操项目：简历压轴、个顶个精彩



【样章】第 17 章：横扫全网 Rocketmq 视频第 2 部曲：工业级 rocketmq 高可用（HA）底层原理和实操

工业级 rocketmq 高可用底层原理，包含：消息消费、同步消息、异步消息、单向消息等不同消息的底层原理和源码实现；消息队列非常底层的主从复制、高可用、同步刷盘、异步刷盘等底层原理。

工业级 rocketmq 高可用底层原理和搭建实操，包含：高可用集群的搭建。

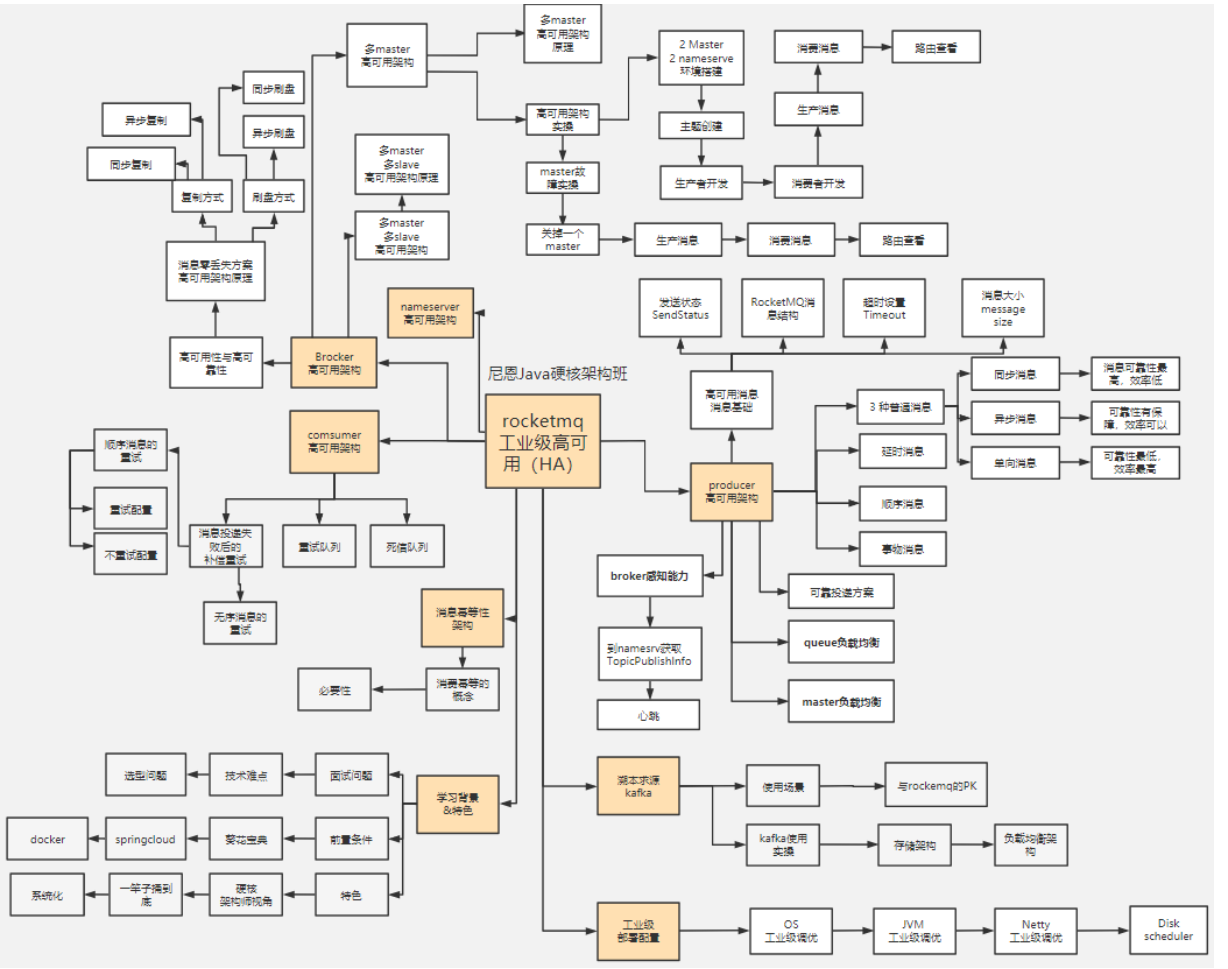
解决以下难题：

1、技术难题：RocketMQ 如何最大限度的保证消息不丢失的呢？RocketMQ 消息如何做到高可靠投递？

2、技术难题：基于消息的分布式事务，核心原理不理解

3、选型难题： kafka or rocketmq ，该娶谁？

下图链接：https://www.processon.com/view/6178e8ae0e3e7416bde9da19



简历优化后的成功涨薪案例（VIP含免费简历优化）


6年小伙收60W年薪
一月速提3大offer
4.24号改简历
5.27号报喜
VIP1239 6年
4月24日 晚上19:10
预期的岗位：60W
预期的岗位：go 和 java 的后端开
4月24日 晚上19:56
秘诀：
简历指导+ 狠狠卷
4月24日 晚上20:50
6年经验 1239
年薪 60W.
21.7 KB
微信电脑版
前几天改 ai，今天改 go
谢谢，我根据这个模式，再整理一下我简历，到时候老师再帮我把把关
ok
4月27日 下午14:37
老师，再帮我看看
准备开始投简历了
简历-6 年后端开发.pdf
210.1 KB
PDF
VIP1239 6年
嗯嗯，我先对着简历准备些东西，然后再开始投吧，反正现在刚刚好五一放假
上午8:01
来还愿啥，3周斩获3个offer，准备入职了
上午8:05
您太牛啦
简历优化后，面试机会太多了，拿完3个offer后，还有许多公司在流程中的都拒绝了
这几天大厂动作不断，联想，阿里都在裁员，您太牛啦
还是老师你强，offer中也达到了预期60w
非常不错
现在都上岸有offer就行，薪酬还能达到预期，已经超级牛掰啦
很多人，连一个面试电话都没有，崩溃的一塌糊涂
抖音上到处是这种
主要是简历优化后，感觉如有神助，每天基本3个面试，除了字节一面没过，其它都通过了
恭喜您
谢谢老师
后续有啥问题，可以找我支撑哈
好嘞，学习圈一直在，要持续提高自己
好的，撸起袖子加油卷，搞技术前途无限好
被裁后转架构，逆涨50%
8年小伙喜提年薪75W
4.16号改简历
5.6号报喜
VIP1236
最近面试了几个一轮游
捞了太多人上岸了
都是你这号
捞我
助力我 一个月时间
绞杀 下钻 打破瓶颈
咱们开始不？
好
4月16日 下午15:04
预期岗位：高级开发、架构
4月16日 下午15:12
预期薪酬：60W
-8 年高级服务端-0404 - 副本(2).
30.2 KB
微信电脑版
4月16日 下午16:19
秘诀：
简历指导+ 狠狠卷
4月16日 下午19:21
通话时长 03:02:25
辛苦老师
VIP1236
上午11:08
尼架，我决定要去上海了。
拿了几个 offer 了？
两个
上海这个是架构师对吧？还有一个呢？
还有个广州高级 Java，待遇 40w 左右，老板比较喜欢我，开了很多绿灯，薪资可以再加，但我还是想闯闯，昨天拒绝了。
年包多少呢？
就之前说的
那都快 80 个 W 了
我 argue 了下，他们控制内部薪酬平衡有点难办到 80，但已经是标出来的上限了。
都快是 高级 java 的 两倍
你撤回了一条消息
75 w 也非常多了，在现在的环境下
除开税，差不了多少，主要是这个方向的潜在价值
关键对你来说，这个是一个 成长机会
是的，寒意还是有的
您是我的贵人
是呢 有个外接大脑就是爽
好好卷，先祝贺您，拿到年薪 75W +
再预祝您，2 年之后，年薪 200W+
谢谢


9年 小伙伴拿到
年薪90W offer
9月11日改简历
11月29日晒offer
秘诀:
简历指导+ 狠狠卷
这个是给你微调的
省略号的地方，需要你再补充一点
上面你留着这个就行
Java 开发 9年-修改.docx
34.1 KB
薪资高 稳
主要的工作是啥?
提升了自己的实力，就不用怕
易所 关于数字货 的 易
po 也有 打盹的时候，该裁员，照样一个不少
年包比 po 多 19w
这么多
po 估计有 70 万
那你不有 90 个 W?
po 给我 68
就是吗


小伙8年经验
年薪60w
7月12日改简历
8月10日晒offer
秘诀:
改简历 + 狠狠卷
明天晚上哈
好哒
恩哥，今晚还改简历吗?
今晚还在外边应酬，估计回去比较晚
要不，咱们延迟到明天，如何
明天白天也行
好的，白天吧，答应别人明天给他们简历了。
OK
那就上午 11 点左右哈
好的
涨幅多少呀?
或者，额度多少呀
之前 36*15，现在这个 39*15
今年行情不太好，还有一些 offer 基本都是平薪，或降薪的。
OK
这个马上来
刚在指导简历
哈哈
稍等哈
辛苦啊恩哥
这次找工作，您的 起到效果了。
我这次复习基本看的都是咱们课程的 。


6年小伙伴
年薪40w
9月6日改简历
9月21日晒offer
秘诀:
简历指导+ 狠狠卷
Java - 6年.docx
24.7 KB
恩哥，简历我改好了，您再帮我看一下
Java - 6年(1).docx
24.5 KB
恩哥，看第二个吧
给你前面调整了一下
涨这多
今年这行情，也算可以了
总包多少呀，让我也陶醉一下
大概 40w 吧
谢谢恩哥的指导和鼓励
是在深圳
深圳的行情尤其难
能有面试电话就不错啦
是的
太牛啦
哈哈哈哈，恩哥的鼓励指导也很重要


5年小伙喜提3个offer
年薪 35个W
5月22日改简历
11月29日晒offer
我简历还没写出来 不然可以给老师您看看
算法方面，可以找点 刷刷，当作复习
项目的话，再好好改造一下
好的 谢谢老师
java 开发-本科统招-2022-5 年经验.docx
36.0 KB
OK
我排好号，回头找你哈
秘诀:
简历指导+ 狠狠卷
尼恩老师晚上好，汇报下最近的 offer 情况。最近面试收到了三个 offer，两个是平薪，一个是跨境电商公司的 offer，涨幅暂时不到 20%。通过这次面试也让我知道自己距离高级开发还有一点点距离，还要再多看才能突破。
最后结合自己的情况，先选择去跨境电商的公司再提升下
之前是 20k*13.5，跨境电商这个是说 23k*（14-15）
恭喜恭喜
就业寒冬，能提到 3 个 offer，已经很 ，已经是高手了
很多小伙伴，面试电话一个都接不到，简历海投 7000 份，只收到 3 次面试机会，没有一个机会拿到最终 offer
您这个年薪，算下来也有 35W 了吧
年终部分还要确认下，大部分人听说只有两个月
这个时间点，拿到这个水平，挺不错的啦
持续加油卷哈
嗯嗯！看看明年自己有没有能力冲击高开
把你我视频都吃透
辛苦老师再帮忙指导下哈


1.5年小伙搞定15K offer
就业寒冬涨100%
5月7日改简历
11月21日晒offer
秘诀：
简历指导+狠狠卷
多谢啦
你好，好多公司都觉得我的简历太简单了，我感觉是不是没法优化了
简历.pdf
397.4 KB
好的
晚上指导你改
推送中台需要加不
我还没做啊，准备八股文暂时没时间搞这些，入职了会开始搞
OK
还有其他的项目吗
一个项目，太少啦
那种练习项目也行
他那个不止两倍吧，我才是两倍，原先7k，现在15k
那也很牛
都翻倍了，都是牛人
正常就30%
慢慢来
下一步可以瞄准25k
其实我工作一年
那你就更牛逼拉
一年经验，搞定15k offer，舍你其谁


卷王逆袭成功案例
6年小伙从很少面试机会到
搞定35K*14薪
3月5日改简历
4月11日拿offer
面试邀请少
小伙很苦恼
理想很丰满
目标35K
一个月拿到了
理想的offer
面试法宝
rocketmq四部曲

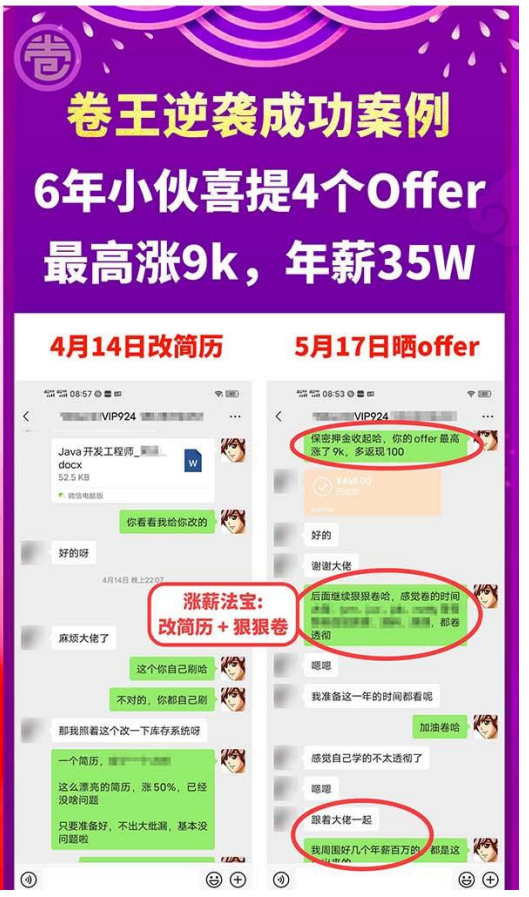

6年 经验小伙伴
喜收25K offer
3月12日改简历
12月1日晒offer
秘诀：
简历指导+狠狠卷
老师
咱们以这个项目为模板改哈
嗯好
项目有多少人，你带领多少人？
你工作几年了？
6年了
项目有多少人，你带领多少人？
语音交流一下
好
你方便吗
OK
不知道往哪个方向发展
改简历挪窝，20 涨到 25 k，高级开发，P7 带的后端团队
任务重，道路远，并不是没有方向


7年经验卷王
薪酬涨30%
7月11日改简历
9月1日晒offer
秘诀：
改简历 + 狠狠卷
咱们开始修改哈
要语音还是打字也行，还没到家
解决过最复杂的难题是啥？说 3 个哈
看看，有没有问题？
没问题
看看，有没有问题？
只能挪窝
下次你喊足一点哦
只要本事好，拿 offer 比较容易的
喊了，但是我流水太差了。。。我这家是补贴，银行不认这玩意，只能在上家的基础上加到 30% 封顶（特别优秀除外，我只是不错。。。）
入职了，最终并不能大幅涨薪，还是旧的太低，只涨了 30%
恭喜一下
现在不比往年
能涨 30% 是大牛
拿到 offer 都算大牛
现在很多人投几百发，连个电话都没有
你已经很


4年经验卷王逆袭
被毕业后，反涨24W
7月改简历
8月30日晒offer
秘诀:
改简历 + 狠狠卷
这就是你的简历
差得太多啦
ok
总共写了四个项目，最近的一年的还没补充上
你是在职，还是离职呢?
离职
原因大概是啥?
这个地方要这样写
项目被终止
方便语音沟通不
ok
是的 经你这么指导，非常重要
老哥 我八月十号开始找工作，今天已经入职了
现金基本持平，股票 +24W
总计涨了多少呢
能涨24W
股票这个吧只能到手了才算
也不错啦
很多小伙伴，面试机会都没有
感谢老哥的指导，继续跟你卷技术
继续狠狠卷哈，马上就技术自由拉

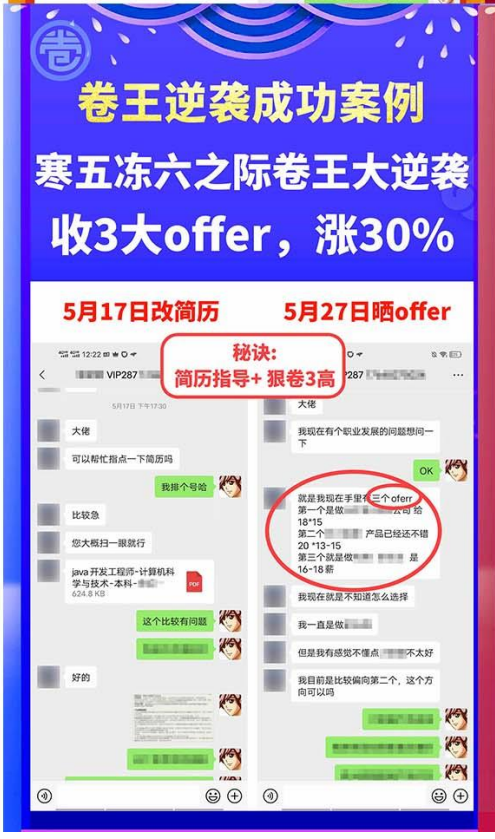

小伙5月份"被毕业"，改简历后
斩获顶级央企Offer
涨薪7000+
5月29日改简历
7月5日晒offer
秘诀:
简历指导+ 狠卷3高
快速看书，就要不求甚解，把目录和 场景大概一下，然后重点的地方，用到的地方，再去回炉
尼恩 我被“ 毕业 ”了
这周末或下周我你改一下简历
毕业了没有关系
ok，发我吧
it行业，跳来跳去，太频繁啦
嗯，其实有点心理准备
简历
不太会写简历
尼恩 我拿到的offer了
涨20%，不还价的
看起来不差钱呀
超过了8000没
平均算下来
7000多
好的
有啥面试的心得吗
可以分享给其他小伙伴的


卷王逆袭成功案例
武汉6年喜收4个优质offer
最高的年薪35W
2月9日改简历
4月15日晒offer
尼恩老师，新年好!
能帮忙修改下简历吗?
金三银四准备跳了
可以的
java 开发-6年-简历-.pdf
拜托了，恩师。希望能拿 25k 回来给你报喜
好，我加一下
还有吗?
面试法宝:
改简历 + 实操
恩哥，决赛圈 offer 了
截图是以我目前认知能写出来的评分了，麻烦帮我参考下
选择大于努力，恩师助我上岸
这么多 offer，我看看哈
都是恩哥指点有方，本来还有个新能源汽车的，35w给拒了，主要太远了
跟着恩哥卷的时间太短了，目前实力也只能到这儿了
这里边有个大数据的，感觉也不错

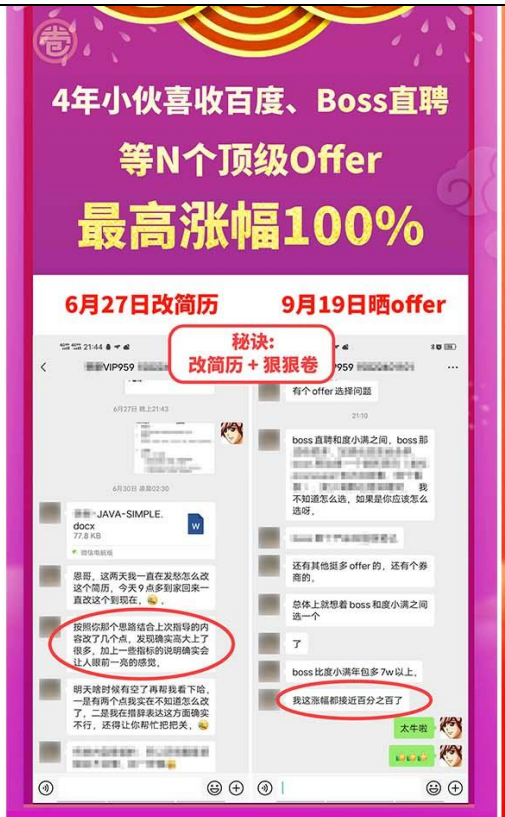

卷王逆袭成功案例
6年小伙喜提4个Offer
最高涨9k，年薪35W
4月14日改简历
5月17日晒offer
Java 开发工程师_.docx
你看看我给你改的
好的呀
麻烦大佬了
这个你自己刷哈
不对的，你都自己刷
那我照着这个改一下库存系统呀
一个简历，
这么漂亮的简历，涨 50%，已经没啥问题
只要准备好，不出大纰漏，基本没问题啦
涨薪法宝:
改简历 + 狠狠卷
保密押金收起哈，你的 offer 最高涨了 9k，多返现 100
好的
谢谢大佬
后面继续狠狠卷哈，感觉卷的时间
透彻
嗯嗯
我准备这一年的时间都看呢
加油卷哈
感觉自己学的不太透彻了
嗯嗯
跟着大佬一起
我周围好几个年薪百万的，都是这


卷王逆袭成功案例
5年经验小伙收2个offer
最高涨薪8k，年薪42W
5月9日改简历
5月30日晒offer
秘诀:
简历指导+ 狠卷3高
以此为样
大家狠狠卷
打造最卷IT社群


卷王逆袭成功案例
非全日制 6年经验卷王
喜提3个Offer，年包30W
5月9日改简历
5月18日晒offer
面试法宝:
改简历 + 狠狠卷


卷王逆袭成功案例
寒五冻六之际卷王大逆袭
收3大offer，涨30%
5月17日改简历
5月27日晒offer
秘诀:
简历指导+ 狠卷3高


卷王逆袭成功案例
4年卷王入职微软，涨50%
3月7日改简历
5月12日晒offer
涨薪法宝:
改简历 + 狠狠卷


4年小伙喜收百度、Boss直聘
等N个顶级Offer
最高涨幅100%
6月27日改简历
9月19日晒offer
秘诀:
改简历＋狠狠卷


卷王逆袭成功案例
4年卷王入收2个offer，涨50%
3月23日改简历
5月12日晒offer
涨薪法宝:
改简历＋狠狠卷


小伙大三暑期很焦虑
跟着尼恩卷一年
校招斩获顶级央企Offer
去年5月19日加入VIP群
今年7月5日晒offer
秘诀:
狠狠卷 书+视频


小伙高中学历
薪酬涨120%
5月6日改简历
7月22日晒offer
秘诀:
改简历＋狠狠卷


卷王逆袭成功案例
非全日制卷王 面试3家
收2个offer 涨薪30%
4月13日改简历
4月21日晒offer
面试法宝：
改简历 + 面试题


5年卷王喜收2大Offer
最高涨5K
5月19日改简历
9月13日晒offer
秘诀：
改简历 + 狠狠卷


卷王逆袭成功案例
3年经验卷王，涨60%
4月16日改简历
5月11日晒offer
涨薪法宝：
改简历 + 狠狠卷


卷王逆袭成功案例
双非二本小伙春招大翻身
喜提9大offer
2月22日改简历
4月13日晒offer
面试法宝：
改简历 + IM实操
9大offer
最高年薪30万

修改简历找尼恩（资深简历优化专家）

- 如果面试表达不好，尼恩会提供 简历优化指导
- 如果项目没有亮点，尼恩会提供 项目亮点指导
- 如果面试表达不好，尼恩会提供 面试表达指导

作为 40 岁老架构师，尼恩长期承担技术面试官的角色：

- **从业以来，“阅历”无数，对简历有着点石成金、改头换面、脱胎换骨**的指导能力。
- **尼恩指导过刚刚就业的小白，也指导过 P8 级的老专家，都指导他们上岸。**

如何联系尼恩。尼恩微信，请参考下面的地址：

语雀：https://www.yuque.com/crazymakercircle/gkkw8s/khigna
码云：https://gitee.com/crazymaker/SimpleCrayIM/blob/master/疯狂创客圈总目录.md