不定期福利第三期 | 测一测你的算法阶段学习成果

time.geekbang.org/column/article/73786

专栏最重要的基础篇马上就要讲完了，不知道你掌握了多少？我从前面的文章中挑选了一些案例，稍加修改，组成了一套测试题。

你先不要着急看答案，自己先想一想怎么解决，测一测自己对之前的知识掌握的程度。如果有哪里卡壳或者不怎么清楚的，可以回过头再复习一下。

正所谓温故知新，这种通过实际问题查缺补漏的学习方法，非常利于你巩固前面讲的知识点，你可要好好珍惜这次机会哦！

实战测试题（一）

假设猎聘网有 10 万名猎头顾问，每个猎头顾问都可以通过做任务（比如发布职位），来积累积分，然后通过积分来下载简历。**假设你是猎聘网的一名工程师，如何在内存中存储这 10 万个猎头 ID 和积分信息，让它能够支持这样几个操作：**

- 根据猎头的 ID 快速查找、删除、更新这个猎头的积分信息；
- 查找积分在某个区间的猎头 ID 列表；
- 查询积分从小到大排在第 x 位的猎头 ID 信息；
- 查找按照积分从小到大排名在第 x 位到第 y 位之间的猎头 ID 列表。

相关章节

17 | 跳表：为什么 Redis 一定要用跳表来实现有序集合？

20 | 散列表（下）：为什么散列表和链表经常会一起使用？

25 | 红黑树：为什么工程中都用红黑树这种二叉树？

题目解析

这个问题既要通过 ID 来查询，又要通过积分来查询，所以，对于猎头这样一个对象，我们需要将其组织成两种数据结构，才能支持这两类操作。

我们按照 ID，将猎头信息组织成散列表。这样，就可以根据 ID 信息快速的查找、删除、更新猎头的信息。我们按照积分，将猎头信息组织成跳表这种数据结构，按照积分来查找猎头信息，就非常高效，时间复杂度是 O(logn)。

我刚刚讲的是针对第一个、第二个操作的解决方案。第三个、第四个操作是类似的，按照排名来查询，这两个操作该如何实现呢？

我们可以对刚刚的跳表进行改造，每个索引结点中加入一个 span 字段，记录这个索引结点到下一个索引结点的包含的链表结点的个数。这样就可以利用跳表索引，快速计算出排名在某一位的猎头或者排名在某个区间的猎头列表。

实际上，这些就是 Redis 中有序集合这种数据类型的实现原理。在开发中，我们并不需要从零开始代码实现一个散列表和跳表，我们可以直接利用 Redis 的有序集合来完成。

实战测试题（二）

电商交易系统中，订单数据一般都会很大，我们一般都分库分表来存储。假设我们分了 10 个库并存储在不同的机器上，在不引入复杂的分库分表中间件的情况下，我们希望开发一个小的功能，能够快速地查询金额最大的前 K 个订单（K 是输入参数，可能是 1、10、1000、10000，假设最大不会

超过 10 万）。**如果你是这个功能的设计开发负责人，你会如何设计一个比较详细的、可以落地执行的设计方案呢？**

为了方便你设计，我先交代一些必要的背景，在设计过程中，如果有其他需要明确的背景，你可以自行假设。

- 数据库中，订单表的金额字段上建有索引，我们可以通过 select order by limit 语句来获取数据库中的数据；
- 我们的机器的可用内存有限，比如只有几百 M 剩余可用内存。希望你的设计尽量节省内存，不要发生 Out of Memory Error。

相关章节

12 | 排序（下）：如何用快排思想在 O(n) 内查找第 K 大元素？

28 | 堆和堆排序：为什么说堆排序没有快速排序快？

29 | 堆的应用：如何快速获取到 Top 10 最热门的搜索关键词？

题目解析

解决这个题目的基本思路我想你应该能想到，就是借助归并排序中的合并函数，这个我们在排序（下）以及堆的应用那一节中讲过。

我们从每个数据库中，通过 select order by limit 语句，各取局部金额最大的订单，把取出来的 10 个订单放到优先级队列中，取出最大值（也就是大顶堆堆顶数据），就是全局金额最大的订单。然后再从这个全局金额最大订单对应的数据库中，取出下一条订单（按照订单金额从大到小排列的），然后放到优先级队列中。一直重复上面的过程，直到找到金额前 K（K 是用户输入的）大订单。

从算法的角度看起来，这个方案非常完美，但是，从实战的角度来说，这个方案并不高效，甚至很低效。因为我们忽略了，数据库读取数据的性能才是这个问题的性能瓶颈。所以，我们要尽量减少 SQL 请求，每次多取一些数据出来，那一次性取出多少才合适呢？这就比较灵活、比较有技巧了。一次性取太多，会导致数据量太大，SQL 执行很慢，还有可能触发超时，而且，我们题目中也说了，内存有限，太多的数据加载到内存中，还有可能导致 Out of Memory Error。

所以，一次性不能取太多数据，也不能取太少数据，到底是多少，还要根据实际的硬件环境做 benchmark 测试去找最合适的。

实战测试题（三）

我们知道，CPU 资源是有限的，任务的处理速度与线程个数并不是线性正相关。相反，过多的线程反而会导致 CPU 频繁切换，处理性能下降。所以，线程池的大小一般都是综合考虑要处理任务的特点和硬件环境，来事先设置的。

当我们向固定大小的线程池中请求一个线程时，如果线程池中没有空闲资源了，这个时候线程池如何处理这个请求？是拒绝请求还是排队请求？各种处理策略又是怎么实现的呢？

相关章节

09 | 队列：队列在线程池等有限资源池中的应用

题目解析

这个问题的答案涉及队列这种数据结构。队列可以应用在任何有限资源池中，用于排队请求，比如数据库连接池等。实际上，对于大部分资源有限的场景，当没有空闲资源时，基本上都可以通过“队列”这种数据结构来实现请求排队。

这个问题的具体答案，在队列那一节我已经讲得非常详细了，你可以回去看看，这里我就不赘述了。

实战测试题（四）

通过 IP 地址来查找 IP 归属地的功能，不知道你有没有用过？没用过也没关系，你现在可以打开百度，在搜索框里随便输一个 IP 地址，就会看到它的归属地。

这个功能并不复杂，它是通过维护一个很大的 IP 地址库来实现的。地址库中包括 IP 地址范围和归属地的对应关系。比如，当我们想要查询 202.102.133.13 这个 IP 地址的归属地时，我们就在地址库中搜索，发现这个 IP 地址落在 [202.102.133.0, 202.102.133.255] 这个地址范围内，那我们就可以将这个 IP 地址范围对应的归属地“山东东营市”显示给用户了。

```
[202.102.133.0, 202.102.133.255] 山东东营市
[202.102.135.0, 202.102.136.255] 山东烟台
[202.102.156.34, 202.102.157.255] 山东青岛
[202.102.48.0, 202.102.48.255] 江苏宿迁
[202.102.49.15, 202.102.51.251] 江苏泰州
[202.102.56.0, 202.102.56.255] 江苏连云港
```


在庞大的地址库中逐一比对 IP 地址所在的区间，是非常耗时的。**假设在内存中有 12 万条这样的 IP 区间与归属地的对应关系，如何快速定位出一个 IP 地址的归属地呢？**

相关章节

15 | 二分查找（上）：如何用最省内存的方式实现快速查找功能？

16 | 二分查找（下）：如何快速定位 IP 对应的省份地址？

题目解析

这个问题可以用二分查找来解决，不过，普通的二分查找是不行的，我们需要用到二分查找的变形算法，查找最后一个小于等于某个给定值的数据。不过，二分查找最难的不是原理，而是实现。要实现一个二分查找的变形算法，并且实现的代码没有 bug，可不是一件容易的事情，不信你自己写写试试。

关于这个问题的解答以及写出 bug free 的二分查找代码的技巧，我们在二分查找（下）那一节有非常详细的讲解，你可以回去看看，我这里就不赘述了。

实战测试题（五）

假设我们现在希望设计一个简单的海量图片存储系统，最大预期能够存储 1 亿张图片，并且希望这个海量图片存储系统具有下面这样几个功能：

- 存储一张图片及其它的元信息，主要的元信息有：图片名称以及一组 tag 信息。比如图片名称叫玫瑰花，tag 信息是{红色，花，情人节}；
- 根据关键词搜索一张图片，比如关键词是“情人节 花”“玫瑰花”；
- 避免重复插入相同的图片。这里，我们不能单纯地用图片的元信息，来比对是否是同一张图片，因为有可能存在名称相同但图片内容不同，或者名称不同图片内容相同的情况。

我们希望自助开发一个简单的系统，不希望借助和维护过于复杂的三方系统，比如数据库（MySQL、Redis 等）、分布式存储系统（GFS、Bigtable 等），并且我们单台机器的性能有限，比如硬盘只有 1TB，内存只有 2GB，**如何设计一个符合我们上面要求，操作高效，且使用机器资源最少的存储系统呢？**

相关章节

21 | 哈希算法（上）：如何防止数据库中的用户信息被脱库？

22 | 哈希算法（下）：哈希算法在分布式系统中有哪些应用？

题目解析

这个问题可以分成两部分，第一部分是根据元信息的搜索功能，第二部分是图片判重。

第一部分，我们可以借助搜索引擎中的倒排索引结构。关于倒排索引我会在实战篇详细讲解，我这里先简要说下。

如题目中所说，一个图片会对应一组元信息，比如玫瑰花对应{红色，花，情人节}，牡丹花对应{白色，花}，我们可以将这种图片与元信息之间的关系，倒置过来建立索引。“花”这个关键词对应{玫瑰花，牡丹花}，“红色”对应{玫瑰花}，“白色”对应{牡丹花}，“情人节”对应{玫瑰花}。

当我们搜索“情人节 花”的时候，我们拿两个搜索关键词分别在倒排索引中查找，“花”查找到了{玫瑰花，牡丹花}，“情人节”查找到了{玫瑰花}，两个关键词对应的结果取交集，就是最终的结果了。

第二部分关于图片判重，我们要基于图片本身来判重，所以可以用哈希算法，对图片内容取哈希值。我们对哈希值建立散列表，这样就可以通过哈希值以及散列表，快速判断图片是否存在。

我这里只说说我的思路，这个问题中还有详细的内存和硬盘的限制。要想给出更加详细的设计思路，还需要根据这些限制，给出一个估算。详细的解答，我都放在在哈希算法（下）那一节里到了，你可以自己回去看。

实战测试题（六）

我们知道，散列表的查询效率并不能笼统地说成是 O(1)。它跟散列函数、装载因子、散列冲突等都有关系。如果散列函数设计得不好，或者装载因子过高，都可能导致散列冲突发生的概率升高，查询效率下降。

在极端情况下，有些恶意的攻击者，还有可能通过精心构造的数据，使得所有的数据经过散列函数之后，都散列到同一个槽里。如果我们使用的是基于链表的冲突解决方法，那这个时候，散列表就会退化为链表，查询的时间复杂度就从 O(1) 急剧退化为 O(n)。

如果散列表中有 10 万个数据，退化后的散列表查询的效率就下降了 10 万倍。更直观点说，如果之前运行 100 次查询只需要 0.1 秒，那现在就需要 1 万秒。这样就有可能因为查询操作消耗大量 CPU 或者线程资源，导致系统无法响应其他请求，从而达到拒绝服务攻击（DoS）的目的。这也就是散列表碰撞攻击的基本原理。

如何设计一个可以应对各种异常情况的工业级散列表，来避免在散列冲突的情况下，散列表性能的急剧下降，并且能抵抗散列碰撞攻击？

相关章节

18 | 散列表（上）：Word 文档中的单词拼写检查功能是如何实现的？

19 | 散列表（中）：如何打造一个工业级水平的散列表？

题目解析

我经常把这道题拿来作为面试题考察候选人。散列表可以说是我们最常用的一种数据结构了，编程语言中很多数据类型，都是用散列表来实现的。尽管很多人能对散列表都知道一二，知道有几种散列表冲突解决方案，知道散列表操作的时间复杂度，但是理论跟实践还是有一定距离的。光知道这些基础的理论并不足以开发一个工业级的散列表。

所以，我在散列表（中）那一节中详细给你展示了一个工业级的散列表要处理哪些问题，以及如何处理的，也就是这个问题的详细答案。

这六道题你回答得怎么样呢？或许你还无法 100% 回答正确，没关系。其实只要你看了解析之后，有比较深的印象，能立马想到哪节课里讲过，这已经说明你掌握得不错了。毕竟想要完全掌握我讲的全部内容还是需要时间沉淀的。对于这门课的学习，你一定不要心急，慢慢来。只要方向对了就都对了，剩下就交给时间和努力吧！

通过这套题，你对自己的学习状况应该有了一个了解。从专栏开始到现在，三个月过去了，我们的内容也更新了大半。**你在专栏开始的时候设定的目标是什么？现在实施得如何了？你可以在留言区给这三个月的学习做个阶段性学习复盘**。重新整理，继续出发！


极客时间
数据结构与算法之美
为工程师量身打造的数据结构与算法私教课
王争
前 Google 工程师
新版升级：点击「 请朋友读 」，10位好友免费读，邀请订阅更有现金奖励。