

讲堂 > 数据结构与算法之美 > 文章详情

29 | 堆的应用：如何快速获取到Top 10最热门的搜索关键词？

2018-11-28 王争



29 | 堆的应用：如何快速获取到Top 10最热门的搜索关键词？

朗读人：修阳 18'55" | 8.67M

搜索引擎的热门搜索排行榜功能你用过吗？你知道这个功能是如何实现的吗？实际上，它的实现并不复杂。搜索引擎每天会接收大量的用户搜索请求，它会把这些用户输入的搜索关键词记录下来，然后再离线地统计分析，得到最热门的 Top 10 搜索关键词。

那请你思考下，**假设现在我们有一个包含 10 亿个搜索关键词的日志文件，如何能快速获取到热门榜 Top 10 的搜索关键词呢？**

这个问题就可以用堆来解决，这也是堆这种数据结构一个非常典型的应用。上一节我们讲了堆和堆排序的一些理论知识，今天我们就来讲一讲，堆这种数据结构几个非常重要的应用：优先级队列、求 Top K 和求中位数。

堆的应用一：优先级队列

首先，我们来看第一个应用场景：优先级队列。

优先级队列，顾名思义，它首先应该是一个队列。我们前面讲过，队列最大的特性就是先进先出。不过，在优先级队列中，数据的出队顺序不是先进先出，而是按照优先级来，优先级最高的，最先出队。

如何实现一个优先级队列呢？方法有很多，但是用堆来实现是最直接、最高效的。这是因为，堆和优先级队列非常相似。一个堆就可以看作一个优先级队列。很多时候，它们只是概念上的区分而已。往优先级队列中插入一个元素，就相当于往堆中插入一个元素；从优先级队列中取出优先级最高的元素，就相当于取出堆顶元素。

你可别小看这个优先级队列，它的应用场景非常多。我们后面要讲的很多数据结构和算法都要依赖它。比如，赫夫曼编码、图的最短路径、最小生成树算法等等。不仅如此，很多语言中，都提供了优先级队列的实现，比如，Java 的 `PriorityQueue`，C++ 的 `priority_queue` 等。

只讲这些应用场景比较空泛，现在，我举两个具体的例子，让你感受一下优先级队列具体是怎么用的。

1. 合并有序小文件

假设我们有 100 个小文件，每个文件的大小是 100MB，每个文件中存储的都是有序的字符串。我们希望将这些 100 个小文件合并成一个有序的大文件。这里就会用到优先级队列。

整体思路有点像归并排序中的合并函数。我们从这 100 个文件中，各取第一个字符串，放入数组中，然后比较大小，把最小的那个字符串放入合并后的大文件中，并从数组中删除。

假设，这个最小的字符串来自于 13.txt 这个小文件，我们就再从这个文件取下一个字符串，并且放到数组中，重新比较大小，并且选择最小的放入合并后的大文件，并且将它从数组中删除。依次类推，直到所有的文件中的数据都放入到大文件为止。

这里我们用数组这种数据结构，来存储从小文件中取出来的字符串。每次从数组中取最小字符串，都需要循环遍历整个数组，显然，这不是很高效。有没有更加高效方法呢？

这里就可以用到优先级队列，也可以说是堆。我们将从小文件中取出来的字符串放入到小顶堆中，那堆顶的元素，也就是优先级队列队首的元素，就是最小的字符串。我们将这个字符串放入到大文件中，并将其从堆中删除。然后再从小文件中取出下一个字符串，放入到堆中。循环这个过程，就可以将 100 个小文件中的数据依次放入到大文件中。

我们知道，删除堆顶数据和往堆中插入数据的时间复杂度都是 $O(\log n)$ ， n 表示堆中的数据个数，这里就是 100。是不是比原来数组存储的方式高效了很多呢？

2. 高性能定时器

假设我们有一个定时器，定时器中维护了很多定时任务，每个任务都设定了一个要触发执行的时间点。定时器每过一个很小的单位时间（比如 1 秒），就扫描一遍任务，看是否有任务到达设

定的执行时间。如果到达了，就拿出来执行。

2018.11.28.17:30	Task A
2018.11.28.19:20	Task B
2018.11.28.15:31	Task C
2018.11.28.13:55	Task D

但是，这样每过 1 秒就扫描一遍任务列表的做法比较低效，主要原因有两点：第一，任务的约定执行时间离当前时间可能还有很久，这样前面很多次扫描其实都是徒劳的；第二，每次都要扫描整个任务列表，如果任务列表很大的话，势必会比较耗时。

针对这些问题，我们就可以用优先级队列来解决。我们按照任务设定的执行时间，将这些任务存储在优先级队列中，队列首部（也就是小顶堆的堆顶）存储的是最先执行的任务。

这样，定时器就不需要每隔 1 秒就扫描一遍任务列表了。它拿队首任务的执行时间点，与当前时间点相减，得到一个时间间隔 T 。

这个时间间隔 T 就是，从当前时间开始，需要等待多久，才会有第一个任务需要被执行。这样，定时器就可以设定在 T 秒之后，再来执行任务。从当前时间点到 $(T-1)$ 秒这段时间里，定时器都不需要做任何事情。

当 T 秒时间过去之后，定时器取优先级队列中队首的任务执行。然后再计算新的队首任务的执行时间点与当前时间点的差值，把这个值作为定时器执行下一个任务需要等待的时间。

这样，定时器既不用间隔 1 秒就轮询一次，也不用遍历整个任务列表，性能也就提高了。

堆的应用二：利用堆求 Top K

刚刚我们学习了优先级队列，我们现在来看，堆的另外一个非常重要的应用场景，那就是“求 Top K 问题”。

我把这种求 Top K 的问题抽象成两类。一类是针对静态数据集合，也就是说数据集合事先确定，不会再变。另一类是针对动态数据集合，也就是说数据集合事先并不确定，有数据动态地加入到集合中。

针对静态数据，如何在一个包含 n 个数据的数组中，查找前 K 大数据呢？我们可以维护一个大小为 K 的小顶堆，顺序遍历数组，从数组中取出数据与堆顶元素比较。如果比堆顶元素大，我们就把堆顶元素删除，并且将这个元素插入到堆中；如果比堆顶元素小，则不做处理，继续遍历数组。这样等数组中的数据都遍历完之后，堆中的数据就是前 K 大数据了。

遍历数组需要 $O(n)$ 的时间复杂度，一次堆化操作需要 $O(\log K)$ 的时间复杂度，所以最坏情况下， n 个元素都入堆一次，所以时间复杂度就是 $O(n\log K)$ 。

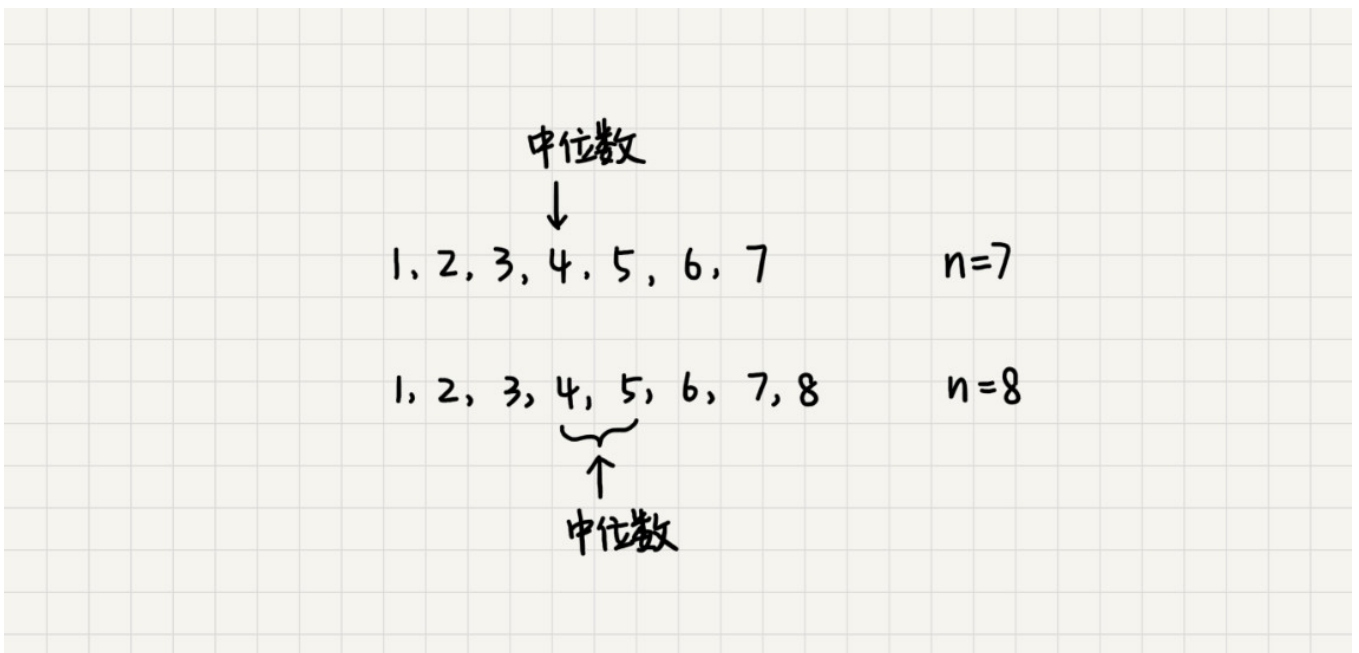
针对动态数据求得 Top K 就是实时 Top K 。怎么理解呢？我举一个例子。一个数据集中有两个操作，一个是添加数据，另一个询问当前的前 K 大数据。

如果每次询问前 K 大数据，我们都基于当前的数据重新计算的话，那时间复杂度就是 $O(n\log K)$ ， n 表示当前的数据的大小。实际上，我们可以一直都维护一个 K 大小的小顶堆，当有数据被添加到集合中时，我们就拿它与堆顶的元素对比。如果比堆顶元素大，我们就把堆顶元素删除，并且将这个元素插入到堆中；如果比堆顶元素小，则不做处理。这样，无论任何时候需要查询当前的前 K 大数据，我们都可以立刻返回给他。

堆的应用三：利用堆求中位数

前面我们讲了如何求 Top K 的问题，现在我们来讲下，如何求动态数据集中的中位数。

中位数，顾名思义，就是处在中间位置的那个数。如果数据的个数是奇数，把数据从小到大排列，那第 $\frac{n}{2} + 1$ 个数据就是中位数；如果数据的个数是偶数的话，那处于中间位置的数据有两个，第 $\frac{n}{2}$ 个和第 $\frac{n}{2} + 1$ 个数据，这个时候，我们可以随意取一个作为中位数，比如取两个数中靠前的那个，就是第 $\frac{n}{2}$ 个数据。



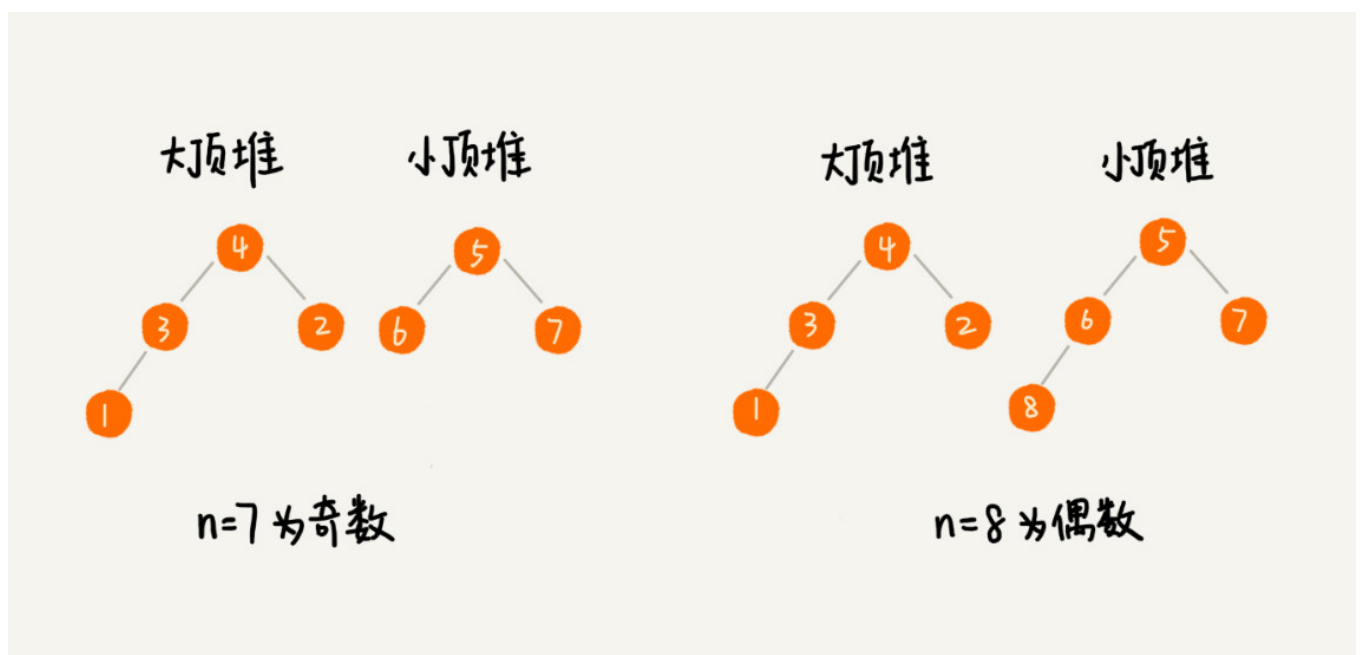
对于一组**静态数据**，中位数是固定的，我们可以先排序，第 $\frac{n}{2}$ 个数据就是中位数。每次询问中位数的时候，我们直接返回这个固定的值就好了。所以，尽管排序的代价比较大，但是边际成本

会很小。但是，如果我们面对的是**动态数据**集合，中位数在不停地变动，如果再用先排序的方法，每次询问中位数的时候，都要先进行排序，那效率就不高了。

借助堆这种数据结构，我们不用排序，就可以非常高效地实现求中位数操作。我们来看看，它是如何做到的？

我们需要维护两个堆，一个大顶堆，一个小顶堆。大顶堆中存储前半部分数据，小顶堆中存储后半部分数据，且小顶堆中的数据都大于大顶堆中的数据。

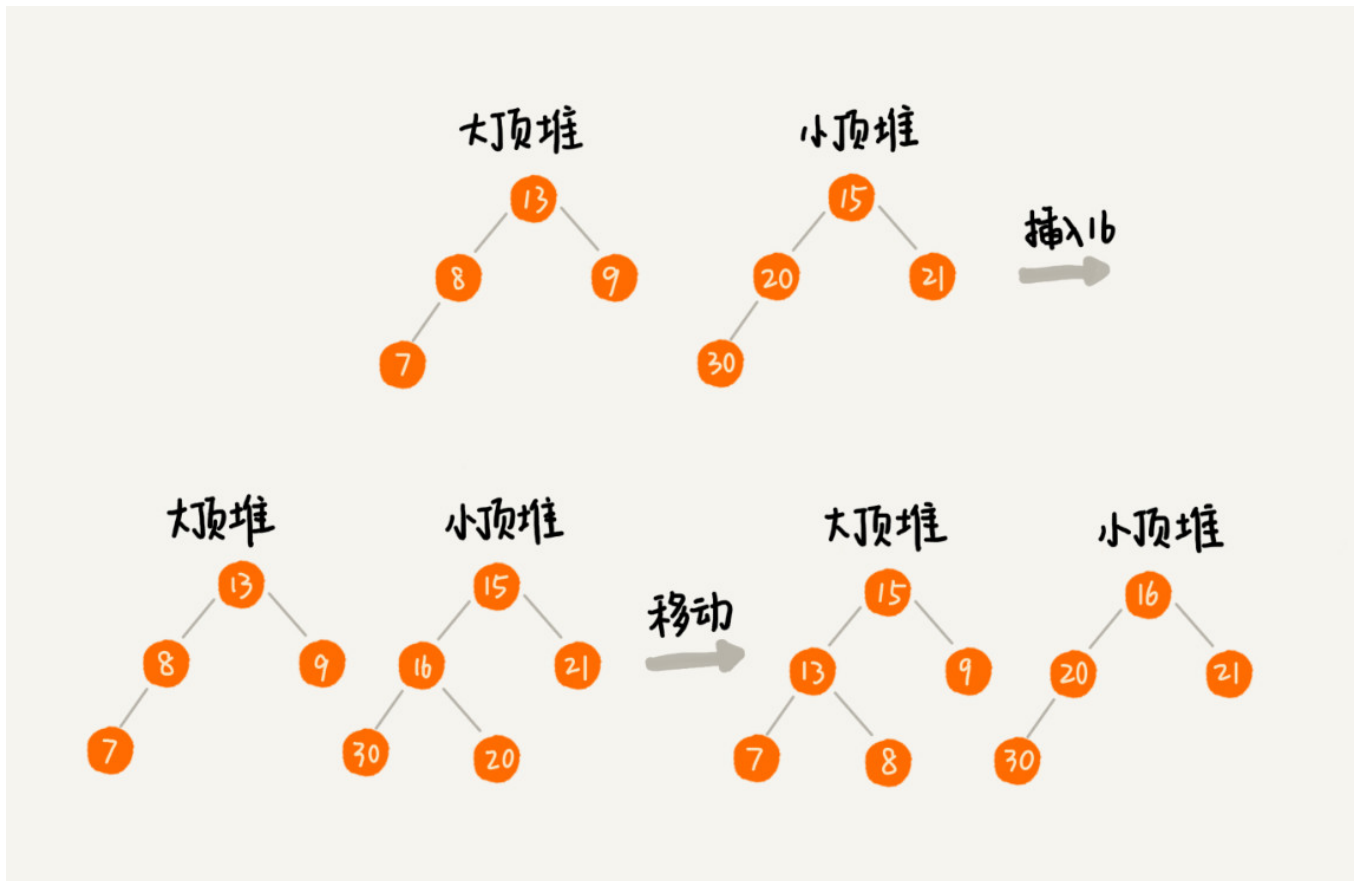
也就是说，如果有 n 个数据， n 是偶数，我们从小到大排序，那前 $\frac{n}{2}$ 个数据存储在大顶堆中，后 $\frac{n}{2}$ 个数据存储在小顶堆中。这样，大顶堆中的堆顶元素就是我们要找的中位数。如果 n 是奇数，情况是类似的，大顶堆就存储 $\frac{n}{2} + 1$ 个数据，小顶堆中就存储 $\frac{n}{2}$ 个数据。



我们前面也提到，数据是动态变化的，当新添加一个数据的时候，我们如何调整两个堆，让大顶堆中的堆顶元素继续是中位数呢？

如果新加入的数据小于等于大顶堆的堆顶元素，我们就将这个新数据插入到大顶堆；如果新加入的数据大于等于小顶堆的堆顶元素，我们就将这个新数据插入到小顶堆。

这个时候就有可能出现，两个堆中的数据个数不符合前面约定的情况：如果 n 是偶数，两个堆中的数据个数都是 $\frac{n}{2}$ ；如果 n 是奇数，大顶堆有 $\frac{n}{2} + 1$ 个数据，小顶堆有 $\frac{n}{2}$ 个数据。这个时候，我们可以从一个堆中不停地将堆顶元素移动到另一个堆，通过这样的调整，来让两个堆中的数据满足上面的约定。



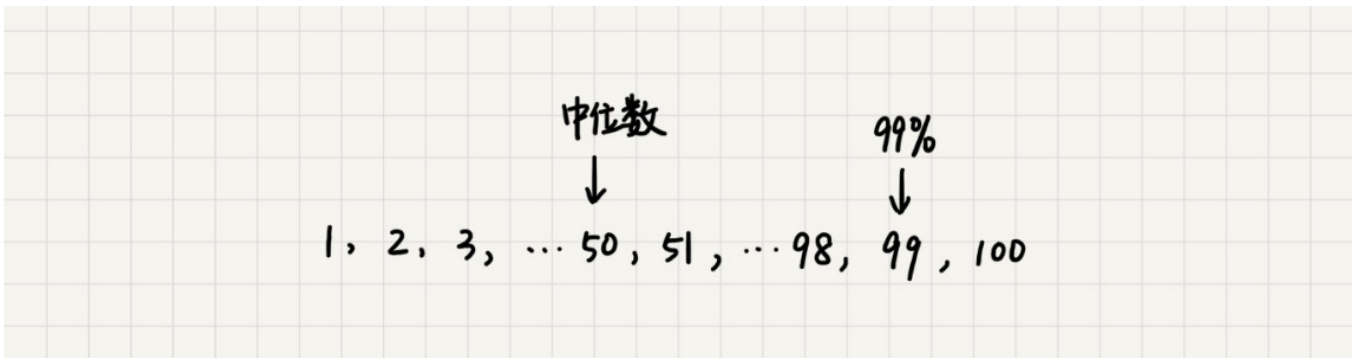
于是，我们就可以利用两个堆，一个大顶堆、一个小顶堆，实现在动态数据集中求中位数的操作。插入数据因为需要涉及堆化，所以时间复杂度变成了 $O(\log n)$ ，但是求中位数我们只需要返回大顶堆的堆顶元素就可以了，所以时间复杂度就是 $O(1)$ 。

实际上，利用两个堆不仅可以快速求出中位数，还可以快速求其他百分位的数据，原理是类似的。还记得我们在“[为什么要学习数据结构与算法](#)”里的这个问题吗？“如何快速求接口的 99% 响应时间？”我们现在就来看下，利用两个堆如何实现。

在开始这个问题的讲解之前，我先解释一下，什么是“99% 响应时间”。

中位数的概念就是将数据从小到大排列，处于中间位置，就叫中位数，这个数据会大于等于前面 50% 的数据。99 百分位数的概念可以类比中位数，如果将一组数据从小到大排列，这个 99 百分位数就是大于前面 99% 数据的那个数据。

如果你还是不太理解，我再举个例子。假设有 100 个数据，分别是 1, 2, 3, ..., 100，那 99 百分位数就是 99，因为小于等于 99 的数占总个数的 99%。



弄懂了这个概念，我们再来看 99% 响应时间。如果有 100 个接口访问请求，每个接口请求的响应时间都不同，比如 55 毫秒、100 毫秒、23 毫秒等，我们把这 100 个接口的响应时间按照从小到大排列，排在第 99 的那个数据就是 99% 响应时间，也叫 99 百分位响应时间。

我们总结一下，如果有 n 个数据，将数据从小到大排列之后，99 百分位数大约就是第 $n \times 99\%$ 个数据，同类，80 百分位数大约就是第 $n \times 80\%$ 个数据。

弄懂了这些，我们再来看如何求 99% 响应时间。

我们维护两个堆，一个大顶堆，一个小顶堆。假设当前总数据的个数是 n ，大顶堆中保存 $n \times 99\%$ 个数据，小顶堆中保存 $n \times 1\%$ 个数据。大顶堆堆顶的数据就是我们要找的 99% 响应时间。

每次插入一个数据的时候，我们要判断这个数据跟大顶堆和小顶堆堆顶数据的大小关系，然后决定插入到哪个堆中。如果这个新插入的数据比大顶堆的堆顶数据小，那就插入大顶堆；如果这个新插入的数据比小顶堆的堆顶数据大，那就插入小顶堆。

但是，为了保持大顶堆中的数据占 99%，小顶堆中的数据占 1%，在每次新插入数据之后，我们都要重新计算，这个时候大顶堆和小顶堆中的数据个数，是否还符合 99:1 这个比例。如果不符合，我们就将一个堆中的数据移动到另一个堆，直到满足这个比例。移动的方法类似前面求中位数的方法，这里我就不啰嗦了。

通过这样的方法，每次插入数据，可能会涉及几个数据的堆化操作，所以时间复杂度是 $O(\log n)$ 。每次求 99% 响应时间的时候，直接返回大顶堆中的堆顶数据即可，时间复杂度是 $O(1)$ 。

解答开篇

学懂了一些应用场景的处理思路，我想你应该能解决开篇的那个问题了。假设现在我们有一个包含 10 亿个搜索关键词的日志文件，如何快速获取到 Top 10 最热门的搜索关键词呢？

处理这个问题，有很多高级的解决方法，比如使用 MapReduce 等。但是，如果我们将处理的场景限定为单机，可以使用的内存为 1GB。那这个问题该如何解决呢？

因为用户搜索的关键词，有很多可能都是重复的，所以我们首先要统计每个搜索关键词出现的频率。我们可以通过散列表、平衡二叉查找树或者其他一些支持快速查找、插入的数据结构，来记录关键词及其出现的次数。

假设我们选用散列表。我们就顺序扫描这 10 亿个搜索关键词。当扫描到某个关键词时，我们去散列表中查询。如果存在，我们就将对应的次数加一；如果不存在，我们就将它插入到散列表，并记录次数为 1。以此类推，等遍历完这 10 亿个搜索关键词之后，散列表中就存储了不重复的搜索关键词以及出现的次数。

然后，我们再根据前面讲的用堆求 Top K 的方法，建立一个大小为 10 的小顶堆，遍历散列表，依次取出每个搜索关键词及对应出现的次数，然后与堆顶的搜索关键词对比。如果出现次数比堆顶搜索关键词的次数多，那就删除堆顶的关键词，将这个出现次数更多的关键词加入到堆中。

以此类推，当遍历完整个散列表中的搜索关键词之后，堆中的搜索关键词就是出现次数最多的 Top 10 搜索关键词了。

不知道你发现了没有，上面的解决思路其实存在漏洞。10 亿的关键词还是很多的。我们假设 10 亿条搜索关键词中不重复的有 1 亿条，如果每个搜索关键词的平均长度是 50 个字节，那存储 1 亿个关键词起码需要 5GB 的内存空间，而散列表因为要避免频繁冲突，不会选择太大的装载因子，所以消耗的内存空间就更多了。而我们的机器只有 1GB 的可用内存空间，所以我们无法一次性将所有的搜索关键词加入到内存中。这个时候该怎么办呢？

我们在哈希算法那一节讲过，相同数据经过哈希算法得到的哈希值是一样的。我们可以哈希算法的这个特点，将 10 亿条搜索关键词先通过哈希算法分片到 10 个文件中。

具体可以这样做：我们创建 10 个空文件 00, 01, 02,, 09。我们遍历这 10 亿个关键词，并且通过某个哈希算法对其求哈希值，然后哈希值同 10 取模，得到的结果就是这个搜索关键词应该被分到的文件编号。

对这 10 亿个关键词分片之后，每个文件都只有 1 亿的关键词，去除掉重复的，可能就只有 1000 万个，每个关键词平均 50 个字节，所以总的大小就是 500MB。1GB 的内存完全可以放得下。

我们针对每个包含 1 亿条搜索关键词的文件，利用散列表和堆，分别求出 Top 10，然后把这 10 个 Top 10 放在一块，然后取这 100 个关键词中，出现次数最多的 10 个关键词，这就是这 10 亿数据中的 Top 10 最频繁的搜索关键词了。

内容小结

我们今天主要讲了堆的几个重要的应用，它们分别是：优先级队列、求 Top K 问题和求中位数问题。

优先级队列是一种特殊的队列，优先级高的数据先出队，而不再像普通的队列那样，先进先出。实际上，堆就可以看作优先级队列，只是称谓不一样罢了。求 Top K 问题又可以分为针对静态数据和针对动态数据，只需要利用一个堆，就可以做到非常高效率的查询 Top K 的数据。求中位数实际上还有很多变形，比如求 99 百分位数据、90 百分位数据等，处理的思路都是一样的，即利用两个堆，一个大顶堆，一个小顶堆，随着数据的动态添加，动态调整两个堆中的数据，最后大顶堆的堆顶元素就是要求的数据。

课后思考

有一个访问量非常大的新闻网站，我们希望将点击量排名 Top 10 的新闻摘要，滚动显示在网站首页 banner 上，并且每隔 1 小时更新一次。如果你是负责开发这个功能的工程师，你会如何实现呢？

欢迎留言和我分享，我会第一时间给你反馈。

 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



©版权归极客邦科技所有，未经许可不得转载

上一篇 28 | 堆和堆排序：为什么说堆排序没有快速排序快？

写留言

精选留言



Miletos

👍 12

“如果新加入的数据小于等于大顶堆的堆顶元素，我们就将这个新数据插入到大顶堆；如果新加入的数据大于等于小顶堆的堆顶元素，我们就将这个新数据插入到小顶堆。”

1. 这里不太对劲，前文中说到，小顶堆的堆顶大于大顶堆的堆顶。

如果新进元素在小顶堆堆顶和大顶堆堆顶元素值之间，没有规定插入哪个堆。

我觉得，是不是只要判断一次就可以了。新进元素值大于等于小顶堆堆顶元素的，插入小顶堆，否则插入大顶堆。

当某一个堆数据过多时再重新移动堆顶元素。

2. 求中位数的源数据中，是否允许重复数据？

2018-11-28

作者回复

1 你说的对 我改下 多谢指正

2 可以重复

2018-11-28



豪华

👍 6

老师，分片求取前十是不是有bug，如果有一个关键词在每一组分片中都是前第十一位，在整个十亿中个数总和是第一位，是不是用分片求出了错误的结果呢？

2018-11-28

作者回复

不会的 相同的关键词经过哈希之后只会到一台机器

2018-11-28



ALAN

👍 1

- 1:建一个散射列表，key为点击网址，value为点击次数。散射列表通过从log中计算得来。
- 2:建一个10个数据的小顶堆，数据值为点击次数，扫描散射列表，新元素次数比堆顶元素大则删除堆顶元素，插入新元素，小则继续扫描散射列表。
- 3:扫描整个散射列表后，即得到top 10点击量，将点击网址存储在数组A中。数组A一个小时更新一次。
- 4:散射列表实时更新，小顶堆也实时更新，以一小时为间隔，将小顶堆结果更新到数组A中。

2018-11-28



美团点评技术团队

👍 1

王老师 第一点合并有序小文件 为什么要用到优先级队列 和 堆还是不理解。两个比最小取出合并，只要两个数组是有序就可以了，快排成有序，从小到大比较合并，不可以吗，为什么要用到优先级队列，方便老师解答下吗

2018-11-28

作者回复

没太看懂你说的 用优先级是为了效率

2018-11-28



吴宇晨

👍 1

可以先用散列表存帖子和点击数的关系，然后每一小时定时遍历散列表用文中的方法，往大小为10的小顶堆插数据？

2018-11-28



蒋礼锐

👍 1

需要一个散列表维护新闻对象，每个新闻对象至少需要维护下列几个字段，1 新闻摘要 2 新闻阅读次数(如果内存有限制分片处理) 3 新闻刷新时间(如果另外有定时器也可不用这个字段)。4 新闻链接

动态维护一个包含10个对象的小顶堆，每次有新闻被浏览时进行，去散列表中找到新闻对应的阅读次数，如果大于小顶堆的堆顶元素，则将堆顶元素删除，并重新堆化。复杂度为 $\log 10$

1小时刷新就可以对比堆顶元素的刷新时间与当前时间间隔。

每1小时获取top10时，将堆排序后输出，复杂度 $10\log 10$ ，

但是有个疑问，因为阅读新闻的人很多，如果同一时间有大量的人在阅读不同的新闻，如何保证数据的一致性呢？也就是这个堆和那个hash表对每个用户来说都一样。想到一种办法是用队列储存同一时间的新闻请求，然后再一个一个事务操作。我是前端，对数据储存这一块不太熟，还请老师或社友指正。

2018-11-28



P@trick

👍 1

思考题：

1. 用散列表存储每个链接的点击数
2. 用优先级队列实现高性能定时器，在初始化和处理函数时更新插入1小时的定时器
3. 利用堆，将散列表中的链接根据点击次数求top10

2018-11-28



行者

👍 1

针对这个动态点击top 10问题，如果继续仅仅统计top 10是有问题的，因为可能当前的top 1在接下来的时间内突然变为了热搜；所以要多统计多一点 top50 或 top100；然后更隔1小时统计增量日志，更新top n，然后在获取最新的top 10。

2018-11-28



他在地城断了弦

👍 0

老师后面会讲AVL树吗？

2018-11-29



老杨同志

👍 0



@美团点评技术团队

你提的问题 “

王老师 第一点合并有序小文件 为什么要用到优先级队列 和 堆还是不理解。两个比最小取出合并，只要两个数组是有序就可以了，快排成有序，从小到大比较合并，不可以吗，为什么要用到优先级队列，方便老师解答下吗 ” 我是这样理解的。

两个有序文件的合并是可以直接比较大小的，类似归并排序的合并。但是老师的方法适合于几十个，上个百个有序文件的合并，多个数字求最大直接遍历是 $O(n)$ ，放到堆里是 $O(\log n)$ 。多个文件两两合并，最后也能合并成一个，但是要生成多个中间临时文件，效率低。

2018-11-29



Kong

0

关于把5G文件分片成10个文件的问题，有点不明白，内存只有1G，是怎么做到把整个文件读到内存中再通过hash算法去分片的？请老师解惑

2018-11-29



Jeremy

0

思考题：

- 1.全量统计摘要和对应的点击率
- 2.用hashheap(小顶堆，按点击率建堆)计算topK；使用hash table定位摘要，用heap对点击率排序
- 3.每小时增量统计摘要和对应的点击率
- 4.用过去一小时的点击率，修改当前hashheap。时间复杂度：找到对应摘要用 $O(1)$ ，增加或修改点击率后heapify用 $O(\log k)$

2018-11-29



纯洁的憎恶

0

用堆实现动态变化的优先级队列，可以在动态保证队列次序的前提下，以 $O(\log n)$ 的时间复杂度完成出队和入队操作。通过大顶堆和小顶堆实现动态查找中位数和百分位数的方法尤为巧妙。

课前思考题与课后思考题的类型近似，解决方法也可以相同。我能想到的新闻与新闻的点击率都是动态变化的，不仅需要实时更新新闻摘要与点击率，还要每过一个小时，做一遍哈希分片、分片TOPK、合并TOPK。有没有更快捷、简便的方法呢？

2018-11-28



守着云开

0

10亿关键词分片之后 每个文件并不一定有1亿的关键词吧 老师

2018-11-28



邵靖隆

0

老师，我问一个关于排序的问题

如果在归并排序中，当切分出的子列长度小于10时不再对其继续递归归并排序，而改用插入

排序。

这样的组合算法，其时间复杂度如何计算？

2018-11-28

作者回复

这个插入排序的部分可以不用计算时间复杂度 比较规模很小

2018-11-28



夏天

0

感觉用堆能解决的问题,换成其他数据结构也能解决

2018-11-28

作者回复

比如？

2018-11-28



猫头鹰爱拿铁

0

思考题：访问量很大，有可能超过Integer.Max。考虑先使用队列存储用户的新闻请求，如果队列数满了就再起一个队列。那么存储用户新闻请求队列的数据结构可以考虑linkedblockingqueue<linkedblockingqueue<News>>。一个队列满了也差不多有48gb数据 ((16+8)*2^31/1024^3, 大概21亿请求) 接着就是对数据进行处理。如果是单机处理，真的很困难，那么得考虑数据量超过多少时序列化，存储到磁盘上。然后再取出数据进行排序。这个方法不是很好，io很影响性能。那么可以换一个方式，考虑对数据分片用多台计算机进行分布式的计算。首先news.hashCode%计算机数量，将news划分到不同的机子上处理，可以利用散列表，计算<news,count>,遍历的时候建立size为10的小顶堆。如果散列表遍历的数量大于堆顶数据，则移除，添加散列表的这个数据。每台计算机这样计算得到小顶堆后拿出自己出现最多的10个news（小顶堆）每隔一小时进行汇总再行比较得到最多的10个news。

2018-11-28



高峰

0

静态数据查找k大的数据 java实现

```
public class HeapTest {
    public static void main(String[] args) {
        Integer[] target = { 2, 3, 5, 8, 11, 22, 97, 98, 33, 44, 55, 66, 77, 88, 99, 100, 12, 13, 14, 15, 16, 17, 18, 19, 20 };
        Queue<Integer> queue = new PriorityQueue<Integer>(5);
        queue.addAll(Arrays.asList(1, 2, 3, 4, 5));
        for (Integer num : target) {
            Integer top = queue.peek();
            if (num >= top) {
                queue.poll();
                queue.add(num);
            }
        }
    }
}
```

```
for (Integer integer : queue) {  
    System.out.println(integer);  
}  
  
}
```

2018-11-28



komo0104

👍 0

每条新闻都在数据库里维护点击量。

每小时把日志中的增量数据一次性统计并更新到数据库。

将更新了点击次数的新闻与原本的top10进行比较，用priorityqueue实现。

2018-11-28



oatlmy

👍 0

老师，请问为什么评价算法性能是根据时间和空间复杂度，而不是别的参数？是因为计算机结构是冯诺依曼体系，除了输入输出设备和控制器，就剩下运算器和存储器了吗？

2018-11-28

作者回复

你理解的没错

2018-11-28