14讲排序优化：如何实现一个通用的、高性能的排序函数

几乎所有的编程语言都会提供排序函数，比如C语言中qsort()，C++ STL中的sort()、stable_sort()，还有Java语言中的Collections.sort()。在平时的开发中，我们也都是直接使用这些现成的函数来实现业务逻辑中的排序功能。那你知道这些排序函数是如何实现的吗？底层都利用了哪种排序算法呢？

基于这些问题，今天我们就来看排序这部分的最后一块内容：**如何实现一个通用的、高性能的排序函数？**

如何选择合适的排序算法？

如果要实现一个通用的、高效率的排序函数，我们应该选择哪种排序算法？我们先回顾一下前面讲过的几种排序算法。

我们前面讲过，线性排序算法的时间复杂度比较低，适用场景比较特殊。所以如果要写一个通用的排序函数，不能选择线性排序算法。

如果对小规模数据进行排序，可以选择时间复杂度是O(n^2)的算法；如果对大规模数据进行排序，时间复杂度是O(nlogn)的算法更加高效。所以，为了兼顾任意规模数据的排序，一般都会首选时间复杂度是O(nlogn)的排序算法来实现排序函数。

时间复杂度是O(nlogn)的排序算法不止一个，我们已经讲过的有归并排序、快速排序，后面讲堆的时候我们还会讲到堆排序。堆排序和快速排序都有比较多的应用，比如Java语言采用堆排序实现排序函数，C语言使用快速排序实现排序函数。

不知道你有没有发现，使用归并排序的情况其实并不多。我们知道，快排在最坏情况下的时间复杂度是O(n^2)，而归并排序可以做到平均情况、最坏情况下的时间复杂度都是O(nlogn)，从这点上看起来很诱人，那为什么它还是没能得到“宠信”呢？

还记得我们上一节讲的归并排序的空间复杂度吗？归并排序并不是原地排序算法，空间复杂度是O(n)。所以，粗略点、夸张点讲，如果要排序100MB的数据，除了数据本身占用的内存之外，排序算法还要额外再占用100MB的内存空间，空间耗费就翻倍了。

前面我们讲到，快速排序比较适合来实现排序函数，但是，我们也知道，快速排序在最坏情况下的时间复杂度是O(n^2)，如何来解决这个“复杂度恶化”的问题呢？

如何优化快速排序？

我们先来看下，为什么最坏情况下快速排序的时间复杂度是O(n^2)呢？我们前面讲过，如果数据原来就是有序的或者接近有序的，每次分区点都选择最后一个数据，那快速排序算法就会变得非常糟糕，时间复杂度就会退化为O(n^2)。实际上，**这种O(n^2)时间复杂度出现的主要原因还是因为我们分区点选的不够合理**。

那什么样的分区点是好的分区点呢？或者说如何来选择分区点呢？

最理想的分区点是：**被分区点分开的两个分区中，数据的数量差不多**。

如果很粗暴地直接选择第一个或者最后一个数据作为分区点，不考虑数据的特点，肯定会出现之前讲的那样，在某些情况下，排序的最坏情况时间复杂度是$O(n^2)$。为了提高排序算法的性能，我们也要尽可能地让每次分区都比较平均。

我这里介绍两个比较常用、比较简单的分区算法，你可以直观地感受一下。

1.三数取中法

我们从区间的首、尾、中间，分别取出一个数，然后对比大小，取这3个数的中间值作为分区点。这样每间隔某个固定的长度，取数据出来比较，将中间值作为分区点的分区算法，肯定要比单纯取某一个数据更好。但是，如果要排序的数组比较大，那“三数取中”可能就不够了，可能要“五数取中”或者“十数取中”。

2.随机法

随机法就是每次从要排序的区间中，随机选择一个元素作为分区点。这种方法并不能保证每次分区点都选的比较好，但是从概率的角度来看，也不大可能会出现每次分区点都选的很差的情况，所以平均情况下，这样选的分区点是比较好的。时间复杂度退化为最糟糕的$O(n^2)$的情况，出现的可能性不大。

好了，我这里也只是抛砖引玉，如果想了解更多寻找分区点的方法，你可以自己课下深入去学习一下。

我们知道，快速排序是用递归来实现的。我们在递归那一节讲过，递归要警惕堆栈溢出。为了避免快速排序里，递归过深而堆栈过小，导致堆栈溢出，我们有两种解决办法：第一种是限制递归深度。一旦递归过深，超过了我们事先设定的阈值，就停止递归。第二种是通过在堆上模拟实现一个函数调用栈，手动模拟递归压栈、出栈的过程，这样就没有了系统栈大小的限制。

举例分析排序函数

为了让你对如何实现一个排序函数有一个更直观的感受，我拿Glibc中的qsort()函数举例说明一下。虽说qsort()从名字上看，很像是基于快速排序算法实现的，实际上它并不仅仅用了快排这一种算法。

如果你去看源码，你就会发现，**qsort()会优先使用归并排序来排序输入数据**，因为归并排序的空间复杂度是O(n)，所以对于小数据量的排序，比如1KB、2KB等，归并排序额外需要1KB、2KB的内存空间，这个问题不大。现在计算机的内存都挺大的，我们很多时候追求的是速度。还记得我们前面讲过的用空间换时间的技巧吗？这就是一个典型的应用。

但如果数据量太大，就跟我们前面提到的，排序100MB的数据，这个时候我们再用归并排序就不合适了。所以，**要排序的数据量比较大的时候，qsort()会改为用快速排序算法来排序**。

那qsort()是如何选择快速排序算法的分区点的呢？如果去看源码，你就会发现，qsort()选择分区点的方法就是“三数取中法”。是不是也并不复杂？

还有我们前面提到的递归太深会导致堆栈溢出的问题，qsort()是通过自己实现一个堆上的栈，手动模拟递归来解决的。我们之前在讲递归那一节也讲过，不知道你还有没有印象？

实际上，qsort()并不仅仅用到了归并排序和快速排序，它还用到了插入排序。在快速排序的过程中，当要排序的区间中，元素的个数小于等于4时，qsort()就退化为插入排序，不再继续用递归来做快速排序，因为我们前面也讲过，在小规模数据面前，**$O(n^2)$时间复杂度的算法并不一定比O(nlogn)的算法执行时间长**。我们现在就来分析下这个说法。

我们在讲复杂度分析的时候讲过，算法的性能可以通过时间复杂度来分析，但是，这种复杂度分析是比较偏理论的，如果我们深究的话，实际上时间复杂度并不等于代码实际的运行时间。

时间复杂度代表的是一个增长趋势，如果画成增长曲线图，你会发现$O(n^2)$比O(nlogn)要陡峭，也就是说增长趋势要更猛一些。但是，我们前面讲过，在大O复杂度表示法中，我们会省略低阶、系数和常数，也就是说，O(nlogn)在没有省略低阶、系数、常数之前可能是O(knlogn + c)，而且k和c有可能还是一个比较大的数。

假设k=1000，c=200，当我们对小规模数据（比如n=100）排序时，n^2的值实际上比knlogn+c还要小。

```
knlogn+c = 1000 * 100 * log100 + 200 远大于10000


n^2 = 100*100 = 10000
```

所以，对于小规模数据的排序，$O(n^2)$的排序算法并不一定比O(nlogn)排序算法执行的时间长。对于小数据量的排序，我们选择比较简单、不需要递归的插入排序算法。

还记得我们之前讲到的哨兵来简化代码，提高执行效率吗？在qsort()插入排序的算法实现中，也利用了这种编程技巧。虽然哨兵可能只是少做一次判断，但是毕竟排序函数是非常常用、非常基础的函数，性能的优化要做到极致。

好了，C语言的qsort()我已经分析完了，你有没有觉得其实也不是很难？基本上都是用了我们前面讲到的知识点，有了前面的知识积累，看一些底层的类库的时候是不是也更容易了呢？

内容小结

今天我带你分析了一下如何来实现一个工业级的通用的、高效的排序函数，内容比较偏实战，而且贯穿了一些前面几节的内容，你要多看几遍。我们大部分排序函数都是采用O(nlogn)排序算法来实现，但是为了尽可能地提高性能，会做很多优化。

我还着重讲了快速排序的一些优化策略，比如合理选择分区点、避免递归太深等等。最后，我还带你分析了一个C语言中qsort()的底层实现原理，希望你对此能有一个更加直观的感受。

课后思考

在今天的内容中，我分析了C语言的中的qsort()的底层排序算法，你能否分析一下你所熟悉的语言中的排序函数都是用什么排序算法实现的呢？都有哪些优化技巧？

欢迎留言和我分享，我会第一时间给你反馈。

特别说明：

专栏已经更新一月有余，我在留言区看到很多同学说，希望给出课后思考题的标准答案。鉴于留言区里本身就有很多非常好的答案，之后我会将我认为比较好的答案置顶在留言区，供需要的同学参考。

如果文章发布一周后，留言里依旧没有比较好的答案，我会把我的答案写出来置顶在留言区。

最后，**希望你把思考的过程看得比标准答案更重要。**

精选留言

刘强

我们的教育让我们对标准答案的依赖太深了，让我们失去了独立思考的能力。深深的感受到了这一点。思考的过程比标准答案更重要，这句话才是关键。

2018-10-22 07:42

杨伟

思考过程比答案重要这句话是不假，但是有答案来验证自己的思考是否准确在初学时期也很重要。

学习知识每个人的理解会不同，有的人可能这么理解有的人可能那样理解。如果没有一个标杆，有些同学就会按照自己错误的理解继续学习下去。

有了标准答案，同学就可以对照答案来反思自己的理解是否正确。也能够从别人的答案中看到更好的解答也是一种学习。

当然自己偷懒不思考，依赖标准答案，那肯定是学不好的

2018-10-22 11:32

Liam

查看了下Arrays.sort的源码，主要采用TimSort算法, 大致思路是这样的：

1 元素个数 < 32, 采用二分查找插入排序(Binary Sort)
2 元素个数 >= 32, 采用归并排序，归并的核心是分区(Run)
3 找连续升或降的序列作为分区，分区最终被调整为升序后压入栈
4 如果分区长度太小，通过二分插入排序扩充分区长度到分区最小阈值
5 每次压入栈，都要检查栈内已存在的分区是否满足合并条件，满足则进行合并
6 最终栈内的分区被全部合并，得到一个排序好的数组

Timsort的合并算法非常巧妙：

1 找出左分区最后一个元素(最大)及在右分区的位置

2 找出右分区第一个元素(最小)及在左分区的位置
3 仅对这两个位置之间的元素进行合并，之外的元素本身就是有序的
2018-10-22 16:56


小确幸
数据库里面的Order BY，用的是什么排序呢?
2018-10-22 09:27


峰
java1.8中的排序，在元素小于47的时候用插入排序，大于47小于286用双轴快排，大于286用timsort归并排序，并在timesort中记录数据的连续的有序段的的位置，若有序段太多，也就是说数据近乎乱序，则用双轴快排，当然快排的递归调用的过程中，若排序的子数组数据数量小，用插入排序。
2018-10-22 10:13


Andrew 陈震
老师，我有一个问题，关于递归太深导致堆栈溢出的问题。对于这个问题，您说一般有两种解决方法，一是设置最深的层数，如果超过层数了，就报错。对于这样的问题，我想排序一个数列，超过了层数，难道就不排了么？我看有留言说，stl中的sort默认是使用快排的，但当递归深度过大时 会转为使用归并排序。但是归并排序也是递归排序啊，如果两种排序都达到了最深层数怎么处理？另外，在排序之前，能否估算出排序是否超过最深层数呢？如果估算不出，那岂不是要先排一遍，发现超过层数，再换用别的。我的想法是设个阈值，不超过阈值，用一种，超过了，用另一种。

第二种应对堆栈溢出的方法是通过在堆上模拟实现一个函数调用栈，手动模拟递归压栈、出栈的过程。这个方法在您的几篇教程里都提到过，但是不详细，您能否稍微详细讲解一下。

谢谢老师
2018-10-22 09:30

作者回复

太深报错也没问题 不过不建议这么处理

归并排序比较稳定 栈的深度是logn 非常小 所以不会堆栈溢出

关于手动模拟栈 你可以看看qsort（）函数的实现

2018-10-23 09:43


姜威
总结：如何实现一个通用的高性能的排序函数?
一、如何选择合适的排序算法?
1.排序算法一览表
时间复杂度 是稳定排序？ 是原地排序?
冒泡排序 O(n^2) 是 是
插入排序 O(n^2) 是 是
选择排序 O(n^2) 否 是
快速排序 O(nlogn) 否 是
归并排序 O(nlogn) 是 否
桶排序 O(n) 是 否
计数排序 O(n+k)，k是数据范围 是 否
基数排序 O(dn)，d是纬度 是 否
2.为什选择快速排序?
1）线性排序时间复杂度很低但使用场景特殊，如果要写一个通用排序函数，不能选择线性排序。
2）为了兼顾任意规模数据的排序，一般会首选时间复杂度为O(nlogn)的排序算法来实现排序函数。
3）同为O(nlogn)的快排和归并排序相比，归并排序不是原地排序算法，所以最优的选择是快排。
二、如何优化快速排序?
导致快排时间复杂度降为O(n)的原因是分区点选择不合理，最理想的分区点是：被分区点分开的两个分区中，数据的数量差不多。如何优化分区点的选择？有2种常用方法，如下：
1.三数取中法
①从区间的首、中、尾分别取一个数，然后比较大小，取中间值作为分区点。
②如果要排序的数组比较大，那“三数取中”可能就不够用了，可能要“5数取中”或者“10数取中”。
2.随机法：每次从要排序的区间中，随机选择一个元素作为分区点。

3.警惕快排的递归发生堆栈溢出，有2中解决方法，如下：
①限制递归深度，一旦递归超过了设置的阈值就停止递归。
②在堆上模拟实现一个函数调用栈，手动模拟递归压栈、出栈过程，这样就没有系统栈大小的限制。
三、通用排序函数实现技巧
1.数据量不大时，可以采取用时间换空间的思路
2.数据量大时，优化快排分区点的选择
3.防止堆栈溢出，可以选择在堆上手动模拟调用栈解决
4.在排序区间中，当元素个数小于某个常数是，可以考虑使用O(n^2)级别的插入排序
5.用哨兵简化代码，每次排序都减少一次判断，尽可能把性能优化到极致
四、思考
1.Java中的排序函数都是用什么排序算法实现的？有有哪些技巧？
2018-10-22 21:46


李靖峰
golang标准库中的Sort用的是快排+希尔排序+插排，数据量大于12时用快排，小于等于12时用6作为gap做一次希尔排序，然后走一遍普通的插排（插排对有序度高的序列效率高）。其中快排pivot的选择做了很多工作不是一两句话可以描述出来，是基于首中尾中值的很复杂的变种
2018-10-29 11:19


Liam
关于快排递归过深的处理的疑惑，以及关于 STL 里的 std::sort 是怎么实现的，可以看我这篇博客：https://liam.page/2018/09/18/std-sort-in-STL/
2018-10-22 12:27


leo
排序的思维导图链接：https://share.weiyun.com/5X17MG3
2018-10-29 12:04


城
qsort中为避免递归调用过深，所以在堆上模拟了栈。不知道是否是将递归调用，改写为循环非递归方式呢？
2018-10-22 08:37

作者回复

是的
2018-10-22 09:53


蛐鸣
看了一下，.NET里面的Array排序实现:
1. 三个以内的，直接比较，交换进行实现
2.大于3个小于16个的，用的是插入排序进行的实现
3.对于大于16，并且深度限制是0的，用的是堆排序实现的
4.对于大于15，并且深度限制不是0的，使用的是快速排序；然后快速排序分区使用的也是三数取中法
2018-11-02 09:47

作者回复

2018-11-02 09:51


等风来
使用快排如何解决不稳定排序的问题？
2018-10-22 09:36

作者回复

并没解决 所以qsort不稳定
2018-10-22 10:10


谢俊
老师，我是算法上的菜鸟，上一篇说快排是原地排序，既然是原地排序，空间复杂度为O(1)，为何需要防止堆栈溢出呢？
2018-11-26 10:14

学习爱好者
王老师，总结8种排序算法的那个图，桶排序不一定是稳定排序吧？比如桶内排序用快排的时候
2018-11-05 23:38

作者回复

嗯嗯 用归并或者插入排序就稳定了

2018-11-06 09:30

favorlm

虽然说思考很重要，但是面试还是需要你实现一种算法。

2018-11-04 20:29

作者回复

留言区点赞最高的就是答案

2018-11-05 09:36

Random.nextName()

Google v8中对QuickSort的实现是:

数据规模在10以内的话使用快排;

数据规模在10到1000之间时选择中点作为pivot进行快排;

数据规模在1000以上时，每隔200到215个数选一个数，将选出来的数排序，选择中间值作为pivot进行快排;

而且还有几个细节:

1是折半的时候用的是位运算;

2是每一次遍历都会分成小于pivot，等于pivot，大于pivot的三个区间;

3是小于pivot和大于pivot这两个区间中数据规模比较小的会递归执行QuickSort，数据规模大的会先通过while循环减小数据规模。

附上源码链接: https://github.com/v8/v8/blob/master/src/js/array.js

2018-10-30 21:41

geektime learn

老师，你之前讲的快排、归并，原理我都理解的很清晰，但是一旦到转换成代码的时候，感觉一脸懵逼，你最开始这是这样吗?

2018-10-23 21:13

作者回复

是有点 毕竟代码是写给机器执行的 多看几遍 再自己默写默写

2018-10-24 23:28

Liam

前面那个说 std::sort 会用归并的你站住。SGISTL 的 std::sort 在数据规模小的时候用改进的插排，在数据规模大的时候用用改进的快排，在陷入恶化之后用堆排，在几乎排好序之后用改进的插排。请告诉我哪里用了归并了……

2018-10-22 09:13

懒猫

简单说下go语言的，大致是两个限制条件：数据长度和递归深度，如果长度大于或等于12，且递归深度大于0时，使用快排，快排在选择分区点数字时用了三数取中法，如果长度大于12且递归深度限制为0时，使用堆排序，如果数据长度小于或等于12时，用的希尔排序，间隔用是6

2018-11-12 17:29