基于计算机程序设计的排序问题分析

姓名学号 饶思莹23121517 范晓蓁 黎倩悦 梁婧亭

(计算机工程与科学学院)

摘 要 我们小组就不同规模、多种构型对测试数据进行算法测试。然后分析算法测试结果，包括算法的时间复杂度、空间复杂度、排序算法的稳定性，以及排序算法对数据构型的“适用性”。以及查找相关资料，优化代码后再测试得出相关结果。在计算机设计中，排序方式较为多样化，因此我们可以结合实际需求合理选择排序方式，优化排序方法。本文即是对以上计算的结果做出一些描述分析。

关键词 计算机程序设计；排序问题；信息技术

**Sorting problem analysis based on**

**computer programming**

Teaching Team of “Programming Training”

(School of Computer Engineering and Science)

**Abstract** Our team conducted algorithm tests on test data of different sizes and configurations. Then the test results of the algorithm are analyzed, including the time complexity of the algorithm, the spatial complexity, the stability of the sorting algorithm, and the "applicability" of the sorting algorithm to the data configuration. And find the relevant information, optimize the code and then test to get the relevant results. In computer design, the sorting way is more diversified, so we can reasonably choose the sorting way according to the actual needs, and optimize the sorting method. This paper is to make some description and analysis of the above calculation results.

**Key words** Computer programming; Sorting problem; Information technology

目录

1. 引言----------------------------------------------P1
2. 小组组织

小组分工--------------------------------------------------------

小组成员个人心得及课程体会--------------------------------------

3、排序算法的理论研究

排序算法的分析

算法的原理------------------------------------------------------------

算法的优缺点----------------------------------------------------------

排序方法的优化意义----------------------------------------------

4、排序算法的实验研究

研究方向--------------------------------------------------------

研究报告

未优化算法

三种算法的时间测试图-----------------------------------------------

三种算法的比较----------------------------------------------------

已优化算法

三种优化代码的时间侧视图--------------------------------------------

三种算法优化原理----------------------------------------------------

5、结语

总结--------------------------------------------------------

致谢---------------------------------------------------------

1 引言

排序是数据处理中的常用基本操作，它通过元素间的比较、交换或移动（多次赋值）实现数据元素按某种顺序进行重新排列。采取不同的比较、交换或移动策略形成了不同的排序算法。不同的排序算法中对数据元素进行的比较次数、赋值次数可能有较大的差别，影响整个排序操作的效率。

2 小组组织

2.1 小组分工

按小组任务量，我们将任务等分。每人负责一种排序代码的检测及优化，最后由小组组长进行整合。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 学号 | 组内情况 | 分工比例 | 任务 |
| 饶思莹 | 23121517 | 组员 | 1 | 冒泡排序就规模构型的时间测量及优化 |
| 范晓蓁 |  | 组员 | 1 | 就规模构型的时间测量及优化 |
| 黎倩悦 |  | 组员 | 1 | 就规模构型的时间测量及优化 |
| 梁婧亭 |  | 组员 | 1 | 就规模构型的时间测量及优化 |

2.2 小组成员的心得体会以及对课程的建议

|  |  |  |
| --- | --- | --- |
| 姓名 | 学习心得 | 课程建议 |
| 饶思莹 | 在计算机程序设计领域，排序问题是一个基础且重要的课题。通过学习排序算法，我深刻体会到了不同规模和多种构型对于排序算法的影响。在课程中我学习了冒泡排序、选择排序、快速排序、等常见的排序算法。每种算法都有其独特的应用场景，了解它们的特点和优缺点对于解决实际问题非常有帮助。同时我了解到评估算法性能的重要性。时间和空间复杂度是衡量算法效率的关键指标，对于设计高效算法具有重要的意义。 | 我建议在课程的讲解中不仅需要有理论的讲解同时还需要实践编程练习的讲解。或者运用一些工具展示排序的过程，可以帮助学生直观的理解算法的工作原理，由浅入深的学习计算机编程语言。 |
| 范晓蓁 |  |  |
| 黎倩悦 |  |  |
| 梁婧亭 |  |  |

3排序算法的理论研究

3.1. 排序算法的分析

3.1.1 算法的原理

冒泡排序

冒泡排序是计算机程序设计中一种常用的排序方法。冒泡排序主要是对相邻的两个元素进行比较，在比较过程中，对较小的数据进行合理的调整，使数据按照从小到大的顺序进行排序。冒泡排序方法可以根据程序中的基本数据情况展开两两比较，不断对数据进行比较，通过比较最终可以得到合理的数据排序结果[2]。 冒泡排序方法操作较为简单，其中只采用了一个辅助单元，借助辅助单元可以实现对数据的对比、排列和控制。在选择冒泡排序方法时，需要注意完成所有数据排序需要的时间。冒泡排序方法的空间复杂度为0 (1),平均时间复杂度为O (n^2)。

选择法排序

在计算机程序设计中，对于排序问题通常会采用选择法排序方法来解决问题，通过应用选择法排序，不仅可以妥善解决排序问题，同时还可以在很大程度上提高计算机程序设计的质量和效果，所以在计算机程序设计中，选择法排序有着广泛的应用。选择法排序主要是先对一组数据进行对比分析，在对比分析中选择其中最小或者最大的数据放在数列的开始位置，一直到所有的数据排列完成[3]。在选择法排序应用过程中需要注意其中的不稳定性，比如每次选择最小的数据放在数列的开始位置时，需要对数列所有数据进行比较，并改变存放位置。选择法排序空间复杂度为0(1),平均时间复杂度为O(n^2)。

快速排序法

快速排序法将原始数据分为两部分，同时要保证一部分数据比另一部分数据大，或者小，分别赋予两部分数据适当的初始值，根据快速排序法的基本思想进行有规则的扫描，通过多次交替扫描，最终使两部分数据相等。当两个数值相同，则代表数据排序工作顺利完成。在计算机程序设计中，快速排序法可以有效提高设计质量，同时降低程序设计的误差发生率。快速排序法的平均时间复杂度为0(1og2n)。

3.1.2 各种算法的优缺点

冒泡排序、快速排序和选择排序各有其优缺点，下面是每种算法的简要总结：

冒泡排序

优点

算法简单：冒泡排序的概念和实现都非常简单，易于理解。

稳定排序：相同值的元素在排序后保持原有的顺序。

适应性：如果数组在排序过程中已经部分有序，冒泡排序可以提前终止。

缺点：

效率低：平均和最坏情况下的时间复杂度都是 \(O(n^2)\)，对于大数据集来说非常慢。

不适用于大型数据集：由于其平方级的时间复杂度，冒泡排序不适合处理大量数据。

快速排序=

优点：

效率高：平均情况下的时间复杂度为 \(O(n \log n)\)，在大多数情况下比冒泡排序和选择排序快。

原地排序：除了递归栈空间外，不需要额外空间。

适应性：可以通过选择合适的枢轴来优化性能，例如使用三数取中法。

缺点：

非稳定排序：相同值的元素可能会在排序过程中改变相对顺序。

最坏情况性能：在最坏的情况下，即输入数组已经是有序或逆序的，快速排序的时间复杂度会退化到 \(O(n^2)\)。

递归使用栈空间：虽然平均情况下递归深度不大，但在最坏情况下，递归可能会导致栈溢出。

选择排序

优点：

算法简单：选择排序的概念和实现都很简单。

原地排序：除了交换元素外，不需要额外的存储空间。

缺点：

效率低：无论最好、最坏还是平均情况下，时间复杂度都是 \(O(n^2)\)，不适合处理大量数据。

非稳定排序：相同值的元素可能会在排序过程中改变相对顺序。

不适应已部分排序的数组：即使数组已经部分有序，选择排序的时间复杂度也不会有所改善。

在选择排序算法时，需要根据具体的应用场景和数据特性来决定使用哪种算法。对于小数据集或者几乎已经排序的数据，冒泡排序可能是一个不错的选择。对于大型数据集，快速排序通常是更优的选择，尽管需要考虑最坏情况下的性能和递归栈空间的问题。选择排序由于其较低的性能，通常不是首选算法，除非有特别的需求。

3.2 排序方法的优化意义

排序方法的优化要保证计算机设计质量，不仅要合理选择最适宜的排序方法，同时还需加强对排序方法的优化，通过优化进一步提高设计质量和效果，比如在程序设计过程中，当选择冒泡排序方法时，为了保证冒泡排序的质量，可以选择标志设置的方法将其写到数据交换代码程序中，这样可以促进冒泡排序工作的顺利开展，同时可以提高排序质量。在选择排序法应用过程中，应该去除无效操作，严格对各个数据的位置进行记录，同时按照特定的规则进行数据查找，这样也可以达到优化选择排序法的目的，进而使排序质量得到提高。在排序方法优化操作过程中，尤其要注意枢纽的选用，只有保证枢纽的功能要求，才能够提高排序质量，这对于提高计算机程序设计的整体质量和效果都具有重要的意义。

4 排序算法的实验研究

4.1 研究方向

我们小组选择4种排序算法冒泡排序、选择排序、快速排序开展深入细致的实验研究。具体研究内容包括如下几个方面。

(1) 设计或修改排序函数，统计该排序算法所进行的数组元素间比较次数、赋值次数（不统计辅助操作，即不统计循环控制变量等的操作次数），并利用参数（不要使用全局变量）“返回”统计结果。

(2) 用不同规模（如：从1024开始，逐次倍增直至65536）、多种构型（如：正序、逆序、均匀分布、正态分布、结构体数组）测试数据进行算法测试。然后分析算法测试结果，包括算法的时间复杂度、空间复杂度、排序算法的稳定性，以及排序算法对数据构型的“适用性”。请特别注意，同一规模、同一构型数据测试不同算法时，要确保测试数据一致。

(3) 对不同存储方式的C-字符串数组开展排序研究，完成Sorts.zip文件中待完成的两个排序函数设计，分析不同存储方式能够进行/不能进行的操作方法。

(4) 所研究的多种排序算法的执行时间比较，不同数据构型、不同算法对数组元素之间所进行的比较次数、赋值次数的比较分析。请用图、表等方式展示不同算法的性能（参见表4及图1），指出不同算法对于特殊构型数据的表现情况；

4.2 研究报告

接下来是我们小组围绕所开展的实验，在研习的基础上完成的一份学术论文形式的实训报告，包括如下内容。

4.2.1未优化排序

4.2.1.1 三种算法的比较

冒泡排序、快速排序和选择排序是三种常见的排序算法，它们在执行时间和性能上有各自的特点。下面是优化前对这三种排序算法执行时间的简要比较

1. 冒泡排序（Bubble Sort）：

时间复杂度：最坏和平均情况下为 \(O(n^2)\)，最好情况下为 \(O(n)\)（已排序的数组）。

执行时间：冒泡排序是一种简单但效率较低的排序算法，特别是对于较大的数据集。它的执行时间随着数组大小的增加而显著增加。

2. 快速排序（Quick Sort）：

时间复杂度：最坏情况下为 \(O(n^2)\)，但这种情况很少发生。平均情况下为 \(O(n \log n)\)，这是它的常见性能表现。

执行时间：快速排序通常比冒泡排序快得多，特别是对于较大的数据集。它是一种高效的排序算法，但在最坏的情况下（输入数组已经是有序的或者逆序的）性能会下降。

3. 选择排序（Selection Sort）：

时间复杂度：无论最好、最坏还是平均情况下，选择排序的时间复杂度都是 \(O(n^2)\)。

执行时间：选择排序的执行时间通常比冒泡排序略好，但仍然不适合处理大型数据集。

在考虑算法的选择时，除了执行时间外，还应该考虑算法的稳定性、空间复杂度和实际应用场景。

总结：

对于小数据集，冒泡排序和选择排序可能还能接受，但对于大型数据集，快速排序通常是更好的选择。

快速排序在实际应用中通常比冒泡排序和选择排序快得多，但需要注意最坏情况的发生。

在实际应用中，快速排序经常被优化以避免最坏情况，并且可以与其它算法如堆排序结合，以提供更好的性能保证。

4.2.1.2 三种算法时间测试图

表4. 正态分布整型数据排序时间（Debug配置版）（秒）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据规模 | 冒泡排序 | 选择排序 | 快速排序 |
| 1024 | 0 | 0.002 | 0 |
| 2048 | 0.01 | 0.006 | 0 |
| 4096 | 0.042 | 0.023 | 0.002 |
| 8192 | 0.195 | 0.093 | 0.003 |
| 16384 | 0.861 | 0.369 | 0.016 |
| 32768 | 3.611 | 1.424 | 0.058 |

图1. 正态分布整型数据排序时间（Debug配置版）（秒）

表4. 均匀分布整型数据排序时间（Debug配置版）（秒）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据规模 | 冒泡排序 | 选择排序 | 快速排序 |
| 1024 | 0.002 | 0.001 | 0 |
| 2048 | 0.009 | 0.005 | 0.001 |
| 4096 | 0.039 | 0.023 | 0.00 |
| 8192 | 0.191 | 0.095 | 0.003 |
| 16384 | 0.861 | 0.359 | 0.009 |
| 32768 | 3.626 | 1.414 | 0.037 |

表4. 完全顺序整型数据排序时间（Debug配置版）（秒）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据规模 | 冒泡排序 | 选择排序 | 快速排序 |
| 1024 | 0.001 | 0.002 | 0.001 |
| 2048 | 0.006 | 0.006 | 0.005 |
| 4096 | 0.023 | 0.022 | 0.023 |
| 8192 | 0.092 | 0.093 | 0.093 |
| 16384 | 0.375 | 0.355 | 0.355 |
| 32768 | 1.484 | 1.412 |  |

表4. 完全逆序整型数据排序时间（Debug配置版）（秒）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据规模 | 冒泡排序 | 选择排序 | 快速排序 |
| 1024 | 0.003 | 0.001 | 0.002 |
| 2048 | 0.011 | 0.006 | 0.006 |
| 4096 | 0.043 | 0.024 | 0.024 |
| 8192 | 0.181 | 0.095 | 0.095 |
| 16384 | 0.698 | 0.374 | 0.36 |
| 32768 | 2.804 | 1.473 |  |

表4. 正态分布双精度浮点型数据排序时间（Debug配置版）（秒）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据规模 | 冒泡排序 | 选择排序 | 快速排序 |
| 1024 | 0.002 | 0.001 | 0 |
| 2048 | 0.008 | 0.006 | 0.00 |
| 4096 | 0.035 | 0.023 | 0.0 |
| 8192 | 0.19 | 0.093 | 0.001 |
| 16384 | 0.824 | 0.358 | 0.002 |
| 32768 | 3.53 | 1.457 | 0.004 |
|  |  |  |  |

表4. 均匀分布双精度浮点型数据排序时间（Debug配置版）（秒）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据规模 | 冒泡排序 | 选择排序 | 快速排序 |
| 1024 | 0.003 | 0.001 | 0 |
| 2048 | 0.009 | 0.005 | 0.001 |
| 4096 | 0.036 | 0.023 | 0.001 |
| 8192 | 0.182 | 0.091 | 0.001 |
| 16384 | 0.825 | 0.364 | 0.003 |
| 32768 | 3.549 | 1.447 | 0.005 |

表4. 完全逆序双精度浮点型数据排序时间（Debug配置版）（秒）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据规模 | 冒泡排序 | 选择排序 | 快速排序 |
| 1024 | 0.003 | 0.001 | 0.002 |
| 2048 | 0.01 | 0.005 | 0.006 |
| 4096 | 0.039 | 0.021 | 0.022 |
| 8192 | 0.157 | 0.086 | 0.088 |
| 16384 | 0.631 | 0.345 | 0.355 |
| 32768 | 2.529 | 1.394 |  |

表4. 完全顺序双精度浮点型数据排序时间（Debug配置版）（秒）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据规模 | 冒泡排序 | 选择排序 | 快速排序 |
| 1024 | 0.002 | 0.001 | 0.002 |
| 2048 | 0.005 | 0.006 | 0.005 |
| 4096 | 0.022 | 0.022 | 0.027 |
| 8192 | 0.087 | 0.091 | 0.091 |
| 16384 | 0.342 | 0.358 | 0.357 |
| 32768 | 1.385 | 1.45 |  |

4.2.2已优化排序

4.2.2.1 三种优化代码时间测试图

4.2.2.2三种算法的优化原理

冒泡排序

优化代码在算法中添加一个标志变量sorted来判断数组是否已经排好序。

优化的原理是：

1. 提前终止：如果在一次完整的内层循环中没有发生任何交换，那么可以认为数组已经排序完成，因此可以提前退出外层循环。这是通过sorted标志变量实现的，如果在一次内层循环中所有元素都没有发生交换，sorted保持为1，表示数组已经排序，因此外层循环的条件i < size && !sorted将不再满足，从而提前终止排序。

2. 减少比较次数：在传统的冒泡排序中，每一轮都会对数组中的所有元素进行比较，即使数组的大部分元素已经处于正确的位置。通过使用sorted标志变量，一旦数组已经排序，算法将不会执行不必要的比较，从而减少了比较次数。

这个优化可以显著提高冒泡排序的效率，尤其是当数组几乎排序好或数组较小的时候。然而，值得注意的是，即使进行了这种优化，冒泡排序的时间复杂度仍然是O(n^2)，因此在处理大量数据时，它仍然不是最高效的排序算法。在实际应用中，通常会使用更高效的排序算法，如快速排序、归并排序或堆排序。

快速排序

代码展示了选择排序的一种变体，它在每次迭代中同时找到最大值和最小值，并将它们分别放到数组的两端。这种优化可以减少迭代的次数，因为每次迭代实际上将两个元素放置到了它们最终的位置上。

优化原理是：

1. 双向选择：传统的选择排序在每次迭代中只找到最小值并将其移动到数组的起始位置。而这个变种算法在每次迭代中同时找到最大值和最小值，将最小值移动到左边，最大值移动到右边。这样，每次迭代可以确定两个元素的最终位置，而不是一个。

2. 减少迭代次数：由于每次迭代都放置了两个元素，所以这个算法的迭代次数是数组长度的一半。对于大小为`n`的数组，传统的选择排序需要`n-1`次迭代，而这个变种只需要`n/2`次迭代。

3. 提高效率：尽管选择排序的时间复杂度仍然是O(n^2)，但这种双向选择的优化可以减少比较和交换的次数，从而在实际应用中提高效率。

需要注意的是，尽管这种优化可以提高选择排序的效率，但它并没有改变选择排序的时间复杂度。对于大型数据集，选择排序仍然不是最优的排序算法，其他算法如快速排序、归并排序或堆排序通常更为高效。此外，这种双向选择排序在数据分布不均匀时可能不会提供显著的性能提升。

快速排序

代码实现了快速排序算法的优化版本，主要优化原理包括：

**小数组优化**：当数组的大小小于等于16时，使用插入排序代替快速排序进行排序。这是因为对于小数组，插入排序的性能可能会优于快速排序，尤其是在最坏情况下快速排序的递归深度可能会成为性能瓶颈。

**三数取中法**：在选择基准（pivot）时，代码中使用了三数取中法来减少最坏情况发生的概率。具体做法是取数组的第一个元素、中间元素和最后一个元素，然后将它们的值进行比较，取中间值作为基准。这样做可以避免在某些特殊情况下（如数组已经排序或接近排序状态）快速排序性能下降的问题。

**尾递归优化：**在快速排序的递归调用中，如果子数组的大小小于等于16，将不再递归调用快速排序，而是使用插入排序。这样可以减少递归的深度，提高效率。

**数组元素交换**：在进行分区操作时，代码中通过交换数组元素而不是通过交换索引的方式来简化代码逻辑，这有助于提高代码的可读性和效率。

**递归终止条件：**在快速排序的递归函数中，当子数组的大小小于等于16时，递归终止，转而调用插入排序函数。

**类型泛化：**代码中提供了针对整数和双精度浮点数的快速排序实现，通过函数重载，可以对不同类型的数组进行排序。

**代码复用：**通过将快速排序的逻辑分解为辅助函数，实现了代码的复用，同时保持了主函数的简洁性。

这些优化措施共同作用，提高了快速排序算法在不同情况下的性能表现，尤其是在处理小数组或者接近有序的数组时。

各研究小组自拟课程实训报告（论文）的题目，要求题目文法合理、达意，切忌笼统含糊。请仿照本文格式撰写报告，报告的各种要件（题目、所有作者姓名、作者单位、摘要、关键词、正文、致谢、参考文献、title、authors、affiliations、abstract、keywords）不要缺少，文中的图、表必须有说明（表的说明排在表的上方、图的说明排在图的下方），独占一行的公式必须有编号（公式居中，编号右对齐）。正文可列出少部分关键程序代码（代码最好用Courier New字体，关键词加粗），所列的程序代码需要有足够的注释，切勿简单地将所有代码粘贴到正文中。

另外，本文作为格式模板主要设置了“样式[[1]](#footnote-1)”、“段落”中的行距选项。采用的中文字体主要有黑体、宋体，英文字体主要有Times New Roman和Courier New。这样，便于更好地控制版面格式。由于Microsoft Word软件版本不同而造成格式上的差异将是被认可的。

3.5 提交作业

以研究小组为单位，将本小组的所有材料（课程实训报告、排序算法程序源代码、测试数据）打包，于夏季学期第四周前（含）上传至“网上教学”平台本课程空间的“分组任务”中。

将课程实训报告（论文）的纸质版，于夏季学期第四周前（含）交指导教师。

【注】在线自动判题系统的解题程序将由教师直接从后台下载，无须同学们打包上传。

4 结语

计算机程序设计中，排序问题是一个需要重点考虑的问题，只有排序问题得到妥善的处理，才能促进计算机程序设计工作的顺利开展，进而提高计算机整体质量和效率。因此需要加强对排序问题的分析和研究，结合计算机程序设计的实际需求，科学合理地选择排序方法。同时，设计人员还需要加强对排序方法的优化， 并加强对多种排序方法的对比和总结，只有这样才能够更好地提高计算机程序设计质量。

**致谢** 在论文的最后，我们小组成员想要表达我们最深的谢意。首先，我们要感谢我们的指导老师，她在整个研究过程中给予我们无私的指导和帮助。老师不仅耐心地解答了我们的疑问，还提供了宝贵的建议和反馈。最后，也非常感谢所有我们的小组成员。每个人的辛勤工作和贡献都是不可或缺的。我们共同学习、讨论和进步，这个论文是我们团队合作的结晶。感谢大家的合作和支持，让我们能够共同完成这个论文。

**参 考 文 献**

[1]朱鹏飞.基于计算机程序设计的排序问题探讨[J].电脑知识与技术，2016,12(33):65-67.

[2]李润荣.计算机程序设计中的排序问题分析及相关阐述[J].电脑知识与技术，2016,12(27):2 02-20 3.

[3]王燕军.计算机程序设计中的排序问题研究[J].电子技术与软件工程，2016(15):255.

[4]张健.计算机程序设计中的排序问题探讨[J].计算机光盘软件与应用，2014,17(14):169-170.

1. [↑](#footnote-ref-1)