

专业课程实验报告

课程名称：算法分析与设计

开课学期： 2021 至 2022 学年 第 1 学期

专业：软件工程 年级班级：19级3班

学生姓名：冯春霖 学号：222019321062074

实验教师：曹严元

计算机与信息科学学院 软件学院

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | | 排序问题 | | | |
| 实验时间 | | 2021 年 9 月 15 日 | 实验类型 | | □验证性 □设计性 □综合性 |
| 一、实验目的   1. 了解并掌握排序的概念，并熟悉常见的几种排序算法； 2. 掌握常见的几种排序算法的基本思想方法； 3. 对问题实例运用不同的排序算法进行求解，并能比较不同方法的效率好坏和适用的应用场景； 4. 设计并实现一种排序算法。   二、实验要求   1. 预习实验指导书及教材的有关内容，查阅相关资料，了解不同排序算法的基本思想； 2. 严格按照实验内容进行实验，培养良好的算法设计和编程的习惯； 3. 认真听讲，服从安排，独立思考并完成实验。 | | | | | |
| 三、实验内容与设计（主要内容，操作步骤、算法描述或程序代码）  1. 查阅相关资料，了解有哪些排序算法，掌握常见的几种排序算法的基本思想  1) 冒泡排序：反复遍历要排序的列表，比较每对相邻的项目，如果它们的顺序与预期中的不一致则交换它们，直到不需要交换  2) 选择排序：遍历列表找到最大(小)的元素，存放到序列首位，再从剩余未排序元素中继续寻找最大(小)的元素放到第一个元素之后，直到所有元素均排序完毕  3) 归并排序：采用分治法，将问题规模缩小对含两个元素的子序列排序，再将有序的子序列合并得到完整的有序序列  4) 插入排序：通过构建有序序列，并从无序序列中取出元素与有序序列比较，选择合适的位置插入，得到新的有序序列  5) 快速排序：在序列中选取一个基准值，将其他值分为小于和大于基准值的两部分，递归的对这两部分继续排序，再将所有部分合并得到完整的有序序列  6) 堆排序：通过构建二叉堆，保证子节点的元素值均大(小)于父节点来进行排序  7) 计数排序：将列表的值存储在额外的数组中并记录其数据量，再将数据按顺序从数组中取出  时空复杂度：   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 排序方法 | T(平均) | T(最坏) | S | 稳定 | 备注 | | 冒泡排序 | O(N^2) | O(N^2) | O(1) | 是 |  | | 选择排序 | O(N^2) | O(N^2) | O(1) | 否 |  | | 归并排序 | O(NlogN) | O(NlogN) | O(N) | 是 |  | | 插入排序 | O(N^2) | O(N^2) | O(1) | 是 |  | | 快速排序 | O(NlogN) | O(N^2) | O(NlogN) | 否 | in-place版空间复杂度为logN | | 堆排序 | O(N^2) | O(N^2) | O(1) | 否 |  | | 计数排序 | O(N+K) | O(N+K) | O(N+K) | 是 | 不是基于比较的排序 |   2. 对下面实例进行排序，用至少4种排序方法进行，写出每种方法的排序过程   |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | | | | | |
| 1) 冒泡排序(由于冒泡排序是N^2算法，仅第一次外层循环展示具体内层循环过程，后续仅展示外层结束后的结果)  黄色代表正在比较并排序的两个元素，绿色代表已经完成排序的元素   |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 3 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 3 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 3 | 1 | 4 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 3 | 1 | 4 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 3 | 1 | 4 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 3 | 1 | 4 | 5 | 2 | 9 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 3 | 1 | 4 | 5 | 2 | 6 | 9 | 5 | 3 | 5 | 8 | 9 | | 1 | 3 | 1 | 4 | 5 | 2 | 6 | 5 | 9 | 3 | 5 | 8 | 9 | | 1 | 3 | 1 | 4 | 5 | 2 | 6 | 5 | 3 | 9 | 5 | 8 | 9 | | 1 | 3 | 1 | 4 | 5 | 2 | 6 | 5 | 3 | 5 | 9 | 8 | 9 | | 1 | 3 | 1 | 4 | 5 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | 9 | | 1 | 3 | 1 | 4 | 5 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | 9 | | 1 | 1 | 3 | 4 | 2 | 5 | 5 | 3 | 5 | 6 | 8 | 9 | 9 | | 1 | 1 | 3 | 4 | 2 | 5 | 5 | 3 | 5 | 6 | 8 | 9 | 9 | | 1 | 1 | 3 | 2 | 4 | 5 | 3 | 5 | 5 | 6 | 8 | 9 | 9 | | 1 | 1 | 2 | 3 | 4 | 3 | 5 | 5 | 5 | 6 | 8 | 9 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 | 8 | 9 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 | 8 | 9 | 9 |   2) 选择排序(仅展示第一次内层循环找到最小值的过程)  蓝色代表当前遍历到的元素，黄色代表当前遍历过程中最小元素，绿色代表已完成排序元素，红色代表被交换到后方的元素   |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 3 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 1 | 4 | 3 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 5 | 9 | 4 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 5 | 9 | 4 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 9 | 4 | 6 | 5 | 5 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 9 | 6 | 5 | 5 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 6 | 9 | 5 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 9 | 6 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 | 9 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 | 9 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 | 8 | 9 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 | 8 | 9 | 9 |   3) 快速排序  黄色：基准值；蓝色：小于等于基准值；红色：大于基准值；绿色：已排序   |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 9 | 6 | 5 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 9 | 6 | 5 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 9 | 6 | 5 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 9 | 6 | 5 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 9 | 6 | 5 | 5 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 9 | 6 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 9 | 6 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 9 | 6 | 8 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 | 8 | 9 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 | 8 | 9 | 9 | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 | 8 | 9 | 9 |   4) 计数排序  计数数组：   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | key | 1 | 2 | 3 | 4 | 5 | 6 | 8 | 9 | | value | 2 | 1 | 2 | 1 | 3 | 1 | 1 | 2 |   从计数数组中取得结果：   |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  | | 1 | 1 | 2 |  |  |  |  |  |  |  |  |  |  | | 1 | 1 | 2 | 3 | 3 |  |  |  |  |  |  |  |  | | 1 | 1 | 2 | 3 | 3 | 4 |  |  |  |  |  |  |  | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 |  |  |  |  | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 |  |  |  | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 | 8 |  |  | | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 6 | 8 | 9 | 9 |   3. 选择你认为最好的一种排序算法，用自己熟悉的编程语言实现。  使用JavaScript语言，编写in-place版本的快速排序算法  环境：VSCode + Node.js 14.14  代码截图：    代码分析：  代码含义如图中注释所示  时间复杂度分析：由于含有未知层数的递归调用(受输入数据影响)，快速排序的时间复杂度平均为O(nlogn)，最坏为O(n^2)  空间复杂度分析：in-place版本的快速排序不需要额外空间开销，仅在递归调用中会有调用栈的内存开销，约为O(logn) | | | | | |
| 四、测试数据和执行结果 （在给定数据下，执行操作、算法和程序的结果，可使用数据、图表、截图等给出）  测试用数组和输出语句截图：    测试用例分别选择：  1. 教师给定的圆周率前13位数组  2. 本人学号  3. 本人QQ号码  控制台输出结果截图(数组格式化换行输出为VSCode自动进行)： | | | | | |
| 五、实验结果分析及总结（对实验的结果是否达到预期进行分析，总结实验的收获和存在的问题等）  通过本次实验，我对排序算法有了较好的掌握，包括：  1. 了解了常用的排序算法，如冒泡排序、选择排序、插入排序等，知道其原理和性质，能用所学知识对其时间复杂度、空间复杂度、是否改变原有顺序等性质进行分析并得出结果，并对其进行比较，在不同场景下能选择不同的排序算法来优化效率；  2. 通过实例对几个常见排序算法的运行过程进行了分析和图形化展示；  3. 使用JavaScript语言实现了一种in-place快速排序算法，使我对快速排序有了更加深刻的认识 | | | | | |
| 教  师  评  阅 | 实验内容和设计（A-E）： | | |  | |
| 操作过程、算法或代码（A-E）： | | |  | |
| 实验结果（A-E）： | | |  | |
| 实验分析和总结（A-E）： | | |  | |
| 实验成绩（A-E）：  反馈评语： | | | | |