**安徽大学集成电路学院**

**《算法设计与分析》第三次报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 算法设计与分析 | | | 日期 |  |
| 专业 | 集成电路设计与集成系统 | 学号 | WB2224186 | 姓名 | 邵宇鹏 |
| 【实验目的】  1. 理解快速排序算法的基本原理和核心思想，包括分治策略的应用和递归算法的实现。  2. 掌握使用Matlab语言编写快速排序算法的程序，通过编程实践加深对算法逻辑的理解。  3. 学会分析快速排序算法的时间复杂度和空间复杂度，理解其在不同数据集上的性能表现。  4. 通过实验数据的记录与处理，验证快速排序算法的正确性和效率，比较不同输入规模下的运行时间。  5. 提高问题解决能力，通过实验过程中可能遇到的问题，学会调试程序和优化算法。 | | | | | |
| 【实验原理（预习）】  实验原理：  快速排序（QuickSort）是一种高效的排序算法，它采用了分治（Divide and Conquer）的策略来对一个序列进行排序。其基本思想是通过一趟排序将待排序的记录分隔成独立的两部分，其中一部分记录的关键字均比另一部分的关键字小，然后分别对这两部分记录继续进行排序，以达到整个序列有序的目的。  快速排序的核心步骤如下：  1. 选择基准（Pivot）：从待排序的序列中选择一个元素作为基准，通常选择第一个或最后一个元素，也可以选择随机元素或采用三数取中法来减少最坏情况的发生。  2. 分割（Partition）：通过一趟排序将序列分割成两部分，使得左边的元素都小于或等于基准，右边的元素都大于基准。这个过程称为分割操作，是快速排序的关键步骤。  3. 递归排序：对分割后的两个子序列分别递归地应用快速排序算法，直到子序列的长度为1或0，此时子序列已经有序。  快速排序的分割操作通常采用以下步骤：   * 设置两个指针，一个指向序列的起始位置（左指针），另一个指向序列的结束位置（右指针）。 * 左指针向右移动，直到找到一个大于基准的元素；右指针向左移动，直到找到一个小于基准的元素。 * 如果左指针的位置小于右指针的位置，则交换这两个指针所指向的元素。 * 重复上述步骤，直到左指针的位置大于或等于右指针的位置。 * 将基准元素与左指针所指向的元素交换，此时基准元素已经放置在其最终的正确位置上。   快速排序的时间复杂度为，在平均情况下表现良好，但在最坏情况下（如输入序列已经有序或逆序），时间复杂度会退化为。为了改善最坏情况下的性能，可以采用随机选择基准或三数取中等优化策略。  通过本次实验，我将深入理解快速排序算法的原理，并通过Matlab编程实现该算法，验证其在不同数据集上的性能表现。 | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 【实验内容与记录（题号、操作步骤、数据记录与处理、附图编号、代码等）】  matlab代码   |  | | --- | | Quicksort.m | | function A = Quicksort(A, p, q)      if p < q          [A, r] = Partition(A, p, q);          A = Quicksort(A, p, r - 1);          A = Quicksort(A, r + 1, q);      end  end |      |  | | --- | | Partition.m | | function [A, j] = Partition(A, p, r)      x = A(r);      i = p - 1;      for j = p:r - 1          if A(j) <= x              i = i + 1;              temp = A(i);              A(i) = A(j);              A(j) = temp;          end      end      temp = A(i + 1);      A(i + 1) = A(r);      A(r) = temp;      j = i + 1;  end |     在编写好排序算法的代码后，需要验证算法编写的是否正确，我重新编写一个m文件如下   |  | | --- | | Input.m | | A = randi(100,1,10);  A=Quicksort(A,1,length(A)) |     得到了正确的结果 |

|  |
| --- |
| 【小结与讨论】  本次实验通过Matlab语言实现了快速排序算法，并对其进行了测试和分析小结与讨论：  我成功地使用Matlab编写了一个快速排序函数，该函数能够对输入的数组进行排序。在实现过程中，我重点关注了基准的选择和分割操作的正确性，确保了算法的正确执行。  算法性能测试：测试了快速排序算法在不同规模和不同分布的数据集上的性能。通过记录排序前后的数组状态和运行时间，我验证了算法的正确性和效率。  时间复杂度分析：实验结果表明，快速排序算法在平均情况下的时间复杂度为，这与理论分析相符。然而，在最坏情况下（如输入数组已经有序），算法的时间复杂度退化为，这提示我在实际应用中可能需要考虑算法的优化。  通过本次实验，我不仅加深了对快速排序算法原理的理解，还提高了编程实现和算法分析的能力。实验过程中的问题解决和讨论也为我提供了宝贵的经验，有助于在未来的学习和研究中更好地应用和优化算法。有一个元素,然后再通过合并有序子序列的方式实现最终排序 |