重 庆 大 学

学 生 实 验 报 告

实验课程名称 算法设计与分析

开课实验室 DS1501

学 院 大数据与软件学院 年级 2023 专业班 软件 01 班

学 生 姓 名 学 号

开 课 时 间 2024 至 2025 学年第 2 学期

总 成 绩	
教师签名	付春雷

大数据与软件学院制

《算法设计与分析》实验报告

开课实验室：DS1501 **2025 年 5 月 5 日**

学院	大数据与软件学院	年级、专业、班		姓名		成绩	
课程名称	算法设计与分析	实验项目名称	分治法实验		指导教师	付春雷	
教师评语							

教师签名：

年 月 日

一、 实验目的

- 掌握分治法的设计思想，包括分治法解决的问题特征、分治法的求解过程。
- 熟练掌握“快速排序”算法的实现，提供输入案例检测算法的正确性，分析算法的时间复杂性。
- 熟练掌握“归并排序”算法的实现，提供输入案例检测算法的正确性，分析算法的时间复杂性。

二、使用仪器、材料

PC 微机；
Windows 操作系统，VSCODE+MINGW64 编译环境（不限）；

三、实验步骤

1. 复习教材第3章分治法知识点，包括分治法的设计思想、分治法解决的问题特征、分治法的求解过程。
2. 阅读和掌握教材“第3章 快速排序”，实现快速排序求解算法，提供输入案例检测算法的正确性，分析算法时间复杂性。 。
3. 阅读和掌握教材“第3章 归并排序”，实现归并排序算法，提供输入案例检测算法的正确性，分析算法时间复杂性。

四、实验过程原始记录(数据、图表、计算等)

快速排序：

```
// 快速排序的分区函数
int partition(int arr[], int lef, int right) {
    int pivot = arr[right];  // 选择最后一个元素作为基准
    int i = (lef - 1);       // i是小于基准的元素的索引

    for (int j = lef; j <= right - 1; j++) {
        // 如果当前元素小于基准
        if (arr[j] < pivot) {
            i++;  // 增加小于基准的元素的索引
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[right]);
    return (i + 1);
}

// 快速排序主函数
void quickSort(int arr[], int lef, int right) {
    if (lef < right) {
        // pi是分区索引，arr[pi]现在在正确位置
        int pi = partition(arr, lef, right);

        // 分别对分区前后的子数组进行排序
        quickSort(arr, lef, pi - 1);
        quickSort(arr, pi + 1, right);
    }
}
```

归并排序：

```
    // 合并临时数组回原数组
    int i = 0;    // 初始化左子数组索引
    int j = 0;    // 初始化右子数组索引
    int k = left; // 初始化合并子数组索引

    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    // 拷贝剩余元素
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
```

```
// 归并排序主函数
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        // 找到中间点
        int mid = left + (right - left) / 2;

        // 递归排序左右两部分
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        // 合并已排序的两部分
        merge(arr, left, mid, right);
    }
}
```

五、实验结果及分析

结果都已对应显示在原始数据记录中，结果都与预期的分析符合。

快速排序

```
请输入数组大小和数组内容（整数数组）：
3 21 12 3
排序后：
3 12 21
```

快速排序的时间复杂度，最好和平均为O(nlogn),最坏情况是O（n^2）

归并排序：

```
请输入数组大小和数组内容（整数数组）：
4 12 21 3 22
排序后：
3 12 21 22
```

归并排序的时间复杂度，最好，平均和最坏都为O(nlogn),空间复杂度为O（n）