華南師範大學

本科学生实验（实践）报告

院　　系：计 算 机 学 院

实验课程：　Java 实验

实验项目：　Git 的使用以及排序

指导老师：　陈俊侠

开课时间：2021 ～ 2022 年度第二学期

专　　业：计算机科学与技术

班　　级：计算机科学与技术二班

学　　生：张泽贤

学　　号：20202131119

华南师范大学教务处

题目

Git的使用以及排序

实验目的

掌握Git的使用方法并熟悉各种排序算法的实现

实验主要硬件软件环境

IntelliJ IDEA

实验内容

1. `MyData`类的实现

代码如下:

```
package Experiment.Exp4;

import org.jetbrains.annotations.NotNull;

import java.security.InvalidParameterException;
import java.util.Arrays;
import java.util.Random;

/**
 * MyData类, 用以存放数据
 * @param <N> 任意实现了Comparable接口的Number的子类
 */
public class MyData<N extends Number & Comparable<N>> {
    private Number[] elementData;

    private Number[] wingMan;

    private final long[] countMain;

    private final long[] countWingMan;

    private final int size;

    public static final int MAIN = 0, WINGMAN = 1;

    /**
     * 构造器
     * @param size 默认大小
     */
    public MyData(int size) {
        this.size = size;
        elementData = new Number[size];
        wingMan = new Number[size];
        countMain = new long[4];
```

```
        countWingMan = new long[4];
        for (int i = 0; i < 4; ++i)
            countMain[i] = 0;
    }

    /**
     * 一个没什么意义的默认构造器, 调用MyData(0)实现
     */
    public MyData() {
        this(0);
    }

    @Deprecated
    public MyData<N> getWingMan() {
        return new MyData<>(size);
    }

    /**
     * 随机填充数组, 不指定范围
     */
    public long fill() {
        long seed = System.currentTimeMillis();
        Random random = new Random(seed);
        for (int i = 0; i < size; ++i) {
            elementData[i] = random.nextInt();
        }
        return seed;
    }

    /**
     * 随机填充数组, 指定随机值的范围
     * @param minValue 随机生成的值的最小范围
     * @param maxValue 随机生成的值的最大范围
     * @return 随机生成器所使用的种子, 便于复现原数组
     */
    public long fill(int minValue, int maxValue) {
        long seed = System.currentTimeMillis();
        Random random = new Random(seed);
        elementData = random.ints(size, minValue,
maxValue).boxed().toArray(Integer[]::new);
        return seed;
    }

    /**
     * 返回数组元素, 忽略检查
     * @param index 元素所在下标
     * @return 强制转换后的T类型实例
     */
    @SuppressWarnings("unchecked")
    private N elementData(int index) {
        return (N) elementData[index];
    }

    /**
     * 用以访问指定下标的辅助数组内容(调用前检查下标合法性)
     * @param index 下标
     * @return 辅助数组下标为index的内容
     */
```

```
    @SuppressWarnings("unchecked")
    private N wingMan(int index) {
        return (N) wingMan[index];
    }

    /**
     * 检查下标是否合法，若否则抛出异常
     * @param index 下标
     */
    private void rangeCheck(int index) {
        if (index < 0 || index >= size)
            throw new IndexOutOfBoundsException();
    }

    /**
     * 获取指定下标的元素(若下标合法)
     * @param index 元素所在下标
     * @return 元素
     */
    public N get(int index) {
        rangeCheck(index);

        ++countMain[0];
        return elementData(index);
    }

    /**
     * 用以访问指定下标的辅助数组内容，访问前检查下标是否合法
     * @param index 下标
     * @return 内容
     */
    public N getWingMan(int index) {
        rangeCheck(index);

        ++countWingMan[0];
        return wingMan(index);
    }

    /**
     * 更改位于给定下标的元素的值(若下标合法)
     * @param index 元素所在下标
     * @param newValue 更改后的值
     * @return 元素原本的值
     */
    public N set(int index, N newValue) {
        rangeCheck(index);

        ++countMain[1];
        N oldValue = elementData(index);
        elementData[index] = newValue;
        return oldValue;
    }

    /**
     * 更改辅助数组中位于给定下标的元素的值(若下标合法)
     * @param index 元素所在下标
     * @param newValue 更改后的值
     * @return 元素原本的值
```

```
     */
    public N setWingMan(int index, N newValue) {
        rangeCheck(index);

        ++countWingMan[1];
        N oldValue = elementData(index);
        wingMan[index] = newValue;
        return oldValue;
    }

    /**
     * 交换位于位置i与j的元素(若i, j均合法)
     * @param i 第一个元素
     * @param j 第二个元素
     */
    public void swap(int i, int j) {
        rangeCheck(i);
        rangeCheck(j);

        ++countMain[2];
        N tmp = elementData(j);
        elementData[j] = elementData[i];
        elementData[i] = tmp;
    }

    /**
     * 交换辅助数组中位于位置i与j的元素(若i, j均合法)
     * @param i 第一个元素
     * @param j 第二个元素
     */
    public void swapWingMan(int i, int j) {
        rangeCheck(i);
        rangeCheck(j);

        ++countWingMan[2];
        N tmp = wingMan(j);
        wingMan[j] = wingMan(i);
        wingMan[i] = tmp;
    }

    /**
     * 调用已经实现的接口比较位于位置i与j的元素(若i, j均合法)
     * @param i 第一个元素
     * @param j 第二个元素
     * @return 比较结果
     */
    public int compare(int i, int j) {
        rangeCheck(i);
        rangeCheck(j);

        ++countMain[3];
        return elementData(i).compareTo(elementData(j));
    }

    /**
     * 比较辅助数组中的元素和主数组中的元素
     * @param posInWingMan 元素位于辅助数组中的位置
     * @param posInMain 元素位于主数组中的位置
```

```
     * @return 比较结果
     */
    public int compareWingManToMain(int posInWingMan, int posInMain) {
        rangeCheck(posInWingMan);
        rangeCheck(posInMain);

        ++countMain[3];
        return wingMan(posInWingMan).compareTo(elementData(posInMain));
    }

    /**
     * 调用已经实现的接口比较辅助数组中位于位置i与j的元素(若i, j均合法)
     * @param i 第一个元素
     * @param j 第二个元素
     * @return 比较结果
     */
    public int compareWingMan(int i, int j) {
        rangeCheck(i);
        rangeCheck(j);

        ++countWingMan[3];
        return wingMan(i).compareTo(wingMan(j));
    }

    /**
     * 获取数组大小
     * @return 数组的大小
     */
    public int getSize() {
        return size;
    }

    /**
     * 将主数组中指定的闭区间[start, end]内的数据拷贝到辅助数组的同等位置中
     * @param start 起点
     * @param end 终点
     */
    public void copyToWingMan(int start, int end) {
        rangeCheck(start);
        rangeCheck(end);

        for (int i = start; i <= end; ++i) {
            setWingMan(i, get(i));
        }
    }

    /**
     * 无参数的copyToWingMan的重载, 将主数组内的所有内容拷贝到辅助数组中
     */
    public void copyToWingMan() {
        copyToWingMan(0, size - 1);
    }

    /**
     * 将辅助数组中指定的闭区间[start, end]内的数据拷贝到主数组的同等位置中
     * @param start 起点
     * @param end 终点
     */
```

```
    public void copyToMain(int start, int end) {
        rangeCheck(start);
        rangeCheck(end);

        for (int i = start; i <= end; ++i) {
            set(i, getWingMan(i));
        }
    }

    /**
     * 无参数的copyToMain的重载, 将辅助数组内的所有内容拷贝到主数组中
     */
    public void copyToMain() {
        copyToMain(0, size - 1);
    }

    /**
     * 检查主数组内容是否与给定数组的内容相等
     * @param sortedArray 给定的数组, 一般是已经排好序的
     * @return 比较结果
     */
    public boolean check(N @NotNull [] sortedArray) {
        if (sortedArray.length != size)
            throw new InvalidParameterException();

        for (int i = 0; i < size; ++i) {
            if (!sortedArray[i].equals(elementData(i)))
                return false;
        }
        return true;
    }

    /**
     * 获取指定数组类型的计数数组
     * @param arrayType 数组类型
     * @return 返回计数数组的一个拷贝
     */
    public long[] getCount(int arrayType) {
        if (arrayType > 1)
            throw new IndexOutOfBoundsException();

        if (arrayType == MAIN) {
            return Arrays.copyOf(countMain, 4);
        }
        return Arrays.copyOf(countWingMan, 4);
    }

    /**
     * 清除所有计数器
     */
    public void clearCount() {
        clearCountMain();
        clearCountWingMan();
    }

    /**
     * 清除主计数器
     */
```

```
    public void clearCountMain() {
        for (int i = 0; i < 4; ++i)
            countMain[i] = 0;
    }

    /**
     * 清除辅助计数器
     */
    public void clearCountWingMan() {
        for (int i = 0; i < 4; ++i)
            countWingMan[i] = 0;
    }
}
```

2. `MySort`类的实现

```
package Experiment.Exp4;

import org.jetbrains.annotations.Contract;
import org.jetbrains.annotations.NotNull;

/**
 * 工具类，不具有任何非静态数据成员，只作为排序方法的包装，无需实例化
 */
public class MySort {
    public static final int BUBBLE = 0, SELECTION = 1, INSERTION = 2, MERGE = 3,
MERGE_RECURSIVE = 4;

    public static final int[] sortType = {BUBBLE, SELECTION, INSERTION, MERGE,
MERGE_RECURSIVE};

    /**
     * 冒泡排序
     * @param array 待排序的数组
     * @param <T> MyData或其派生类
     * @return 一个Result类的实例，记录了结果
     */
    @Contract("_ -> new")
    public static <N extends Number & Comparable<N>, T extends MyData<N>>
@NotNull Result BubbleSort(@NotNull T array) {
        final long startTime = System.currentTimeMillis();
        int size = array.getSize();
        for (int i = 0; i < size; ++i) {
            int pos = 0;
            while (pos < size - 1) {
                if (array.compare(pos, pos + 1) > 0) {
                    array.swap(pos, pos + 1);
                }
                ++pos;
            }
        }
        return new Result(array, startTime);
    }

    /**
     * 选择排序
```

```
     * @param array 待排序的数组
     * @param <T> MyData或其派生类
     * @return 一个Result类的实例, 记录了结果
     */
    @Contract("_ -> new")
    public static <N extends Number & Comparable<N>, T extends MyData<N>>
@NotNull Result SelectionSort(@NotNull T array) {
        final long startTime = System.currentTimeMillis();
        int size = array.getSize();

        array.copyToWingMan();
        for (int i = 0; i < size; ++i) {
            int smallest = i;
            for (int j = i + 1; j < size; ++j) {
//                if (array.getWingMan(smallest).compareTo(array.get(j)) > 0)
                if (array.compareWingMan(smallest, j) > 0)
                    smallest = j;
            }
            array.swapWingMan(i, smallest);
        }
        array.copyToMain();
        return new Result(array, startTime);
    }

    /**
     * 插入排序
     * @param array 待排序的数组
     * @param <T> MyData或其派生类
     * @return 一个Result类的实例, 记录了结果
     */
    @Contract("_ -> new")
    public static <N extends Number & Comparable<N>, T extends MyData<N>>
@NotNull Result InsertionSort(@NotNull T array) {
        final long startTime = System.currentTimeMillis();
        int size = array.getSize();

        array.setWingMan(0, array.get(0));
        for (int i = 1; i < size; ++i) {
            //如果当前数字小于有序数组内的所有数, 则将有序数组全部后移一个并将这个数插入至有序
数组的开头
            //if (array.getWingMan(0).compareTo(array.get(i)) >= 0) {
            if (array.compareWingManToMain(0, i) >= 0) {
                for (int k = i - 1; k >= 0; --k) {
                    array.swapWingMan(k, k + 1);
                }
                array.setWingMan(0, array.get(i));
                continue;
            }

            for (int j = 0; j < i; ++j) {
                //否则一直寻找, 直到第j个数小于待插入数字
                if (array.compareWingManToMain(j, i) < 0) {
                    //且j已经是有序数组的最后一个, 则直接将其插入至有序数组末尾
                    if (j == i - 1) {
                        array.setWingMan(i, array.get(i));
                        break;
                    }
                    //或者第j个数小于待插入数字且第j+1个数大于等于待插入数字
```

```
                    //则将第j+1个数及之后的所有的数字往后移一位，再将待插入数字插入至下标
j+1
                    if (array.compareWingManToMain(j + 1, i) >= 0) {
                        for (int k = i - 1; k > j; --k) {
                            array.swapWingMan(k, k + 1);
                        }
                        array.setWingMan(j + 1, array.get(i));
                        break;
                    }
                }
            }
        }

        array.copyToMain();
        return new Result(array, startTime);
    }

    /**
     * 从外部调用的归并排序方法，并不实现任何功能，用以分离递归部分
     * @param array 待排序数组
     * @param <T> MyData或其派生类
     * @return 一个Result类的实例，记录了结果
     */
    public static <N extends Number & Comparable<N>, T extends MyData<N>>
@NotNull Result MergeSortRecursive(@NotNull T array) {
        final long startTime = System.currentTimeMillis();
        MergeSortWorker(array, 0, array.getSize() - 1);
        return new Result(array, startTime);
    }

    /**
     * 归并排序递归实现的工作函数
     * @param array 待排序数组
     * @param start 需要排序的区间起始位置
     * @param end 需要排序的区间末尾位置
     * @param <T> MyData或其派生类
     */
    private static <N extends Number & Comparable<N>, T extends MyData<N>> void
MergeSortWorker(T array, int start, int end) {
        //递归边界条件，如果数组长度为1则无需排序
        if (start == end)
            return;

        //辅助变量，用来待排序区间的记录长度
        final int length = end - start + 1;
        //将待排序区间分为长度相等或相差1的两个区间，左区间的端点分别为ls和le，右区间的端点为
rs和re(end)
        //同时ls和rs记录左右区间未排序的第一个数的位置
        int ls = start, le = start + length / 2 - 1, rs = le + 1;

        //首先将左区间和右区间分别排序
        MergeSortWorker(array, ls, le);
        MergeSortWorker(array, rs, end);

        //而后分别比较左区间和右区间的第一个待排序的数，将较小的插入到有序数组内
        for (int i = 0; i < length; ++i) {
            //如果左区间或右区间已经取完，则直接将另外一个区间内剩余的数直接按顺序插入
            if (ls > le) {
```

```
                array.setWingMan(start + i, array.get(rs++));
                continue;
            }
            if (rs > end) {
                array.setWingMan(start + i, array.get(ls++));
                continue;
            }

            //否则比较左区间和右区间的第一个待排序的数，将更小的那个插入到有序数组内
            if (array.compare(ls, rs) <= 0) {
                array.setWingMan(start + i, array.get(ls++));
            } else {
                array.setWingMan(start + i, array.get(rs++));
            }
        }

        //最后将排序好的有序数组拷贝到主数组中
        array.copyToMain(start, end);
    }

    /**
     * 非递归的归并排序
     * @param array 待排序数组
     * @param <T> MyData或其派生类
     * @return 一个Result类的实例，记录了结果
     */
    @Contract("_ -> new")
    public static <N extends Number & Comparable<N>, T extends MyData<N>>
@NotNull Result MergeSortNonRecursive(@NotNull T array) {
        final long startTime = System.currentTimeMillis();
        int len = 2, size = array.getSize();
        //从大小为2的区间开始排序，直到排序的区间的大小与数组大小相等或更大时进行最后一次排序，
而后终止循环
        //因此当排序区间大小len >= size时将flag标记为true，进行最后一次排序
        //排序结束后，若flag == true，则退出循环
        boolean flag = false;

        while (true) {
            //首先枚举各个待排序区间的起点start
            for (int start = 0; start < size; start += len) {
                //由于可能出现最后一个待排序区间的终点大于数组终点的情况，即len > size的情
况
                //因此在每个小区间内比较时需将比较的终点设置为start+len-1和size-1中较小的
那个
                final int end = Math.min(start + len - 1, size - 1);
                //剩余部分与递归实现没有差别，不再赘述
                int ls = start, le = ls + len / 2 - 1, rs = le + 1;
                if (start == end)
                    continue;
                for (int j = 0; j < Math.min(len, size - start); ++j) {
                    if (ls > le) {
                        array.setWingMan(start + j, array.get(rs++));
                        continue;
                    }
                    if (rs > end) {
                        array.setWingMan(start + j, array.get(ls++));
                        continue;
                    }
```

```
                    if (array.compare(ls, rs) <= 0) {
                        array.setWingMan(start + j, array.get(ls++));
                    } else {
                        array.setWingMan(start + j, array.get(rs++));
                    }
                }

                array.copyToMain(start, end);
            }

            //排序完成后将len乘以2
            len *= 2;
            //如果flag == true则结束循环
            //即此时已经完成了一次len >= size(因为此时flag == true)的排序
            //因此可以直接结束循环
            if (flag)
                break;
            //否则, 如果原本的len * 2 >= size, 则需进行最后一次排序
            //将flag设置为true, 完成最后一次排序后退出
            if (len >= size)
                flag = true;
        }
        return new Result(array, startTime);
    }

    /**
     * 通过给定的排序类型调用对应函数进行排序
     * @param array 待排序数组
     * @param operationType 排序类型, 输入值应在闭区间[0, 4]之中
     * @param <T> MyData或其派生类
     * @return 一个Result类的实例, 记录了结果
     */
    public static <N extends Number & Comparable<N>, T extends MyData<N>>
@NotNull Result sort(@NotNull T array, int operationType) {
        switch (operationType) {
            case BUBBLE:
                return BubbleSort(array);
            case SELECTION:
                return SelectionSort(array);
            case INSERTION:
                return InsertionSort(array);
            case MERGE:
                return MergeSortNonRecursive(array);
            case MERGE_RECURSIVE:
                return MergeSortRecursive(array);
            default:
                throw new IndexOutOfBoundsException();
        }
    }
}
```

3. `Result`类的实现

```
package Experiment.Exp4;
```

```java
import org.jetbrains.annotations.NotNull;

/**
 * 结果类，便于输出结果
 */
class Result {
    private final long[] countMain;

    private final long[] countWingMan;

    private final long timeCostMillis;

    public static final int MAIN = 0, WINGMAN = 1;

    public static final int GET = 0, SET = 1, SWAP = 2, COMPARE = 3;

    /**
     * 构造函数，在MySort中使用
     * @param array 被排序的数组
     * @param startTimeMillis 排序操作的最开始的时间
     * @param <T> MyData类或其派生类
     */
    <T extends MyData<?>> Result(@NotNull T array, long startTimeMillis) {
        countMain = array.getCount(MAIN);
        countWingMan = array.getCount(WINGMAN);
        timeCostMillis = System.currentTimeMillis() - startTimeMillis;
    }

    /**
     * 获取指定类型的指定操作类型的计数
     * @param arrayType 数组类型
     * @param operationType 操作类型
     * @return 返回计数
     */
    public long getCount(int arrayType, int operationType) {
        if (arrayType > 1 || operationType > 3)
            throw new IndexOutOfBoundsException();

        if (arrayType == MAIN) {
            return countMain[operationType];
        }
        return countWingMan[operationType];
    }

    /**
     * 获取指定操作的计数的和
     * @param operationType 操作类型
     * @return 总数
     */
    public long getCount(int operationType) {
        if (operationType > 3)
            throw new IndexOutOfBoundsException();

        return countMain[operationType] + countWingMan[operationType];
    }

    /**
     * 获取排序所花费时间
```

```
     * @return 排序所花费时间
     */
    public long getTimeCostMillis() {
        return timeCostMillis;
    }
}
```

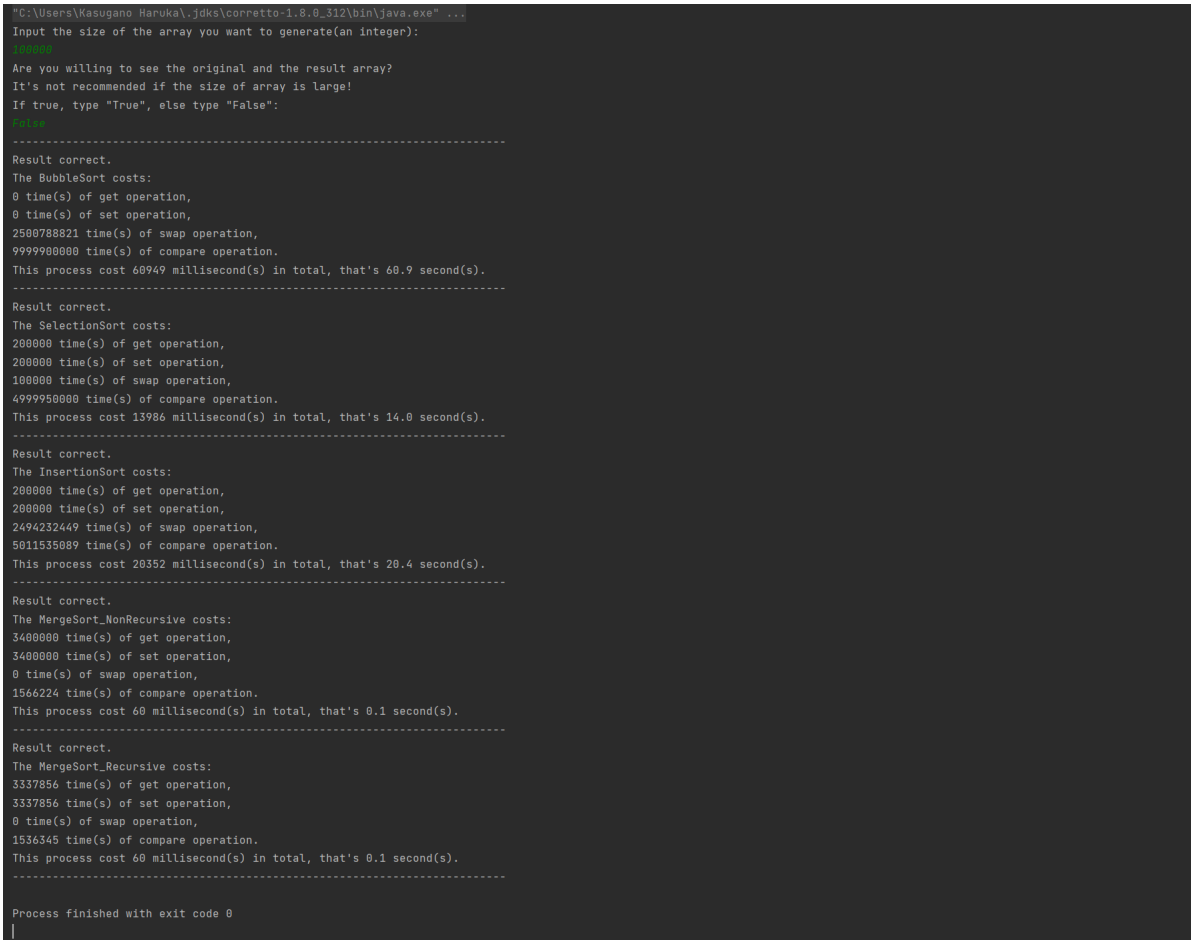
4. `MyTest`类的实现

```
package Experiment.Exp4;

import org.jetbrains.annotations.NotNull;

import java.util.Random;
import java.util.Scanner;

public class MyTest {
    static Scanner scanner = new Scanner(System.in);
    static final String[] strings =
            {"BubbleSort", "SelectionSort", "InsertionSort",
"MergeSort_NonRecursive", "MergeSort_Recursive"};

    /**
     * 用以输出排序的各种结果
     * @param result MySort返回的结果
     */
    static void output(@NotNull Result result) {
        System.out.printf("%d time(s) of get operation,\n" +
                "%d time(s) of set operation,\n" +
                "%d time(s) of swap operation,\n" +
                "%d time(s) of compare operation.\n" +
                "This process cost %d millisecond(s) in total, that's %.1f
second(s).\n",
                result.getCount(Result.GET), result.getCount(Result.SET),
                result.getCount(Result.SWAP), result.getCount(Result.COMPARE),
                result.getTimeCostMillis(), 1.0 * result.getTimeCostMillis() /
1000);
    }

    /**
     * 用以输出原本的数组
     * @param array 原本的数组
     */
    static void output(Integer @NotNull [] array) {
        for (int i : array) {
            System.out.printf("%d ", i);
        }
        System.out.println();
    }

    /**
     * 将排序后的MyData类输出
     * @param array MyData类的实例，应已排序
     */
    static void output(@NotNull MyData<Integer> array) {
        for (int i = 0; i < array.getSize(); ++i) {
```

```
            System.out.printf("%d ", array.get(i));
        }
        System.out.println();
    }

    /**
     * 工作函数，用于精简主函数
     * @param sort 排序类型
     * @param arraySize 数组大小
     * @param output 是否将原数组和排序后的数组输出，如果值为True则输出
     */
    static void worker(int sort, int arraySize, boolean output) {
        MyData<Integer> array = new MyData<>(arraySize);
        final int minValue = 0, maxValue = 10000;

        /*
        通过传回的种子创造Random实例，调用ints()方法获得IntStream，再调用boxed()方法获得
Stream<Integer>，最后调用toArray(Integer[]::new)获得原数组，
        使用Stream<Integer>的sorted()方法获取正确排序后的Stream<Integer>便于确认排序是否
正确
        每次生成原数组都应使用新的Random实例，否则无法复原
         */
        long seed = array.fill(minValue, maxValue);
        if (output)
            output(new Random(seed).ints(arraySize, minValue,
maxValue).boxed().toArray(Integer[]::new));
        Integer[] sortedArray = new Random(seed).ints(arraySize, minValue,
maxValue).boxed().sorted().toArray(Integer[]::new);

        Result result = MySort.sort(array, sort);
        if (output)
            output(array);
        if (array.check(sortedArray))
            System.out.println("Result correct.");
        else
            System.out.println("Result incorrect.");

        //输出排序所使用的各种操作的次数
        System.out.printf("The %s costs: \n", strings[sort]);
        output(result);
        System.out.println("----------------------------------------------------
----------------------");
    }

    public static void main(String[] args) {
        System.out.println("Input the size of the array you want to generate(an
integer): ");
        final int arraySize = scanner.nextInt();

        System.out.println("Are you willing to see the original and the result
array?\n" +
                "It's not recommended if the size of array is large!\n" +
                "If true, type \"True\", else type \"False\": ");
        final boolean output = scanner.nextBoolean();
        System.out.println("----------------------------------------------------
----------------------");

        for (int sort : MySort.sortType) {
```

```
                worker(sort, arraySize, output);
            }
        }
    }
```

5. 运行截图



6. 使用Git提交代码

```
PS C:\Users\Kasugano Haruka\IdeaProjects\JavaHomework> git add .
warning: LF will be replaced by CRLF in
doc/Experiment4/jquery/external/jquery/jquery.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in doc/Experiment4/jquery/jquery-3.5.1.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in doc/Experiment4/jquery/jquery-ui.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in doc/Experiment4/jquery/jquery-ui.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in doc/Experiment4/jquery/jquery-
ui.min.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in doc/Experiment4/jquery/jquery-ui.min.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in doc/Experiment4/jquery/jquery-
ui.structure.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in doc/Experiment4/jquery/jquery-
ui.structure.min.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in doc/Experiment4/jquery/jszip-
utils/dist/jszip-utils-ie.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in doc/Experiment4/jquery/jszip-
utils/dist/jszip-utils-ie.min.js.
The file will have its original line endings in your working directory
```

```
warning: LF will be replaced by CRLF in doc/Experiment4/jquery/jszip-
utils/dist/jszip-utils.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in doc/Experiment4/jquery/jszip-
utils/dist/jszip-utils.min.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in
doc/Experiment4/jquery/jszip/dist/jszip.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in
doc/Experiment4/jquery/jszip/dist/jszip.min.js.
The file will have its original line endings in your working directory
PS C:\Users\Kasugano Haruka\IdeaProjects\JavaHomework> git commit -m "Add doc."
[master 30b89cb] Add doc.
 73 files changed, 45400 insertions(+), 8 deletions(-)
 create mode 100644 doc/Experiment4/Experiment/Exp4/MyData.html
 create mode 100644 doc/Experiment4/Experiment/Exp4/MySort.html
 create mode 100644 doc/Experiment4/Experiment/Exp4/MyTest.html
 create mode 100644 doc/Experiment4/Experiment/Exp4/Result.html
 create mode 100644 doc/Experiment4/Experiment/Exp4/class-use/MyData.html
 create mode 100644 doc/Experiment4/Experiment/Exp4/class-use/MySort.html
 create mode 100644 doc/Experiment4/Experiment/Exp4/class-use/MyTest.html
 create mode 100644 doc/Experiment4/Experiment/Exp4/class-use/Result.html
 create mode 100644 doc/Experiment4/Experiment/Exp4/package-summary.html
 create mode 100644 doc/Experiment4/Experiment/Exp4/package-tree.html
 create mode 100644 doc/Experiment4/Experiment/Exp4/package-use.html
 create mode 100644 doc/Experiment4/allclasses-index.html
 create mode 100644 doc/Experiment4/allclasses.html
 create mode 100644 doc/Experiment4/allpackages-index.html
 create mode 100644 doc/Experiment4/constant-values.html
 create mode 100644 doc/Experiment4/deprecated-list.html
 create mode 100644 doc/Experiment4/element-list
 create mode 100644 doc/Experiment4/help-doc.html
 create mode 100644 doc/Experiment4/index-files/index-1.html
 create mode 100644 doc/Experiment4/index-files/index-10.html
 create mode 100644 doc/Experiment4/index-files/index-11.html
 create mode 100644 doc/Experiment4/index-files/index-12.html
 create mode 100644 doc/Experiment4/index-files/index-2.html
 create mode 100644 doc/Experiment4/index-files/index-3.html
 create mode 100644 doc/Experiment4/index-files/index-4.html
 create mode 100644 doc/Experiment4/index-files/index-5.html
 create mode 100644 doc/Experiment4/index-files/index-6.html
 create mode 100644 doc/Experiment4/index-files/index-7.html
 create mode 100644 doc/Experiment4/index-files/index-8.html
 create mode 100644 doc/Experiment4/index-files/index-9.html
 create mode 100644 doc/Experiment4/index.html
 create mode 100644 doc/Experiment4/jquery/external/jquery/jquery.js
 create mode 100644 doc/Experiment4/jquery/images/ui-
bg_glass_55_fbf9ee_1x400.png
 create mode 100644 doc/Experiment4/jquery/images/ui-
bg_glass_65_dadada_1x400.png
 create mode 100644 doc/Experiment4/jquery/images/ui-
bg_glass_75_dadada_1x400.png
 create mode 100644 doc/Experiment4/jquery/images/ui-
bg_glass_75_e6e6e6_1x400.png
 create mode 100644 doc/Experiment4/jquery/images/ui-
bg_glass_95_fef1ec_1x400.png
```

```
 create mode 100644 doc/Experiment4/jquery/images/ui-bg_highlight-
soft_75_cccccc_1x100.png
 create mode 100644 doc/Experiment4/jquery/images/ui-icons_222222_256x240.png
 create mode 100644 doc/Experiment4/jquery/images/ui-icons_2e83ff_256x240.png
 create mode 100644 doc/Experiment4/jquery/images/ui-icons_454545_256x240.png
 create mode 100644 doc/Experiment4/jquery/images/ui-icons_888888_256x240.png
 create mode 100644 doc/Experiment4/jquery/images/ui-icons_cd0a0a_256x240.png
 create mode 100644 doc/Experiment4/jquery/jquery-3.5.1.js
 create mode 100644 doc/Experiment4/jquery/jquery-ui.css
 create mode 100644 doc/Experiment4/jquery/jquery-ui.js
 create mode 100644 doc/Experiment4/jquery/jquery-ui.min.css
 create mode 100644 doc/Experiment4/jquery/jquery-ui.min.js
 create mode 100644 doc/Experiment4/jquery/jquery-ui.structure.css
 create mode 100644 doc/Experiment4/jquery/jquery-ui.structure.min.css
 create mode 100644 doc/Experiment4/jquery/jszip-utils/dist/jszip-utils-ie.js
 create mode 100644 doc/Experiment4/jquery/jszip-utils/dist/jszip-utils-
ie.min.js
 create mode 100644 doc/Experiment4/jquery/jszip-utils/dist/jszip-utils.js
 create mode 100644 doc/Experiment4/jquery/jszip-utils/dist/jszip-utils.min.js
 create mode 100644 doc/Experiment4/jquery/jszip/dist/jszip.js
 create mode 100644 doc/Experiment4/jquery/jszip/dist/jszip.min.js
 create mode 100644 doc/Experiment4/member-search-index.js
 create mode 100644 doc/Experiment4/member-search-index.zip
 create mode 100644 doc/Experiment4/overview-tree.html
 create mode 100644 doc/Experiment4/package-search-index.js
 create mode 100644 doc/Experiment4/package-search-index.zip
 create mode 100644 doc/Experiment4/resources/glass.png
 create mode 100644 doc/Experiment4/resources/x.png
 create mode 100644 doc/Experiment4/script.js
 create mode 100644 doc/Experiment4/search.js
 create mode 100644 doc/Experiment4/stylesheet.css
 create mode 100644 doc/Experiment4/type-search-index.js
 create mode 100644 doc/Experiment4/type-search-index.zip
 rename "experimentReport/20202131119 \345\274\240\346\263\275\350\264\244
\347\254\254\344\272\214\346\254\241\350\257\225\351\252\214.pdf" =>
"experimentReport/20202131119 \345\274\240\346\263\275\350\264\244
\347\254\254\344\272\214\346\254\241\345\256\236\351\252\214.pdf"(100%)
PS C:\Users\Kasugano Haruka\IdeaProjects\JavaHomework> git push github
Enumerating objects: 102, done.
Counting objects: 100% (102/102), done.
Delta compression using up to 16 threads
Compressing objects: 100% (85/85), done.
Writing objects: 100% (93/93), 302.24 KiB | 1.02 MiB/s, done.
Total 93 (delta 38), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (38/38), completed with 7 local objects.
To github.com:Jobove/JavaHomework.git
   df2978a..30b89cb  master -> master
PS C:\Users\Kasugano Haruka\IdeaProjects\JavaHomework> git push gitee
Enumerating objects: 102, done.
Counting objects: 100% (102/102), done.
Delta compression using up to 16 threads
Compressing objects: 100% (85/85), done.
Writing objects: 100% (93/93), 302.24 KiB | 2.32 MiB/s, done.
Total 93 (delta 38), reused 0 (delta 0), pack-reused 0
remote: Powered by GITEE.COM [GNK-6.2]
To gitee.com:Kasugano_Haruka/JavaHomework.git
   df2978a..30b89cb  master -> master
```

小结

通过妥善使用Git可以便捷地将代码上传到远程服务器托管, 便于代码版本迭代的管理.