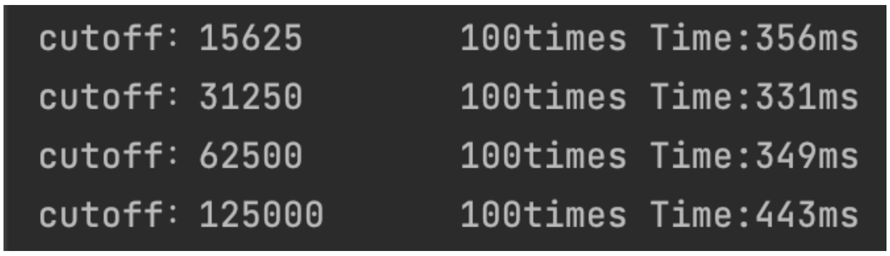
Program Structures and Algorithms
Spring 2023(SEC –8)

NAME: Junlong Qiao
NUID: 002784609

Task: Assignment 5(Parallel Sorting)
Implement a parallel sorting algorithm such that each partition of the array is sorted in parallel. Considering two different schemes for deciding whether to sort in parallel.

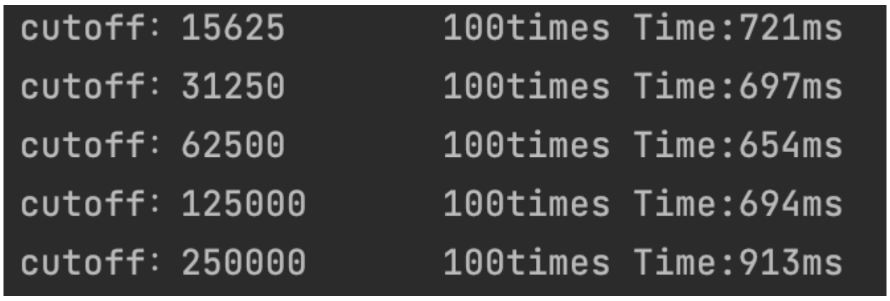
Relationship Conclusion:
Sorting arrays of length 125000 to 4000000 respectively, it is found that the longest time is spent when the cutoff is the length of the array (without use Parallel Sorting), the sorting speed is also longer when the cutoff is too small, and the fastest when the cutoff is a quarter of the length of the array.

Evidence to support that conclusion: Degree of parallelism:8
Array length：125000



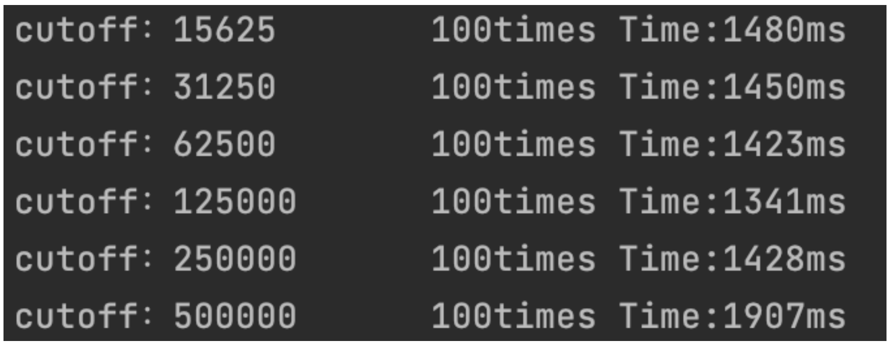
```
cutoff: 15625       100times Time:356ms
cutoff: 31250       100times Time:331ms
cutoff: 62500       100times Time:349ms
cutoff: 125000      100times Time:443ms
```

Array length：250000



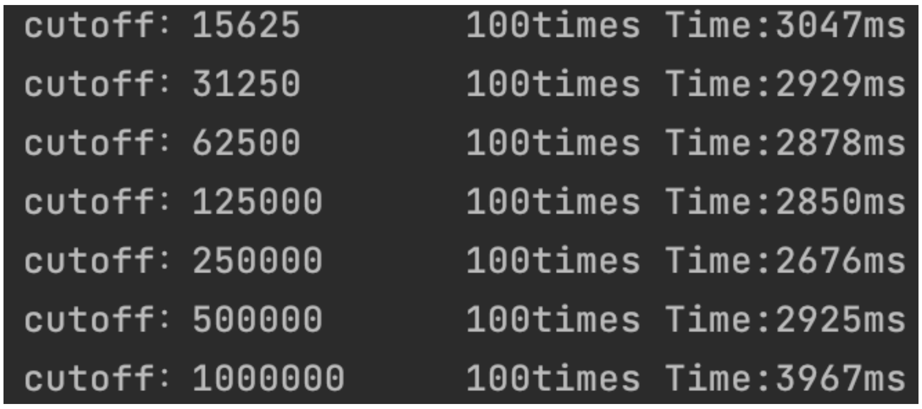
```
cutoff: 15625       100times Time:721ms
cutoff: 31250       100times Time:697ms
cutoff: 62500       100times Time:654ms
cutoff: 125000      100times Time:694ms
cutoff: 250000      100times Time:913ms
```

Array length：500000



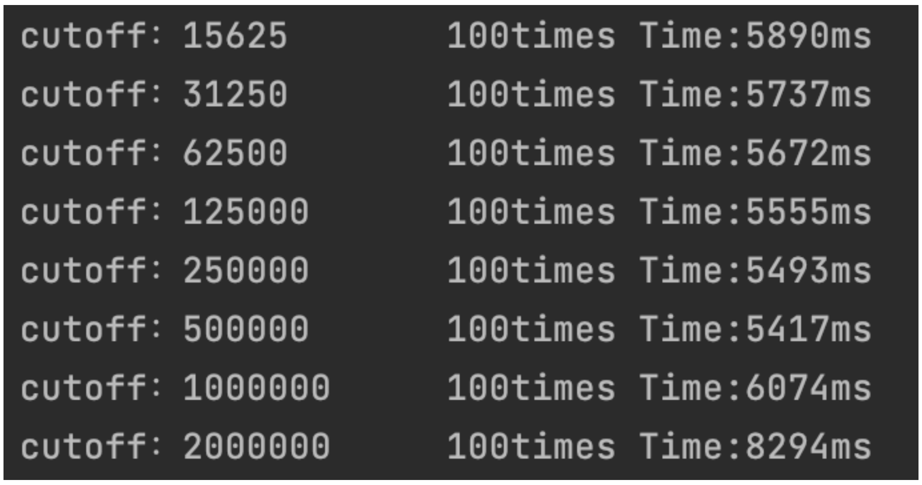
```
cutoff: 15625       100times Time:1480ms
cutoff: 31250       100times Time:1450ms
cutoff: 62500       100times Time:1423ms
cutoff: 125000      100times Time:1341ms
cutoff: 250000      100times Time:1428ms
cutoff: 500000      100times Time:1907ms
```

Array length：1000000



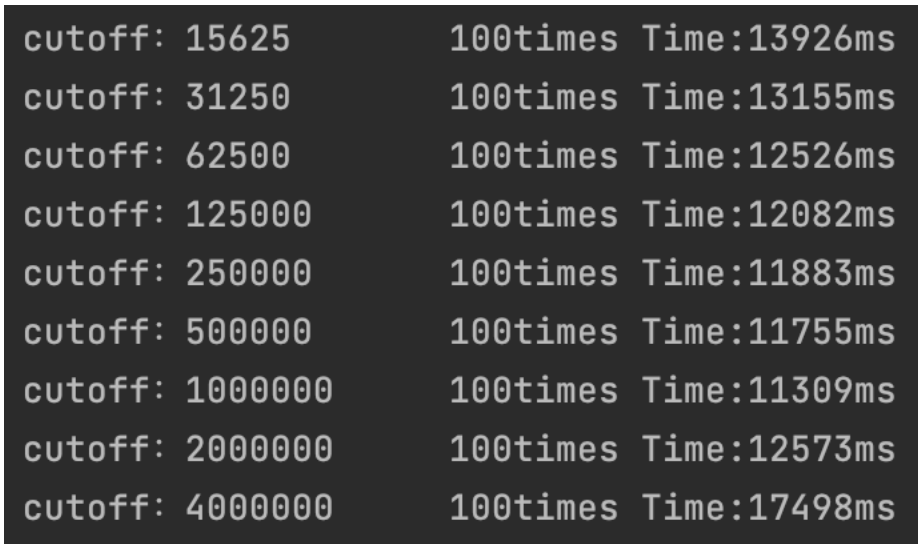
```
cutoff: 15625        100times Time:3047ms
cutoff: 31250        100times Time:2929ms
cutoff: 62500        100times Time:2878ms
cutoff: 125000       100times Time:2850ms
cutoff: 250000       100times Time:2676ms
cutoff: 500000       100times Time:2925ms
cutoff: 1000000      100times Time:3967ms
```

Array length：2000000



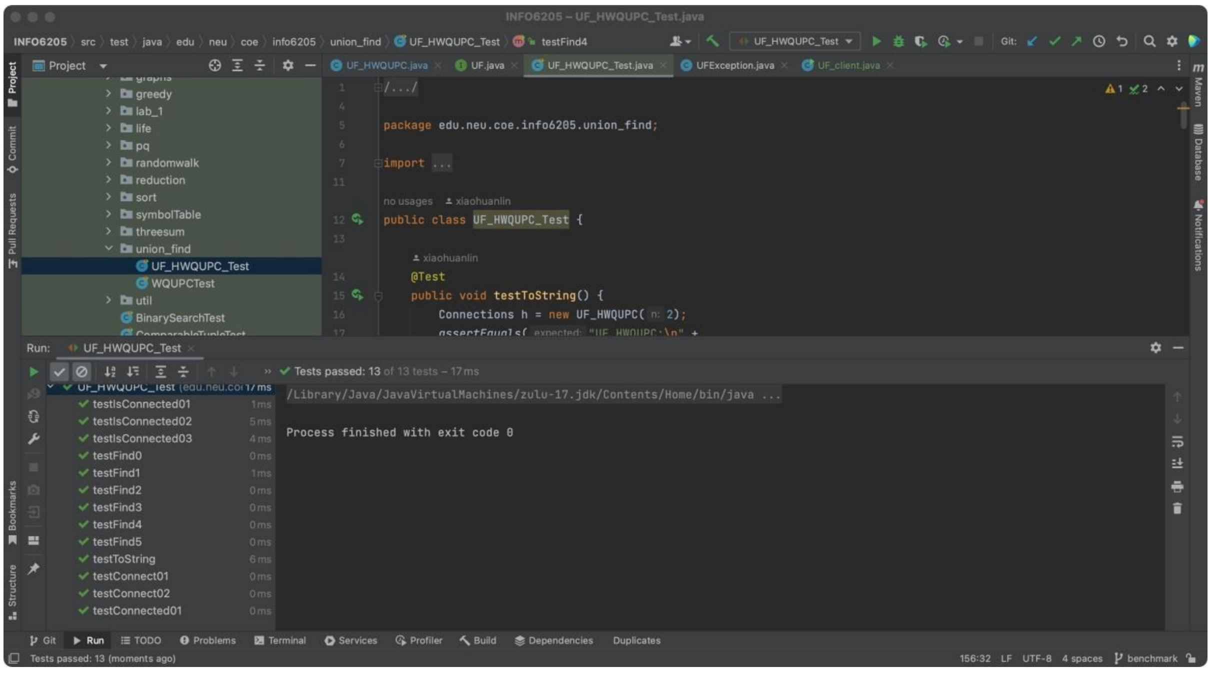
```
cutoff: 15625        100times Time:5890ms
cutoff: 31250        100times Time:5737ms
cutoff: 62500        100times Time:5672ms
cutoff: 125000       100times Time:5555ms
cutoff: 250000       100times Time:5493ms
cutoff: 500000       100times Time:5417ms
cutoff: 1000000      100times Time:6074ms
cutoff: 2000000      100times Time:8294ms
```

Array length：4000000



```
cutoff: 15625        100times Time:13926ms
cutoff: 31250        100times Time:13155ms
cutoff: 62500        100times Time:12526ms
cutoff: 125000       100times Time:12082ms
cutoff: 250000       100times Time:11883ms
cutoff: 500000       100times Time:11755ms
cutoff: 1000000      100times Time:11309ms
cutoff: 2000000      100times Time:12573ms
cutoff: 4000000      100times Time:17498ms
```

Unit Test Screenshots:



```
INFO6205 – UF_HWQUPC_Test.java
INFO6205 › src › test › java › edu › neu › coe › info6205 › union_find › UF_HWQUPC_Test › testFind4
```

```java
/.../

package edu.neu.coe.info6205.union_find;

import ...

no usages  xiaohuanlin
public class UF_HWQUPC_Test {

     xiaohuanlin
    @Test
    public void testToString() {
        Connections h = new UF_HWQUPC( n: 2);
        assertEquals( expected: "UF_HWQUPC:\n" +
```

```
Run: UF_HWQUPC_Test
✓ Tests passed: 13 of 13 tests – 17 ms

✓ UF_HWQUPC_Test (edu.neu.coe 17 ms
  ✓ testIsConnected01   1 ms
  ✓ testIsConnected02   5 ms
  ✓ testIsConnected03   4 ms
  ✓ testFind0           0 ms
  ✓ testFind1           1 ms
  ✓ testFind2           0 ms
  ✓ testFind3           0 ms
  ✓ testFind4           0 ms
  ✓ testFind5           0 ms
  ✓ testToString        6 ms
  ✓ testConnect01       0 ms
  ✓ testConnect02       0 ms
  ✓ testConnected01     0 ms

/Library/Java/JavaVirtualMachines/zulu-17.jdk/Contents/Home/bin/java ...

Process finished with exit code 0

Tests passed: 13 (moments ago)                156:32  LF  UTF-8  4 spaces  benchmark
```

Code:
Main.java

```java
package edu.neu.coe.info6205.sort.par;

import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import java.util.concurrent.ForkJoinPool;


/**
 * This code has been fleshed out by Ziyao Qiao. Thanks very much.
 * CONSIDER tidy it up a bit.
 */
public class Main {

    public static void main(String[] args) {
        //processArgs(args);
        System.setProperty("java.util.concurrent.ForkJoinPool.common.parallelism", "8");
        System.out.println("Degree of parallelism: " +
ForkJoinPool.getCommonPoolParallelism());
        Random random = new Random();
        int[] len = {125000,250000,500000,1000000,2000000,4000000};
        int[] array = new int[len[5]];
        ArrayList<Long> timeList = new ArrayList<>();
        int cut[] =
{125000,15625,31250,62500,125000,250000,500000,1000000,2000000,4000000};
        for (int j = 0; j < 10; j++) {
            ParSort.cutoff = cut[j]; //1000 * (j + 1);
            //for (int i = 0; i < array.length; i++) array[i] = random.nextInt(10000000);
            long time;
            long startTime = System.currentTimeMillis();
            for (int t = 0; t < 100; t++) {
                for (int i = 0; i < array.length; i++) array[i] = random.nextInt(10000000);
                ParSort.sort(array, 0, array.length);
            }
            long endTime = System.currentTimeMillis();
            time = (endTime - startTime);
            timeList.add(time);


            System.out.println("cutoff：" + (ParSort.cutoff) + "\t\t100times Time:" + time +
"ms");

        }
        try {
            FileOutputStream fis = new FileOutputStream("./src/result.csv");
            OutputStreamWriter isr = new OutputStreamWriter(fis);
            BufferedWriter bw = new BufferedWriter(isr);
            int j = 0;
            for (long i : timeList) {
                String content = (double) 10000 * (j + 1) / 2000000 + "," + (double) i / 10 + "\n";
                j++;
                bw.write(content);
                bw.flush();
            }
            bw.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void processArgs(String[] args) {
        String[] xs = args;
        while (xs.length > 0)
            if (xs[0].startsWith("-")) xs = processArg(xs);
    }

    private static String[] processArg(String[] xs) {
        String[] result = new String[0];
        System.arraycopy(xs, 2, result, 0, xs.length - 2);
        processCommand(xs[0], xs[1]);
        return result;
    }

    private static void processCommand(String x, String y) {
        if (x.equalsIgnoreCase("N")) setConfig(x, Integer.parseInt(y));
        else
            // TODO sort this out
            if (x.equalsIgnoreCase("P")) {
                //noinspection ResultOfMethodCallIgnored
            }
                ForkJoinPool.getCommonPoolParallelism();
    }

    private static void setConfig(String x, int i) {
        configuration.put(x, i);
    }

    @SuppressWarnings("MismatchedQueryAndUpdateOfCollection")
    private static final Map<String, Integer> configuration = new HashMap<>();


}
```