

Implementation code:

1. Modify initial number of j to modify the number of cutoff:

```

package edu.neu.coe.info6205.sort.par;
import ...;

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

    public static void main(String[] args) {
        processArgs(args);
        System.out.println("Degree of parallelism: " + ForkJoinPool.getCommonPoolParallelism());
        Random random = new Random();
        int[] array = new int[20000000];
        ArrayList<Long> timelist = new ArrayList<>();
        for (int j = 40; j < 100; j++) {
            ParSort.cutoff = 10000 * (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 < 10; t++) {
                for (int i = 0; i < array.length; i++) array[i] = random.nextInt(10000000);
            }
        }
    }
}

```

2. ForkJoinPool implements Executor:

```

INFO6205 / src / main / java / edu / neu / coe / info6205 / sort / par / ParSort.java
Project Full Requests Commit
Main.java gitignore ParSort.java
/*
 * TODO tidy it up a bit.
 */
class ParSort {

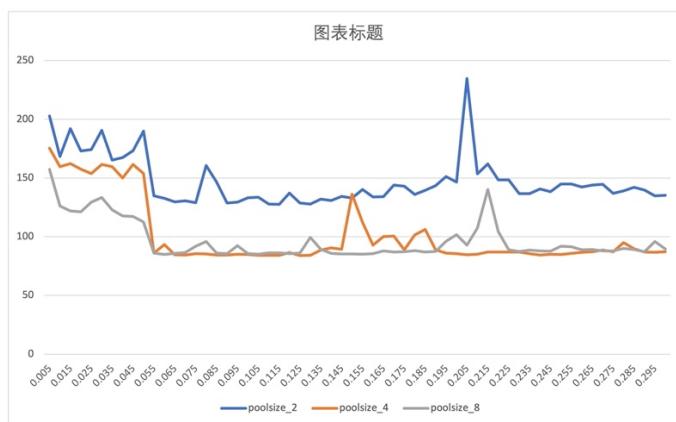
    public static int cutoff = 1000;
    private static int threadCount = 8;
    static ForkJoinPool myPool = new ForkJoinPool(threadCount);

    private static CompletableFuture<int[]> parsort(int[] array, int from, int to) {
        return CompletableFuture.supplyAsync(
            () -> {
                int[] result = new int[to - from];
                // TO IMPLEMENT
                System.arraycopy(array, from, result, destPos: 0, result.length);
                sort(result, from: 0, to: to - from);
                return result;
            }, myPool
        );
    }
}

```

Conclusion:

1. Since my laptop is based on 4-CPU, so the actual working thread number is under 8. Trying to set the thread number to 2, 4 and 8, and change the value of cutoff, the export data is in poolSize2.csv, poolSize4.csv and poolSize8.csv, and the line chart is at below:



when the thread number is 8, the parallel sort costs least time. And when array size is 2000000, the value of cutoff is between 0.5-0.55 (the actual number is between 500000-510000).

2. Trying to change the size of array to 2000000, 3000000, 4000000, 5000000 and 6000000, when array size is large enough (more than 3000000), the cut off becomes more obvious, and the larger the array size is, the larger the value of cut off is.

