import java.util.Random;

import java.util.concurrent.ForkJoinPool;

import java.util.concurrent.RecursiveTask;

public class MaxFind extends RecursiveTask<Integer> {

private static final int SEQUENTIAL\_THRESHOLD = 100;

private final int[] data;

private final int start;

private final int end;

public MaxFind(int[] data, int start, int end) {

this.data = data;

this.start = start;

this.end = end;

}

public MaxFind(int[] data) {

this(data, 0, data.length);

}

protected Integer compute() {

final int length = end - start;

if (length < SEQUENTIAL\_THRESHOLD) {

return computeDirectly();

}

final int split = length / 2;

final MaxFind left = new MaxFind(data, start, start + split);

left.fork();

final MaxFind right = new MaxFind(data, start + split, end);

return Math.max(right.compute(), left.join());

}

private Integer computeDirectly() {

System.out.println(Thread.currentThread() + " is searching array index: " + start + " to " + end);

int max = Integer.MIN\_VALUE;

for (int i = start; i < end; i++) {

if (data[i] > max) {

max = data[i];

}

}

return max;

}

public static void main(String[] args) {

// create a random data set

final int[] data = new int[200];

final Random random = new Random();

for (int i = 0; i < data.length; i++) {

data[i] = random.nextInt(350);

}

// submit the task to the pool

final ForkJoinPool pool = new ForkJoinPool(4);

final MaxFind finder = new MaxFind(data);

System.out.println("The max value is: "+pool.invoke(finder));

}

}