1 – Andrei Abramchuk

Conditions

1. Create 100 random binary-chromosomes each with 1000 genes.
2. Fitness is the number of “1” in one chromosome – the more the better.
3. Select 2 chromosomes at random from the better half of the population.
4. Create a child chromosome by a one-point-crossover.
5. Give the child a mutation with a probability of 1/1000 = 0.001.
6. Repeat from 2. to 5. 100 times and create the next generation.
7. Repeat 6. until the fitness value does not change any more.
8. Show the result:

(1) Desplay the best chromosome in the 1st, an intermediate & final generation.

(2) Desplay the best and average fitness vs. generation.

Source code

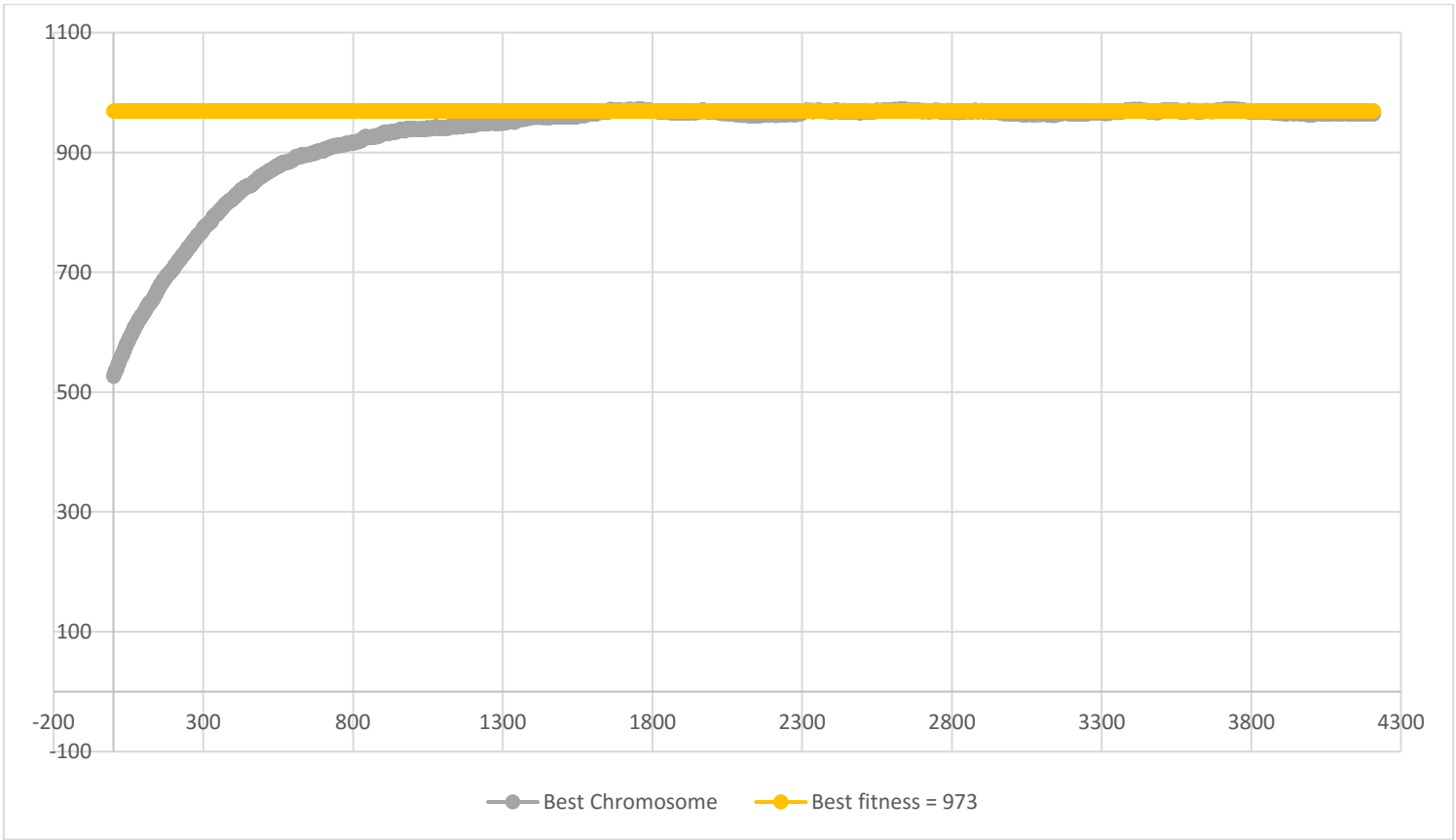
```
import java.io.*;
import java.util.*;

public class AllOneProblem {

    public ArrayList<ArrayList<Boolean>> binChrMas2 = new
ArrayList<ArrayList<Boolean>>();
public ArrayList<Integer> maxFitness = new ArrayList<Integer> ();
    public Integer generation = 0;

    public static void main(String[] args) {
                AllOneProblem obj = new AllOneProblem();
        obj.random();
                boolean tmp = false;
        int genNumber = 1;
        while (tmp != true) {
            obj.fun2();
            int sch = obj.end();
            genNumber++;
            if (sch < 1) tmp = true;
            //if (tmp != false) break;
        }
    }

    public void random() {
        ArrayList<Boolean> binChr = new ArrayList<Boolean>();
        Random rand = new Random();
        for (int a = 0; a < 100; a++) {
            for (int b = 0; b < 1000; b++) {
                binChr.add(rand.nextBoolean());
            }
            binChrMas2.add(binChr);
        }
    }

    public void fun2() {
        Random rand = new Random();
        ArrayList<Integer> fitnessMas = new ArrayList<Integer>();
        Map<Integer, Integer> map = new HashMap<>();
        for (int a = 0; a < binChrMas2.size(); a++) {
            int fitnessChr = 0;
            for (int b = 0; b < 1000; b++) {
                if (binChrMas2.get(a).get(b) != false) fitnessChr++;
            }
            fitnessMas.add(fitnessChr);
            map.put(a, fitnessChr);
        }
        Collections.sort(fitnessMas, new Comparator<Integer>() {
            public int compare(Integer o1, Integer o2) {
                return o2 - o1;
            }
        });
        if (fitnessMas.get(0).equals(Collections.max(fitnessMas)))
            System.out.println("Sorting was successful. Max fitness: " +
fitnessMas.get(0));
```

```
        maxFitness.add(fitnessMas.get(0));
        ArrayList<ArrayList<Boolean>> newGen = new ArrayList<ArrayList<Boolean>>();
        for (int i = 0; i < 50; i++) {
            int mama = fitnessMas.get(rand.nextInt(50)),
                    papa = fitnessMas.get(rand.nextInt(50));
            mama = getValue(map, mama);
            papa = getValue(map, papa);
            int cros_gen = rand.nextInt(998) + 1;
            ArrayList<Boolean> firstChild = new ArrayList<Boolean>();
            for (int j = 0; j < cros_gen; j++)
                firstChild.add(binChrMas2.get(mama).get(j));
            for (int j = cros_gen; j < 1000; j++)
                firstChild.add(binChrMas2.get(papa).get(j));
            newGen.add(firstChild);
            ArrayList<Boolean> seconChild = new ArrayList<Boolean>();
            for (int j = 0; j < cros_gen; j++)
                seconChild.add(binChrMas2.get(papa).get(j));
            for (int j = cros_gen; j < 1000; j++)
                seconChild.add(binChrMas2.get(mama).get(j));
            newGen.add(seconChild);
        }
        binChrMas2 = new ArrayList<ArrayList<Boolean>>(newGen);
        for (int i = 0; i < binChrMas2.size(); i++)
            for (int j = 0; j < binChrMas2.get(0).size(); j++)
                if (rand.nextInt(1000) == 769) {
                    if (binChrMas2.get(i).get(j) != true) {
                        ArrayList<Boolean> t = new
ArrayList<Boolean>(binChrMas2.get(i));
                        t.set(j, true);
                        binChrMas2.set(i, t);
                    } else {
                        ArrayList<Boolean> t = new
ArrayList<Boolean>(binChrMas2.get(i));
                        t.set(j, false);
                        binChrMas2.set(i, t);
                    }
                }

        Integer average = 0;
        for (int i = 0; i < 100; i++)
            average += fitnessMas.get(i);
        average /= 100;
        System.out.println("Average value: " + average);
        generation++;
        write(fitnessMas.get(0).toString(), average.toString(),
generation.toString());
    }

    public void write(String maxfitness, String average, String generation) {

        try {
            PrintStream printStream1 = new PrintStream(new
FileOutputStream("D:\\maxfitness.txt", true), true);
            PrintStream printStream2 = new PrintStream(new
FileOutputStream("D:\\average.txt", true), true);
            PrintStream printStream3 = new PrintStream(new
FileOutputStream("D:\\generation.txt", true), true);

            try {
                //Записываем текст
                printStream1.println(maxfitness);
                printStream2.println(average);
                printStream3.println(generation);
            } finally {
                //После чего мы должны закрыть файл, Иначе файл не запишется
                printStream1.close();
                printStream2.close();
                printStream3.close();
            }
        } catch(IOException e) {
            throw new RuntimeException(e);
        }
    }

    public static Integer getValue(Map<Integer, Integer> map, Integer value) {
        Set<Map.Entry<Integer, Integer>> entrySet = map.entrySet();
        Integer desiredFitness = value;//что хотим найти
        Integer key = 0;
        for (Map.Entry<Integer, Integer> pair : entrySet) {
```

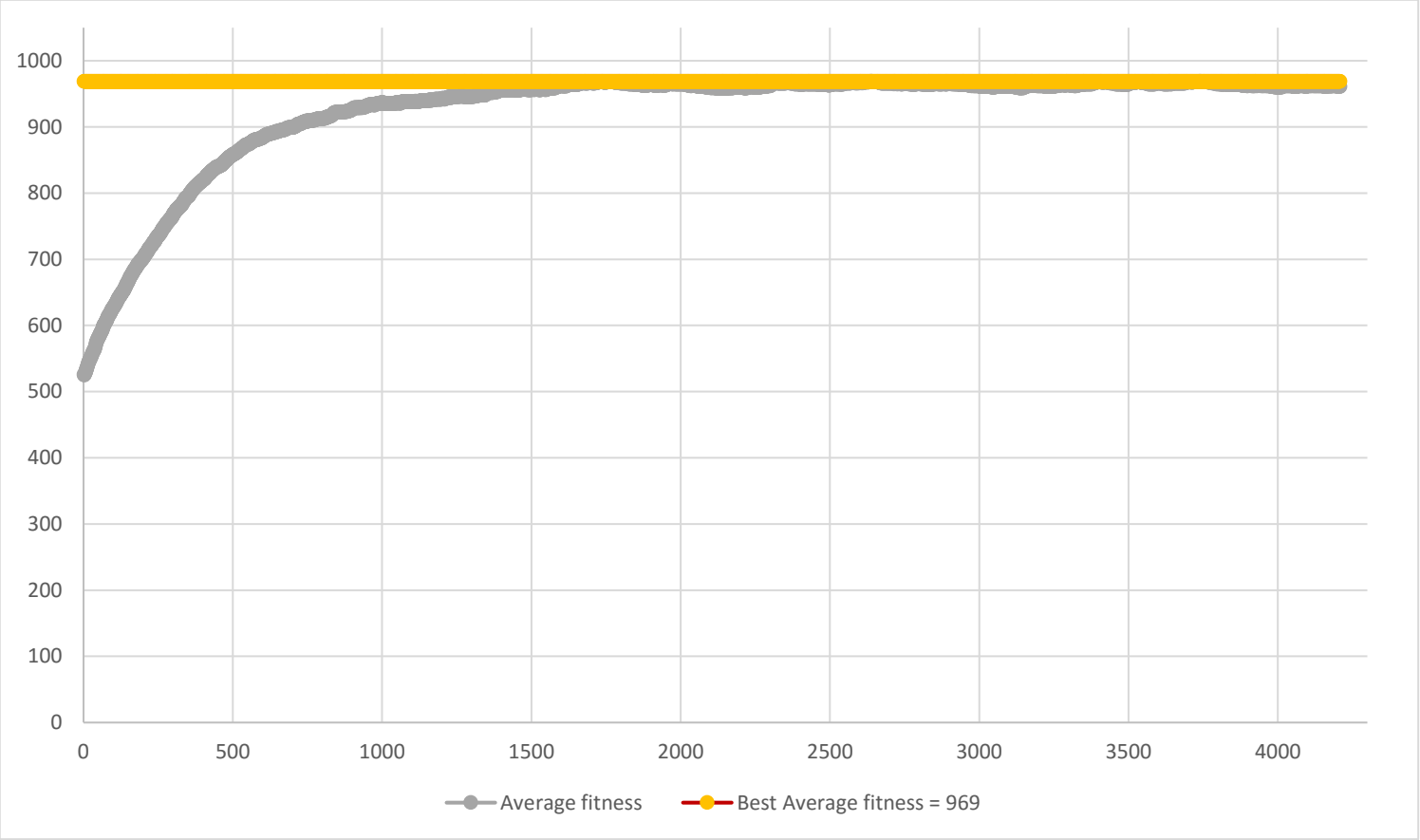
```
                if (desiredFitness.equals(pair.getValue())) {
                    key = pair.getKey();// нашли наше значение и возвращаем  ключ
                }
            }
            return key;
        }

        public int end() {
            int sch = 0;
            if (maxFitness.size() > 300) {
                ArrayList<Integer> temp = new ArrayList<Integer> ();
                for (int b = maxFitness.size() - 200; b < maxFitness.size(); b++) {
                    temp.add(maxFitness.get(b));
                }
                int m = Collections.max(temp), n = Collections.min(temp);
                int k = m - n;
                if (k > 3)
                    sch++;
            }
            else {
                for (int a = 0; a < binChrMas2.size(); a++) {
                    for (int b = 0; b < 1000; b++) {
                        if (binChrMas2.get(a).get(b) != true) {
                            sch++;
                            break;
                        }
                    }
                    if (sch > 0) break;
                }
            }
            return sch;
        }
    }
```

Results

(1) Best chromosome: Start fitness: 526, Best fitness: 973.



(2) Fitness graph:



Best average fitness: 969.

Starting average fitness: 526.

Generations: 4208.