1 exercise

Andrey Cheslow

```
package siit_1;
import java.util.Random;
/**
 *
 * @author Andrey
 */
public class SIIT_1 {
   public static void main(String[] args) {
      int generation[][]=new int[100][1000];
      int next_generation[][]=new int[100][1000];
      int number_of_choices[] =new int [100];
      Random rnd=new Random();
      //make first generation
      for(int i=0;i<100;i++){
         for(int j=0;j<1000;j++){
           generation[i][j]=(rnd.nextInt(2));
         }}
      double probability[] = new double[100];
      int fitness[] = new int[100];
      int sum_fit=0;
      while (fitness[0]<1000){
         for(int i=0;i<100;i++){
            fitness[i]=0;
         }
         for(int i=0;i<100;i++){
            number_of_choices[i]=0;
         }
         //calculating fitness
         for(int i=0;i<100;i++){
            for(int j=0;j<1000;j++){
               if (generation[i][j]==1) fitness[i]++;
            } }
         sum_fit=0;
         for(int j=0;j<100;j++){
               sum_fit+=fitness[j];
            }
         //sort generation according to ascending
         sort(fitness,generation);
         //calculating probability
         for(int j=0;j<100;j++){
               probability[j]=(double)(fitness[j])/sum_fit;
            }
         System.out.println(fitness[0]+" "+sum_fit/100);
         make_new_gen(probability,generation,next_generation,number_of_choices);
         /*for(int i=0;i<100;i++){
            System.out.println(number_of_choices[i]);
         }*/
```

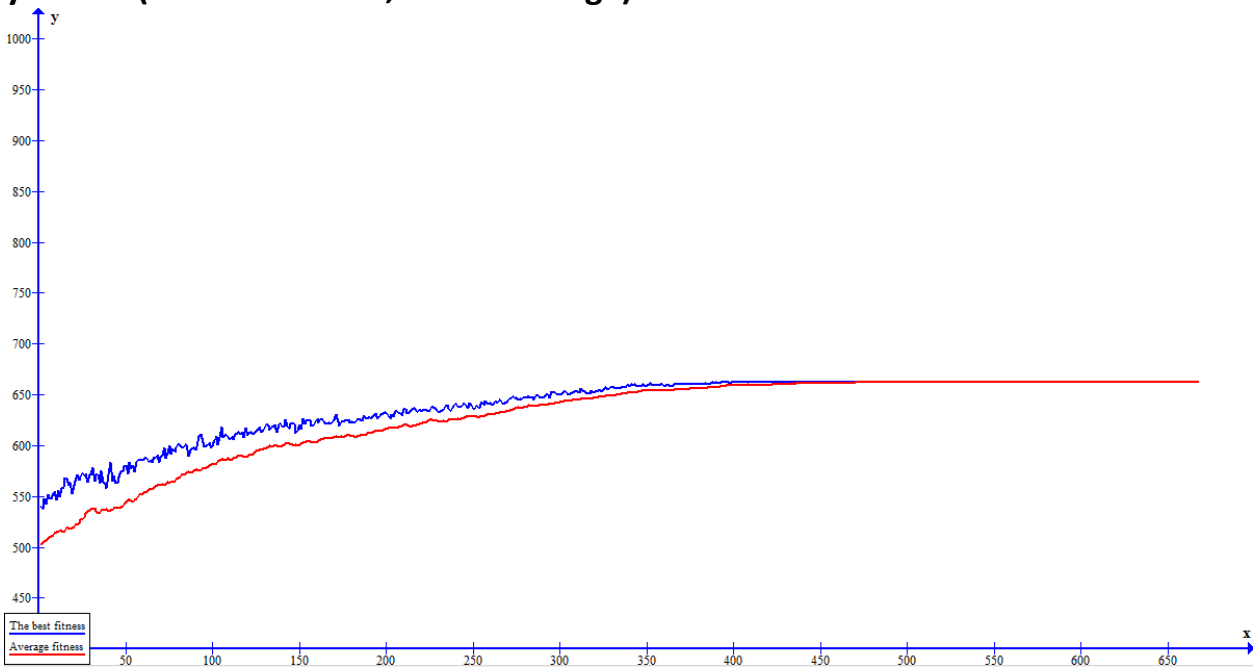
```
        generation=next_generation;
      }
    }
    public static void sort(int[] fitness,int[][] generation){
      for(int i=0;i<99;i++){
        for(int j=i+1;j<100;j++){
          if (fitness[j]>fitness[i]){
            int temp=fitness[i];
            fitness[i]=fitness[j];
            fitness[j]=temp;
            int temps[]=generation[i];
              generation[i] = generation[j];
              generation[j] = temps;
          }
        }
      }
    }
    public static void make_new_gen(double[] probability,int[][] generation,int[][]
next_generation, int[] number_of_choices){
      Random rnd= new Random();
      for(int i=0;i<100;i++){
        /*double p1=(double)(rnd.nextInt(100))/100;
        double p2=(double)(rnd.nextInt(100))/100;
        int nump1=0,nump2=0;
        int k=0;
        while(p1>0){
          if((p1-probability[k])>0){k++;p1-=probability[k];}
          else {nump1=k;p1-=probability[k];}
        }
         k=0;
        while(p2>0){
          if((p2-probability[k])>0){k++;p2-=probability[k];}
          else {nump2=k;p2-=probability[k];}
        }*/
        //truncate method
        int nump1=rnd.nextInt(50);
        int nump2=rnd.nextInt(50);
        //number_of_choices[nump1]++;
        //number_of_choices[nump2]++;
        int orrr=rnd.nextInt(1000);
        for(int f=0;f<1000;f++){
          //uniform crossover
          if(orrr>f)
          next_generation[i][f]=generation[nump1][f];
          else next_generation[i][f]=generation[nump2][f];
        }
      }
    }}
```

Graph:

Roulette selection and uniform crossover.

x-generation

y-fitness(blue— the best, red— average)



Roulette selection and one-point crossover.

x-generation

y-fitness (blue— the best, red— average)