

1 exercise – Igor Kondraashuk

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) Display the best chromosome in the 1st, an intermediate & final generation. (2) Display the best and average fitness vs. generation.

Source code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApplication1
{
    class GA
    {
        List<List<Boolean>> arrBinChrom;
        public List<double> generationFitness;
        public List<double> theBestInGen;
        List<double> theAverInGen;
        double theBestFirst;
        double theBestLast;
        public GA()
        {
            theAverInGen = new List<double>();
            theBestInGen = new List<double>();
            generationFitness = new List<double>();
            Random random = new Random();
            arrBinChrom = new List<List<bool>>();
            for(int i=0;i<100;i++)
            {
                List<Boolean> tempBinChrom= new List<bool>();
                for (int j=0;j<1000;j++)
                {
                    if (random.Next(2) == 0)
                        tempBinChrom.Add(false);
                    else
                        tempBinChrom.Add(true);
                }
                arrBinChrom.Add(tempBinChrom);
            }
            generationFitness.Add(getFitnessGen(arrBinChrom));
            theBestFirst = getTheBestFitnessGen(arrBinChrom);
            theBestInGen.Add(theBestFirst);
            int k = 0;
            do
            {
                List<List<Boolean>> tempGen = getNextGen(arrBinChrom);
                generationFitness.Add(getFitnessGen(tempGen));
                theBestInGen.Add(getTheBestFitnessGen(tempGen));
                arrBinChrom = tempGen;
                k++;
            }
            while (theBestFirst != theBestInGen[k]);
        }
    }
}
```

```

    }
    while (!isReady(generationFitness));
    theBestLast = getTheBestFitnessGen(arrBinChrom);
}
Boolean isReady(List<double> fitness)
{
    if (fitness.Count < 100)
        return false;
    else
    {
        for(int i=fitness.Count-100;i<fitness.Count;i++)
        {
            if ((int)fitness[fitness.Count - 100] != (int)fitness[i])
                return false;
        }
        return true;
    }
}
List<List<Boolean>> getNextGen(List<List<Boolean>> arrBinChrom)
{
    //Getting fitness
    List<int> fitnessGen = new List<int>();
    for(int i=0;i<arrBinChrom.Count;i++)
    {
        fitnessGen.Add(getFitness(arrBinChrom[i]));
    }
    //Sorting by fitness
    for(int i=0;i<fitnessGen.Count;i++)
    {
        for(int j=fitnessGen.Count-1;j>i;j--)
        {
            if(fitnessGen[j]<fitnessGen[j-1])
            {
                int temp = fitnessGen[j];
                fitnessGen[j] = fitnessGen[j - 1];
                fitnessGen[j - 1] = temp;
                List<Boolean> tempGen = arrBinChrom[j];
                arrBinChrom[j] = arrBinChrom[j - 1];
                arrBinChrom[j - 1] = tempGen;
            }
        }
    }
    //Getting children
    Random random = new Random();
    List<List<Boolean>> newGen = new List<List<bool>>();
    for (int i=0;i<50;i++)
    {
        int numberFather= random.Next(50, 100);
        int numberMother = random.Next(50, 100);
        int poinCrossover = random.Next(0,1000);
        List<Boolean> firstChild = new List<bool>();
        for (int j = 0; j < poinCrossover; j++)
        {
            firstChild.Add(arrBinChrom[numberFather][j]);
        }
        for (int j = poinCrossover; j < 1000; j++)
        {
            firstChild.Add(arrBinChrom[numberMother][j]);
        }
        newGen.Add(firstChild);
        List<Boolean> seconChild = new List<bool>();
        for (int j = 0; j < poinCrossover; j++)
        {
            seconChild.Add(arrBinChrom[numberMother][j]);
        }
    }
}

```

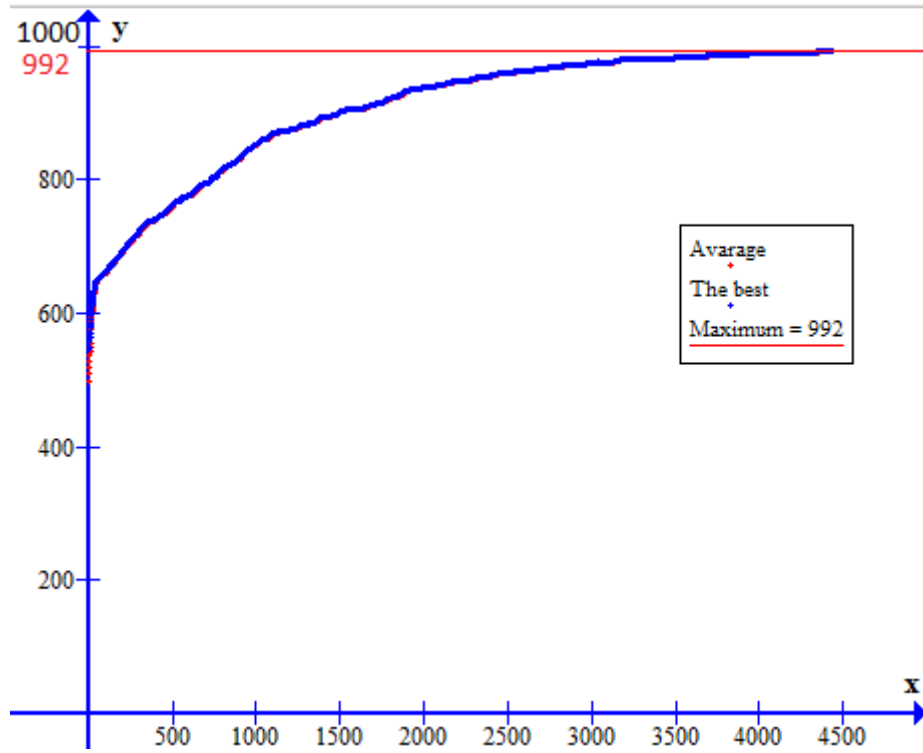
```

        }
        for (int j = poinCrossover; j < 1000; j++)
        {
            seconChild.Add(arrBinChrom[numberFather][j]);
        }
        newGen.Add(seconChild);
        //Mutation
        for(int j=0;j<newGen.Count;j++)
        {
            int prob = random.Next(0, 1000);
            if(prob==777)
            {
                int number = random.Next(1000);
                if (newGen[j][number])
                    newGen[j][number] = false;
                else
                    newGen[j][number] = true;
            }
        }
    }
    return newGen;
}
int getFitness(List<Boolean> binChrom)
{
    int fitness = 0;
    for(int i=0;i<binChrom.Count;i++)
    {
        if (binChrom[i])
            fitness++;
    }
    return fitness;
}
double getFitnessGen(List<List<Boolean>> gen)
{
    int sum = 0;
    for(int i=0;i<gen.Count;i++)
    {
        sum += getFitness(gen[i]);
    }
    return (double)sum / 100;
}

double getTheBestFitnessGen(List<List<Boolean>> gen)
{
    int max = 0;
    for (int i = 0; i < gen.Count; i++)
    {
        int fit=getFitness(gen[i]);
        if (fit > max)
            max = fit;
    }
    return max;
}
}
}

```

Result



First the best is 497

Last the best is 992