

KIRILL ZABRODSKY II-11 LAB-3 SIIT

Source code:

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

namespace lab3_siit_kirill_ii_11
{
    class GA
    {
        Random random;
        List<Population> historyOfPopulation;
        public GA()
        {
            random = new Random();
            List<Net> Nets = new List<Net>();
            for(int i=0;i<100;i++)
            {
                List<List<double>> w1 = new List<List<double>>();
                List<double> w2 = new List<double>();
                for(int j=0;j<5;j++)
                {
                    w2.Add(2 * random.NextDouble() - 1);
                    List<double> tempW1 = new List<double>();
                    for (int t = 0; t < 5; t++)
                        tempW1.Add(2 * random.NextDouble() - 1);
                    w1.Add(tempW1);
                }
                Nets.Add(new Net(w1,w2));
            }
            Population startPopulation = new Population(Nets);
            historyOfPopulation = new List<Population>();
            historyOfPopulation.Add(startPopulation);
            int k = 0;
            while(historyOfPopulation[k].theBestFitness!=32)
            {
                Thread.Sleep(10);

                historyOfPopulation.Add(getNextPupulation(historyOfPopulation[k]));
            }
        }
    }
}
```

KIRILL ZABRODSKY II-11 LAB-3 SIIT

```

        k++;
    }
}

private List<Net> population;

private Population getNextPupulation(Population
parentPopulation)
{
    List<Net> childrenPopulationNets = new List<Net>();

    for (int i=0;i<50;i++)
    {
        List<Net> childrenNets = getChildren(
            parentPopulation.Nets[random.Next() % 50+50],
            parentPopulation.Nets[random.Next() % 50+50]
        );
        childrenPopulationNets.AddRange(childrenNets);
    }
    for (int k = 0; k < childrenPopulationNets.Count; k++)
    {
        for (int i = 0; i <
childrenPopulationNets[k].w1.Count; i++)
        {
            for (int j = 0; j <
childrenPopulationNets[k].w1[i].Count; j++)
            {
                int prob = random.Next(0, 1000);
                if (prob == 777)
                {
                    childrenPopulationNets[k].w1[i][j] =
2*random.NextDouble() + 1;
                }
            }
        }
        for (int j = 0; j <
childrenPopulationNets[k].w2.Count; j++)
        {
            int prob = random.Next(0, 1000);
            if (prob == 777)
            {
                childrenPopulationNets[k].w2[j] = 2 *
random.NextDouble() + 1;
            }
        }
    }

    //Sorting by fitness
    for (int i = 0; i < childrenPopulationNets.Count; i++)

```

KIRILL ZABRODSKY II-11 LAB-3 SIIT

```

        {
            for (int j = childrenPopulationNets.Count - 1; j > i;
j--)
            {
                if (childrenPopulationNets[j].fitness <
childrenPopulationNets[j - 1].fitness)
                {
                    Net tempNet = childrenPopulationNets[j];
                    childrenPopulationNets[j] =
childrenPopulationNets[j - 1];
                    childrenPopulationNets[j - 1] = tempNet;
                }
            }
        }
        Population childrenPopulation = new
Population(childrenPopulationNets);
        return childrenPopulation;
    }

    Boolean isReady(List<Population> historyOfPopulation)
    {
        if (historyOfPopulation[historyOfPopulation.Count -
1].theBestFitness == 32)
            return true;
        else
            return false;
        //if (historyOfPopulation.Count < 100)
        //    return false;
        //else
        //{
        //    for (int i = historyOfPopulation.Count - 100; i <
historyOfPopulation.Count; i++)
        //    {
        //        if
(historyOfPopulation[historyOfPopulation.Count - 100].avarageFitness
!= historyOfPopulation[i].avarageFitness)
        //            return false;
        //    }
        //    return true;
        //}
    }

    private List<Net> getChildren(Net father, Net mother)
    {
        List<Net> childrenNets = new List<Net>();
        Random random = new Random();
        List<Boolean> mask = new List<bool>();
        List<List<double>> firstW1 = new List<List<double>>();
        List<List<double>> secondW1 = new List<List<double>>();
    }

```

KIRILL ZABRODSKY II-11 LAB-3 SIIT

```
List<double> firstW2 = new List<double>();
List<double> secondW2 = new List<double>();
for (int i=0;i<father.w1.Count;i++)
{
    for(int j=0;j< father.w1[i].Count;j++)
    {
        List<double> firstTempW1 = new List<double>();
        List<double> secondTempW1 = new List<double>();
        if (random.Next() % 2 == 0)
        {
            firstTempW1.Add(father.w1[i][j]);
            secondTempW1.Add(mother.w1[i][j]);
        }
        else
        {
            firstTempW1.Add(mother.w1[i][j]);
            secondTempW1.Add(father.w1[i][j]);
        }
        father.w1.Add(firstTempW1);
        mother.w1.Add(secondTempW1);
    }
}
for (int j = 0; j < father.w2.Count; j++)
{
    if (random.Next() % 2 == 0)
    {
        firstW2.Add(father.w2[j]);
        secondW2.Add(mother.w2[j]);
    }
    else
    {
        firstW2.Add(mother.w2[j]);
        secondW2.Add(father.w2[j]);
    }
}

childrenNets.Add(new Net (firstW1,firstW2));
childrenNets.Add(new Net (secondW1,secondW2));
return childrenNets;
}
}
}

Net.cs
using System;
```

KIRILL ZABRODSKY II-11 LAB-3 SIIT

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

namespace lab3_siit_kirill_ii_11
{
    class Net
    {
        public List<double> etalonInput2;
        public List<double> etalonResult2;
        public List<List<double>> w1;
        public List<double> w2;
        public int fitness;
        double[,] etalonInput = new double[,]
        {
            {-1, -1, -1, -1, -1 },
            {-1, -1, -1, -1, 1},
            {-1, -1, -1, 1, -1},
            {-1, -1, -1, 1, 1},
            {-1, -1, 1, -1, -1},
            {-1, -1, 1, -1, 1},
            {-1, -1, 1, 1, -1},
            {-1, -1, 1, 1, 1},
            {-1, 1, -1, -1, -1},
            {-1, 1, -1, -1, 1},
            {-1, 1, -1, 1, -1},
            {-1, 1, -1, 1, 1},
            {-1, 1, 1, -1, -1},
            {-1, 1, 1, -1, 1},
            {-1, 1, 1, 1, -1},
            {-1, 1, 1, 1, 1},
            {1, -1, -1, -1, -1},
            {1, -1, -1, -1, 1},
            {1, -1, -1, 1, -1},
            {1, -1, -1, 1, 1},
            {1, -1, 1, -1, -1},
            {1, -1, 1, -1, 1},
            {1, -1, 1, 1, -1},
            {1, -1, 1, 1, 1},
            {1, 1, -1, -1, -1},
            {1, 1, -1, -1, 1},
            {1, 1, -1, 1, -1},
        }
    }
}
```

KIRILL ZABRODSKY II-11 LAB-3 SIIT

```
{1, 1, -1, 1, 1},
{1, 1, 1, -1, -1},
{1, 1, 1, -1, 1},
{1, 1, 1, 1, -1},
{1, 1, 1, 1, 1}};
double[] etalonResult = new double[]
1, 0, 0, {1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0,
1, 1, 0, 0, 1, 0, 1, 1, 0};
public Net(List<List<double>> w1, List<double> w2)
{
    this.w1 = w1;
    this.w2 = w2;
    fitness = getFitness(w1, w2);
}
private int getFitness(List<List<double>> w1, List<double> w2)
{
    int count = 0;
    for (int k = 0; k < 32; k++)
    {
        List<double> tempLayout = new List<double>();
        for (int i = 0; i < w1.Count; i++)
        {
            double tempWeight = 0;
            for (int j = 0; j < 5; j++)
            {
                tempWeight += etalonInput[k, i] * w1[i][j];
            }
            tempLayout.Add(tempWeight);
        }
        double final = 0;
        for (int i = 0; i < tempLayout.Count; i++)
        {
            final += tempLayout[i] * w2[i];
        }
        double afterActivation;
        if (final > 0)
            afterActivation = 1;
        else
            afterActivation = -1;
        if (afterActivation == etalonResult[k])
            count++;
    }
    return count;
}
```

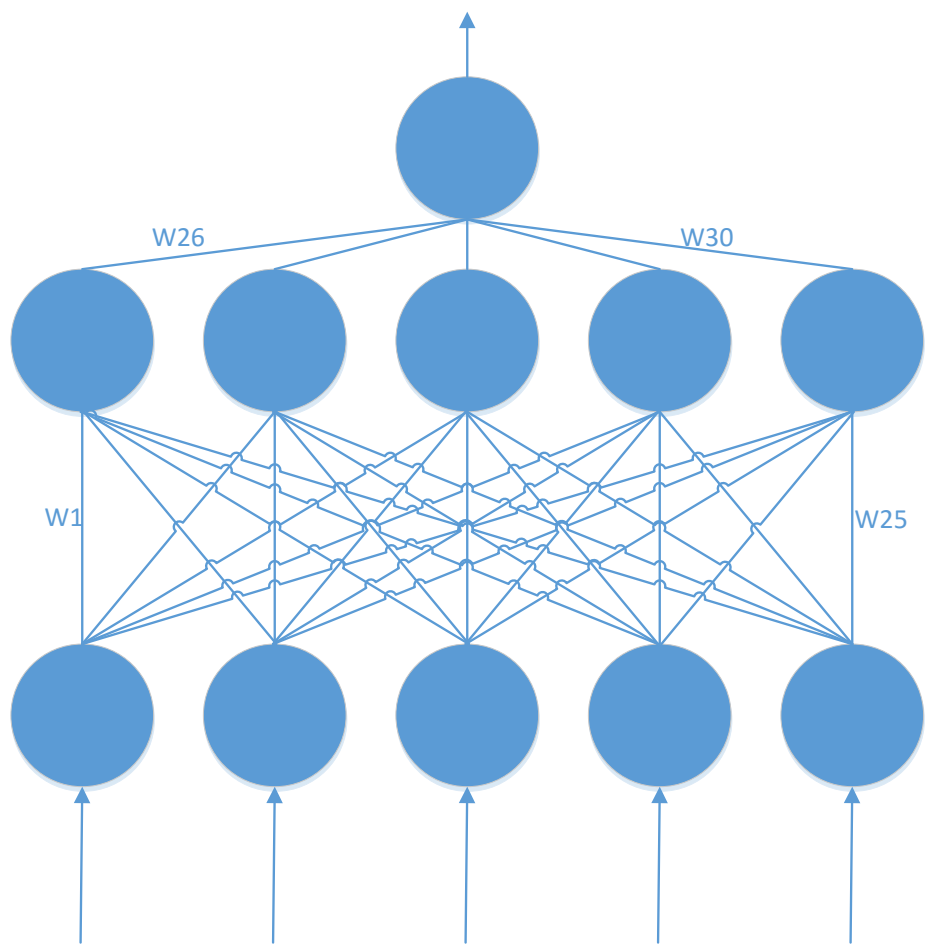
KIRILL ZABRODSKY II-11 LAB-3 SIIT

```
    }  
    }  
}
```

Population.cs

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace lab3_siit_kirill_ii_11  
{  
    class Population  
    {  
        public List<Net> Nets;  
        public int theBestFitness;  
        public int avarageFitness;  
        public Population(List<Net> Nets)  
        {  
            this.Nets = Nets;  
            theBestFitness = getTheBestFitness(Nets);  
            avarageFitness = getAvarageFitness(Nets);  
        }  
        public int getTheBestFitness(List<Net> Nets)  
        {  
            int bestFitness = 0;  
            for (int i = 0; i < Nets.Count; i++)  
                if (bestFitness < Nets[i].fitness)  
                    bestFitness = Nets[i].fitness;  
            return bestFitness;  
        }  
        public int getAvarageFitness(List<Net> Nets)  
        {  
            int avarageFitness = 0;  
            for (int i = 0; i < Nets.Count; i++)  
                avarageFitness += Nets[i].fitness;  
            avarageFitness /= Nets.Count;  
            return avarageFitness;  
        }  
    }  
}
```

Neural network model:



Weight	Value
W1	0,64736
W2	0,976
W3	0,621
W4	0,935
W5	-0,9306
W6	0,438
W7	0,8001
W8	0,7912
W9	0,9146
W10	0,6823
W11	0,7412
W12	0,601
W13	-0,498
W14	-0,814
W15	0,6954

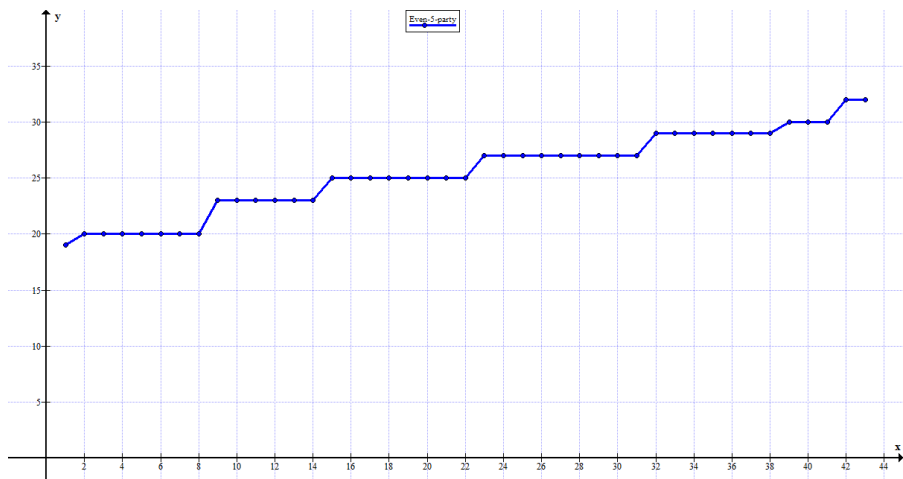
W16	0,0689
W17	0,6957
W18	0,4035
W19	-0,2287
W20	0,8953
W21	0,8547
W22	-0,3124
W23	-0,7156
W24	-0,26108
W25	0,3478
W26	0,521478
W27	-0,45633
W28	-0,8012
W29	0,80123
W30	0,7415

KIRILL ZABRODSKY II-11 LAB-3 SIIT

Table of reference values:

-1	-1	-1	-1	-1
-1	-1	-1	-1	1
-1	-1	-1	1	-1
-1	-1	-1	1	1
-1	-1	1	-1	-1
-1	-1	1	-1	1
-1	-1	1	1	-1
-1	-1	1	1	1
-1	1	-1	-1	-1
-1	1	-1	-1	1
-1	1	-1	1	-1
-1	1	-1	1	1
-1	1	1	-1	-1
-1	1	1	-1	1
-1	1	1	1	-1
-1	1	1	1	1
1	-1	-1	-1	-1
1	-1	-1	-1	1
1	-1	-1	1	-1
1	-1	-1	1	1
1	-1	1	-1	-1
1	-1	1	-1	1
1	-1	1	1	-1
1	-1	1	1	1
1	1	-1	-1	-1
1	1	-1	-1	1
1	1	-1	1	-1
1	1	-1	1	1
1	1	1	-1	-1
1	1	1	-1	1
1	1	1	1	-1
1	1	1	1	1

KIRILL ZABRODSKY II-11 LAB-3 SIIT



iteration	1	10	26	35	44	reference
fitness	19	23	27	29	32	values
	1	1	1	1	1	1
	-1	-1	-1	-1	-1	-1
	1	1	1	1	-1	-1
	1	1	1	1	1	1
	-1	-1	-1	-1	-1	-1
	-1	-1	-1	-1	1	1
	1	1	1	1	1	1
	-1	-1	-1	-1	-1	-1
	1	1	1	1	-1	-1
	1	1	1	1	1	1
	1	1	1	1	1	1
	1	1	1	-1	-1	-1
	1	1	1	1	1	1
	-1	-1	-1	-1	-1	-1
	1	1	1	-1	-1	-1
	1	1	1	1	1	1
	-1	-1	-1	-1	-1	-1
	-1	-1	1	1	1	1
	1	1	1	1	1	1
	-1	-1	-1	-1	-1	-1
	-1	-1	1	1	1	1
	-1	-1	-1	-1	-1	-1
	-1	-1	-1	-1	-1	-1
	-1	-1	-1	-1	-1	-1
	-1	-1	1	1	1	1
	1	1	1	1	1	1
	-1	-1	-1	-1	-1	-1
	1	1	-1	-1	-1	-1
	1	1	1	1	1	1
	1	-1	-1	-1	-1	-1
	-1	1	1	1	1	1
	-1	1	1	1	1	1
	1	-1	-1	-1	-1	-1