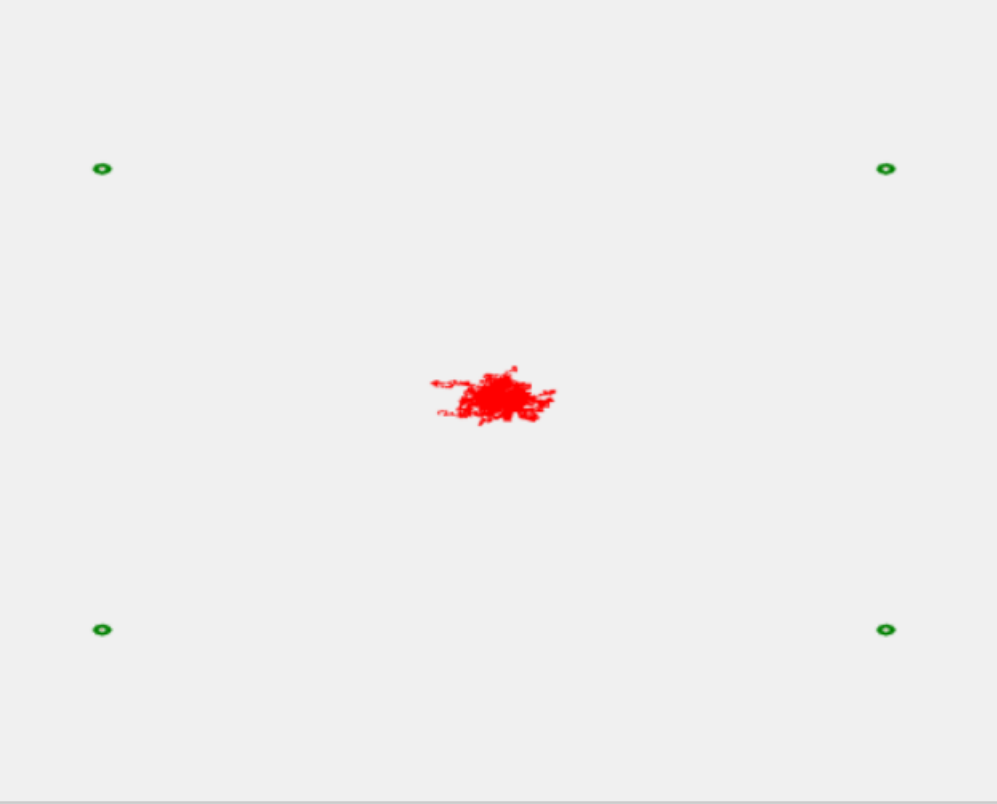
Exercise 3 1.
Simulate 100 dogs in the gridworld (0,0)-(1000,1000) with sausage at (800,800) (800,200), (200,800),(200,200) where all dogs start from (500,500) looking for the sausage by:

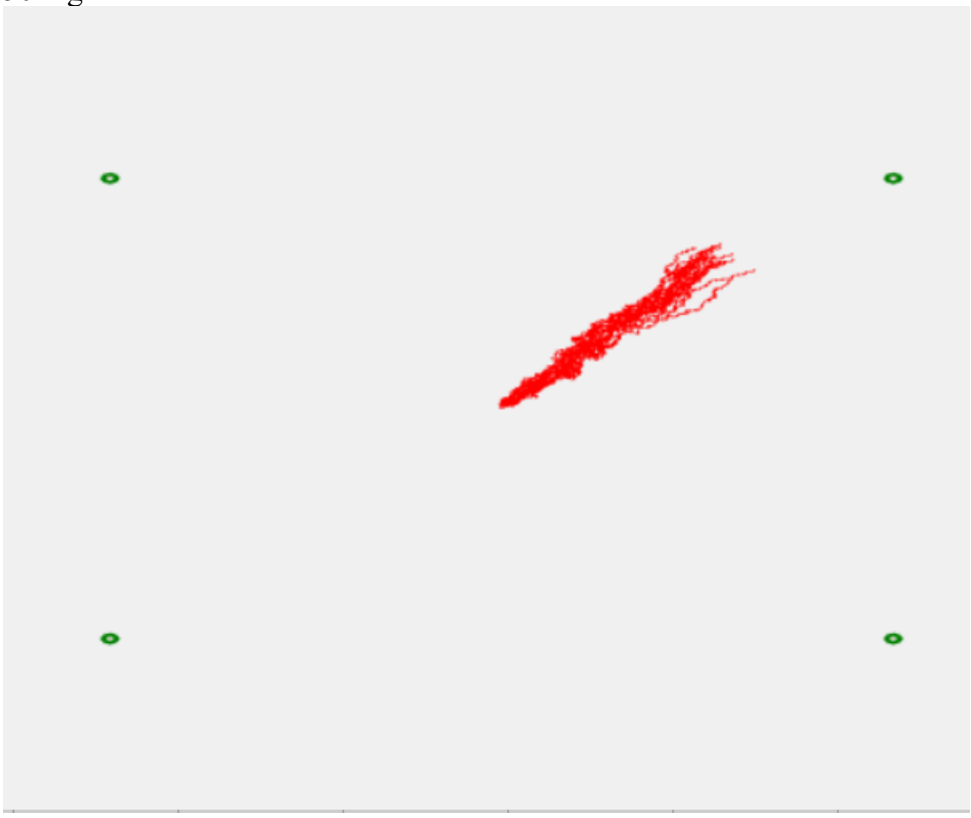
(1) assuming all dog has a chromosome whose each of 1000 genes are 1, 2, 3 or 4,
(2) moving step by step according to his/her chromosome,
(3) with 1, 2, 3, 4 meaning a movement toward up, down, right, left, respectively.

First generation:



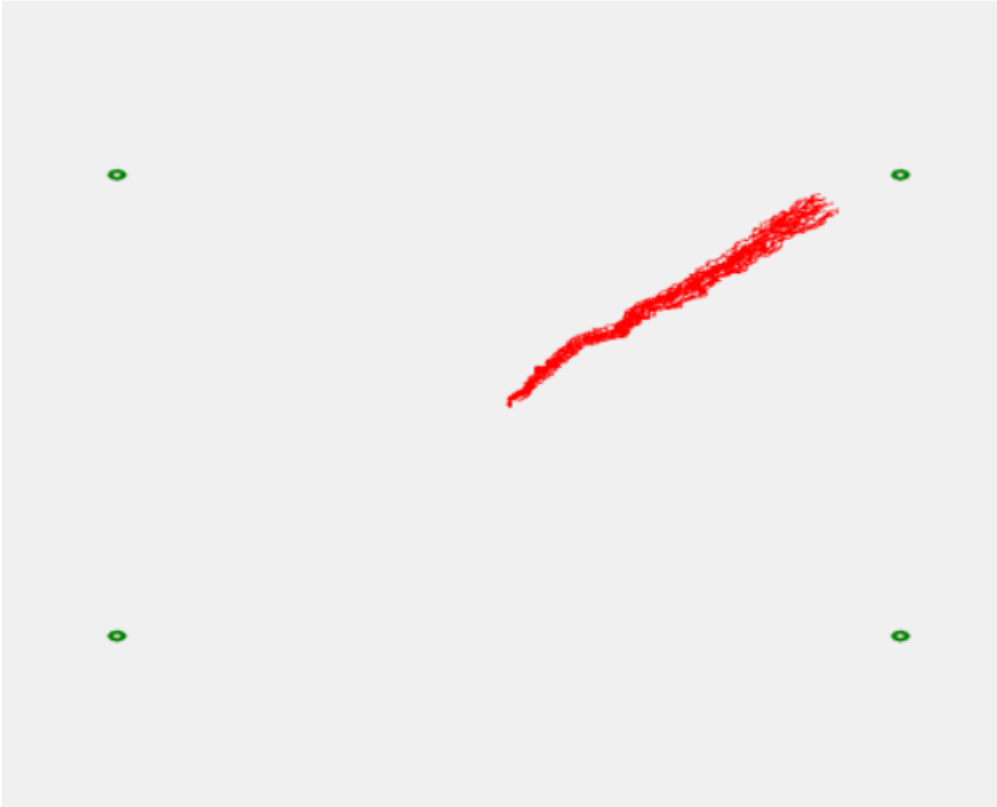
| Generation №1 | | | |
|---|---|---|---|
| Dog № | fitness | share fitness | dogs arround |
| 1 | 550 | 1540 | 8 |
| 2 | 566 | 2716,8 | 12 |
| 3 | 544 | 1501,44 | 7 |
| 4 | 568 | 3839,68 | 14 |
| 5 | 564 | 3790,08 | 16 |
| 6 | 586 | 3914,48 | 15 |
| 7 | 572 | 2425,28 | 12 |
| 8 | 578 | 4739,6 | 17 |
| 9 | 564 | 3180,96 | 12 |
| 10 | 578 | 4485,28 | 17 |
| 11 | 566 | 3531,84 | 16 |
| 12 | 582 | 3934,32 | 15 |
| 13 | 598 | 4927,52 | 16 |
| 14 | 540 | 864 | 5 |
| 15 | 586 | 4992,72 | 16 |
| 16 | 580 | 4640 | 15 |
| 17 | 578 | 3468 | 16 |
| 18 | 582 | 3678,24 | 14 |
| 19 | 544 | 1610,24 | 8 |
| 20 | 554 | 2770 | 11 |

50th generation:



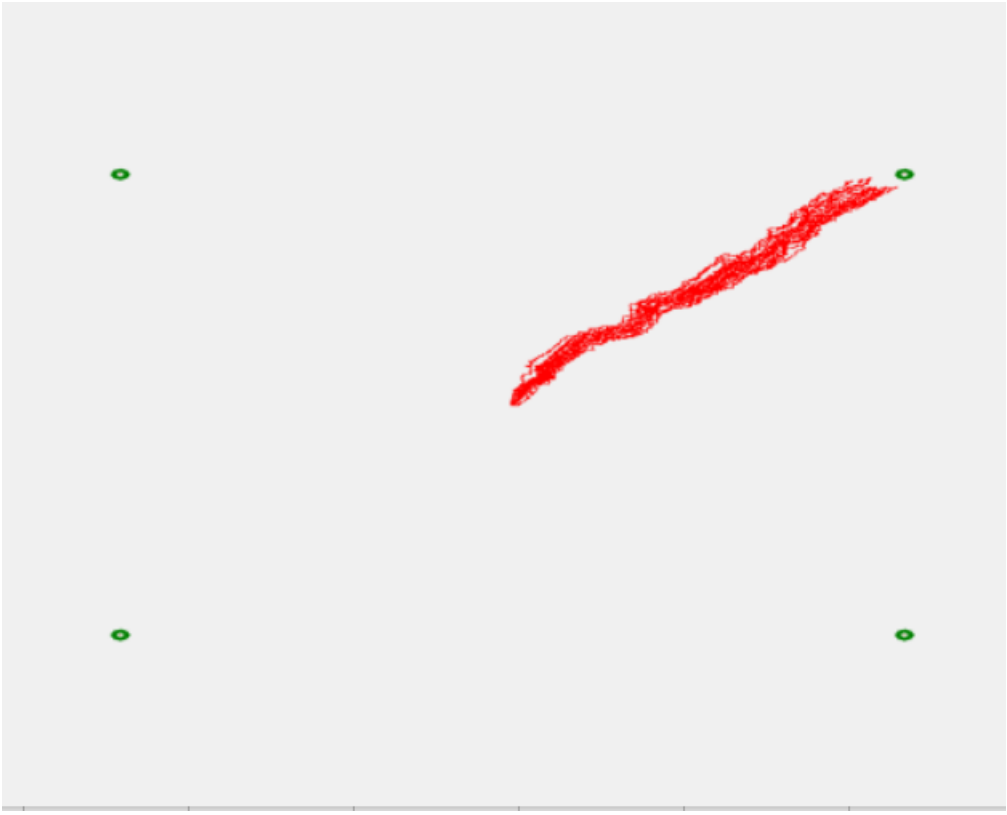
| Generation №50 | | | |
|---|---|---|---|
| Dog № | fitness | share fitness | dogs arround |
| 1 | 238 | 3284,4 | 20 |
| 2 | 234 | 3013,92 | 19 |
| 3 | 240 | 3235,2 | 20 |
| 4 | 252 | 2731,68 | 18 |
| 5 | 252 | 1340,64 | 15 |
| 6 | 238 | 2951,2 | 20 |
| 7 | 240 | 2956,8 | 20 |
| 8 | 240 | 3264 | 19 |
| 9 | 228 | 2635,68 | 20 |
| 10 | 236 | 2756,48 | 18 |
| 11 | 222 | 2610,72 | 19 |
| 12 | 226 | 804,56 | 11 |
| 13 | 222 | 2495,28 | 18 |
| 14 | 240 | 3158,4 | 20 |
| 15 | 244 | 3162,24 | 19 |
| 16 | 246 | 3247,2 | 20 |
| 17 | 244 | 1903,2 | 18 |
| 18 | 240 | 3302,4 | 20 |
| 19 | 218 | 2293,36 | 18 |
| 20 | 222 | 2592,96 | 20 |

125th generation:



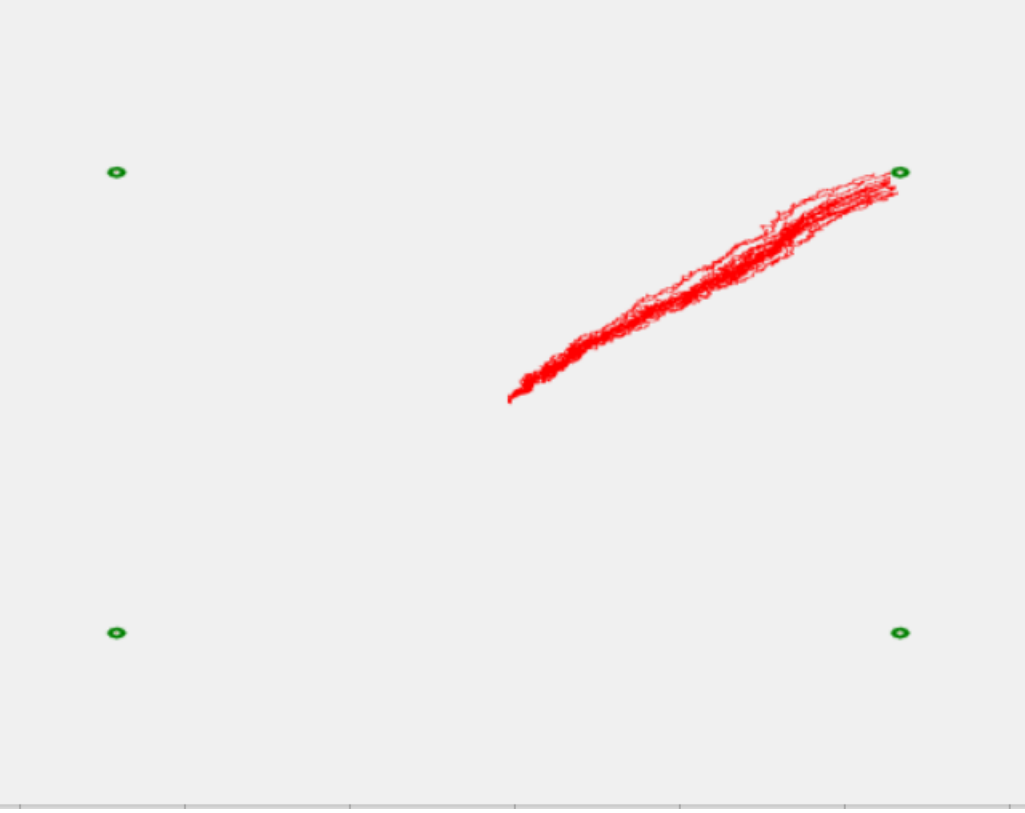
| Generation №125 | | | |
|---|---|---|---|
| Dog № | fitness | share fitness | dogs arround |
| 1 | 92 | 1111,36 | 20 |
| 2 | 110 | 1548,8 | 20 |
| 3 | 92 | 1310,08 | 20 |
| 4 | 116 | 1484,8 | 20 |
| 5 | 96 | 1098,24 | 20 |
| 6 | 102 | 1272,96 | 20 |
| 7 | 106 | 1509,44 | 20 |
| 8 | 104 | 1480,96 | 20 |
| 9 | 88 | 1203,84 | 20 |
| 10 | 102 | 1501,44 | 20 |
| 11 | 114 | 1523,04 | 20 |
| 12 | 102 | 1232,16 | 20 |
| 13 | 102 | 1150,56 | 20 |
| 14 | 88 | 1189,76 | 20 |
| 15 | 94 | 1353,6 | 20 |
| 16 | 98 | 1340,64 | 20 |
| 17 | 106 | 1127,84 | 20 |
| 18 | 94 | 1376,16 | 20 |
| 19 | 88 | 1077,12 | 20 |
| 20 | 94 | 1270,88 | 20 |

200th generation:



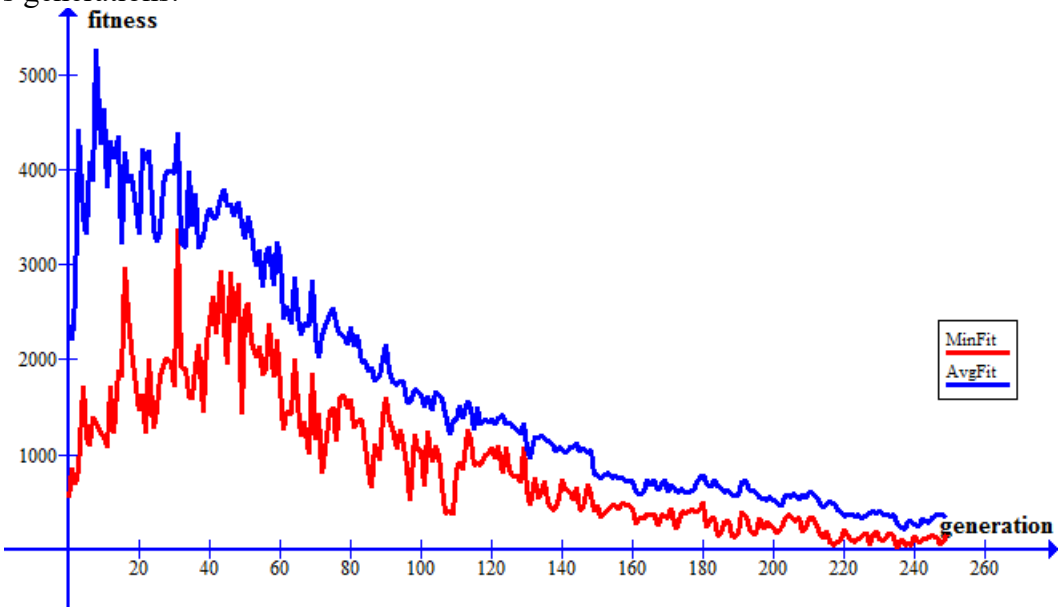
| Generation №200 | | | |
|---|---|---|---|
| Dog № | fitness | share fitness | dogs arround |
| 1 | 48 | 453,12 | 20 |
| 2 | 38 | 586,72 | 20 |
| 3 | 52 | 607,36 | 20 |
| 4 | 30 | 360 | 20 |
| 5 | 38 | 434,72 | 20 |
| 6 | 30 | 422,4 | 20 |
| 7 | 44 | 594,88 | 20 |
| 8 | 38 | 586,72 | 20 |
| 9 | 38 | 516,8 | 20 |
| 10 | 38 | 544,16 | 20 |
| 11 | 36 | 547,2 | 20 |
| 12 | 22 | 246,4 | 20 |
| 13 | 36 | 524,16 | 20 |
| 14 | 42 | 628,32 | 20 |
| 15 | 40 | 592 | 20 |
| 16 | 44 | 672,32 | 20 |
| 17 | 42 | 598,08 | 20 |
| 18 | 42 | 641,76 | 20 |
| 19 | 54 | 734,4 | 20 |
| 20 | 52 | 728 | 20 |

250th generation:



| Generation №250 | | | |
|---|---|---|---|
| Dog № | fitness | share fitness | dogs arround |
| 1 | 24 | 347,52 | 20 |
| 2 | 18 | 247,68 | 20 |
| 3 | 28 | 344,96 | 20 |
| 4 | 22 | 212,96 | 20 |
| 5 | 36 | 504 | 20 |
| 6 | 8 | 85,12 | 20 |
| 7 | 34 | 443,36 | 20 |
| 8 | 14 | 176,96 | 20 |
| 9 | 24 | 347,52 | 20 |
| 10 | 44 | 503,36 | 20 |
| 11 | 56 | 510,72 | 20 |
| 12 | 24 | 332,16 | 20 |
| 13 | 42 | 554,4 | 20 |
| 14 | 40 | 390,4 | 20 |
| 15 | 20 | 284,8 | 20 |
| 16 | 32 | 435,2 | 20 |
| 17 | 22 | 306,24 | 20 |
| 18 | 24 | 337,92 | 20 |
| 19 | 38 | 525,92 | 20 |
| 20 | 38 | 498,56 | 20 |

Fitness vs generations:




Source code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;


namespace Siit_3
{
   public partial class Form1 : Form
   {
      Bitmap bit = new Bitmap(1000, 1000);
      public Form1()
      {
         InitializeComponent();

         pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;


      }

      private void button1_Click(object sender, EventArgs e)
      {
         StreamWriter avgFitFile = new StreamWriter("averageFit.txt");
         StreamWriter maxFitFile = new StreamWriter("maxFit.txt");
         StreamWriter numGenFile = new StreamWriter("numGen.txt");
         StreamWriter tableFile = new StreamWriter("Table.txt");
```

```
        StreamWriter tablenum = new StreamWriter("Num.txt");
        generation old_gens = new generation();
        old_gens.randomize();
        old_gens.setFitness();
        old_gens.setProbability();
        double maxFit = 0;
        int numGeneration = 0;
        for (int j = 0; (j < 1000) && (numGeneration<250); numGeneration++)
        {
           bit = new Bitmap(1000, 1000);
           for (int i = 0; i < generation.numChromo; i++)
           {
              PaintWay(old_gens.GetChromo(i));
           }
           this.Refresh();
           if(numGeneration==0 || numGeneration==50 || numGeneration==125 ||
numGeneration==200 || numGeneration ==249)
           MessageBox.Show("gav gav!");
           numGenFile.WriteLine(numGeneration.ToString());
           Console.WriteLine(old_gens.bestFitness() + " " + old_gens.getAverageFit() + " " +
old_gens.getBestArround());
           //if (old_gens.bestFitness() == 0) break;
           List<int[]> new_tmp = new List<int[]>();
           old_gens.Sort();                                    //for truncate
           for (int i = 0; i < generation.numChromo; i++)
           {
              new_tmp.Add(old_gens.newChild());
           }
           old_gens.WriteTable(tableFile, tablenum);
           generation new_gens = new generation(new_tmp);
           old_gens = new_gens;
           old_gens.setFitness();
           old_gens.setProbability();
           avgFitFile.WriteLine(old_gens.getAverageFit().ToString());
           maxFitFile.WriteLine(old_gens.bestFitness().ToString());
           if (old_gens.bestFitness() > maxFit)
           {
              maxFit = old_gens.bestFitness();
              j = 0;
           }
           else j++;

        }
        for (int i = 0; i < generation.numChromo; i++)
        {
           PaintWay(old_gens.GetChromo(i));
        }
        tablenum.Close();
        tableFile.Close();
        numGenFile.Close();
        avgFitFile.Close();
        maxFitFile.Close();
```

```
        }
        private void PaintWay(int[] chromo)
        {
            double location_x = 500, location_y = 500;

           bit.SetPixel(200, 200, Color.Red);
            bit.SetPixel(200, 800, Color.Red);
            bit.SetPixel(800, 200, Color.Red);
            bit.SetPixel(800, 800, Color.Red);
            Graphics graph = Graphics.FromImage(bit);
            Pen my_pen = new Pen(Color.Green);
            my_pen.Width = 10;
            graph.DrawEllipse(my_pen, 800 - 2, 800 - 2, 4, 4);
            graph.DrawEllipse(my_pen, 800 - 2, 200 - 2, 4, 4);
            graph.DrawEllipse(my_pen, 200 - 2, 800 - 2, 4, 4);
            graph.DrawEllipse(my_pen, 200 - 2, 200 - 2, 4, 4);
            foreach (int i in chromo)
            {
                if (i == 1)
                {
                    location_y += 1;
                    if (location_y == 1000) location_y = 1;
                    bit.SetPixel((int)location_x,(int) location_y, Color.Red);
                }
                else if (i == 2)
                {
                    location_x += 1;
                    if (location_x == 1000) location_x = 1;
                    bit.SetPixel((int)location_x, (int)location_y, Color.Red);
                }
                else if (i == 3)
                {
                    location_y -= 1;
                    if (location_y == 0) location_y = 999;
                    bit.SetPixel((int)location_x, (int)location_y, Color.Red);
                }

                else
                {
                    location_x -= 1;
                    if (location_x == 0) location_x = 999;
                    bit.SetPixel((int)location_x, (int)location_y, Color.Red);
                }

            }
            pictureBox1.Image = bit;
        }
    }
}
```

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

namespace Siit_3
{
   class generation
   {
      static public int numChromo = 20;
      static public int numGens = 1000;
      List<int[]> gens;
      List<int> fitness { get; }
      List<double> sharedFitness;
      List<float> probability { get; }
      List<int> chromSelect;
      List<int[]> location;
      List<int> arround;
      public int sousage_x1 = 800, sousage_x2 = 800, sousage_x3 = 200, sousage_x4 = 200;
      public int sousage_y1 = 800, sousage_y2 = 200, sousage_y3 = 800, sousage_y4 = 200;
      public double averagefitness = 0f;
      StreamWriter fitOut = new StreamWriter("fitOut.txt");
      StreamWriter sharefitOut = new StreamWriter("sharefitOut.txt");
      StreamWriter arrOut = new StreamWriter("arrOut.txt");

      Random mutat = new Random();
      int rando = 0;

      public generation()
      {
         gens = new List<int[]>();
         fitness = new List<int>();
         probability = new List<float>();
         chromSelect = new List<int>();
         location = new List<int[]>();
         sharedFitness = new List<double>();
         arround = new List<int>();

         for (int j = 0; j < numChromo; j++)
         {
            int[] gen = new int[numGens];
            location.Add(new int[] { 500,500});
            gens.Add(gen);
            fitness.Add(0);
            arround.Add(0);
            sharedFitness.Add(0);
            probability.Add(0f);
            chromSelect.Add(0);
         }
```

```
}

public generation(List<int[]> new_gens)
{
   gens = new List<int[]>();
   fitness = new List<int>();
   probability = new List<float>();
   chromSelect = new List<int>();
   location = new List<int[]>();
   sharedFitness = new List<double>();
   arround = new List<int>();
   gens = new_gens;
   for (int j = 0; j < numChromo; j++)
   {
      location.Add(new int[] { 500, 500 });
      fitness.Add(0);
      sharedFitness.Add(0);
      arround.Add(0);
      probability.Add(0f);
      chromSelect.Add(0);
   }
}

public void randomize()
{
   Random rand = new Random();
   for (int i = 0; i < numChromo; i++)
   {
      for (int j = 0; j < numGens; j++)
      {

         gens[i][j] = rand.Next() % 4 + 1;
      }
   }
}
public void setFitness()
{
   for (int i = 0; i < numChromo; i++)
   {
      for (int j = 0; j < numGens; j++)            //1 - up || y++
      {                                    //2 - right || x++
         if (gens[i][j] == 1)
         {

            location[i][1]++;
            if (location[i][1] == 1000)
               location[i][1] = 0;
         }                                 //3 - down || y--
         else if (gens[i][j] == 2)
         {

            location[i][0]++;
```

```
                if (location[i][0] == 1000)
                    location[i][0] = 0;
            }  //4 - left || x--
            else if (gens[i][j] == 3)
            {
                location[i][1]--;
                if (location[i][1] == 0)
                    location[i][1] = 1000;
            }
            else {
                location[i][0]--;
                if (location[i][0] == 0)
                    location[i][0] = 1000;
                    }
        }
        List<int> fit = new List<int>();
        fit.Add( Math.Abs(sousage_x1 - location[i][0]) + Math.Abs(sousage_y1 - location[i][1]));
        fit.Add(Math.Abs(sousage_x2 - location[i][0]) + Math.Abs(sousage_y2 - location[i][1]));
        fit.Add(Math.Abs(sousage_x3 - location[i][0]) + Math.Abs(sousage_y3 - location[i][1]));
        fit.Add(Math.Abs(sousage_x4 - location[i][0]) + Math.Abs(sousage_y4 - location[i][1]));
        fitness[i] = fit.Min();
        fitOut.WriteLine(fitness[i]);
      }
      for(int i = 0; i < numChromo; i++)
      {
        double sum = 0;
        for(int j = 0; j < numChromo; j++)
        {
          int tmp = Math.Abs(location[i][0] - location[j][0]) + Math.Abs(location[i][1] -
location[j][1]);
          if (tmp < 50)
          {
            arround[i]++;
            sum += (1 - (double)tmp / 50);
          }
        }
        sharedFitness[i] = (fitness[i])*sum;
        sharefitOut.WriteLine(sharedFitness[i]);
        arrOut.WriteLine(arround[i]);
      }
      fitOut.Close();
      sharefitOut.Close();
      arrOut.Close();
    }

    public void setProbability()
    {
      double mass = 0;
      for (int i = 0; i < numChromo; i++)
      {
        mass += sharedFitness[i];
      }
```

```
    averagefitness = mass / numChromo;
    for (int i = 0; i < numChromo; i++)
    {
        probability[i] = (float)fitness[i] / (float)mass;
    }
}
public int[] newChild()
{

    Random rand = new Random(DateTime.Now.TimeOfDay.Milliseconds + rando);
    rando++;
    if (rando == 10000000) rando = 0;
    int rand_num = rand.Next(numChromo/2);
    float sum = 0f;
    int[] chrom_1 = new int[numGens], chrom_2 = new int[numGens];

    //for (int i = 0; i < 100; i++)
    //{
    //    sum += probability[i] * numGens000000;
    //    if (rand_num <= sum)
    //    {
    //        chromSelect[i]++;
    //        chrom_1 = gens[i];
    //        break;
    //    }

    //}
    chrom_1 = gens[rand_num];                    // for truncate
    sum = 0f;
    rand_num = rand.Next(numChromo/2);
    //for (int i = 0; i < 100; i++)
    //{
    //    sum += probability[i] * numGens000000;
    //    if (rand_num <= sum)
    //    {
    //        chromSelect[i]++;
    //        chrom_2 = gens[i];
    //        break;
    //    }
    //}
    chrom_2 = gens[rand_num];                     // for truncate



    int[] new_chrom = new int[numGens];

    //uniform crossover
    for (int i = 0; i < numGens; i++)
    {
        if (rand.Next() % 2 == 1) new_chrom[i] = chrom_1[i];
        else new_chrom[i] = chrom_2[i];
    }
```

```
    //one point crossover
    //int point = rand.Next() % numGens;
    //for (int i = 0; i < numGens; i++)
    //{
    //    if (i < point) new_chrom[i] = chrom_1[i];
    //    else new_chrom[i] = chrom_2[i];
    //}
    Mutation(new_chrom);
    return new_chrom;
}
public double bestFitness()
{
    return sharedFitness.Min();
}

public void Sort()
{
    for (int i = 0; i < numChromo - 1; i++)
    {
        bool swapped = false;
        for (int j = 0; j < numChromo - i - 1; j++)
        {
            if (sharedFitness[j] > sharedFitness[j + 1])
            {
                int[] tmp_gen = gens[j];
                gens[j] = gens[j + 1];
                gens[j + 1] = tmp_gen;

                int tmp_fit = fitness[j];
                fitness[j] = fitness[j + 1];
                fitness[j + 1] = tmp_fit;

                double tmp_shr_fit = sharedFitness[j];
                sharedFitness[j] = sharedFitness[j + 1];
                sharedFitness[j + 1] = tmp_fit;
                swapped = true;
            }

        }
        if (!swapped) break;
    }
}
public double getAverageFit()
{
    return averagefitness;
}
public void WriteTable(StreamWriter file1, StreamWriter file2)
{
    for (int i = 0; i < numChromo; i++)
    {
        file1.WriteLine(chromSelect[i].ToString());
```

```
            file2.WriteLine(i.ToString());
        }
        file1.WriteLine();
        file1.WriteLine();
    }
    public int[] GetMaxChromo()
    {
        return gens[0];
    }
    public int[] GetChromo(int index)
    {
        return gens[index];
    }

    private void Mutation(int[] chromo)
    {
        for(int i = 0;i<numGens;i++)
        {
            if(mutat.Next()%50 == 1)
            {
                int tmp = mutat.Next() % 4 + 1;
                if (tmp == chromo[i]) chromo[i] = (tmp + 1) % 4 + 1;
            }
        }
    }
    public int getBestArround()
    {
        return arround.Min();
    }

  }
}
```