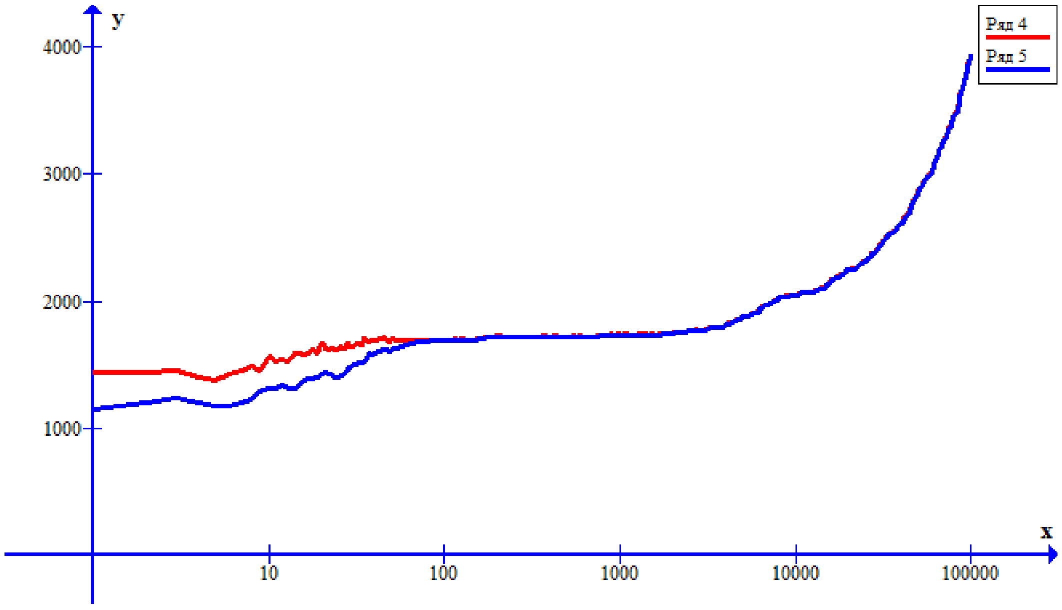

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace siit_4
{
    class Program
    {
        static void Main(string[] args)
        {
            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.RandomizeStatic();
            old_gens.randomize();
            old_gens.setFitness();
            old_gens.setProbability();

            double maxFit = 0;
            int numGeneration = 0;
            for (int j = 0; (j < 1000) && (numGeneration < 100000); numGeneration++)
            {

                numGenFile.WriteLine(numGeneration.ToString());
                Console.WriteLine(old_gens.bestFitness() + " " + old_gens.getAverageFit());
                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.price,old_gens.valume);
                old_gens = new_gens;
                old_gens.setFitness();
                old_gens.setProbability();
                avgFitFile.WriteLine(old_gens.getAverageFit().ToString());
                maxFitFile.WriteLine(old_gens.bestFitness().ToString());
                //Console.ReadKey();
                //if (old_gens.bestFitness() > maxFit)
                //{
                //    maxFit = old_gens.bestFitness();
                //    j = 0;
                //}
                //else j++;

            }
            Console.ReadKey();
            tablenum.Close();
            tableFile.Close();
            numGenFile.Close();
            avgFitFile.Close();
            maxFitFile.Close();
        }
    }
}


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

namespace siit_4
{
    class generation
    {
        static public int numChromo = 40;
        static public int numGens = 20;
        static public int maxValume = 10000;
        List<int[]> gens;
        List<int> fitness { get; }
        List<float> probability { get; }
```

```
 List<int> chromSelect;
 public List<int> price = new List<int>(numGens);
 public List<int> valume = new List<int>(numGens);  //объём
 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>();

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

 public generation(List<int[]> new_gens, List<int> p, List<int> v)
 {
    gens = new List<int[]>();
    fitness = new List<int>();
    probability = new List<float>();
    chromSelect = new List<int>();

    gens = new_gens;
    price = p;
    valume = v;
    for (int j = 0; j < numChromo; j++)
    {
       fitness.Add(0);
       probability.Add(0f);
       chromSelect.Add(0);
    }
 }

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

                gens[i][j] = rand.Next() % 100;

            }
            for (int z = 0; z < numGens; z++)
            {
                _valume += gens[i][z] * valume[z];
            }
            if (_valume < maxValume) break;
            _valume = 0;
        }
    }


}
public void setFitness()
{
    int _valume = 0;
    for (int i = 0; i < numChromo; i++)
    {
        for (int j = 0; j < numGens; j++)
        {
            fitness[i] += gens[i][j] * price[j];
            _valume += gens[i][j] * valume[j];
        }
        if (_valume > maxValume) fitness[i] = 0;
        _valume = 0;
    }

}

public void setProbability()
{
    double mass = 0;
    for (int i = 0; i < numChromo; i++)
    {
        mass += fitness[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 fitness.Max();
}

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


         }

      }
      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() % 20 == 1)
                {
                    int tmp = mutat.Next() % 100;
                    if (tmp == chromo[i]) chromo[i] = (tmp + 1) % 100;
                }
            }
        }

        public void RandomizeStatic()
        {
            Random rand = new Random();
            for (int i = 0; i < numGens; i++)
            {
                price.Add(rand.Next() % 10+1);
                valume.Add(rand.Next() % 10+1);
            }
            }

    }
}
```