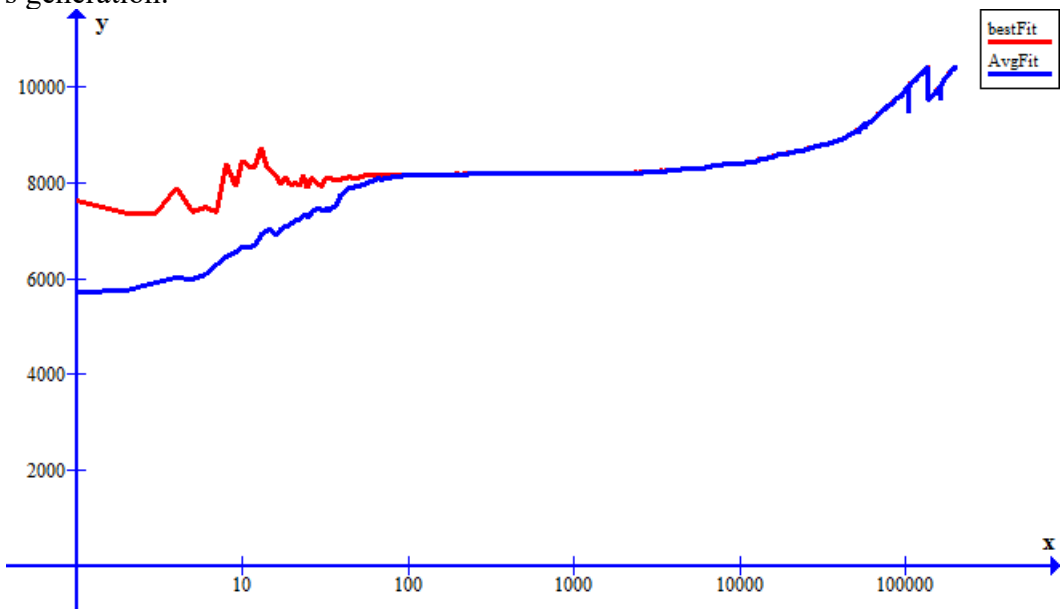
The problem of filling a backpack

We have a volume of the backpack, the price of items, the volume of items and the number of items
We need to maximize the price, minimize the volume so that the volume of the backpack is not exceeded.
We need to show the top five backpacks to the first, last, and three intermediate generation

Fitness vs generation:



First generation:

Price	Volume	#1	#2	#3	#4	#5
7	1	97	66	58	42	27
9	4	13	60	62	82	78
2	5	28	48	36	37	88
2	1	20	53	54	61	31
9	8	19	1	51	60	20
10	6	15	4	29	87	34
10	9	77	81	20	66	44
8	3	12	30	28	24	24
3	9	10	68	97	14	32
9	8	52	77	24	4	34
8	4	95	69	16	58	74
9	1	54	15	20	96	91
10	6	52	9	36	57	55
8	3	22	68	34	48	7
1	3	30	77	12	61	20
6	5	48	2	32	35	77
3	3	95	72	83	38	42
1	9	17	78	74	36	71
6	8	67	18	68	58	67
8	10	39	37	2	5	98
	Volume of all	4900	4292	4102	4528	4566
	Price of all	5853	5308	4715	6659	6565

50000 generation:

Price	Volume	#1	#2	#3	#4	#5
7	1	67	67	67	67	67
9	4	75	75	75	75	75
2	5	86	86	86	86	86
2	1	82	82	82	82	82
9	8	66	66	66	66	66
10	6	61	61	61	61	61
10	9	89	89	89	89	89
8	3	81	81	81	81	81
3	9	99	99	99	99	99
9	8	99	99	99	99	99
8	4	99	99	99	99	99
9	1	82	82	82	82	82
10	6	88	88	88	88	88
8	3	99	99	99	99	99
1	3	31	31	31	31	31
6	5	99	99	99	99	99
3	3	81	81	81	81	81
1	9	86	86	86	86	86
6	8	65	65	65	65	65
8	10	61	61	61	61	61
	Volume of all	8538	8538	8538	8538	8538
	Price of all	10444	10444	10444	10444	10444

100000 generation:

Price	Volume	#1	#2	#3	#4	#5
7	1	80	80	80	80	80
9	4	86	86	86	86	86
2	5	99	99	99	99	99
2	1	95	95	95	95	95
9	8	73	73	73	73	73
10	6	75	75	75	75	75
10	9	99	99	99	99	99
8	3	90	90	90	90	90
3	9	11	11	11	11	11
9	8	99	99	99	99	99
8	4	99	99	99	99	99
9	1	94	94	94	94	94
10	6	99	99	99	99	99
8	3	99	99	99	99	99
1	3	39	39	39	39	39
6	5	6	6	6	6	6
3	3	95	95	95	95	95
1	9	95	95	95	95	95
6	8	80	80	80	80	80
8	10	71	71	71	71	71
	Volume of all	8118	8118	8118	8118	8118
	Price of all	10686	10686	10686	10686	10686

150000 generation:

Price	Volume	#1	#2	#3	#4	#5
7	1	90	90	90	90	90
9	4	98	98	98	98	98
2	5	99	99	99	99	99
2	1	99	99	99	99	99
9	8	89	89	89	89	89
10	6	86	86	86	86	86
10	9	99	99	99	99	99
8	3	95	95	95	95	95
3	9	22	22	22	22	22
9	8	99	99	99	99	99
8	4	99	99	99	99	99
9	1	99	99	99	99	99
10	6	99	99	99	99	99
8	3	99	99	99	99	99
1	3	45	45	45	45	45
6	5	24	24	24	24	24
3	3	99	99	99	99	99
1	9	99	99	99	99	99
6	8	89	89	89	89	89
8	10	80	80	80	80	80
	Volume of all	8811	8811	8811	8811	8811
	Price of all	11500	11500	11500	11500	11500

200000 generation:

Price	Volume	#1	#2	#3	#4	#5
7	1	99	99	99	99	99
9	4	99	99	99	99	99
2	5	99	99	99	99	99
2	1	8	8	8	8	8
9	8	96	96	96	96	96
10	6	99	99	99	99	99
10	9	99	99	99	99	99
8	3	99	99	99	99	99
3	9	32	32	32	32	32
9	8	99	99	99	99	99
8	4	99	99	99	99	99
9	1	99	99	99	99	99
10	6	99	99	99	99	99
8	3	99	99	99	99	99
1	3	54	54	54	54	54
6	5	33	33	33	33	33
3	3	7	7	7	7	7
1	9	0	0	0	0	0
6	8	98	98	98	98	98
8	10	84	84	84	84	84
	Volume of all	7986	7986	7986	7986	7986
	Price of all	11419	11419	11419	11419	11419

```
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");
         StreamWriter gensFile = new StreamWriter("gens.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 < 200000); numGeneration++)
         {

            numGenFile.WriteLine(numGeneration.ToString());
            Console.WriteLine(old_gens.bestFitness() + " " + old_gens.getAverageFit() + " " +
old_gens.volume.Max());
            if (old_gens.bestFitness() == 0) break;
            List<int[]> new_tmp = new List<int[]>();
            old_gens.Sort();                                    //for truncate
            if(numGeneration == 0
               || numGeneration == 50000
               || numGeneration == 100000
               || numGeneration == 150000
               || numGeneration == 199999)
            {
               old_gens.GetFivebest();
               generation.indexOfMax++;
            }
            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;
    static public int indexOfMax = 0;
    static public int countOfObject = 100;
    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 List<int> volume = new List<int>(numChromo);
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);
      volume.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);
      volume.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() % countOfObject;

            }
            for (int z = 0; z < numGens; z++)
            {
                _valume += gens[i][z] * valume[z];
            }
            if (_valume < maxValume) break;
            else volume[i] = _valume;
            _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;
            volume[i] = _valume;
        }
        else volume[i] = _valume;
        _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() % countOfObject;
            if (tmp == chromo[i]) chromo[i] = (tmp + 1) % countOfObject;
        }
    }
}

public void RandomizeStatic()
{
    Random rand = new Random();
    StreamWriter priceFile = new StreamWriter("price.txt");
    StreamWriter valumeFile = new StreamWriter("valume.txt");
    for (int i = 0; i < numGens; i++)
    {
        price.Add(rand.Next() % 10 + 1);
        valume.Add(rand.Next() % 10 + 1);
        priceFile.WriteLine(price[i]);
        valumeFile.WriteLine(valume[i]);
    }
    priceFile.Close();
    valumeFile.Close();
}

public void GetFivebest()
{

    StreamWriter gensFile = new StreamWriter("gens"+indexOfMax+".txt");
    StreamWriter volumeFile = new StreamWriter("volume" + indexOfMax + ".txt");
    StreamWriter priceFile = new StreamWriter("prive" + indexOfMax + ".txt");
    for(int i = 0; i < 5; i++)
    {
        for(int j = 0; j<numGens;j++)
        gensFile.WriteLine(gens[i][j]);
        gensFile.WriteLine();
        volumeFile.WriteLine(volume[i]);
        priceFile.WriteLine(fitness[i]);
```

```
            }
            gensFile.Close();
            volumeFile.Close();
            priceFile.Close();
        }
    }
}
```