27.10 Roma Rudsky

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;
using System.Text;
using System.Threading.Tasks;
namespace CIIT4
{
   class Program
   {
      static void Main(string[] args)
      {
         int[] prices = new int[20];
         int[] volumes = new int[20];
         StreamWriter pricewriter = new StreamWriter("prices.txt");
         StreamWriter volumeswriter = new StreamWriter("volumes.txt");
         Random random = new Random(DateTime.Now.Millisecond);
         for (int i = 0; i < 20; i++)
         {
            prices[i] = random.Next() % 10 + 1;
            volumes[i] = random.Next() % 10 + 1;
            pricewriter.WriteLine(prices[i]);
            volumeswriter.WriteLine(volumes[i]);
         }
         pricewriter.Close();
         volumeswriter.Close();
         int maxBagVolume = 0;
         List<Bag> bags = new List<Bag>();
         for (int i = 0; i < 40; i++)
         {
            Bag bag = new Bag(prices, volumes, random);
            bags.Add(bag);
            //bag.ShowBag();


            if (maxBagVolume < bag.totalVolume)
               maxBagVolume = bag.totalVolume;
         }
         Console.WriteLine(bags[39].totalPrice);
         Console.WriteLine(maxBagVolume);


         var sortedBags = bags.OrderBy(x => x.totalPrice).ToList();
         bags = sortedBags;

         //while (true)
         StreamWriter swr;
         StreamWriter fitness = new StreamWriter("fit.txt");
         for (int q = 0; q < 1000; q++)
         {
            double avg = 0;
            double Allfit = 0;
            List<Bag> newBags = new List<Bag>();
            for (int i = 0; i < 40; i++)
            {
               int j = random.Next(30, 39);
               int k = random.Next(30, 39);
               Bag newBag = bags[j].ChooseItems(bags[k], random);
               //newBag.ShowBag();
               Allfit += newBag.totalPrice;
```

```
                newBags.Add(newBag);
            }


            bags = newBags.OrderBy(x => x.totalVolume < 1000 ? x.totalPrice : 0).ToList();
            avg = Allfit / 40;
            fitness.Write(bags[39].totalPrice + " ");
            fitness.WriteLine(avg);
            if (q == 0 || q == 5 || q == 50 || q == 500|| q == 999)
            {
                swr = new StreamWriter(q + ".txt");
                for (int i = 0; i < 40; i++)
                {
                    swr.WriteLine(bags[i]);
                }
                swr.Close();
            }


            Console.WriteLine("Price: " + bags[39].totalPrice);
            Console.WriteLine("Volume: " + bags[39].totalVolume);
        }
        fitness.Close();

        Console.ReadKey();
    }
  }
}

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

namespace CIIT4
{
  class Bag
  {
      int[] prices;
      int[] volumes;
      int[] counts;
      public int totalVolume { get; set; }
      public int totalPrice { get; set; }

      public Bag(int[] prices, int[] volumes, Random random)
      {
          this.prices = prices;
          this.volumes = volumes;
          counts = new int[20];
          //Random random = new Random(DateTime.Now.Millisecond);
          for (int i = 0; i < prices.Length; i++)
          {
              counts[i] = random.Next() % 13 + 1;

              totalPrice += counts[i] * prices[i];
              totalVolume += counts[i] * volumes[i];
          }
      }
      public Bag(int[] prices, int[] volumes, int[] counts)
      {
          this.prices = prices;
          this.volumes = volumes;
          this.counts = counts;
          for (int i = 0; i < prices.Length; i++)
```

```
            {
                totalPrice += counts[i] * prices[i];
                totalVolume += counts[i] * volumes[i];
            }
        }
        public Bag ChooseItems(Bag bag, Random random)
        {
            //Random random = new Random(DateTime.Now.Millisecond);
            int[] newCounts = new int[20];
            for (int i = 0; i < prices.Length; i++)
            {
                int key = random.Next() % 1;
                if (key == 0)
                    newCounts[i] = this.counts[i];
                else
                    newCounts[i] = bag.counts[i];
            }
            newCounts[random.Next(0, 19)] = random.Next(1, 10);

                return new Bag(prices, volumes, newCounts);
        }

        public override string ToString()
        {
            StringBuilder sb = new StringBuilder();
            for (int i = 0; i < 20; i++)
            {
                sb.Append(counts[i] + " ");
            }
            sb.Append(totalVolume + " ");
            sb.Append(totalPrice);
            return sb.ToString();
        }
        public void ShowBag()
        {
            Console.WriteLine("Total Price: " + totalPrice);
            Console.WriteLine("Total Volume: " + totalVolume);
            Console.WriteLine("_____");
        }
    }
}
```

Graphs:

Min fitness = 930.
Max fitness = 1102.
Generations = 1000.

Price = 1 – 10

Volume = 1 - 10

Items	1	2	3	4	5	6	7	8	9	10
Price	1	7	8	9	9	7	9	8	1	8
Volu	4	4	1	9	10	8	4	6	5	4

me										

Items	11	12	13	14	15	16	17	18	19	20
Price	4	3	5	7	6	2	1	1	6	6
Volume	3	9	7	4	6	7	10	1	6	3

5 better chromosomes in generation:

Max size of a knapsack <= 1000.

1st generation:

Chromosomes	1	2	3	4	5
Items count	4	2	13	9	4
	13	12	13	13	13
	6	6	10	4	6
	4	13	1	13	9
	9	12	11	9	9
	1	3	6	2	1
	6	8	10	13	6
	12	7	1	4	12
	13	3	6	10	13
	8	8	12	6	8
	13	8	5	11	13
	8	9	10	9	8
	8	7	4	7	8
	10	6	9	13	10
	11	4	11	2	11
	11	12	7	4	11
	11	7	10	10	11
	3	1	3	1	4
	8	10	11	13	8

	9	12	13	10	9
Total Volume	939	897	897	929	985
Total Price	884	899	904	917	930

5st generation:

Chromosomes	**1**	**2**	**3**	**4**	**5**
Items count	9	9	9	9	9
	13	13	13	13	13
	4	4	4	4	4
	13	13	13	13	13
	9	9	9	9	9
	2	2	4	6	4
	13	13	13	13	13
	6	4	4	4	4
	10	10	9	10	7
	6	8	9	6	6
	11	11	11	11	11
	7	7	9	9	9
	9	9	7	7	7
	13	13	13	13	13
	2	2	2	2	8
	9	9	4	9	4
	9	9	9	9	9
	1	1	1	6	1
	13	13	13	13	13
	10	10	10	10	10
Total Volume	962	958	942	991	956
Total Price	946	946	953	959	963

50st generation:

Chromosomes	1	2	3	4	5
Items count	8	8	8	8	5
	13	13	13	13	13
	9	9	8	9	9
	13	13	13	13	13
	9	9	9	9	9
	9	9	9	9	9
	13	13	13	13	13
	8	8	8	8	8
	8	6	8	6	8
	9	9	9	9	9
	11	11	11	11	11
	6	6	9	6	6
	8	8	8	8	8
	13	13	13	13	13
	7	8	8	8	9
	7	7	7	7	7
	2	2	2	5	2
	7	5	5	5	5
	13	13	13	13	13
	10	10	10	10	10
Total Volume	969	963	999	993	967
Total Price	1089	1091	1094	1094	1096

500st generation:

Chromosomes	1	2	3	4	5
Items count	2	3	3	2	3
	13	13	13	13	13
	9	9	9	9	9
	13	13	13	13	13
	9	9	9	9	9
	9	9	9	9	9
	13	13	13	13	13
	9	9	9	9	9
	1	3	3	6	3
	9	9	9	9	9
	9	9	9	9	9
	5	5	5	5	7
	8	8	8	8	8
	13	13	13	13	13
	9	9	9	9	9
	9	8	8	7	8
	7	7	7	8	7
	3	3	3	3	3
	13	13	13	13	13
	10	10	10	10	10
Total Volume	973	980	980	994	998
Total Price	1090	1091	1091	1092	1097

1000st generation:

Chromosomes	1	2	3	4	5
Items count	7	7	7	7	7
	13	13	13	13	13
	9	9	9	9	9
	13	13	13	13	13
	9	9	9	9	9
	8	9	9	9	9
	13	13	13	13	13
	9	9	9	9	9
	2	2	2	2	2
	9	9	9	9	9
	9	9	9	8	9
	7	8	9	9	9
	9	6	8	9	9
	13	13	13	13	13
	9	9	8	8	8
	9	9	9	9	9
	3	4	3	3	3
	4	9	4	4	4
	13	13	13	13	13
	10	10	10	10	10
Total Volume	976	987	989	993	996
Total Price	1097	1098	1099	1100	1102

Graphs:

Min fitness = 930.
Max fitness = 1102.
Generations = 1000.