Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
федерального государственного бюджетного образовательного учреждения высшего образования
**«Владимирский государственный университет
Имени Александра Григорьевича и Николая Григорьевича Столетовых»
(МИВлГУ)**

Факультет ИТ

Кафедра ПИн

ЛАБОРАТОРНАЯ РАБОТА №1

по САОД

Тема Алгоритмы сортировки линейных коллекций данных

Руководитель

Привезенцев Д.Г.
(фамилия, инициалы)

(подпись) (дата)

Студент ПИн-119
(группа)

Лямина И.А.
(фамилия, инициалы)

(подпись) (дата)

Муром 2021

Лабораторная работа №1

Тема: алгоритмы сортировки линейных коллекций данных.

Цель: необходимо изучить ряд известных алгоритмов сортировки.

Задание на лабораторную работу

1. Реализовать три метода сортировки коллекций согласно таблице 1.

2. Реализовать управляющую программу(ы), включающую:

a. ввод исходных данных: из файла, с консоли, генерацией случайных чисел (способа ввода данных предусмотреть в программе путем ввода выбора пользователя, при этом размер массива тоже указывается во время выполнения программы);

b. ввод на экран исходных данных;

c. выбор направления сортировки;

d. выбор коллекции (массив, связанный список);

e. сортировку коллекции;

f. вывод на экран результата работы;

g. замер времени выполнения сортировки.

3. Реализовать вывод отладочной информации:

a. количество сравнений двух элементов;

b. количество перестановок двух элементов.

4. Выполнить исследование реализованных трех алгоритмах на разных коллекциях:

a. Выполнить замеры времени для коллекций размера 10, 100, 1000, 10000, 100000, 1000000 элементов.

b. Заполнить таблицы 2-4 для каждого анализируемого алгоритма.

c. Построить графики зависимости времени сортировки, числа перестановок, числа сравнений элементов от размера коллекции для каждого анализируемого алгоритма.

d. Сделать выводы по таблицам и графику.

Алгоритмы сортировки

1)Метод сортировки перемешиванием;

2)Метод сортировка Шелла;

3)Метод сортировки слиянием;

Выполнение задания:

Листинг программы:

```
using System;
using System.Diagnostics;

namespace LB_1
{
   class Program
   {
      //Метод сортировки перемешиванием
      static void ShakerSort(int[] myint, out ulong n_c, out ulong n_p)
      {
         n_c = 0;
         n_p = 0;
         int left = 0, right = myint.Length - 1;

         while (left < right)
         {
            for (int i = left; i < right; i++)
            {
               n_c++;
               if (myint[i] > myint[i + 1])
               {
                  int glass = myint[i];
                  myint[i] = myint[i + 1];
                  myint[i + 1] = glass;

                  n_p++;
               }
            }
            right--;

            for (int i = right; i > left; i--)
            {
               n_c++;
               if (myint[i - 1] > myint[i])
               {
                  int glass = myint[i];
                  myint[i] = myint[i - 1];
                  myint[i - 1] = glass;
```

```
                n_p++;
            }
        }
        left++;
    }
}

//Метод сортировка Шелла
static void ShellSort(int[] myint, out ulong n_c, out ulong n_p)
{
    n_c = 0;
    n_p = 0;
    for (int s = myint.Length / 2; s > 0; s /= 2)
    {
        for (int i = 0; i < myint.Length; i++)
        {
            for (int j = i + s; j < myint.Length; j += s)
            {
                n_c++;
                if (myint[i] > myint[j])
                {
                    int glass = myint[i];
                    myint[i] = myint[j];
                    myint[j] = glass;

                    n_p++;
                }
            }
        }
    }
}

//сортировка слиянием
static int[] MergeSort(int[] array, ref ulong n_c, ref ulong n_p)
{
    n_c = 0;
    n_p = 0;
    return MergeSort(array, 0, array.Length - 1, ref n_c, ref n_p);
}
static int[] MergeSort(int[] array, int lowIndex, int highIndex, ref ulong n_c, ref ulong n_p)
{
    n_c++;
    if (lowIndex < highIndex)
    {
        var middleIndex = (lowIndex + highIndex) / 2;
        MergeSort(array, lowIndex, middleIndex, ref n_c, ref n_p);
        MergeSort(array, middleIndex + 1, highIndex, ref n_c, ref n_p);
        Merge(array, lowIndex, middleIndex, highIndex, ref n_c, ref n_p);
    }

    return array;
}
//метод для слияния массивов
static void Merge(int[] array, int lowIndex, int middleIndex, int highIndex, ref ulong n_c, ref ulong n_p)
{
    var left = lowIndex;
    var right = middleIndex + 1;
    var tempArray = new int[highIndex - lowIndex + 1];
    var index = 0;

    while ((left <= middleIndex) && (right <= highIndex))
    {
        n_c++;
        if (array[left] < array[right])
        {
```

```
            n_p++;
            tempArray[index] = array[left];
            left++;
        }
        else
        {
            n_p++;
            tempArray[index] = array[right];
            right++;
        }

        index++;
    }

    for (var i = left; i <= middleIndex; i++)
    {
        n_p++;
        tempArray[index] = array[i];
        index++;
    }

    for (var i = right; i <= highIndex; i++)
    {
        n_p++;
        tempArray[index] = array[i];
        index++;
    }

    for (var i = 0; i < tempArray.Length; i++)
    {
        n_p++;
        array[lowIndex + i] = tempArray[i];
    }
}




static void Main(string[] args)
{
    void ShowArr(int[] arr)
    {
        for (int i = 0; i < arr.Length; i++)
        {
            Console.WriteLine("arr[" + i + "] = " + arr[i]);
        }
    }
    void CreatArray_3(int[] arr_1, int[] arr_2, int[] arr_3)
    {
        var rnd = new Random();
        for (int i = 0; i < arr_1.Length; i++)
        {
            arr_1[i] = rnd.Next(0, 100);
        }
        Array.Copy(arr_1, arr_2, arr_1.Length);
        Array.Copy(arr_1, arr_3, arr_1.Length);
    }

    int[] arr_1_1 = new int[10];
    int[] arr_1_2 = new int[10];
    int[] arr_1_3 = new int[10];
    CreatArray_3(arr_1_1, arr_1_2, arr_1_3);
    int[] arr_2_1 = new int[100];
    int[] arr_2_2 = new int[100];
    int[] arr_2_3 = new int[100];
```

```
CreatArray_3(arr_2_1, arr_2_2, arr_2_3);
int[] arr_3_1 = new int[1000];
int[] arr_3_2 = new int[1000];
int[] arr_3_3 = new int[1000];
CreatArray_3(arr_3_1, arr_3_2, arr_3_3);
int[] arr_4_1 = new int[10000];
int[] arr_4_2 = new int[10000];
int[] arr_4_3 = new int[10000];
CreatArray_3(arr_4_1, arr_4_2, arr_4_3);
int[] arr_5_1 = new int[100000];
int[] arr_5_2 = new int[100000];
int[] arr_5_3 = new int[100000];
CreatArray_3(arr_5_1, arr_5_2, arr_5_3);
int[] arr_6_1 = new int[1000000];
int[] arr_6_2 = new int[1000000];
int[] arr_6_3 = new int[1000000];
CreatArray_3(arr_6_1, arr_6_2, arr_6_3);

ulong n_c_1_1;
ulong n_p_1_1;
ulong n_c_1_2;
ulong n_p_1_2;
ulong n_c_1_3 = 0;
ulong n_p_1_3 = 0;
ulong n_c_2_1;
ulong n_p_2_1;
ulong n_c_2_2;
ulong n_p_2_2;
ulong n_c_2_3 = 0;
ulong n_p_2_3 = 0;
ulong n_c_3_1;
ulong n_p_3_1;
ulong n_c_3_2;
ulong n_p_3_2;
ulong n_c_3_3 = 0;
ulong n_p_3_3 = 0;
ulong n_c_4_1;
ulong n_p_4_1;
ulong n_c_4_2;
ulong n_p_4_2;
ulong n_c_4_3 = 0;
ulong n_p_4_3 = 0;
ulong n_c_5_1;
ulong n_p_5_1;
ulong n_c_5_2;
ulong n_p_5_2;
ulong n_c_5_3 = 0;
ulong n_p_5_3 = 0;
ulong n_c_6_1;
ulong n_p_6_1;
ulong n_c_6_2;
ulong n_p_6_2;
ulong n_c_6_3 = 0;
ulong n_p_6_3 = 0;

Stopwatch stopWatch = new Stopwatch();
//Метод сортировки перемешиванием
stopWatch.Start();
ShakerSort(arr_1_1, out n_c_1_1, out n_p_1_1);
stopWatch.Stop();
TimeSpan ts_1_1 = stopWatch.Elapsed;

stopWatch.Restart();
ShakerSort(arr_2_1, out n_c_2_1, out n_p_2_1);
stopWatch.Stop();
TimeSpan ts_2_1 = stopWatch.Elapsed;
```

```
stopWatch.Restart();
ShakerSort(arr_3_1, out n_c_3_1, out n_p_3_1);
stopWatch.Stop();
TimeSpan ts_3_1 = stopWatch.Elapsed;

stopWatch.Restart();
ShakerSort(arr_4_1, out n_c_4_1, out n_p_4_1);
stopWatch.Stop();
TimeSpan ts_4_1 = stopWatch.Elapsed;

stopWatch.Restart();
ShakerSort(arr_5_1, out n_c_5_1, out n_p_5_1);
stopWatch.Stop();
TimeSpan ts_5_1 = stopWatch.Elapsed;

stopWatch.Restart();
ShakerSort(arr_6_1, out n_c_6_1, out n_p_6_1);
stopWatch.Stop();
TimeSpan ts_6_1 = stopWatch.Elapsed;

//Метод сортировка Шелла
stopWatch.Restart();
ShellSort(arr_1_2, out n_c_1_2, out n_p_1_2);
stopWatch.Stop();
TimeSpan ts_1_2 = stopWatch.Elapsed;

stopWatch.Start();
ShellSort(arr_2_2, out n_c_2_2, out n_p_2_2);
stopWatch.Stop();
TimeSpan ts_2_2 = stopWatch.Elapsed;

stopWatch.Restart();
ShellSort(arr_3_2, out n_c_3_2, out n_p_3_2);
stopWatch.Stop();
TimeSpan ts_3_2 = stopWatch.Elapsed;

stopWatch.Start();
ShellSort(arr_4_2, out n_c_4_2, out n_p_4_2);
stopWatch.Stop();
TimeSpan ts_4_2 = stopWatch.Elapsed;

stopWatch.Restart();
ShellSort(arr_5_2, out n_c_5_2, out n_p_5_2);
stopWatch.Stop();
TimeSpan ts_5_2 = stopWatch.Elapsed;

stopWatch.Restart();
ShellSort(arr_6_2, out n_c_6_2, out n_p_6_2);
stopWatch.Stop();
TimeSpan ts_6_2 = stopWatch.Elapsed;

//сортировка слиянием
stopWatch.Restart();
MergeSort(arr_1_3, ref n_c_1_3, ref n_p_1_3);
stopWatch.Stop();
TimeSpan ts_1_3 = stopWatch.Elapsed;

stopWatch.Restart();
MergeSort(arr_2_3, ref n_c_2_3, ref n_p_2_3);
stopWatch.Stop();
TimeSpan ts_2_3 = stopWatch.Elapsed;

stopWatch.Restart();
MergeSort(arr_3_3, ref n_c_3_3, ref n_p_3_3);
stopWatch.Stop();
```

```
        TimeSpan ts_3_3 = stopWatch.Elapsed;

        stopWatch.Restart();
        MergeSort(arr_4_3, ref n_c_4_3, ref n_p_4_3);
        stopWatch.Stop();
        TimeSpan ts_4_3 = stopWatch.Elapsed;

        stopWatch.Restart();
        MergeSort(arr_5_3, ref n_c_5_3, ref n_p_5_3);
        stopWatch.Stop();
        TimeSpan ts_5_3 = stopWatch.Elapsed;

        stopWatch.Restart();
        MergeSort(arr_6_3, ref n_c_6_3, ref n_p_6_3);
        stopWatch.Stop();
        TimeSpan ts_6_3 = stopWatch.Elapsed;

        TimeSpan Comparison_Min(TimeSpan ts_1, TimeSpan ts_2, TimeSpan ts_3)
        {
           if (TimeSpan.Compare(ts_1,ts_2)==-1 & TimeSpan.Compare(ts_1, ts_3) == -1) return ts_1;
           else if (TimeSpan.Compare(ts_2, ts_1) == -1 & TimeSpan.Compare(ts_2, ts_3) == -1) return ts_2;
           else return ts_3;
        }
        TimeSpan Comparison_Max(TimeSpan ts_1, TimeSpan ts_2, TimeSpan ts_3)
        {
           if (TimeSpan.Compare(ts_1, ts_2) == 1 & TimeSpan.Compare(ts_1, ts_3) == 1) return ts_1;
           else if (TimeSpan.Compare(ts_2, ts_1) == 1 & TimeSpan.Compare(ts_2, ts_3) == 1) return ts_2;
           else return ts_3;
        }
        TimeSpan Average(TimeSpan ts_1, TimeSpan ts_2, TimeSpan ts_3)
        {
           return TimeSpan.FromMilliseconds((ts_1.TotalMilliseconds + ts_2.TotalMilliseconds + ts_3.TotalMilliseconds) /
3);
        }

        Console.WriteLine("{0,28}{1,17}{2,17}{3,17}{4,17}{5,17}", 10, 100, 1000, 10000, 100000, 1000000);
        Console.WriteLine("Алгоритм 1  " + ts_1_1 + " " + ts_2_1 + " " + ts_3_1 + " " + ts_4_1 + " " + ts_5_1 + " " +
ts_6_1);
        Console.WriteLine("Алгоритм 2  " + ts_1_2 + " " + ts_2_2 + " " + ts_3_2 + " " + ts_4_2 + " " + ts_5_2 + " " +
ts_6_2);
        Console.WriteLine("Алгоритм 3  " + ts_1_3 + " " + ts_2_3 + " " + ts_3_3 + " " + ts_4_3 + " " + ts_5_3 + " " +
ts_6_3);
        Console.WriteLine("----------------------------------------------------------------------------------------------------------------------
");
        Console.WriteLine("Мин.знач    " + Comparison_Min(ts_1_1, ts_1_2, ts_1_3) + " " + Comparison_Min(ts_2_1,
ts_2_2, ts_2_3) + " " + Comparison_Min(ts_3_1, ts_3_2, ts_3_3) + " " + Comparison_Min(ts_4_1, ts_4_2, ts_4_3) + " " +
Comparison_Min(ts_5_1, ts_5_2, ts_5_3) + " " + Comparison_Min(ts_6_1, ts_6_2, ts_6_3) );
        Console.WriteLine("Макс.знач    " + Comparison_Max(ts_1_1, ts_1_2, ts_1_3) + " " + Comparison_Max(ts_2_1,
ts_2_2, ts_2_3) + " " + Comparison_Max(ts_3_1, ts_3_2, ts_3_3) + " " + Comparison_Max(ts_4_1, ts_4_2, ts_4_3) + " " +
Comparison_Max(ts_5_1, ts_5_2, ts_5_3) + " " + Comparison_Max(ts_6_1, ts_6_2, ts_6_3) );
        Console.WriteLine("Ср.знач          {0,16}{1,17}{2,17}{3,17}{4,17}{5,17}", Average(ts_1_1, ts_1_2, ts_1_3),
Average(ts_2_1, ts_2_2, ts_2_3), Average(ts_3_1, ts_3_2, ts_3_3), Average(ts_4_1, ts_4_2, ts_4_3), Average(ts_5_1,
ts_5_2, ts_5_3), Average(ts_6_1, ts_6_2, ts_6_3) );




        ulong Comparison_Min_Int(ulong item_1, ulong item_2, ulong item_3)
        {
           if (item_1 < item_2 & item_1 < item_3) return item_1;
           else if (item_2 < item_1 & item_2 < item_3) return item_2;
           else return item_3;
        }
        ulong Comparison_Max_Int(ulong item_1, ulong item_2, ulong item_3)
        {
```

```
            if (item_1 > item_2 & item_1 > item_3) return item_1;
            else if (item_2 > item_1 & item_2 > item_3) return item_2;
            else return item_3;
        }
        ulong Average_Int(ulong item_1, ulong item_2, ulong item_3)
        {
            return (item_1 + item_2 + item_3) / 3;
        }

        Console.WriteLine("\n\nЧисло сравнений значений элементов");
        Console.WriteLine("{0,16}{1,10}{2,10}{3,10}{4,20}{5,30}", 10, 100, 1000, 10000, 100000, 1000000);
        Console.WriteLine("-----------------------------------------------------------------------------------------------------------------------
");
        Console.WriteLine("Алгоритм 1   {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", n_c_1_1, n_c_2_1, n_c_3_1, n_c_4_1,
n_c_5_1, n_c_6_1 );
        Console.WriteLine("Алгоритм 2   {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", n_c_1_2, n_c_2_2, n_c_3_2, n_c_4_2,
n_c_5_2, n_c_6_2 );
        Console.WriteLine("Алгоритм 3   {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", n_c_1_3, n_c_2_3, n_c_3_3, n_c_4_3,
n_c_5_3, n_c_6_3 );
        Console.WriteLine("-----------------------------------------------------------------------------------------------------------------------
");
        Console.WriteLine("Мин.знач     {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", Comparison_Min_Int(n_c_1_1, n_c_1_2,
n_c_1_3), Comparison_Min_Int(n_c_2_1, n_c_2_2, n_c_2_3), Comparison_Min_Int(n_c_3_1, n_c_3_2, n_c_3_3),
Comparison_Min_Int(n_c_4_1, n_c_4_2, n_c_4_3), Comparison_Min_Int(n_c_5_1, n_c_5_2, n_c_5_3),
Comparison_Min_Int(n_c_6_1, n_c_6_2, n_c_6_3) );
        Console.WriteLine("Макс.знач    {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", Comparison_Max_Int(n_c_1_1, n_c_1_2,
n_c_1_3), Comparison_Max_Int(n_c_2_1, n_c_2_2, n_c_2_3), Comparison_Max_Int(n_c_3_1, n_c_3_2, n_c_3_3),
Comparison_Max_Int(n_c_4_1, n_c_4_2, n_c_4_3), Comparison_Max_Int(n_c_5_1, n_c_5_2, n_c_5_3),
Comparison_Max_Int(n_c_6_1, n_c_6_2, n_c_6_3) );
        Console.WriteLine("Ср.знач      {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", Average_Int(n_c_1_1, n_c_1_2, n_c_1_3),
Average_Int(n_c_2_1, n_c_2_2, n_c_2_3), Average_Int(n_c_3_1, n_c_3_2, n_c_3_3), Average_Int(n_c_4_1, n_c_4_2,
n_c_4_3), Average_Int(n_c_5_1, n_c_5_2, n_c_5_3), Average_Int(n_c_6_1, n_c_6_2, n_c_6_3) );

        Console.WriteLine("\n\nЧисло перестановок элементов");
        Console.WriteLine("{0,16}{1,10}{2,10}{3,10}{4,20}{5,30}", 10, 100, 1000, 10000, 100000, 1000000);
        Console.WriteLine("-----------------------------------------------------------------------------------------------------------------------
");
        Console.WriteLine("Алгоритм 1   {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", n_p_1_1, n_p_2_1, n_p_3_1, n_p_4_1,
n_p_5_1, n_p_6_1 );
        Console.WriteLine("Алгоритм 2   {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", n_p_1_2, n_p_2_2, n_p_3_2, n_p_4_2,
n_p_5_2, n_p_6_2 );
        Console.WriteLine("Алгоритм 3   {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", n_p_1_3, n_p_2_3, n_p_3_3, n_p_4_3,
n_p_5_3, n_p_6_3 );
        Console.WriteLine("-----------------------------------------------------------------------------------------------------------------------
");
        Console.WriteLine("Мин.знач     {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", Comparison_Min_Int(n_p_1_1, n_p_1_2,
n_p_1_3), Comparison_Min_Int(n_p_2_1, n_p_2_2, n_p_2_3), Comparison_Min_Int(n_p_3_1, n_p_3_2, n_p_3_3),
Comparison_Min_Int(n_p_4_1, n_p_4_2, n_p_4_3), Comparison_Min_Int(n_p_5_1, n_p_5_2, n_p_5_3),
Comparison_Min_Int(n_p_6_1, n_p_6_2, n_p_6_3) );
        Console.WriteLine("Макс.знач    {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", Comparison_Max_Int(n_p_1_1, n_p_1_2,
n_p_1_3), Comparison_Max_Int(n_p_2_1, n_p_2_2, n_p_2_3), Comparison_Max_Int(n_p_3_1, n_p_3_2, n_p_3_3),
Comparison_Max_Int(n_p_4_1, n_p_4_2, n_p_4_3), Comparison_Max_Int(n_p_5_1, n_p_5_2, n_p_5_3),
Comparison_Max_Int(n_p_6_1, n_p_6_2, n_p_6_3) );
        Console.WriteLine("Ср.знач      {0,4}{1,10}{2,10}{3,10}{4,20}{5,30}", Average_Int(n_p_1_1, n_p_1_2, n_p_1_3),
Average_Int(n_p_2_1, n_p_2_2, n_p_2_3), Average_Int(n_p_3_1, n_p_3_2, n_p_3_3), Average_Int(n_p_4_1, n_p_4_2,
n_p_4_3), Average_Int(n_p_5_1, n_p_5_2, n_p_5_3), Average_Int(n_p_6_1, n_p_6_2, n_p_6_3) );

        Console.ReadKey();
    }
  }
}
```

Выбрать C:\Users\user\Documents\Институт\СиАОД\LB_1\LB_1\bin\Debug\LB_1.exe

```
                        10              100             1000            10000           100000          1000000
Алгоритм 1  00:00:00.0005300 00:00:00.0000692 00:00:00.0083616 00:00:00.8373259 00:00:51.6490010 01:27:31.2588186
Алгоритм 2  00:00:00.0004083 00:00:00.0004675 00:00:00.0043090 00:00:00.5318015 00:00:45.6592432 01:15:20.5434520
Алгоритм 3  00:00:00.0010780 00:00:00.0000439 00:00:00.0003463 00:00:00.0038426 00:00:00.0493225 00:00:00.4765772
-------------------------------------------------------------------------------------------------------------------
Мин.знач    00:00:00.0004083 00:00:00.0000439 00:00:00.0003463 00:00:00.0038426 00:00:00.0493225 00:00:00.4765772
Макс.знач   00:00:00.0010780 00:00:00.0004675 00:00:00.0083616 00:00:00.8373259 00:00:51.6490010 01:27:31.2588186
Ср.знач     00:00:00.0010000         00:00:00 00:00:00.0040000 00:00:00.4580000 00:00:32.4530000 00:54:17.4260000


Число сравнений значений элементов
             10       100      1000     10000           100000                1000000
-------------------------------------------------------------------------------------------------------------------
Алгоритм 1    45      4950    499500  49995000       4999950000             499999500000
Алгоритм 2    70      7919    798131  98176009       8331077776             804477914612
Алгоритм 3    41       737     10693    140213          1732347                 20619289
-------------------------------------------------------------------------------------------------------------------
Мин.знач      41       737     10693    140213          1732347                 20619289
Макс.знач     70      7919    798131  98176009       8331077776             804477914612
Ср.знач       52      4535    436108  49437074       4444253374             434832677967


Число перестановок элементов
             10       100      1000     10000           100000                1000000
-------------------------------------------------------------------------------------------------------------------
Алгоритм 1    25      2273    247768  24818218       2477245543             247417481162
Алгоритм 2    17       363      5416     68845           703121                  7973274
Алгоритм 3    68      1344     19952    267232          3337856                 39902848
-------------------------------------------------------------------------------------------------------------------
Мин.знач      17       363      5416     68845           703121                  7973274
Макс.знач     68      2273    247768  24818218       2477245543             247417481162
Ср.знач       36      1326     91045   8384765        827095506               82488452428
```

Рисунок 1 – скриншот работы программы

Алгоритм 1 - сортировка перемешиванием

Алгоритм 2 - сортировка Шелла

Алгоритм 3 - сортировка слиянием

	10	100	1000	10000	100000	1000000
сортировка перемешиванием	0,00053	0,0000692	0,0083616	0,8373259	51,649001	5251,258819
сортировка Шелла	0,0004083	0,0004675	0,004309	0,5318015	45,6592432	4520,543452
сортировка слиянием	0,001078	0,0000439	0,0003463	0,0038426	0,0493225	0,4765772

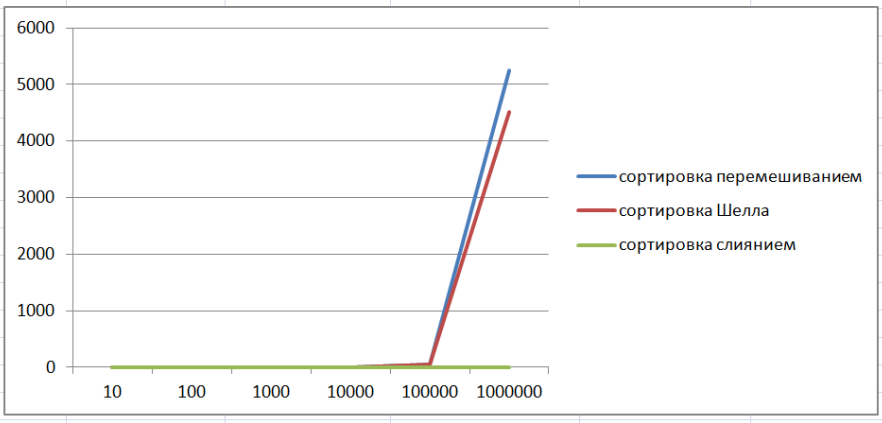

Рисунок 2 – графики зависимости времени сортировки элементов от размера коллекции

	10	100	1000	10000	100000	1000000
сортировка перемешиванием	45	4950	499500	49995000	4999950000	5E+11
сортировка Шелла	70	7919	798131	98176009	8331077776	8,04478E+11
сортировка слиянием	41	737	10693	140213	1732347	20619289

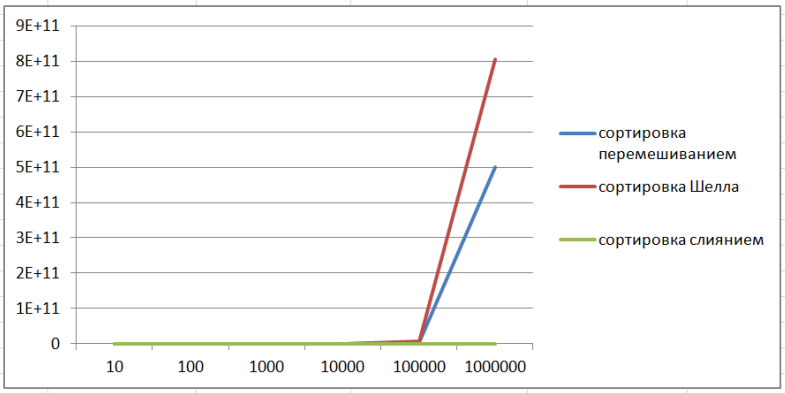

Рисунок 3 – графики зависимости числа перестановок элементов от размера коллекции

	10	100	1000	10000	100000	1000000
сортировка перемешиванием	25	2273	247768	24818218	2477245543	2,47417E+11
сортировка Шелла	17	363	5416	68845	703121	7973274
сортировка слиянием	68	1344	19952	267232	3337856	39902848

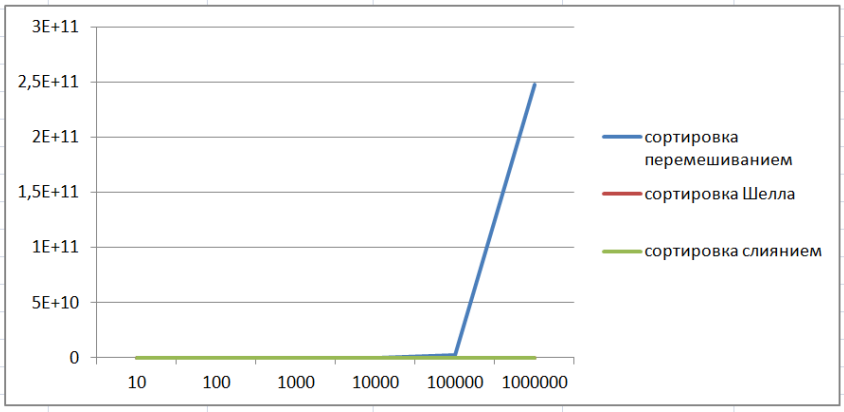

Рисунок 3 – графики зависимости числа сравнений элементов от размера коллекции

Вывод по таблицам и графикам: по времени сортировка перемешиванием самая медленная, а сортировка слиянием самая быстрая. Больше всего перестановок элементов было при сортировки Шелла, а меньше всего при сортировки слиянием. Больше всего сравнений элементов было при сортировки перемешиванием, а меньше всего при сортировки Шелла.

Вывод: в ходе данной лабораторной работы были изучены некоторые алгоритмы сортировки коллекций.