Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Чувашский государственный университет им. И.Н.Ульянова».
Кафедра вычислительной техники.
Предмет: Объектно-ориентированное программирование

Лабораторная работа №5.
Исследование методов сортировки.

Вариант 4.

Выполнил: Васильев Егор Юрьевич
студент группы ИВТ-41-20
Проверил: Павлов.Л.А

Чебоксары, 2022.

Цель работы: ознакомление с методами сортировки, получение практических навыков программирования задач сортировки, получение навыков экспериментальных исследований алгоритмов.

3. Сортировка вставками в связанный список.

```
Ссылок: 0
public void InsertionSort()
{
    Node current = _head;

    while (current != null)
    {
        Node Next = current.Next;
        SortedInsert(current);
        current = Next;
    }

    _head = _sorted;
}
```

4. Сортировка Шелла.

```
Ссылок: 0
public static void ShellSort(this Array array)
{
    var step = array.Length / 2;

    while (step > 0)
    {
        for (int i = step; i < array.Length; i++)
        {
            var value = array.GetValue(i);
            int j;

            for (j = i - step; (j >= 0) && (Compare(array.GetValue(j), value) > 0); j -= step)
            {
                array.SetValue(array.GetValue(j), j + step);
            }

            array.SetValue(value, j + step);
        }

        step /= 2;
    }
}
```

9. Цифровая обменная сортировка.

```
Ссылок: 0
public static void RadixSort(this int[] array)
{
    var arraySize = array.Length;
    var tempArray = new int[arraySize];

    var numOfBits = 4;
    var numOfBitsOfInt = 32;

    var countArr = new int[1 << numOfBits];
    var prefix = new int[1 << numOfBits];

    var numOfGroups = (int)Math.Ceiling((double)numOfBitsOfInt / (double)numOfBits);
    var mask = (1 << numOfBits) - 1;

    for (int c = 0, shift = 0; c < numOfGroups; c++, shift += numOfBits)
    {
        for (int j = 0; j < countArr.Length; j++)
        {
            countArr[j] = 0;
        }

        for (int i = 0; i < arraySize; i++)
        {
            countArr[(array[i] >> shift) & mask]++;
        }

        prefix[0] = 0;
        for (int i = 1; i < countArr.Length; i++)
        {
            prefix[i] = prefix[i - 1] + countArr[i - 1];
        }

        for (int i = 0; i < arraySize; i++)
        {
            tempArray[prefix[(array[i] >> shift) & mask]++] = array[i];
        }

        tempArray.CopyTo(array, 0);
    }
}
```

7. Быстрая сортировка (рекурсивный вариант).

```
Ссылок: 2
public static void QuickSort(this Array array, int startIndex = 0, int endIndex = -1)
{
    if (endIndex < 0)
    {
        endIndex = array.Length - 1;
    }

    if (startIndex >= endIndex)
    {
        return;
    }

    var middleIndex = (endIndex - startIndex) / 2 + startIndex;
    var currentIndex = startIndex;

    array.Swap(startIndex, middleIndex);

    for (int i = startIndex + 1; i <= endIndex; ++i)
    {
        if (Compare(array.GetValue(i), array.GetValue(startIndex)) <= 0)
        {
            array.Swap(++currentIndex, i);
        }
    }

    array.Swap(startIndex, currentIndex);

    QuickSort(array, startIndex, currentIndex - 1);
    QuickSort(array, currentIndex + 1, endIndex);
}
```

11. Пирамидальная сортировка.

```
Ссылок: 0
public static void HeapSort(this Array array)
{
    for (int i = array.Length / 2 - 1; i >= 0; i--)
    {
        array.Heapify(array.Length, i);
    }
    for (int i = array.Length - 1; i >= 0; i--)
    {
        array.Swap(0, i);

        array.Heapify(i, 0);
    }
}
```

12. Сортировка подсчетом (перечислением).

```
public static void CountSort(this int[] array)
{
    var max = array.Max();
    var min = array.Min();

    int[] countArr = new int[max - min + 1];

    for (int i = 0; i < countArr.Length; i++)
    {
        countArr[i] = 0;
    }

    for (int i = 0; i < array.Length; i++)
    {
        countArr[array[i] - min]++;
    }

    var index = 0;
    for (int i = min; i <= max; i++)
    {
        while (countArr[i - min]-- > 0)
        {
            array[index] = i;
            array.SetValue(i, index);
            index++;
        }
    }
}
```

15. Сортировка простым двухпутевым слиянием.

```
Ссылок: 3
public static void MergeSort(this Array array, int left = 0, int right = -1)
{
    if (right < 0)
    {
        right = array.Length - 1;
    }

    if (right > left)
    {
        int mid = (right + left) / 2;
        array. MergeSort(left, mid);
        array.MergeSort(mid + 1, right);

        array.Merge(left, mid + 1, right);
    }
}
```

17. Сортировка слиянием списков для простого двухпутевого слияния.

```
Ссылок: 3
public Node MergeSort(Node node)
{
    if (node == null || node.Next == null)
    {
        return node;
    }

    Node second = Split(node);
    node = MergeSort(node);
    second = MergeSort(second);

    return Merge(node, second);
}
```

Результаты работы программы.

Слева – количество сравнений, справа – кол-во пересылок.

Метод сортировки	1000		2000	
Сортировка вставками в связанный список	239158	239413	998169	998628
Сортировка Шелла	15392	15392	37175	37175
Быстрая сортировка (рекурсия)	6345	5665	16141	14807
Быстрая сортировка (без рекурсии)	11100	2087	26099	4543
Цифровая обменная сортировка	8120	8000	16120	16000
Пирамидальаня сортировка	19808	9094	44476	20152
Сортировка подсчетом	1000	1000	2000	2000
Сортировка простым двухпутевым слиянием	9976	9976	21952	21952
Сортировка слиянием списков для простого двухпутегого слияния	8674	999	19425	1999

3000		4000		5000	
2276308	2276716	3999431	4000094	6314182	6314896
60783	60783	86031	86031	115188	115188
24444	22421	36374	33656	42808	39441
44178	7262	60366	9916	74446	12938
24120	24000	32120	32000	40120	40000
71246	32104	99171	44358	128014	57154
3000	3000	4000	4000	4997	5000
34904	34904	47904	47904	61808	61808
30856	2999	42818	3999	55215	4999

Вывод: ознакомился с методами сортировки, получил практические навыки программирования задач сортировки, получил навыки экспериментальных исследований алгоритмов.