Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Ордена Трудового Красного Знамени федеральное государственное
бюджетное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и
информационных технологий

Лабораторная работа №1
на тему: «Методы сортировки»

Выполнила:
Студентка группы БФИ1902
Струкова А.В.
Вариант 18

Проверил:

Москва, 2021 г.

Оглавление

1. Цель работы 3
2. Задание на лабораторную работу 3
3. Ход лабораторной работы 3
3.1 Задание 1 3
3.2 Задание 2 3
3.3 Задание 3 4
3.3.1 Сортировка выбором 4
3.3.2 Сортировка вставкой 4
3.3.3 Сортировка обменом 5
3.3.4 Сортировка Шелла 5
3.3.5 Быстрая сортировка 6
3.3.6 Пирамидальная сортировка 6
4. Листинг программы 7
5. Результат работы программы 9
Список использованных источников 11

1. Цель работы

Цель работы: рассмотреть и изучить основные методы сортировки.

2. Задание на лабораторную работу

1) Вывести на экран приветствие: «Hello, World!»
2) Написать генератор случайных матриц (многомерных), который принимает опциональные параметры m, n, min_limit, max_limit, где m и n указывают размер матрицы, а min_limit и max_limit – минимальное и максимальное значение для генерируемого числа. По умолчанию при отсутствии параметров принимать следующие значения:

   m = 50;

   n = 50;

   min_limit = -250;

   max_limit = 1000 + (номер своего варианта)
3) Реализовать методы сортировки строк в соответствии с заданием. Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки.

   Необходимо использовать следующие методы: выбором, вставкой, обменом, Шелла, турнирная, быстрая, пирамидальная
4) Создать публичный репозиторий на github и запушить выполненное задание.

3. Ход лабораторной работы

3.1 Задание 1

Чтобы вывести на экран приветствие: «Hello, World!» используем следующую строку:

```
System.out.println("Hello, World!");
```

3.2 Задание 2

Для того, чтобы написать генератор случайных матриц воспользуемся двойным циклом for и просчитаем каждый элемент матрицы в соответствии с формулой:

```
Mas[i][j] = (int) (Math.random() * (max - min + 1) + min);
```

Затем пройдемся еще раз по двойному циклу и заполним получившимися элементами матрицу.

3.3 Задание 3

Поэтапно реализуем все необходимые методы сортировки.

3.3.1 Сортировка выбором

Сортировка выбором разделяет массив на сортированный и несортированный подмассивы. Сортированный подмассив формируется вставкой минимального элемента не отсортированного подмассива в конец сортированного, заменой.

Для реализации мы в каждой итерации предполагаем, что первый неотсортированный элемент минимален и итерируем по всем оставшимся элементам в поисках меньшего. После нахождения текущего минимума неотсортированной части массива меняем его местами с первым элементом, и он уже становится частью отсортированного массива.

Временная сложность:

При поиске минимума для длины массива проверяются все элементы, поэтому сложность равна O(n). Поиск минимума для каждого элемента массива равен O(n^2).

3.3.2 Сортировка вставкой

Этот алгоритм разделяет оригинальный массив на сортированный и несортированный подмассивы. Длина сортированной части равна 1 в начале и соответствует первому (левому) элементу в массиве. После этого остается итерировать массив и расширять отсортированную часть массива одним элементом с каждой новой итерацией.

После расширения новый элемент помещается на свое место в отсортированном подмассиве. Это происходит путём сдвига всех элементов вправо, пока не встретится элемент, который не нужно двигать.

Временная сложность:

В случае убывания массива каждая итерация сдвигает отсортированный массив на единицу O(n). Придется делать это для каждого элемента в каждом массиве, что приведет к сложности равной O(n ^ 2).

3.3.3 Сортировка обменом

Метод сортировки, который многие обычно осваивают раньше других из-за его исключительной простоты, называется пузырьковой сортировкой (bubble sort), в рамках которой выполняются следующие действия: проход по файлу с обменом местами соседних элементов, нарушающих заданный порядок, до тех пор, пока файл не будет окончательно отсортирован. Основное достоинство пузырьковой сортировки заключается в том, что его легко реализовать в виде программы. Для понимания и реализации этот алгоритм — простейший, но эффективен он лишь для небольших массивов. Сложность алгоритма: O(n^2). Суть алгоритма пузырьковой сортировки состоит в сравнении соседних элементов и их обмене, если они находятся не в надлежащем порядке. Неоднократно выполняя это действие, мы заставляем наибольший элемент "всплывать" к концу массива. Следующий проход приведет к всплыванию второго наибольшего элемента, и так до тех пор, пока после n-1 итерации массив не будет полностью отсортирован.

3.3.4 Сортировка Шелла

Идея метода заключается в сравнение разделенных на группы элементов последовательности, находящихся друг от друга на некотором расстоянии. Изначально это расстояние равно d или N/2, где N — общее число элементов. На первом шаге каждая группа включает в себя два элемента расположенных

друг от друга на расстоянии N/2; они сравниваются между собой, и, в случае необходимости, меняются местами. На последующих шагах также происходят проверка и обмен, но расстояние d сокращается на d/2, и количество групп, соответственно, уменьшается. Постепенно расстояние между элементами уменьшается, и на d=1 проход по массиву происходит в последний раз.

3.3.5 Быстрая сортировка

Выбираем один элемент массива в качестве стержня и сортируем остальные элементы вокруг (меньшие элементы налево, большие направо). Так соблюдается правильная позиция самого «стержня».

Затем рекурсивно повторяем сортировку для правой и левой частей.

3.3.6 Пирамидальная сортировка

Общая идея пирамидальной сортировки заключается в том, что сначала строится пирамида из элементов исходного массива, а затем осуществляется сортировка элементов.

Выполнение алгоритма разбивается на два этапа.

1 этап

Построение пирамиды. Определяем правую часть дерева, начиная с n/2-1 (нижний уровень дерева). Берем элемент левее этой части массива и просеиваем его сквозь пирамиду по пути, где находятся меньшие его элементы, которые одновременно поднимаются вверх; из двух возможных путей выбираете путь через меньший элемент.

2 этап

Сортировка на построенной пирамиде. Берем последний элемент массива в качестве текущего. Меняем верхний (наименьший) элемент массива и текущий местами. Текущий элемент (он теперь верхний) просеиваем сквозь n-1 элементную пирамиду. Затем берем предпоследний элемент и т.д.

В худшем случае требуется $n \cdot \log_2 n$ шагов, сдвигающих элементы.

4. Листинг программы

```
<!DOCTYPE HTML>
<html>

<body>

<p>Перед скриптом...</p>
<script>

alert ("Hello, world");



function selectionSort(A1)       // A - массив, который нужно
{                                // отсортировать по возрастанию.
    var n = A1.length;
    for (var i = 0; i < n-1; i++)
     { var min = i;
       for (var j = i+1; j < n; j++)
        { if (A1[j] < A1[min]) min = j; }
       var t = A1[min]; A1[min] = A1[ i ]; A1[ i ] = t;
     }
    return A1;    // На выходе сортированный по возрастанию массив A.
}
let A1 = [5, 2, 4, 6, 1, 3];
selectionSort(A1);
alert (A1);

function Vstavka(A2)       // A - массив, который нужно
{                                // отсортировать по возрастанию.
    var n = A2.length;
    for (var i = 0; i < n; i++)
     { var v = A2[ i ], j = i-1; //v-след. эл-т
       while (j >= 0 && A2[j] > v) //если тек. эл-т больше след. меняем,
                                   //то местами
        { A2[j+1] = A2[j]; j--; }
       A2[j+1] = v;
     }
    return A2;    // На выходе сортированный по возрастанию массив A.
}
let A2 = [1, 8, 5, 4];
Vstavka(A2);
alert (A2);

function BubbleSort(A3)       // A - массив, который нужно
{                             // отсортировать по возрастанию.
    var n = A3.length;
    for (var i = 0; i < n-1; i++)
     { for (var j = 0; j < n-1-i; j++)
        { if (A3[j+1] < A3[j])
```

```
           { var t = A3[j+1]; A3[j+1] = A3[j]; A3[j] = t; }
        }
     }
    return A3;    // На выходе сортированный по возрастанию массив A.
}
let A3 = [1, 2, 5, 4, 3] ;
BubbleSort(A3);
alert (A3);


function Shell(A4)
{
   {
    const l = A4.length;
    let gap = Math.floor(l / 2);
    while (gap >= 1) {
        for (let i = gap; i < l; i++) {
            const current = A4[i]; //тек эл-т
            let j = i; //индекс эл-та
            while (j > 0 && A4[j - gap] > current) {
                A4[j] = A4[j - gap]; //
                j -= gap; //уменьшаем на шаг
            }
            A4[j] = current;
        }
        gap = Math.floor(gap / 2);
    }
    return A4;
};
}
let A4 = [10, 9, 8, 6, 7] ;
Shell(A4);
alert (A4);

function Piramida(A5)  //** ПОСТОЯННО УМЕНЬШАЕМ
{
    if (A5.length == 0) return [];
    var n = A5.length, i = Math.floor(n/2), j, k, t;
    while (true)
    { if (i > 0) t = A5[--i];
      else { n--;
             if (n == 0) return A5;
             t = A5[n];  A5[n] = A5[0];
           }
      j = i;  k = j*2+1;
      while (k < n)
       { if (k+1 < n && A5[k+1] > A5[k]) k++;
         if (A5[k] > t)
          { A5[j] = A5[k];  j = k;  k = j*2+1; }
         else break;
```

```
       }
      A5[j] = t;
    }
}
let A5 = [10, 9, 8, 6, 5] ;
Piramida(A5);
alert (A5);

function quicksort(array) {

if (array.length < 1) {
    return array;    // базовый случай
} else {
     let pivotIndex = Math.floor(array.length / 2);
let pivot = array[pivotIndex];
let less = [];
let greater = [];
  for ( i =0; i < array.length; i++) {
    if ( i ==  pivotIndex) continue;
    if (array[i] <= pivot) {
      less.push(array[i]);
    } else {
      greater.push(array[i]);
    }

  }
     let result = [];
return result.concat(quicksort(less), pivot, quicksort(greater));
}
};

let array = [1, 7, 2, 3, 4] ;
//quicksort(A6);
alert (quicksort(array));

</script>
<p>...После скрипта.</p>
</body>

</html>
```

5. Результат работы программы

Рисунок 1 - Hello, world

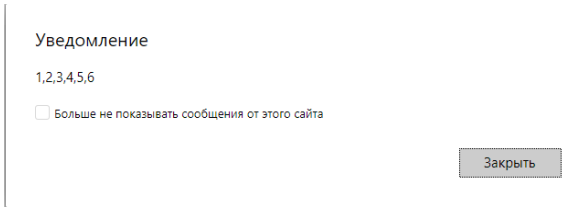

Рисунок 2 - Сортировка выбором

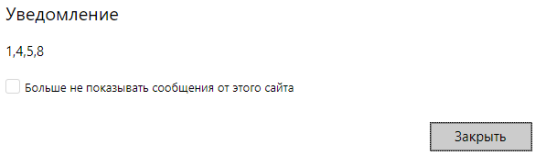

Рисунок 3 - Сортировка вставками

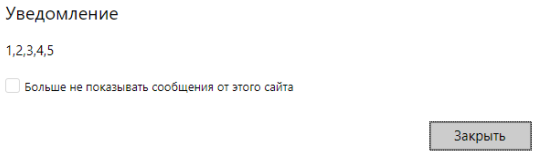

Рисунок 4 - Сортировка обменом

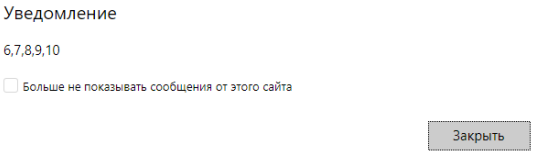

Рисунок 5 - Сортировка Шелла

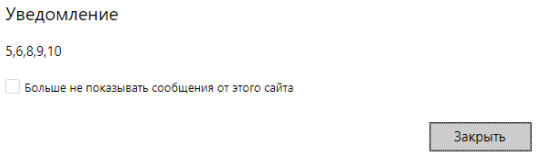

Рисунок 6 - Сортировка пирамидальная

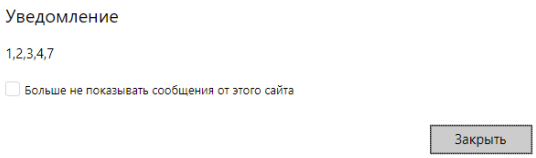

Рисунок 7 - Сортировка быстрая

Вывод: Мы рассмотрели и изучили основные методы сортировки, такие как сортировка выбором, вставкой, обменом, Шелла, быстрая сортировка и пирамидальная.

Список использованных источников

1) ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчёт о научно-исследовательской работе. Структура и правила оформления

2) ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления