Министерство цифрового развития, связи и массовых коммуникаций Государственное образовательного учреждение высшего образования

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Лабораторная работа № 1

по дисциплине «Структура и алгоритмы обработки данных»

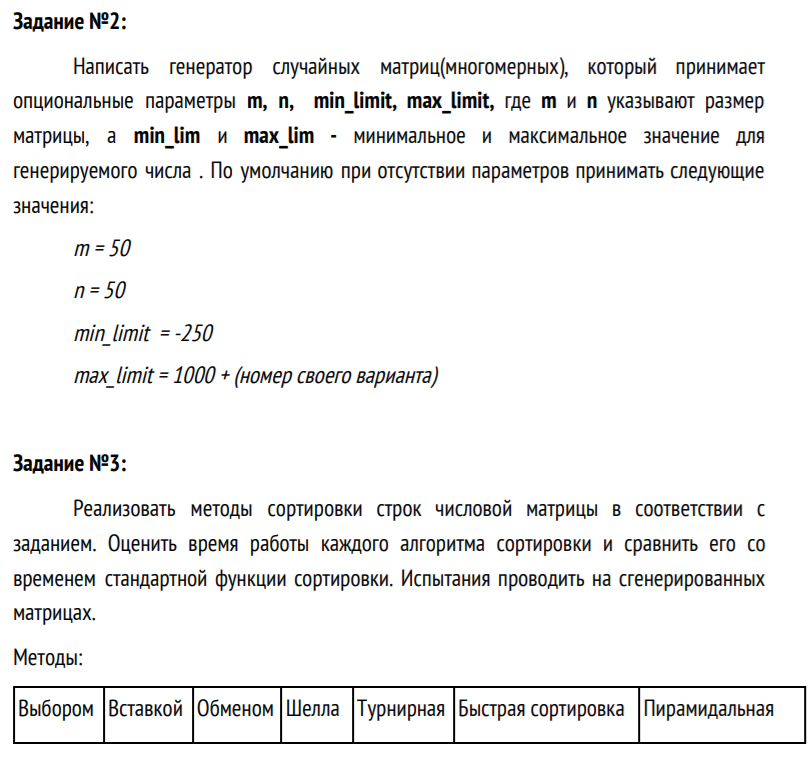
«Методы сортировки»

Выполнил студент группы БФИ-1901:

Бардюк Д. В.

Москва 2021

# Задание



# Код программы

module.exports = {

    generateMatrix(

      m = 50,

      n = 50,

      minLimit = -250,

      maxLimit = 1010

    ) {

      const matrix = new Array(m);

      for (let i = 0; i < matrix.length; i++) {

        matrix[i] = [];

        for (let j = 0; j < n; j++) {

          matrix[i][j] = minLimit + Math.floor(Math.random() \* (maxLimit - minLimit + 1));

        }

      }

      return matrix;

    },

    flattenMatrix(matrix) {

      return matrix.reduce((flatArray, row) => [...flatArray, ...row], []);

    },

    getMatrixFromArray(array, m, n) {

      const matrix = new Array(m);

      for (let i = 0; i < matrix.length; i++) {

        matrix[i] = array.slice(i \* m, (i + 1) \* n);

      }

      return matrix;

    },

    swap(array, first, second) {

      [array[first], array[second]] = [array[second], array[first]];

    }

  }

const Utils = require("./Utils");

module.exports = {

  nativeSort(matrix) {

    const array = Utils.flattenMatrix(matrix);

    array.sort((first, second) => second < first ? 1 : -1);

    return Utils.getMatrixFromArray(

      array,

      matrix.length,

      matrix[0].length

    );

  },

  selectionSort(matrix) {

    const array = Utils.flattenMatrix(matrix);

    for (let i = 0; i < array.length; i++) {

      let min = i;

      for (let j = i + 1; j < array.length; j++) {

        if (array[j] < array[min]) {

          min = j;

        }

      }

      if (min !== i) {

        Utils.swap(array, i, min);

      }

    }

    return Utils.getMatrixFromArray(

      array,

      matrix.length,

      matrix[0].length

    );

  },

  insertionSort(matrix) {

    const array = Utils.flattenMatrix(matrix);

    for (let i = 1; i < array.length; i++) {

      const key = array[i];

      let j = i - 1;

      while (j >= 0 && array[j] > key) {

        array[j + 1] = array[j];

        j--;

      }

      array[j + 1] = key;

    }

    return Utils.getMatrixFromArray(

      array,

      matrix.length,

      matrix[0].length

    );

  },

  bubbleSort(matrix) {

    const array = Utils.flattenMatrix(matrix);

    for (let i = 0; i < array.length; i++) {

      for (let j = 0; j < array.length; j++) {

        if (array[j] > array[j + 1]) {

          Utils.swap(array, j, j + 1);

        }

      }

    }

    return Utils.getMatrixFromArray(

      array,

      matrix.length,

      matrix[0].length

    );

  },

  shellSort(matrix) {

    const array = Utils.flattenMatrix(matrix);

    let gap = Math.floor(array.length / 2);

    while (gap > 0) {

      for (let i = gap; i < array.length; i++) {

        const key = array[i];

        let j = i;

        while (j >= gap && array[j - gap] > key) {

          array[j] = array[j - gap];

          j -= gap;

        }

        array[j] = key;

      }

      gap = Math.floor(gap / 2);

    }

    return Utils.getMatrixFromArray(

      array,

      matrix.length,

      matrix[0].length

    );

  },

  quickSort(matrix) {

    const array = Utils.flattenMatrix(matrix);

    function \_sort(array) {

      if (array.length < 2) {return array; }

      const pivot = array[0];

      const left = [];

      const right = [];

      for (let i = 1; i < array.length; i++) {

        if (pivot > array[i]) {

          left.push(array[i]);

        } else {

          right.push(array[i]);

        }

      }

      return [...\_sort(left), pivot, ...\_sort(right)];

    }

    return Utils.getMatrixFromArray(

      \_sort(array),

      matrix.length,

      matrix[0].length

    );

  },

  heapSort(matrix) {

    const array = Utils.flattenMatrix(matrix);

    const length = array.length;

    function \_heapify(array, length, i) {

      let largest = i;

      let left = i \* 2 + 1;

      let right = left + 1;

      if (left < length && array[left] > array[largest]) {

        largest = left;

      }

      if (right < length && array[right] > array[largest]) {

        largest = right;

      }

      if (largest !== i) {

        Utils.swap(array, i, largest);

        \_heapify(array, length, largest);

      }

    }

    for (let i = Math.floor(length / 2 - 1); i >= 0; i--) {

      \_heapify(array, length, i);

    }

    for (let k = length - 1; k >= 0; k--) {

      Utils.swap(array, 0, k);

      \_heapify(array, k, 0);

    }

    return Utils.getMatrixFromArray(

      array,

      matrix.length,

      matrix[0].length

    );

  }

}

const Utils = require("./Utils");

const Sorts = require("./Sorts");

function compareWithNativeSort(matrix, sort) {

  console.log("Initial matrix: ", matrix);

  const start = Date.now();

  const sortedMatrix = sort(matrix);

  const end = Date.now();

  const startNative = Date.now();

  Sorts.nativeSort(matrix);

  const endNative = Date.now();

  console.log("Sorted matrix: ", sortedMatrix);

  console.log(`User's sort: ${end - start} ms`);

  console.log(`Native sort: ${endNative - startNative} ms`);

}

const matrix = Utils.generateMatrix();

    console.log("\n<--- Selection sort --->");

    compareWithNativeSort(matrix, Sorts.selectionSort);

    console.log("\n<--- Insertion sort --->");

    compareWithNativeSort(matrix, Sorts.insertionSort);

    console.log("\n<--- Bubble sort --->");

    compareWithNativeSort(matrix, Sorts.bubbleSort);

    console.log("\n<--- Shell sort --->");

    compareWithNativeSort(matrix, Sorts.shellSort);

    console.log("\n<--- Quick sort --->");

    compareWithNativeSort(matrix, Sorts.quickSort);

    console.log("\n<--- Heap sort --->");

    compareWithNativeSort(matrix, Sorts.heapSort);

# Результат работы

На рисунке 1 представлен результат работы программы

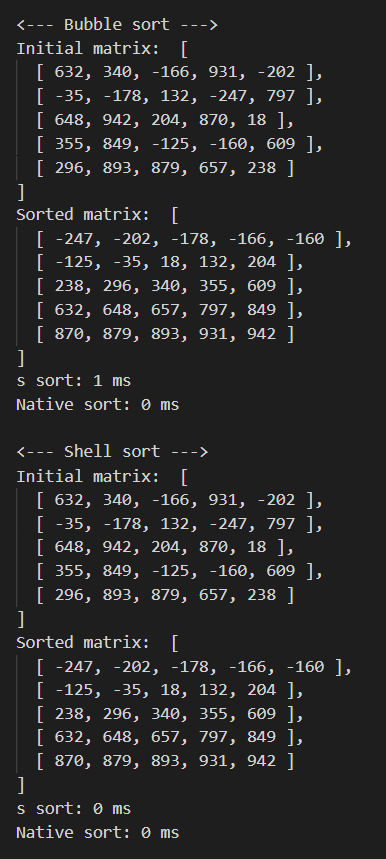
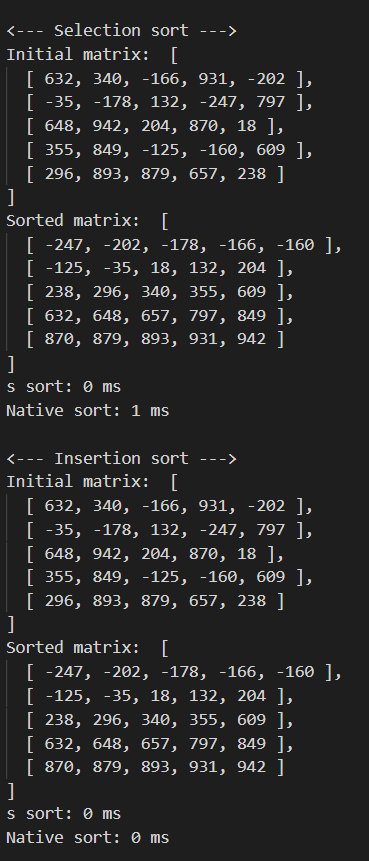


Рисунок – 1 Отсортированные массивы

**Вывод:** в ходе выполнения данной работы я узнал о методах сортировки, их плюсах и минусах, сложностях алгоритмов, написал каждый из этих алгоритмов на языке программирования.