Отчет по лабораторной работе

по дисциплине «Структуры и алгоритмы обработки данных»

на тему:

«Методы сортировки»

Выполнил:

Студент группы БСТ1902

Игнатов В.С.

Вариант №5

Москва 2020

Оглавление

Задания на лабораторную работу

1. Вывести в консоль «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_lim и max_lim - минимальное и максимальное значение для генерируемого числа. По умолчанию при отсутствии параметров принимать следующие значения:

3. Реализовать методы сортировки строк числовой матрицы в соответствии с заданием. Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки. Испытания проводить на сгенерированных матрицах.

4. Создать публичный репозиторий на GitHub.

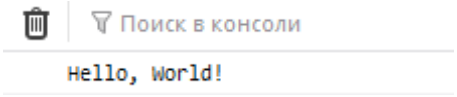
Ход работы

Задание 1

Выведем в консоль сообщение:

```
console.log("Hello, World!");
```

Результат команды:



Задание 2

Реализуем функцию генерации матрицы:

```
function randomMatrix(m = 50, n = 50, minLim = -250, maxLim = 1005) {
    let matrix = [];
    for (let i = 1; i <= m; i++) {
        let matrixInner = [];

        for (let j = 1; j <= n; j++) {
            let elem = Math.floor(minLim + Math.random() * (maxLim + 1 - minLim))
;
            matrixInner.push(elem);
        }
```

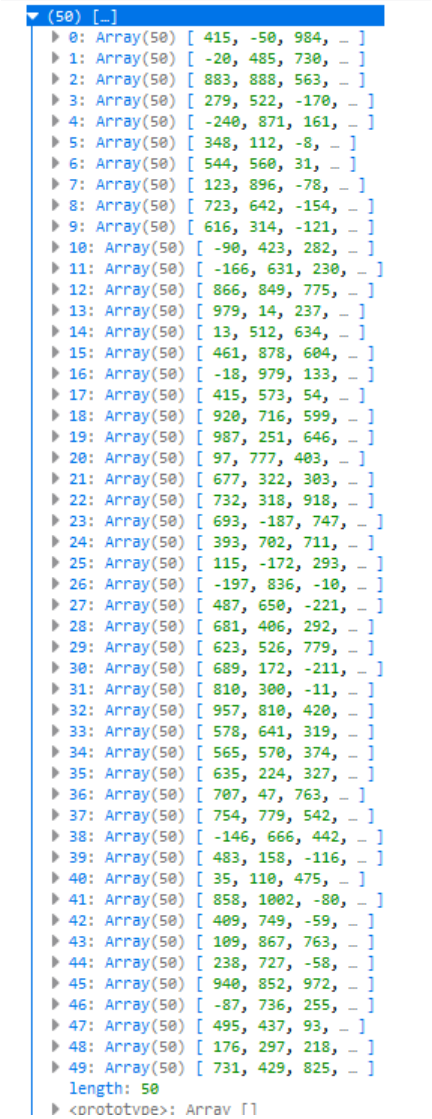
```
        matrix.push(matrixInner);
    }

    return matrix
}
```

Вызов функции матрицы:

```
let matrix = randomMatrix();
console.log(matrix);
```

Результат вывода:



Задание 3

Сортировка выбором

```
function selectSort(arr) {
    for (let i = 0; i < arr.length - 1; i++) {
        let MinElemIndex = i;

        for (let j = i + 1; j < arr.length; j++) {
            if (arr[MinElemIndex] > arr[j]) {
                MinElemIndex = j;
            }
        }

        if (MinElemIndex !== i) {
            [arr[i], arr[MinElemIndex]] = [arr[MinElemIndex], arr[i]];
        }
    }
    return arr;
};
```

Сортировка вставками

```
function insertSort(arr) {
    for (let i = 1; i < arr.length; i++) {
        const current = arr[i];
        let j = i;

        while (j > 0 && arr[j - 1] > current) {
            arr[j] = arr[j - 1];
            j--;
        }

        arr[j] = current;
    }

    return arr;
};
```

Сортировка обменом

```
function swapSort(arr) {
    for (let i = 0; i < arr.length - 1; i++) {
        for (let j = 0; j < arr.length - 1 - i; j++) {
            if (arr[j] > arr[j + 1]) {
                let swap = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = swap;
```

```
            }
        }
    }

    return arr;
}
```

Сортировка Шелла

```
function ShellSort(arr) {
    let gap = Math.floor(arr.length / 2);

    while (gap >= 1) {
        for (let i = gap; i < arr.length; i++) {
            const current = arr[i];
            let j = i;

            while (j > 0 && arr[j - gap] > current) {
                arr[j] = arr[j - gap];
                j -= gap;
            }

            arr[j] = current;
        }

        gap = Math.floor(gap / 2);
    }

    return arr;
};
```

Быстрая сортировка

```
function quickSort(arr) {
    if (arr.length < 2) return arr;

    let less = [],
        more = [],
        pivot = arr[0];

    for (let i = 1; i < arr.length; i++) {
        if (arr[i] < pivot) {
            less.push(arr[i])
        } else {
            more.push(arr[i]);
        };
    }
```

```
    return quickSort(less).concat(pivot, quickSort(more));
}
```

Пирамидальная сортировка

```
function HeapSort(arr) {
    let n = arr.length,
        i = Math.floor(n / 2),
        j, k, t;

    while (true) {
        if (i > 0) t = arr[--i];
        else {
            n--;

            if (n == 0) return arr;
            t = arr[n];
            arr[n] = arr[0];
        }

        j = i;
        k = j * 2 + 1;

        while (k < n) {
            if (k + 1 < n && arr[k + 1] > arr[k]) k++;
            if (arr[k] > t) {
                arr[j] = arr[k];
                j = k;
                k = j * 2 + 1;
            }
            else break;
        }

        arr[j] = t;
    }
}
```

Встроенная сортировка

```
function includeSort(arr) {
   return arr.sort((a, b) => a - b);
}
```

Турнирная сортировка

```
function TournamentSort(arr) {
    let three = Array(2 * (arr.length + arr.length % 2));
    let index = three.length - arr.length + arr.length % 2;

    for (let i = index; i < three.length; i++) {
        tree[i] = i - index, array[i - index];
    }

    for (let j = 0; j < arr.length; j++) {
        let n = arr.length;
        index = three.length - arr.length  + arr.length % 2;

        while (index > -1) {
            n = (n + 1) / 2;

            for (let i = 0; i < n; i++) {
                let iCopy = Math.max(index + i * 2, 1);

                if (three[iCopy] != null && three[iCopy + 1] != null) {
                    if (three[iCopy][1] < three[iCopy + 1][1]) {
                        three[iCopy / 2] = three[iCopy];
                    } else {
                        three[iCopy / 2] = three[iCopy + 1]
                    }
                } else {
                    three[iCopy / 2] = (three[iCopy] != null) ? three[iCopy] : th
ree[iCopy + 1];
                }
            }

            index -= n;
        }

        index = three[0][0]
        let x = three[0][1]
        arr[j] = x;
        three[three.length - arr.length - arr.length % 2 + index] = null;
    }
}
```

Результат выполнения сортировок

```
Сортировка выбором: 4мс - таймер закончился
Сортировка вставками: 2мс - таймер закончился
Сортировка обменом: 1мс - таймер закончился
Сортировка Шелла: 2мс - таймер закончился
Быстрая сортировка: 3мс - таймер закончился
Пирамидальная сортировка: 1мс - таймер закончился
Турнирная сортировка: 2мс - таймер закончился
Встроенная сортировка: 0мс - таймер закончился
```

Вывод

Я изучил методы сортировки и реализовал их на практике.