Министерство цифрового развития, связи и массовых коммуникаций

Ордена Трудового Красного Знамени федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Отчёт по лабораторной работе № 1

«Методы сортировки»

по дисциплине «Структуры и алгоритмы обработки данных»

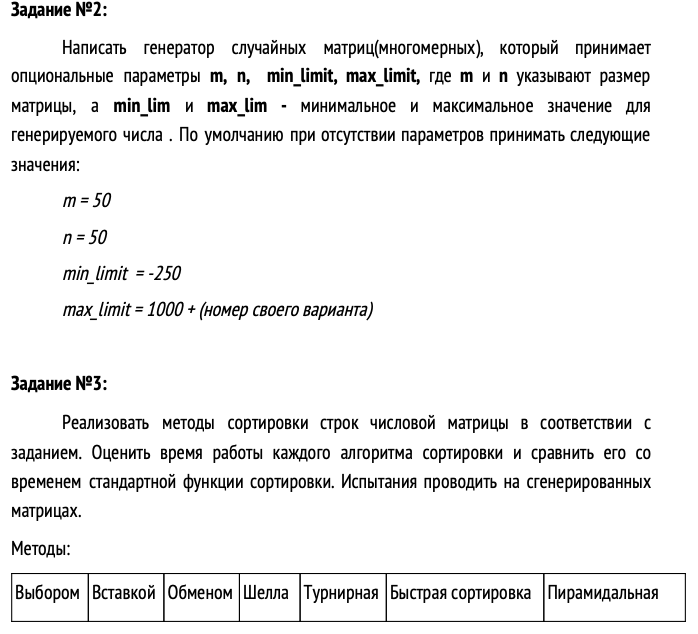
Выполнил: студент группы БВТ1902

Холмовский Олег Васильевич

Москва

2021

Задание



Код программы

const createMatrix = (m = 50, n = 50, min\_limit = -250, max\_limit = 1000) => {

let matrix = [];

*for* (let i = 0; i < m; i++) {

matrix[i] = [];

*for* (let j = 0; j < n; j++) {

matrix[i][j] =

Math.floor(Math.random() \* (max\_limit - min\_limit + 1)) + min\_limit;

}

}

*return* matrix;

};

const selectionSort = (matrix) => {

console.log('old matrix selectionSort:', matrix);

let newMatrix = [];

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

*for* (let i = 0; i < matrix[z].length - 1; i++) {

let indexMin = i;

*for* (let j = i + 1; j < matrix[z].length; j++) {

*if* (matrix[z][indexMin] > matrix[z][j]) {

indexMin = j;

}

}

*if* (indexMin !== i) {

[matrix[z][i], matrix[z][indexMin]] = [

matrix[z][indexMin],

matrix[z][i],

];

}

}

newMatrix[z] = matrix[z];

}

*return* newMatrix;

};

const insertionSort = (matrix) => {

console.log('old matrix insertionSort:', matrix);

let newMatrix = [];

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

*for* (let i = 1; i < matrix[z].length; i++) {

const current = matrix[z][i];

let j = i;

*while* (j > 0 && matrix[z][j - 1] > current) {

matrix[z][j] = matrix[z][j - 1];

j--;

}

matrix[z][j] = current;

}

newMatrix[z] = matrix[z];

}

*return* newMatrix;

};

const bubbleSort = (matrix) => {

console.log('old matrix bubbleSort:', matrix);

let newMatrix = [];

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

let swapped;

*do* {

swapped = false;

*for* (let i = 0; i < matrix[z].length; i++) {

*if* (matrix[z][i] > matrix[z][i + 1]) {

let tmp = matrix[z][i];

matrix[z][i] = matrix[z][i + 1];

matrix[z][i + 1] = tmp;

swapped = true;

}

}

} *while* (swapped);

newMatrix[z] = matrix[z];

}

*return* newMatrix;

};

const shellSort = (matrix) => {

console.log('old matrix shellSort:', matrix);

let newMatrix = [];

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

*for* (

let gap = Math.floor(matrix[z].length / 2);

gap > 0;

gap = Math.floor(gap / 2)

) {

*for* (let i = gap; i < matrix[z].length; i += 1) {

let temp = matrix[z][i];

let j;

*for* (j = i; j >= gap && matrix[z][j - gap] > temp; j -= gap) {

matrix[z][j] = matrix[z][j - gap];

}

matrix[z][j] = temp;

}

}

newMatrix[z] = matrix[z];

}

*return* newMatrix;

};

Array.prototype.swap = *function* (a, b) {

let tmp = *this*[a];

*this*[a] = *this*[b];

*this*[b] = tmp;

};

const sink = (array, i, max) => {

let big\_index, childl, childr;

*while* (i < max) {

big\_index = i;

childl = 2 \* i + 1;

childr = childl + 1;

*if* (childl < max && array[childl] > array[big\_index]) big\_index = childl;

*if* (childr < max && array[childr] > array[big\_index]) big\_index = childr;

*if* (big\_index == i) *return*;

array.swap(i, big\_index);

i = big\_index;

}

};

const build\_heap = (array) => {

let index = Math.floor(array.length / 2 - 1);

*while* (index >= 0) {

sink(array, index, array.length);

index--;

}

};

const heapSort = (matrix) => {

console.log('old matrix heapSort:', matrix);

let newMatrix = [];

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

build\_heap(matrix[z]);

let end = matrix[z].length - 1;

*while* (end >= 0) {

matrix[z].swap(0, end);

sink(matrix[z], 0, end);

end -= 1;

}

newMatrix[z] = matrix[z];

}

*return* newMatrix;

};

const quickSort = (matrix) => {

console.log('old matrix quickSort:', matrix);

let newMatrix = [];

const sort = (arr) => {

*if* (arr.length < 2) *return* arr;

let min = 1;

let max = arr.length - 1;

let rand = Math.floor(min + Math.random() \* (max + 1 - min));

let pivot = arr[rand];

const left = [];

const right = [];

arr.splice(arr.indexOf(pivot), 1);

arr = [pivot].concat(arr);

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

*if* (pivot > arr[i]) {

left.push(arr[i]);

} *else* {

right.push(arr[i]);

}

}

*return* sort(left).concat(pivot, sort(right));

};

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

newMatrix[z] = sort(matrix[z]);

}

*return* newMatrix;

};

console.log(

'new matrix selectionSort:',

selectionSort(createMatrix(5, 5, 0, 10))

);

console.log(

'new matrix insertionSort:',

insertionSort(createMatrix(5, 5, 0, 10))

);

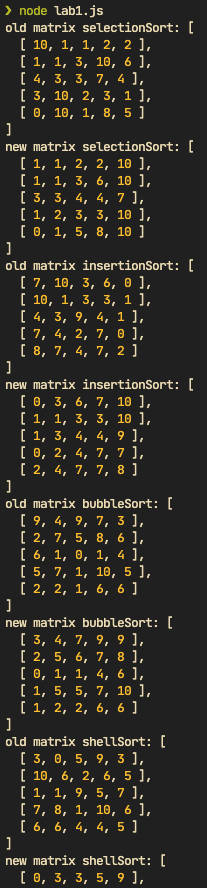
console.log('new matrix bubbleSort:', bubbleSort(createMatrix(5, 5, 0, 10)));

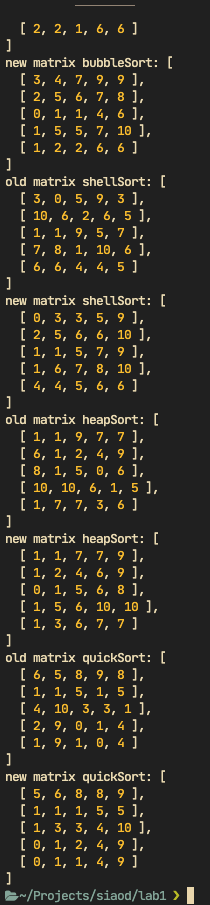
console.log('new matrix shellSort:', shellSort(createMatrix(5, 5, 0, 10)));

console.log('new matrix heapSort:', heapSort(createMatrix(5, 5, 0, 10)));

console.log('new matrix quickSort:', quickSort(createMatrix(5, 5, 0, 10)));

Cнимки экрана работы программы





Вывод

В ходе выполнения лабораторной работы, мной были получены навыки реализации различных алгоритмов сортировки на примере сортировки числовых матриц.