Федеральное агентство связи

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Отчет по лабораторной работе

по дисциплине «СИАОД»

на тему: «Алгоритмы сортировки»

Выполнил: студент группы БВТ1802

Сурин В.И.

Руководитель:

Кутейников Иван Алексеевич

Москва 2020

Цель работы: реализовать метод сортировки числовой матрицы в соответствии с индивидуальным заданием. Для все вариантов добавить реализацию быстрой сортировки. Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки, используемой в выбранном языке программирования.

Реализовать пирамидальную сортировку.

Выполнение работы:

```
import java.util.Random;
import java.util.Scanner;
import java.util.Arrays;

public class Main {
    static void heapify(int arr[], int n, int i) {
        int largest = i; // Инициализируем наибольший элемент как корень
        int l = 2 * i + 1; // левый = 2*i + 1
        int r = 2 * i + 2; // правый = 2*i + 2

        // Если левый дочерний элемент больше корня
        if (l < n && arr[l] > arr[largest])
            largest = l;

        // Если правый дочерний элемент больше, чем самый большой элемент на данный
момент
        if (r < n && arr[r] > arr[largest])
            largest = r;
        // Если самый большой элемент не корень
        if (largest != i) {
            int swap = arr[i];
            arr[i] = arr[largest];
            arr[largest] = swap;

            // Рекурсивно преобразуем в двоичную кучу затронутое поддерево
            heapify(arr, n, largest);
        }
    }

    public static void heapSort(int arr[]) {
        int n = arr.length;

        // Построение кучи (перегруппируем массив)
        for (int i = n / 2 - 1; i >= 0; i--)
            heapify(arr, n, i);

        // Один за другим извлекаем элементы из кучи
        for (int i = n - 1; i >= 0; i--) {
            // Перемещаем текущий корень в конец
            int temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;

            // Вызываем процедуру heapify на уменьшенной куче
            heapify(arr, i, 0);
        }
    }
```

```
public static void quickSort(int[] array, int low, int high) {
    if (array.length == 0)
        return;
    if (low >= high)
        return;
    int middle = low + (high - low) / 2;
    int opora = array[middle];
    int i = low, j = high;
    while(i <= j) {
        while (array[i] < opora)
            i++;
        while(array[j] > opora)
            j--;
        if (i <= j) {
            int temp = array[i];
            array[i] = array[j];
            array[j] = temp;
            i++;
            j--;
        }
    }
    if (low < j)
        quickSort(array, low, j);
    if (high > i)
        quickSort(array, i, high);
}

public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int n;
    System.out.println("Input matrix size");
    n = in.nextInt();

    int[][] array1 = new int[n][n];
    int[][] array2 = new int[n][n];
    int[][] array3 = new int[n][n];

    Random random = new Random();

    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++) {
            array1[i][j] = random.nextInt(100);
            array2[i][j] = array1[i][j];
            array3[i][j] = array1[i][j];
        }

    long before = System.currentTimeMillis();
    for (int i = 0; i < n; i++) {
        Arrays.sort(array1[i]);
    }
    long after = System.currentTimeMillis();
    System.out.println("Time JavaSort in millis: " + (after - before));

    before = System.currentTimeMillis();
    for (int i = 0; i < n; i++) {
        heapSort(array2[i]);
    }
    after = System.currentTimeMillis();
    System.out.println("Time HeapSort in millis: " + (after - before));

    before = System.currentTimeMillis();
    for (int i = 0; i < n; i++) {
```

```
            quickSort(array3[i], 0, n - 1);
        }
        after = System.currentTimeMillis();
        System.out.println("Time QuickSort in millis: " + (after - before));
    }
}
```

Тесты программы:

```
Input matrix size
3000
Time JavaSort in millis: 809
Time HeapSort in millis: 2064
Time QuickSort in millis: 1030
```

```
Input matrix size
5000
Time JavaSort in millis: 1710
Time HeapSort in millis: 3590
Time QuickSort in millis: 2250
```

Размер / Тип сорт.	1000x1000	2000x2000	3000x3000	5000x5000
JavaSort	310мс	380мс	809мс	1710мс
QuickSort	509мс	350мс	1030мс	2250мс
HeapSort	320мс	520мс	2064мс	3590мс

Вывод

В ходе работы были изучены реализации методов сортировки, было оценено время работы каждого алгоритма и сравнено время работы используемых сортировок.