

Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации

Бюджетное образовательное учреждение высшего образования
Московский технический университет связи и информатики

Кафедра «Математической кибернетики и информационных технологий»

Дисциплина «Структуры и алгоритмы обработки данных»

Отчёт по лабораторной работе №1

«Методы сортировки»

Выполнил студент
Группы БФИ1901
Курбатов А.О.

Проверил
Кутейников И. А.

Москва 2021

Задание

Лабораторная работа №1

Выполнил студент(ка) группы ГРУППА ФИО

Задание №1

Описание условия задания

```
In [1]: print("Hello, World!")  
Hello, World!
```

Задание №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:

$$m = 50$$

$$n = 50$$

$$\text{min_limit} = -250$$

$$\text{max_limit} = 1000 + (\text{номер своего варианта})$$

Задание №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.

Методы:

Выбором	Вставкой	Обменом	Шелла	Турнирная	Быстрая сортировка	Пирамидальная
---------	----------	---------	-------	-----------	--------------------	---------------

Листинг

```
public static void main(String[] args) {

    // Задание 1

    System.out.println("Лабораторная работа №1 \n" + "Выполнил студент группы  
БФИ1901 Курбатов А. О. \n" + "Задание №1");
    System.out.println("Hello, World!");

    // Задание 2

    System.out.println("\n" + "Задание №2" + "\n");
    int m = 10, n = 10, min_limit = -250, max_limit = 1011;
    Random rand = new Random();
    int[][] mass = new int[m][n];
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            mass[i][j] = min_limit + rand.nextInt(max_limit - min_limit + 1);
            System.out.print(mass[i][j] + " ");
        }
        System.out.println("\n");
    }

    public static void selectionSort(int[][] rezmass, int m, int n) {
        int index = -1;
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                int min = rezmass[i][j];
                for (int k = j + 1; k < n; k++) {
                    if (rezmass[i][k] < min) {
                        index = k;
                        min = rezmass[i][k];
                    }
                }
                if (index != -1) {
                    rezmass[i][index] = rezmass[i][j];
                    rezmass[i][j] = min;
                }
                index = -1;
            }
        }

        public static void insertionSort(int[][] rezmass, int m, int n) {
            for (int i = 0; i < m; i++){
                for (int j = 1; j < n; j++) {
                    int selected = rezmass[i][j];
                    int k = j - 1;
                    while (k >= 0 && selected < rezmass[i][k]) {
                        rezmass[i][k + 1] = rezmass[i][k];
                        k--;
                    }
                    rezmass[i][k + 1] = selected;
                }
            }
        }
    }
}
```

```

public static void bubbleSort(int[][] rezmass, int m, int n) {
    int swap;
    long t = System.currentTimeMillis();
    boolean k = true;
    while (k == true) {
        k = false;
        for (int i = 0; i < m; i++) {
            for (int j = n - 1; j > 0; j--) {
                if (rezmass[i][j] > rezmass[i][j - 1]) {
                    swap = rezmass[i][j];
                    rezmass[i][j] = rezmass[i][j - 1];
                    rezmass[i][j - 1] = swap;
                    k = true;
                }
            }
        }
    }
}

public static void sort(int[][] rezmass, int m, int n) {
    int h = n / 2;
    while(h >= 1) {
        ShellSort(rezmass, h, m, n);
        h = h/3;
    }
}

public static void ShellSort(int[][] rezmass, int h, int m, int n) {
    int swap;
    for (int i = 0; i < m; i++) {
        for (int j = h; j < n; j++) {
            for (int k = j; k >= h; k = k - h) {
                if (rezmass[i][k] < rezmass[i][k - h]) {
                    swap = rezmass[i][k];
                    rezmass[i][k] = rezmass[i][k - h];
                    rezmass[i][k - h] = swap;
                }
            }
        }
    }
}

public static void quickSort(int[][] rezmass, int low, int high, int m) {
    for (int i = 0; i < m; i++) {
        if (rezmass.length == 0)
            return;

        if (low >= high)
            return;

        int middle = low + (high - low) / 2;
        int opora = rezmass[i][middle];

        int l = low, h = high;
        while (l <= h) {
            while (rezmass[i][l] < opora) {
                l++;
            }

```

```

        while (rezmass[i][h] > opora) {
            h--;
        }

        if (l <= h) {
            int temp = rezmass[i][l];
            rezmass[i][l] = rezmass[i][h];
            rezmass[i][h] = temp;
            l++;
            h--;
        }
    }
    if (low < h)
        quickSort(rezmass, low, h, m);

    if (high > l)
        quickSort(rezmass, l, high, m);
}

public static void heapSort(int[][] rezmass, int m) {
    for(int i = 0; i < m; i++) {
        int k = rezmass.length;

        for (int j = k / 2 - 1; j >= 0; j--)
            heapify(rezmass, k, j, i);

        for (int j = k - 1; j >= 0; j--) {
            int temp = rezmass[i][0];
            rezmass[i][0] = rezmass[i][j];
            rezmass[i][j] = temp;
            heapify(rezmass, j, 0, i);
        }
    }
}

static void heapify(int [][] rezmass, int k, int j, int i)
{
    int largest = j;
    int l = 2*j + 1;
    int r = 2*j + 2;

    if (l < k && rezmass[i][l] > rezmass[i][largest])
        largest = l;

    if (r < k && rezmass[i][r] > rezmass[i][largest])
        largest = r;

    if (largest != j)
    {
        int swap = rezmass[i][j];
        rezmass[i][j] = rezmass[i][largest];
        rezmass[i][largest] = swap;
        heapify(rezmass, k, largest, i);
    }
}

```

```

public static void main(String[] args) {

    // Задание 3

    System.out.println("Задание №3" + "\n");

    int[][] rezmass = new int[m][n];
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            rezmass[i][j] = mass[i][j];
        }
    }

    // Сортировка выбором

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            rezmass[i][j] = mass[i][j];
        }
    }

    System.out.println("Сортировка выбором" + "\n");
    long t = System.currentTimeMillis();
    selectionSort(rezmass, m, n);
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(rezmass[i][j] + " ");
        }
        System.out.println("\n");
    }
    System.out.println("Время: ");
    System.out.println(((double)System.currentTimeMillis()-t) + " ms");

    // Сортировка вставками

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            rezmass[i][j] = mass[i][j];
        }
    }
    System.out.println("Сортировка вставками " + "\n");

    t = System.currentTimeMillis();
    insertionSort(rezmass, m, n);
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(rezmass[i][j] + " ");
        }
        System.out.println("\n");
    }
    System.out.println("Время: ");
    System.out.println(((double)System.currentTimeMillis()-t) + " ms");

    // Сортировка пузырьком

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            rezmass[i][j] = mass[i][j];
        }
    }

```

```

    }
    System.out.println("Сортировка пузырьком " + "\n");

    t = System.currentTimeMillis();
    bubbleSort(rezmass, m, n);
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(rezmass[i][j] + " ");
        }
        System.out.println("\n");
    }
    System.out.println("Время: ");
    System.out.println(((double)System.currentTimeMillis()-t) + " ms");

    // Сортировка Шелла

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            rezmass[i][j] = mass[i][j];
        }
    }

    System.out.println("Сортировка Шелла" + "\n");
    t = System.currentTimeMillis();
    sort(rezmass, m, n);
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(rezmass[i][j] + " ");
        }
        System.out.println("\n");
    }
    System.out.println("Время: ");
    System.out.println(((double)System.currentTimeMillis()-t) + " ms");

    // Быстрая сортировка

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            rezmass[i][j] = mass[i][j];
        }
    }

    int low = 0;
    int high = mass.length - 1;

    System.out.println("Быстрая сортировка" + "\n");

    t = System.currentTimeMillis();
    quickSort(rezmass, low, high, m);

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(rezmass[i][j] + " ");
        }
        System.out.println("\n");
    }
    System.out.println("Время: ");
    System.out.println(((double)System.currentTimeMillis()-t) + " ms");

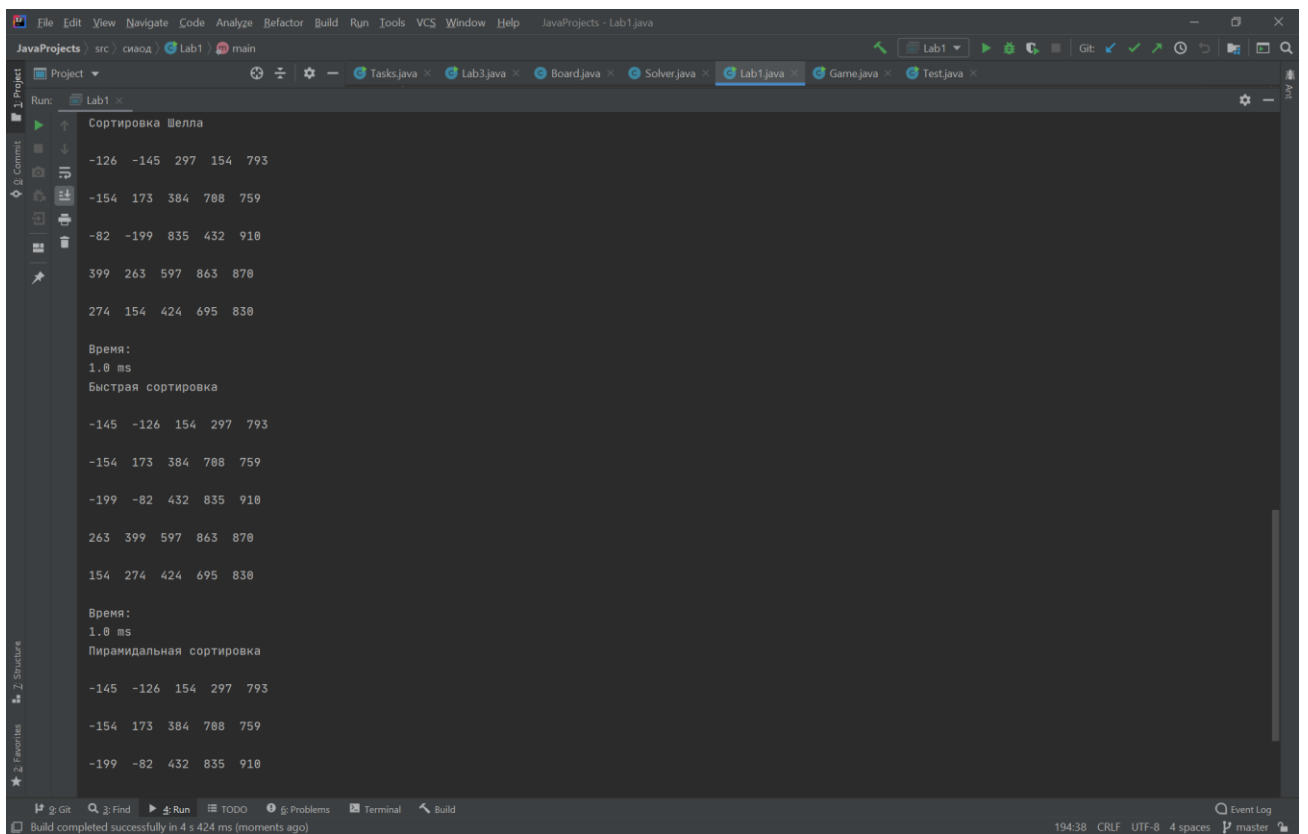
```

```
// Пирамидальная сортировка

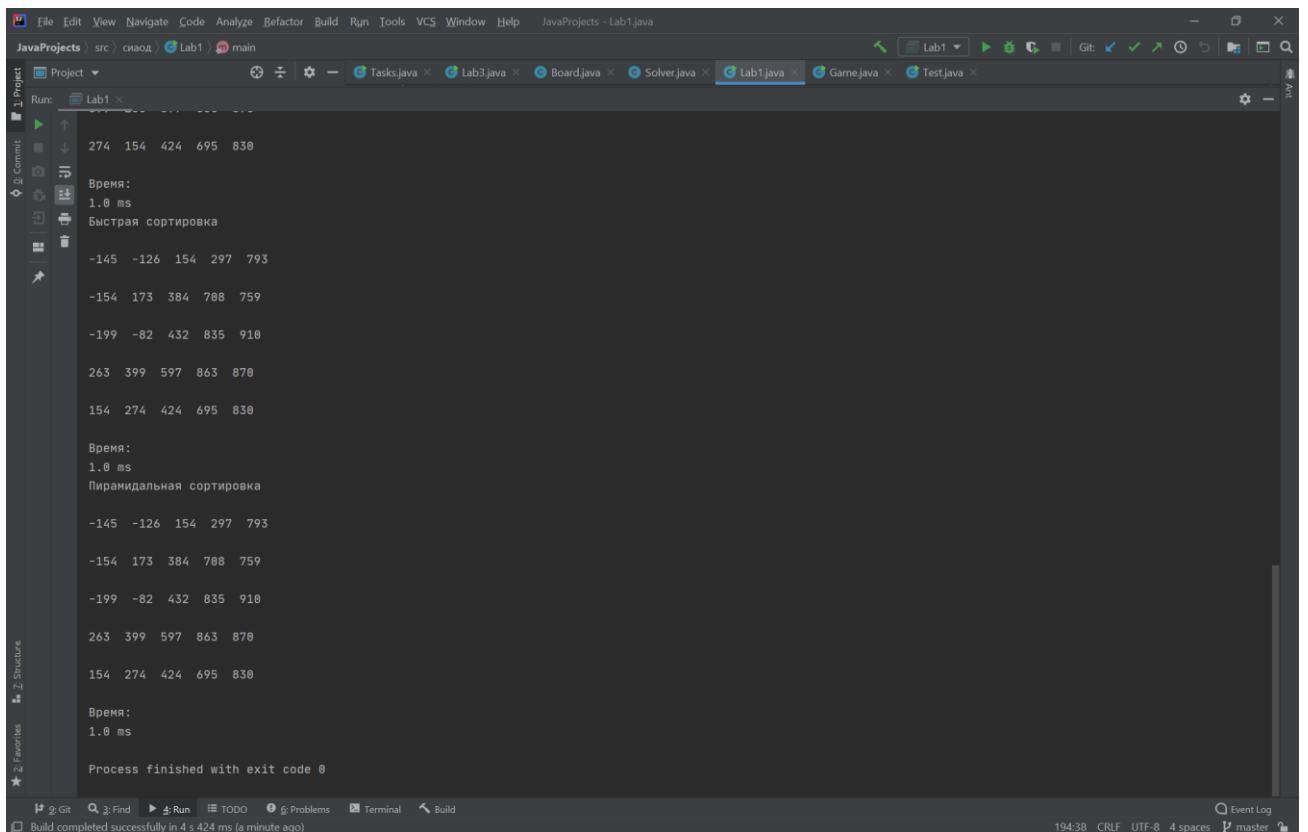
for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
        rezmass[i][j] = mass[i][j];
    }
}
System.out.println("Пирамидальная сортировка" + "\n");
t = System.currentTimeMillis();
heapSort(rezmass, m);

for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
        System.out.print(rezmass[i][j] + " ");
    }
    System.out.println("\n");
}
System.out.println("Время: ");
System.out.println(((double)System.currentTimeMillis()-t) + " ms");
}
```

Результат выполнения работы



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help JavaProjects - Lab1.java
JavaProjects src / src / Lab1 / main
Run: Lab1
Сортировка Шелла
-126 -145 297 154 793
-154 173 384 788 759
-82 -199 835 432 918
399 263 597 863 878
274 154 424 695 838
Время:
1.0 ms
Быстрая сортировка
-145 -126 154 297 793
-154 173 384 788 759
-199 -82 432 835 918
263 399 597 863 878
154 274 424 695 838
Время:
1.0 ms
Пирамидальная сортировка
-145 -126 154 297 793
-154 173 384 788 759
-199 -82 432 835 918
Build completed successfully in 4 s 424 ms (moments ago) 194:38 CRLF UTF-8 4 spaces master
```



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help JavaProjects - Lab1.java
JavaProjects src / src / Lab1 / main
Run: Lab1
274 154 424 695 838
Время:
1.0 ms
Быстрая сортировка
-145 -126 154 297 793
-154 173 384 788 759
-199 -82 432 835 918
263 399 597 863 878
154 274 424 695 838
Время:
1.0 ms
Пирамидальная сортировка
-145 -126 154 297 793
-154 173 384 788 759
-199 -82 432 835 918
263 399 597 863 878
154 274 424 695 838
Время:
1.0 ms
Process finished with exit code 0
Build completed successfully in 4 s 424 ms (a minute ago) 194:38 CRLF UTF-8 4 spaces master
```

Вывод

В ходе данной лабораторной работы мы изучили методы сортировки, реализовали их на языке программирования java и сравнили скорость этих методов.