**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Информатики

****

**Отчет по лабораторной работе № 1**

по дисциплине «СиАОД»

на тему:

«**Методы сортировки**»

Выполнила: студентка группы БВТ1802

Лаврухина Елена Павловна

Руководитель:

Кутейников Иван Алексеевич

Москва 2020

Цель работы

Реализовать заданный метод сортировки числовой матрицы в соответствии с индивидуальным заданием. Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки, используемой в выбранном языке программирования.

Выполнение

Код программы

**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStreamReader;  
**import** java.util.Random;  
**import** java.util.Arrays;  
  
**public class** main {  
 **public static double** *timer*=0.0;  
 **public static void** main(String[] args) **throws** IOException {  
 BufferedReader bufferedReader = **new** BufferedReader(**new** InputStreamReader(System.***in***));  
 System.***out***.print(**"Введите количество строк: "**);  
 String s1 = bufferedReader.readLine();  
 System.***out***.print(**"Введите количество столбцов: "**);  
 String s2 = bufferedReader.readLine();  
 **int** a = Integer.*parseInt*(s1);  
 **int** b = Integer.*parseInt*(s2);  
 **int**[][] matrixA = **new int**[a][b];  
 Random random = **new** Random();  
 **for** (**int** i=0; i<a; i++) {  
 **for** (**int** j=0; j<b; j++) {  
 matrixA[i][j] = random.ints(0,(100+1)).findFirst().getAsInt();  
 **if** (j+1==b)  
 System.***out***.println(matrixA[i][j]);  
 **else** System.***out***.print(matrixA[i][j]+**"\t"**);  
 }  
 }  
 **int**[][] matrixAS = *clone*(matrixA,a,b);  
 **int**[][] matrixAQ = *clone*(matrixA,a,b);  
 **int**[][] matrixSS = *clone*(matrixA,a,b);  
 System.***out***.println(**"№1"**);  
 *time*();  
 matrixAS=*Sort*(matrixAS,a,b);  
 *time*();  
 **for** (**int** i=0; i<a; i++) {  
 **for** (**int** j=0; j<b; j++) {  
 **if** (j+1==b)  
 System.***out***.println(matrixAS[i][j]);  
 **else** System.***out***.print(matrixAS[i][j]+**"\t"**);  
 }  
 }  
 System.***out***.println(**"№2"**);  
 *time*();  
 *quickSort*(matrixAQ,a,b);  
 *time*();  
 **for** (**int** i=0; i<a; i++) {  
 **for** (**int** j=0; j<b; j++) {  
 **if** (j+1==b)  
 System.***out***.println(matrixAQ[i][j]);  
 **else** System.***out***.print(matrixAQ[i][j]+**"\t"**);  
 }  
 }  
 System.***out***.println(**"№3"**);  
 *time*();  
 *standardSort*(matrixSS,a);  
 *time*();  
 **for** (**int** i=0; i<a; i++) {  
 **for** (**int** j=0; j<b; j++) {  
 **if** (j+1==b)  
 System.***out***.println(matrixSS[i][j]);  
 **else** System.***out***.print(matrixSS[i][j]+**"\t"**);  
 }  
 }  
 }  
 **public static int**[][] Sort(**int**[][] matrixAS, **int** a, **int** b) {  
 **int** min, temp, mj = 0, mi = 0;  
 **for** (**int** m = 0; m < a; m++) {  
 **for** (**int** i = 0; i < b - 1; i++) {  
 min = i;  
 **for** (**int** j = i + 1; j < b; j++) {  
 **if** (matrixAS[m][j] < matrixAS[m][min]) {  
 min = j;}  
 }temp = matrixAS[m][min];  
 matrixAS[m][min] = matrixAS[m][i];  
 matrixAS[m][i] = temp;  
 }  
 }  
 **return** matrixAS;  
 }  
 **public static void** quickSort(**int**[][] matr,**int** a,**int** b){  
 **for** (**int** i=0; i<a; i++){  
 *quickSortLine*(matr[i],0,b-1);  
 }  
 }  
 **public static void** quickSortLine(**int**[] matr, **int** low, **int** high){  
 **if** (matr.**length** == 0)  
 **return**;  
 **if** (low >= high)  
 **return**;  
 **int** middle = low + (high - low) / 2;  
 **int** opora = matr[middle];  
  
 **int** i = low, j = high;  
 **while** (i <= j) {  
 **while** (matr[i] < opora) {  
 i++;  
 }  
 **while** (matr[j] > opora) {  
 j--;  
 }  
 **if** (i <= j) {**int** temp = matr[i];  
 matr[i] = matr[j];  
 matr[j] = temp;  
 i++;  
 j--;  
 }  
 }  
 **if** (low < j)  
 *quickSortLine*(matr, low, j);  
  
 **if** (high > i)  
 *quickSortLine*(matr, i, high);  
 }  
 **public static void** standardSort(**int**[][] matr,**int** a) {  
 **for** (**int** i=0;i<a;i++){  
 Arrays.*sort*(matr[i]);  
 }  
 }  
 **public static void** time(){  
 **if** (*timer*==0) *timer*=System.*nanoTime*();  
 **else** {  
 **double** val = (**double**)(System.*nanoTime*()-*timer*);  
 System.***out***.println(val/1000000);  
 *timer*=0;  
 }  
 }  
 **public static int**[][] clone(**int**[][] matr, **int** a, **int** b) {  
 **int**[][] newMatr = **new int**[a][b];  
 **for** (**int** i=0; i<a; i++) {  
 **for** (**int** j = 0; j < b; j++) {  
 newMatr[i][j]=matr[i][j];  
 }  
 }  
 **return** newMatr;  
 }  
}

Скриншот работы программы

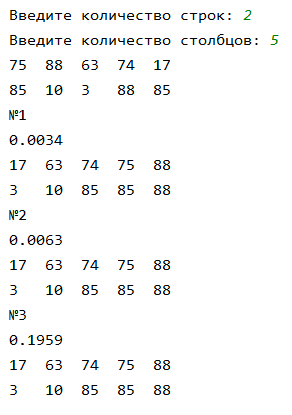


Рисунок 1 – Работа программы

Сравнение времени работы используемых сортировок

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Количество столбцов** | | 5 | 25 | 125 | 625 | 3125 |
| **Количество строк** | | 2 | 2 | 2 | 2 | 2 |
| **Время** | **Choice** | 0.0034 | 0.0228 | 0.3447 | 4.4273 | 16.8009 |
| **Quick** | 0.0063 | 0.0173 | 0.1083 | 0.316 | 0.9044 |
| **Standard** | 0.1959 | 0.2778 | 0.4497 | 0.5016 | 1.2971 |

Вывод

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