

Федеральное агентство связи
Ордена Трудового Красного Знамени федеральное государственное
бюджетное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и
информационных технологий

Лабораторная работа №1
по дисциплине: «Структуры и алгоритмы обработки данных»
на тему: «Методы сортировки»

Выполнил студент
группы БФИ1902
Гусев Н. С.
Проверил:
Мкртчян Г. М.

Москва, 2020 г.

Оглавление

| | |
|---|---|
| 1. Цель работы..... | 3 |
| 2. Задание на лабораторную работу | 3 |
| 3. Листинг программы | 3 |

1. Цель работы

Цель работы: рассмотреть работу методов сортировки строк числовой матрицы: Выбором, вставкой, обменом, Шелла, турнирная, быстрая сортировка, пирамидальная.

2. Задание на лабораторную работу

- 1) Написать программу, которая выводит текст «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.
- 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.
- 4) Создать публичный репозиторий на GitHub и запустить выполненное задание.

3. Листинг программы

```
import java.util.Arrays;

public class Main {

    public static int m = 20; // строки
    public static int n = 20; // столбцы
    public static int min_lim = -250; // Минимальный элемент
    public static int max_lim = 1003; // Максимальный элемент

    public static void main(String[] args) {
        System.out.println("1) Hello World!");
        System.out.println();

        int[][] a = new int[m][n];
        Random(a);

        System.out.println("2) Матрица до преобразований");
        showMatrix(a);
        System.out.println();

        System.out.println("3) Системная сортировка:");
        long start0 = System.nanoTime();
        for(int i = 0; i < n; i++) {
            int[] c = a[i];
```

```

        Arrays.sort(c);
        System.out.println(Arrays.toString(c));
    }
    long finish0 = System.nanoTime();
    long elapsed0 = finish0 - start0;
    System.out.println("Прошло времени, нс: " + elapsed0);
    System.out.println();

    System.out.println("Сортировка выбором:");
    long start1 = System.nanoTime();
    for (int i = 0; i < n; i++)
        selectionSort(a[i]);
    long finish1 = System.nanoTime();
    long elapsed1 = finish1 - start1;
    System.out.println("Прошло времени, нс: " + elapsed1);
    System.out.println();

    System.out.println("Сортировка вставками:");
    long start2 = System.nanoTime();
    for (int i = 0; i < n; i++)
        InsertionSort(a[i]);
    long finish2 = System.nanoTime();
    long elapsed2 = finish2 - start2;
    System.out.println("Прошло времени, нс: " + elapsed2);
    System.out.println();

    System.out.println("Сортировка обменом:");
    long start3 = System.nanoTime();
    for (int i = 0; i < n; i++)
        bubbleSort(a[i]);
    long finish3 = System.nanoTime();
    long elapsed3 = finish3 - start3;
    System.out.println("Прошло времени, нс: " + elapsed3);
    System.out.println();

    System.out.println("Сортировка Шелла:");
    long start4 = System.nanoTime();
    for (int i = 0; i < n; i++)
        Shell(a[i]);
    long finish4 = System.nanoTime();
    long elapsed4 = finish4 - start4;
    System.out.println("Прошло времени, нс: " + elapsed4);
    System.out.println();

    System.out.println("Быстрая сортировка:");
    long start5 = System.nanoTime();
    for (int i = 0; i < n; i++) {
        quicksort(a[i], 0, m - 1);
        System.out.println(Arrays.toString(a[i]));
    }
    long finish5 = System.nanoTime();
    long elapsed5 = finish5 - start5;
    System.out.println("Прошло времени, нс: " + elapsed5);
    System.out.println();

    System.out.println("Пирамидальная сортировка:");
    long start6 = System.nanoTime();
    for (int i = 0; i < n; i++)
        heapSort(a[i]);
    long finish6 = System.nanoTime();
    long elapsed6 = finish6 - start6;
    System.out.println("Прошло времени, нс: " + elapsed6);
    System.out.println();

```

```

        if (elapsed1 <= elapsed2 && elapsed1 <= elapsed3 && elapsed1 <=
elapsed4 && elapsed1 <= elapsed5 && elapsed1 <= elapsed6 && elapsed1 <=
elapsed0) {
            System.out.println("Сортировка выбором - самая быстрая
сортировка");
        } else if (elapsed2 <= elapsed1 && elapsed2 <= elapsed3 && elapsed2
<= elapsed4 && elapsed2 <= elapsed5 && elapsed2 <= elapsed6 && elapsed2 <=
elapsed0) {
            System.out.println("Сортировка вставками - самая быстрая
сортировка");
        } else if (elapsed3 <= elapsed2 && elapsed3 <= elapsed1 && elapsed3
<= elapsed4 && elapsed3 <= elapsed5 && elapsed3 <= elapsed6 && elapsed3 <=
elapsed0) {
            System.out.println("Сортировка обменом - самая быстрая
сортировка");
        } else if (elapsed4 <= elapsed1 && elapsed4 <= elapsed2 && elapsed4
<= elapsed3 && elapsed4 <= elapsed5 && elapsed4 <= elapsed6 && elapsed4 <=
elapsed0) {
            System.out.println("Сортировка Шелла - самая быстрая
сортировка");
        } else if (elapsed5 <= elapsed1 && elapsed5 <= elapsed2 && elapsed5
<= elapsed3 && elapsed5 <= elapsed4 && elapsed5 <= elapsed6 && elapsed5 <=
elapsed0) {
            System.out.println("Быстрая сортировка - самая быстрая
сортировка");
        } else if (elapsed6 <= elapsed1 && elapsed6 <= elapsed2 && elapsed6
<= elapsed3 && elapsed6 <= elapsed4 && elapsed6 <= elapsed5 && elapsed6 <=
elapsed0) {
            System.out.println("Пирамидальная сортировка - самая быстрая
сортировка");
        } else {
            System.out.println("Системная сортировка - самая быстрая
сортировка");
        }
    }

    // Генерация случайных чисел
    public static void Random(int[][] a) {
        for(int i = 0; i < m; i++) {
            for(int j = 0; j < n; j++) {
                a[i][j] = min_lim + (int) (Math.random() * ((max_lim - min_lim)
+ 1));
            }
        }
    }

    // Вывод матрицы
    public static void showMatrix(int[][] a){
        for(int i=0; i < m; i++){
            System.out.print("[ ");
            for(int j=0; j < n; j++){
                System.out.print(a[i][j] + " ");
            }
            System.out.print("]");
            System.out.println();
        }
    }

    // Сортировка вставкой
    public static void InsertionSort(int[] array)
    {
        int[] b = array;
        for (int i = 1; i < array.length; i++)

```

```

    {
        int j;
        int buf = b[i];
        for (j = i - 1; j >= 0; j--)
        {
            if (b[j] < buf)
                break;
            b[j + 1] = b[j];
        }
        b[j + 1] = buf;
    }
    System.out.println(Arrays.toString(b));
}

// Сортировка выбором
public static void selectionSort(int[] a) {
    int[] b = a;
    int tmp;
    for(int i = 0; i < m; i++)
    {
        int pos = i;
        tmp = b[i];
        for(int j = i + 1; j < m; j++)
        {
            if (b[j] < tmp)
            {
                pos = j;
                tmp = b[j];
            }
        }
        b[pos] = b[i];
        b[i] = tmp;
    }
    System.out.println(Arrays.toString(b));
}

// Сортировка обменом, или пузырьком
public static void bubbleSort(int[] a) {
    int[] b = a;
    boolean sorted = false;
    int temp;
    while(!sorted) {
        sorted = true;
        for (int i = 0; i < b.length - 1; i++) {
            if (b[i] > b[i+1]) {
                temp = b[i];
                b[i] = b[i+1];
                b[i+1] = temp;
                sorted = false;
            }
        }
    }
    System.out.println(Arrays.toString(b));
}

// Сортировка Шелла
public static void Shell(int[] a) {
    int[] b = a;
    for (int step = n / 2; step > 0; step /= 2) {
        for (int i = step; i < n; i++) {
            for (int j = i - step; j >= 0 && b[j] > b[j + step]; j -=
step) {
                int x = b[j];

```

```

        b[j] = b[j + step];
        b[j + step] = x;
    }
}

System.out.println(Arrays.toString(b));
}

// Быстрая сортировка
public static int partition (int[] a, int start, int end)
{
    int marker = start;
    for ( int i = start; i <= end; i++ )
    {
        if (a[i] <= a[end] )
        {
            int temp = a[marker]; // swap
            a[marker] = a[i];
            a[i] = temp;
            marker += 1;
        }
    }
    return marker - 1;
}

public static void quicksort(int[] a, int start, int end) {
    int[] b = a;
    if ( start >= end ) {
        return;
    }
    int pivot = partition (b, start, end);
    quicksort (b, start, pivot-1);
    quicksort (b, pivot+1, end);
}

// Пирамидальная сортировка
static void heapify(int[] a, int length, int i) {
    int leftChild = 2*i+1;
    int rightChild = 2*i+2;
    int largest = i;

    // если левый дочерний больше родительского
    if (leftChild < length && a[leftChild] > a[largest]) {
        largest = leftChild;
    }

    // если правый дочерний больше родительского
    if (rightChild < length && a[rightChild] > a[largest]) {
        largest = rightChild;
    }

    // если должна произойти замена
    if (largest != i) {
        int temp = a[i];
        a[i] = a[largest];
        a[largest] = temp;
        heapify(a, length, largest);
    }
}

public static void heapSort(int[] a) {
    int[] b = a;
    // Строим кучу
    int length = b.length;
    // проходим от первого без ответвлений к корню

```

```

    for (int i = length / 2-1; i >= 0; i--)
        heapify(b, length, i);
    for (int i = length-1; i >= 0; i--) {
        int temp = b[0];
        b[0] = b[i];
        b[i] = temp;
        heapify(b, i, 0);
    }
    System.out.println(Arrays.toString(b));
}
}

```

4. Результат работы программы

```

Run: Main x
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
1) Hello World!

2) Матрица до преобразований
[ 809 120 75 931 1000 517 115 386 571 711 148 420 264 477 812 211 581 54 683 186 ]
[ 668 101 231 266 298 -102 779 906 641 -32 -157 364 -155 528 206 925 121 629 612 883 ]
[ 169 703 289 -194 782 936 733 -16 516 867 573 122 495 425 217 558 951 605 931 -200 ]
[ -41 651 837 561 402 472 326 282 830 712 318 923 -165 -185 11 857 120 51 5 309 ]
[ 419 68 491 41 955 620 423 244 636 751 658 71 -72 818 858 557 380 810 342 891 ]
[ 628 244 546 497 331 657 265 405 401 776 680 -116 260 616 2 533 552 -177 434 -2 ]
[ 210 572 -130 107 215 767 633 438 72 877 442 -156 894 687 122 -210 883 -75 -35 -183 ]
[ 98 -170 329 753 204 369 293 240 431 -177 -43 585 595 -60 954 387 652 595 939 745 ]
[ 356 801 90 854 720 2 689 933 -44 205 74 820 488 474 -70 727 211 814 813 -89 ]
[ 872 984 524 997 -49 286 477 726 327 593 616 69 -59 366 -161 921 -157 17 -221 22 ]
[ 103 764 484 388 -112 519 784 -93 -50 -171 -11 507 624 995 -26 672 -65 560 12 327 ]
[ 748 698 -92 478 452 703 949 -134 634 316 507 783 593 -63 966 989 734 715 485 1 ]
[ 280 -213 18 359 301 201 -208 769 990 -212 977 669 548 259 359 720 548 13 308 167 ]
[ 370 -34 254 950 378 344 -77 -166 681 914 500 192 24 941 688 164 -171 973 224 738 ]
[ 457 -76 -214 -79 526 627 1 -230 373 902 704 816 123 806 986 284 122 624 992 -250 ]
[ 314 299 261 840 591 127 -221 774 -39 -122 717 223 -225 257 871 213 235 527 148 692 ]
[ 67 465 104 288 163 -229 -181 120 783 16 -120 304 -117 449 348 -23 867 -178 50 44 ]
[ 355 214 674 24 711 891 -81 172 -111 532 -46 -235 -170 749 48 -191 696 334 -33 727 ]
[ -1 756 371 530 122 -68 974 148 -47 -48 238 20 461 -209 526 862 651 284 32 311 ]
[ 82 318 75 1003 305 727 565 674 405 437 677 229 71 748 536 712 867 493 781 515 ]

3) Системная сортировка:
[54, 75, 115, 120, 148, 186, 211, 264, 386, 420, 477, 517, 571, 581, 683, 711, 809, 812, 931, 1000]
[-157, -155, -102, -32, 101, 121, 206, 231, 266, 298, 364, 528, 612, 629, 641, 668, 779, 883, 906, 925]
[-200, -194, -16, 122, 169, 217, 289, 425, 495, 516, 558, 573, 605, 703, 733, 782, 867, 931, 936, 951]
[-185, -165, -41, 5, 11, 51, 120, 282, 309, 318, 326, 402, 472, 561, 651, 712, 830, 837, 857, 923]
[-72, 41, 68, 71, 244, 342, 380, 419, 423, 491, 557, 620, 636, 658, 751, 810, 818, 858, 891, 955]
[-177, -116, -2, 2, 244, 260, 265, 331, 401, 405, 434, 497, 533, 546, 552, 616, 628, 657, 680, 776]
[-210, -183, -156, -130, -75, -35, 72, 107, 122, 210, 215, 438, 442, 572, 633, 687, 767, 877, 883, 894]
[-177, -170, -60, -43, 98, 204, 240, 293, 329, 369, 387, 431, 585, 595, 595, 652, 745, 753, 939, 954]
[-89, -70, -44, 2, 74, 90, 205, 211, 356, 474, 488, 689, 720, 727, 801, 813, 814, 820, 854, 933]
[-221, -161, -157, -59, -49, 17, 22, 69, 286, 327, 366, 477, 524, 593, 616, 726, 872, 921, 984, 997]
[-171, -112, -93, -65, -50, -26, -11, 12, 103, 327, 388, 484, 507, 519, 560, 624, 672, 764, 784, 995]
[-134, -92, -63, 1, 316, 452, 478, 485, 507, 593, 634, 698, 703, 715, 734, 748, 783, 949, 966, 989]
[-213, -212, -208, 13, 18, 167, 201, 259, 280, 301, 308, 359, 359, 548, 548, 669, 720, 769, 977, 990]
[-171, -166, -77, -34, 24, 164, 192, 224, 254, 344, 370, 378, 500, 681, 688, 738, 914, 941, 950, 973]
[-250, -230, -214, -79, -76, 1, 122, 123, 284, 373, 457, 526, 624, 627, 704, 806, 816, 902, 986, 992]

```



```
[-89, -70, -44, 2, 74, 90, 205, 211, 356, 474, 488, 689, 720, 727, 801, 813, 814, 820, 854, 933]
[-221, -161, -157, -59, -49, 17, 22, 69, 286, 327, 366, 477, 524, 593, 616, 726, 872, 921, 984, 997]
[-171, -112, -93, -65, -50, -26, -11, 12, 103, 327, 388, 484, 507, 519, 560, 624, 672, 764, 784, 995]
[-134, -92, -63, 1, 316, 452, 478, 485, 507, 593, 634, 698, 703, 715, 734, 748, 783, 949, 966, 989]
[-213, -212, -208, 13, 18, 167, 201, 259, 280, 301, 308, 359, 359, 548, 548, 669, 720, 769, 977, 990]
[-171, -166, -77, -34, 24, 164, 192, 224, 254, 344, 370, 378, 500, 681, 688, 738, 914, 941, 950, 973]
[-250, -230, -214, -79, -76, 1, 122, 123, 284, 373, 457, 526, 624, 627, 704, 806, 816, 902, 986, 992]
[-225, -221, -122, -39, 127, 148, 213, 223, 235, 257, 261, 299, 314, 527, 591, 692, 717, 774, 840, 871]
[-229, -181, -178, -120, -117, -23, 16, 44, 50, 67, 104, 120, 163, 288, 304, 348, 449, 465, 783, 867]
[-235, -191, -170, -111, -81, -46, -33, 24, 48, 172, 214, 334, 355, 532, 674, 696, 711, 727, 749, 891]
[-209, -68, -48, -47, -1, 20, 32, 122, 148, 238, 284, 311, 371, 461, 526, 530, 651, 756, 862, 974]
[71, 75, 82, 229, 305, 318, 405, 437, 493, 515, 536, 565, 674, 677, 712, 727, 748, 781, 867, 1003]
Прошло времени, нс: 636000

Пирамидальная сортировка:
[54, 75, 115, 120, 148, 186, 211, 264, 386, 420, 477, 517, 571, 581, 683, 711, 809, 812, 931, 1000]
[-157, -155, -102, -32, 101, 121, 206, 231, 266, 298, 364, 528, 612, 629, 641, 668, 779, 883, 906, 925]
[-200, -194, -16, 122, 169, 217, 289, 425, 495, 516, 558, 573, 605, 703, 733, 782, 867, 931, 936, 951]
[-185, -165, -41, 5, 11, 51, 120, 282, 309, 318, 326, 402, 472, 561, 651, 712, 830, 837, 857, 923]
[-72, 41, 68, 71, 244, 342, 380, 419, 423, 491, 557, 620, 636, 658, 751, 810, 818, 858, 891, 955]
[-177, -116, -2, 2, 244, 260, 265, 331, 401, 405, 434, 497, 533, 546, 552, 616, 628, 657, 680, 776]
[-210, -183, -156, -130, -75, -35, 72, 107, 122, 210, 215, 438, 442, 572, 633, 687, 767, 877, 883, 894]
[-177, -170, -60, -43, 98, 204, 240, 293, 329, 369, 387, 431, 585, 595, 595, 652, 745, 753, 939, 954]
[-89, -70, -44, 2, 74, 90, 205, 211, 356, 474, 488, 689, 720, 727, 801, 813, 814, 820, 854, 933]
[-221, -161, -157, -59, -49, 17, 22, 69, 286, 327, 366, 477, 524, 593, 616, 726, 872, 921, 984, 997]
[-171, -112, -93, -65, -50, -26, -11, 12, 103, 327, 388, 484, 507, 519, 560, 624, 672, 764, 784, 995]
[-134, -92, -63, 1, 316, 452, 478, 485, 507, 593, 634, 698, 703, 715, 734, 748, 783, 949, 966, 989]
[-213, -212, -208, 13, 18, 167, 201, 259, 280, 301, 308, 359, 359, 548, 548, 669, 720, 769, 977, 990]
[-171, -166, -77, -34, 24, 164, 192, 224, 254, 344, 370, 378, 500, 681, 688, 738, 914, 941, 950, 973]
[-250, -230, -214, -79, -76, 1, 122, 123, 284, 373, 457, 526, 624, 627, 704, 806, 816, 902, 986, 992]
[-225, -221, -122, -39, 127, 148, 213, 223, 235, 257, 261, 299, 314, 527, 591, 692, 717, 774, 840, 871]
[-229, -181, -178, -120, -117, -23, 16, 44, 50, 67, 104, 120, 163, 288, 304, 348, 449, 465, 783, 867]
[-235, -191, -170, -111, -81, -46, -33, 24, 48, 172, 214, 334, 355, 532, 674, 696, 711, 727, 749, 891]
[-209, -68, -48, -47, -1, 20, 32, 122, 148, 238, 284, 311, 371, 461, 526, 530, 651, 756, 862, 974]
[71, 75, 82, 229, 305, 318, 405, 437, 493, 515, 536, 565, 674, 677, 712, 727, 748, 781, 867, 1003]
Прошло времени, нс: 719000

Сортировка вставками - самая быстрая сортировка

Process finished with exit code 0
```

5. Вывод

Я рассмотрел различные методы сортировки строк числовой матрицы.