

|  |  |
| --- | --- |
| **Министерство образования и науки**  **Российской Федерации**  **Государственное образовательное учреждение высшего образования «МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ**  **ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  **(МАДИ)»** |  |

**Кафедра «Высшая математика»**

**Отчет по дисциплине «Структуры и алгоритмы обработки данных»**

**Лабораторная работа №4**

**«Методы поиска и модификация двумерных массивов»**

**Выполнил:**

Учебная группа 1бПМ\_\_\_\_\_\_\_\_\_  
ФИО Кузнецов А-С.О.\_\_\_\_\_

**Принял:**

Должность \_\_\_\_\_

Звание \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ФИО \_ Кутейников И.А.\_\_\_\_\_\_

«\_\_\_ »\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г.

Москва 2023

Цель: определить двумерный массив целых чисел (например,

размера 30), заполнить его случайными числами (в диапазоне от A до B) или

ввести его элементы с клавиатуры и определить его характеристики в

соответствии с вариантом.

Программа должна содержать следующие функции:

1 Инициализация элементов двумерного массива числами, вводимыми

с клавиатуры;

2 Инициализация элементов двумерного массива случайными числами;

3 Вывод двумерного массива на экран;

4 Сортировка двумерного массива по строкам;

4.1 Сортировка в соответствии с вариантом;

4.2 Собственная реализация быстрой сортировки;

4.3 Сортировка встроенной функцией сортировки;

4.4 Сравнение времени работы сортировок;

5 Поиск значения в строке массива;

4.1 Последовательный поиск;

4.2 Бинарный поиск;

4.3 Фибоначчиев поиск;

4.4 Интерполяционный поиск;

4.5 Поиск встроенной функцией

4.4 Сравнение времени алгоритмов поиска;

6 Решение задачи уровня В;

7 Решение задачи уровня С (опционально);

8 Выход из программы.

Реализовать заданные методы поиска значений в числовом двумерном

массиве в соответствии с индивидуальным заданием. Оценить время работы

каждого алгоритма поиска.

Требуется определить двумерный массив целых чисел a (например,

размера 30), заполнить его случайными числами (в диапазоне от A до B) или

ввести его элементы с клавиатуры.

B5.

Квадратный

непересекающихся прямоугольников в квадрате, причем каждый

прямоугольник заполнен своим числом, отличным от нуля, а промежутки

между ними заполнены нулями. Определить количество прямоугольников их

общую площадь.

Код:(Здесь приведен не весь код, а только важные части)

static int[][] fillByHand (int[][]mass)  
{  
 int str,stb;//строчки и столбцы, ввод матрицы в ручную  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Число строк =");  
 str=in.nextInt();  
 System.*out*.print("Число столбцов =");  
 stb=in.nextInt();  
 mass= new int[str][stb];  
 for (int i = 0; i<mass.length;i++)  
 {  
 for (int j=0;j<mass[i].length;j++){  
 System.*out*.println("Введите элемент матрицы:");  
 mass[i][j]=in.nextInt();  
 }  
 }  
 return mass;  
}  
static int[][] fillByHandCase6(int[][]mass)  
{  
 int str;//строчки и столбцы, ввод матрицы в ручную  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Задайте размер матрицы :");  
 str=in.nextInt();  
 mass= new int[str][str];  
 for (int i = 0; i<mass.length;i++)  
 {  
 for (int j=0;j<mass[i].length;j++){  
 System.*out*.print("Введите элемент матрицы :"+'('+"Str"+j+"Stb"+i+')');  
 mass[i][j]=in.nextInt();  
 }  
 }  
 return mass;  
}  
public static int[][] FillByRandom(int string, int colons){  
 Scanner in = new Scanner(System.*in*);  
 Random rand = new Random();  
 int [][] matrix = new int[string][colons];  
 System.*out*.print("Введите минимальное значение: ");  
 int left = in.nextInt();  
 int right = left-1;  
 while (right < left){  
 System.*out*.print("Введите максимальное значение: ");  
 right = in.nextInt();  
 if (right < left){  
 System.*out*.println("Минимальное не может быть больше максимального.");  
 }  
 }  
 for (int i = 0; i< matrix.length;i++){  
 for (int j = 0; j < matrix[i].length;j++){  
 matrix[i][j] = rand.nextInt(right - left+1) + left ;  
 }  
 }  
 return matrix;  
}  
  
static int[][] fillByRandcase6 (int[][]mass,int min, int max)  
{  
 int str;  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Задайте размер матрицы :");  
 str=in.nextInt();  
 mass= new int[str][str];  
 for (int i = 0; i<mass.length;i++)  
 {  
 for (int j=0;j<mass[i].length;j++){  
 mass[i][j]=ThreadLocalRandom.*current*().nextInt(min, max + 1);  
 }  
 }  
 return mass;  
}  
static int [] BubbleSort(int [] mass){//сортировка пузырьком  
 int buf = 0;  
 boolean swap;  
 for (int i = 0; i < mass.length; i++) {  
 swap= false;  
 for (int j = 1;j < (mass.length-i); j++) {  
 if (mass[j-1]>mass[j])  
 {  
 buf=mass[j];  
 mass[j]=mass[j-1];  
 mass[j-1]=buf;  
 swap= true;  
 }  
 }  
 if (swap==false) {  
 break;}  
 }  
 return mass;  
}  
static void QuickSort(int [] mass, int left , int right){//быстрая сортировка  
 if (mass.length==0) {  
 return;  
 }  
 if (left>=right) {  
 return;  
 }  
 int mid= left +(right- left)/2;  
 int op = mass[mid];  
 int i= left, j=right;  
 while(i<=j)  
 {  
 while (mass[i]<op){  
 i++;  
 }  
 while(mass[j]>op){  
 j--;  
 }  
 if (i<=j){  
 int buf= mass[i];  
 mass[i]=mass[j];  
 mass[j]=buf;  
 i++;  
 j--;  
 }  
 if (left<j){  
 *QuickSort*(mass,left,j);  
 }  
 if (right>i){  
 *QuickSort*(mass,i,right);  
 }  
 }  
}  
public static int binarySearch(int mass9[], int elementToSearch) {  
  
 int firstIndex = 0;  
 int lastIndex = mass9.length - 1;  
 while(firstIndex <= lastIndex) {  
 int middleIndex = (firstIndex + lastIndex) / 2;  
 if (mass9[middleIndex] == elementToSearch) {  
 return middleIndex;  
 }  
 else if (mass9[middleIndex] < elementToSearch)  
 firstIndex = middleIndex + 1;  
 else if (mass9[middleIndex] > elementToSearch)  
 lastIndex = middleIndex - 1;  
 }  
 return -1;  
}  
public static int interpolationSearch(int[] arr, int target) {  
 int low = 0;  
 int high = arr.length - 1;  
  
 while (low <= high && target >= arr[low] && target <= arr[high]) {  
 // Calculate the position with interpolation formula  
 int pos = low + ((target - arr[low]) \* (high - low)) / (arr[high] - arr[low]);  
  
 if (arr[pos] == target)  
 return pos;  
 else if (arr[pos] < target)  
 low = pos + 1;  
 else  
 high = pos - 1;  
 }  
  
 return -1;  
}  
  
public static void fibonacci(int[] f){  
 f[0]=0;  
 f[1]=1;  
 for(int i=2;i<f.length;++i){  
 f[i]=f[i-1]+f[i-2];  
 }  
  
}  
public static int FibonacInsearch(int[] mass, int x){  
 int left=0, right=mass.length-1;  
 int k=0;  
 int FIB\_MAX = 20;  
 int[] f = new int[FIB\_MAX];  
 *fibonacci*(f);  
  
 while(mass.length>f[k]-1){  
 k++;  
 }  
  
 int[] tmp = new int[f[k]-1];  
 System.*arraycopy* (mass, 0, tmp, 0, mass.length); // Копируем элемент в tmp  
 for (int i = mass.length; i <f [k] -1; ++ i) {// Значения после справа такие же  
 tmp[i]=mass[right];  
 }  
  
 while(left<=right){  
 int mid = left+f[k-1]-1;  
 if(x<mass[mid]){  
 right=mid-1;  
 k-=1;  
 }  
 else if(x>mass[mid]){  
 left=mid+1;  
 k-=2;  
 }  
 else{  
 if(mid<mass.length)  
 return mid;  
 else // находим x в расширении и возвращаем последний нижний индекс  
 return mass.length-1;  
  
 }  
 }  
 return -1;  
  
}  
private static Integer[] append(Integer[] arr, int element) {  
 List<Integer> list = new ArrayList<>(Arrays.*asList*(arr));  
 list.add(element);  
  
 return list.toArray(new Integer[0]);  
}  
public static int findIndex(int[] arr, int num) {  
 for (int i = 0; i < arr.length; i++) {  
 if (arr[i] == num) {  
 return i;  
 }  
 }  
 return -1;  
}

Integer arr[] = new Integer[0];  
do {  
 System.*out*.println("Создать матрицу вручную.");  
 System.*out*.println("Создать матрицу рандомно.");  
 System.*out*.println("Решение задачи B.");  
 System.*out*.println("Выход.");  
 System.*out*.print("Пункт меню :");  
 localans2 = in.nextInt();  
 switch (localans2) {  
 case 1:  
 array = *fillByHandCase6*(array);  
 break;  
 case 2:  
 System.*out*.print("Задайте диапазон.");  
 System.*out*.print("Левая граница = ");  
 int min=in.nextInt();  
 System.*out*.print("Правая граница = ");  
 int max= in.nextInt();  
 array = *fillByRandcase6*(array,min,max);  
 for (int i = 0; i < array.length; i++) {  
 for (int j = 0; j < array[i].length; j++) {  
 System.*out*.print(" "+array[i][j]);  
 }  
 System.*out*.println(" ");  
 }  
 break;  
 case 3:  
 // создана квадратная матрица и необходимо найти число прямоугольников и их общую площадь.  
 int k = 0;  
 int o = 0;  
 int l = 0;  
 boolean flag1 = false;  
 arr = *append*(arr, 0);  
 for (int p = 0; p < array.length; ) {  
 for (; l < array.length; l++) {  
 arr = *append*(arr, array[p][l]);  
 if (l == array.length - 1) {  
 if (p < array.length - 1) {  
 l = -1;  
 p++;  
 } else  
 flag1 = true;  
 }  
 }  
 if (flag1 == true)  
 break;  
 }  
 int[] arrs = new int[arr.length];  
 for (int i = 0; i < arr.length; i++) {  
 arrs[i] = arr[i];  
 }  
 arrs = Arrays.*stream*(arrs).distinct().toArray();  
 Arrays.*sort*(arrs);  
 int q = 1;  
 int count = 0;  
 int count1=0;  
 int S=0;  
 boolean flag = false;  
 int[][] array1 = array.clone();  
 for (int i = 0; i < array.length; ) {  
 for (int n = 0; n < array.length; ) {  
 boolean b = true;  
 boolean b1 = true;  
 boolean b2 = true;  
 boolean b3 = true;  
 boolean b4 = true;  
 boolean b5 = true;  
 boolean b6 = true;  
 int array1length = 0;  
 Integer[] ai = new Integer[0];  
 ai = *append*(ai, 0);  
 Integer[] an = new Integer[0];  
 an = *append*(an, 0);  
 Integer[] an2 = new Integer[0];  
 an2 = *append*(an2, 0);  
 Integer[] ai2 = new Integer[0];  
 ai2 = *append*(ai2, 0);  
 if(count1<1) {  
 if (arrs[q] != array[i][n]) {  
 do {  
 for (int i1 = 0; i1 < array.length; ) {  
 for (int n1 = 0; n1 < array.length; ) {  
 if (array1[i1][n1] != arrs[q]) {  
 n1++;  
 if (array1[i1][n1] != arrs[q] && n1 == array1.length - 1) {  
 i1++;  
 n1 = 0;  
 if (array1[i1][n1] == arrs[q])  
 b5 = false;  
 }  
 if (array1[i1][n1] == arrs[q]) {  
 b5 = false;  
 n = n1;  
 i = i1;  
 array1[i][n] = 0;  
 break;  
 }  
 }  
 }  
 if (b5 == false)  
 break;  
 }  
 }  
 while (b5);  
 }  
 }  
 if ((n < array.length - 1) && (arrs[q] == array[i][n + 1])) {  
 do {  
 array1[i][n] = 0;  
 n++;  
 an = *append*(an, n);  
 an2 = *append*(an2, n);  
 array1[i][n] = 0;  
 if (n == (array.length - 1) || arrs[q] != array[i][n + 1])  
 b = false;  
 }  
 while (b);  
 }  
 if (i < array.length - 1) {  
 if (arrs[q] == array[i + 1][n]) {  
 do {  
 i++;  
 ai = *append*(ai, i);  
 ai2 = *append*(ai2, i);  
 an = *append*(an, n);  
 an2 = *append*(an2, n);  
 array1[i][n] = 0;  
 if (i % 2 != 0&&n>1) {  
 do {  
 if (n > 0 && arrs[q] == array[i][n - 1]) {  
 n--;  
 an = *append*(an, n);  
 an2 = *append*(an2, n);  
 array1[i][n] = 0;  
 if ((n > 0 && arrs[q] != array[i][n - 1]) || n == 0) {  
 b1 = false;  
 }  
 } else {  
 b1 = false;  
 }  
 }  
 while (b1);  
 }  
 if (i % 2 == 0&&n>1) {  
 do {  
 if ((n < (array.length - 1)) && arrs[q] == array[i][n + 1]) {  
 n++;  
 an = *append*(an, n);  
 an2 = *append*(an2, n);  
 array1[i][n] = 0;  
 if ((n < array.length - 1 && arrs[q] != array[i][n + 1]) || n == (array.length - 1)) {  
 b2 = false;  
 }  
 } else {  
 b2 = false;  
 }  
 }  
 while (b2);  
 }  
 if ((i == (array.length - 1) || arrs[q] != array[i + 1][n]))  
 b3 = false;  
 }  
 while (b3);  
 }  
 }  
 int[] ai1 = new int[ai.length];  
 int[] an1 = new int[an.length];  
 for (int w = 0; w < ai.length; w++) {  
 ai1[w] = ai[w];  
 }  
 for (int v = 0; v < an.length; v++) {  
 an1[v] = an[v];  
 }  
 if ((ai1.length \* ai.length) != (an1.length)) {  
 count++;  
 if(ai1.length==an1.length) {  
 S += (ai1.length);  
 }  
 else{  
 S+= ai1.length\*(an1.length/ ai1.length);  
 }  
 }  
 if(ai1.length==1&&an1.length==1){  
 array1[i][n] = 0;  
 }  
 boolean proverka = true;  
 for (int i2 = 0; i2 < array1.length; ) {  
 for (int n2 = 0; n2 < array1.length; ) {  
 if (arrs[q] != array1[i2][n2]) {  
 do {  
 if (n2 < array1.length - 1) {  
 n2++;  
 }  
 if (arrs[q] != array1[i2][n2] && n2 == array1.length - 1) {  
 if (i2 < array1.length - 1) {  
 i2++;  
 n2 = 0;  
 } else {  
 proverka = false;  
 }  
 }  
 if (arrs[q] == array1[i2][n2]) {  
 proverka = false;  
 array1length = 1;  
 count1++;  
 i = i2;  
 n = n2;  
 array1[i][n] = 0;  
 }  
 if (n2 == (array1.length - 1) && i2 == array1.length - 1) {  
 proverka = false;  
 }  
 }  
 while (proverka);  
 }  
 if (proverka == false)  
 break;  
 }  
 if (proverka == false)  
 break;  
 }  
 if (array1length == 0) {  
 q++;  
 i = 0;  
 n = 0;  
 count1=0;  
 }  
 if (q == (arrs.length)) {  
 flag = true;  
 break;  
 }  
 }  
 if (flag == true) {  
 break;  
 }  
 }  
 System.*out*.println("Количество прямоугольников:"+count);  
 System.*out*.println("Общая площадь всех прямоугольников:"+S);  
 System.*out*.println("hasdhdoa");  
 break;  
 case 4:  
 break;  
 }  
}  
while (localans2!=4);

Результат:

1.Задать матрицу руками.

2.Задать матрицу рандомно.

3.Вывести матрицу в консоль.

4.Сортировка матрицы по строкам.

5.Поиск значения в строке массива.

6.Решение задачи уровня В.

7.Решение задачи уровня С.

8.Выход из программы.

Пункт меню: 2

Введите число строк: 3

Введите число столбцов:

4

Введите минимальное значение: 1

Введите максимальное значение: 101

Пункт меню: 3

91 25 37 100

54 79 71 74

84 43 89 85

Пункт меню: 4

1.Сортировка пузырьком.

2.Быстрая сортировка.

3.Сортировка встроенной функцией.

4.Сравнение времени сортировок.

Пункт меню сортировок: 1

SortTime(buble)=0

BubleSort- complete

25 37 91 100

54 71 74 79

43 84 85 89

Пункт меню сортировок: 2

SortTime(quick)=0

QuickSort- complete

25 37 91 100

54 71 74 79

43 84 85 89

Пункт меню сортировок: 3

SortTime(funcInJava)=0

Sort- complete

25 37 91 100

54 71 74 79

43 84 85 89

1.Задать матрицу руками.

2.Задать матрицу рандомно.

3.Вывести матрицу в консоль.

4.Сортировка матрицы по строкам.

5.Поиск значения в строке массива.

6.Решение задачи уровня В.

7.Решение задачи уровня С.

8.Выход из программы.

Пункт меню: 5

1.Последовательный поиск.

2.Бинарный поиск.

3.Интерполяционный поиск.

4.Фибоначчиев поиск.

5.Поиск встроенной функцией.

6.Сравнение времени работы алгоритмов поиска.

Пункт меню поиска: 1

Введите элемент, который хотите найти: 12

Введенный вами элемент не найден.

Пункт меню поиска: 2

Строка:1|Столбец:4

Пункт меню поиска: 3

Введите элемент который хотите найти : 100

Строка:1|Столбец:4

Пункт меню поиска: 4

Введите элемент который хотите найти: 100

Строка:1|Столбец:4

1.Задать матрицу руками.

2.Задать матрицу рандомно.

3.Вывести матрицу в консоль.

4.Сортировка матрицы по строкам.

5.Поиск значения в строке массива.

6.Решение задачи уровня В.

7.Решение задачи уровня С.

8.Выход из программы.

Пункт меню: 6

Создать матрицу вручную.

Создать матрицу рандомно.

Решение задачи B.

Выход.

Пункт меню :1

Задайте размер матрицы:

3

Введите элемент матрицы :(Str0Stb0)1

Введите элемент матрицы :(Str1Stb0)1

Введите элемент матрицы :(Str2Stb0)1

Введите элемент матрицы :(Str0Stb1)1

Введите элемент матрицы :(Str1Stb1)1

Введите элемент матрицы :(Str2Stb1)1

Введите элемент матрицы :(Str0Stb2)2

Введите элемент матрицы :(Str1Stb2)2

Введите элемент матрицы :(Str2Stb2)3

Создать матрицу вручную.

Создать матрицу рандомно.

Решение задачи B.

Выход.

Пункт меню :3

Количество прямоугольников:2

Общая площадь всех прямоугольников:8

Вывод:

В ходе лабораторной работы было получено понимание, как работать с матрицами на языке программирования Java, а именно:

1. Как создавать матрицу
2. Как построчно сортировать матрицу
3. Как искать значение в матрице
4. Как работать с определенного вида матрицами

Также было получено лучшее понимание того, как работать с циклами (do/while) и просто (while).