Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени федеральное государственное

бюджетное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и

информационных технологий

Лабораторная работа №2

на тему: «Методы поиска»

Выполнила:

Студентка группы БФИ1902

Кареева В.В.

Вариант 5

Проверил:

Москва, 2021 г.

Оглавление

[1. Цель работы 3](#_Toc72408691)

[2. Задание на лабораторную работу 3](#_Toc72408692)

[2.1. Задание №1: 3](#_Toc72408693)

[2.2. Задание №2: 3](#_Toc72408694)

[2.3. Задание № 3: 3](#_Toc72408695)

[3. Листинг программы 3](#_Toc72408696)

[4. Результат работы программы 9](#_Toc72408697)

[Список использованных источников 11](#_Toc72408698)

# **1. Цель работы**

Цель работы: рассмотреть и изучить основные методы поиска.

# **2. Задание на лабораторную работу**

Реализовать методы поиска в соответствии с заданием. Организовать генерацию начального набора случайных данных. Для всех вариантов добавить реализацию добавления, поиска и удаления элементов. Оценить время работы каждого алгоритма поиска и сравнить его со временем работы стандартной функции поиска, используемой в выбранном языке программирования.

## 2.1. Задание №1:

|  |  |  |  |
| --- | --- | --- | --- |
| Бинарный поиск | Бинарное дерево | Фибоначчиев | Интерполяционный |

## 2.2. Задание №2:

|  |  |  |
| --- | --- | --- |
| Простое рехэширование | Рехэширование с помощью  псевдослучайных чисел | Метод цепочек |

## 2.3. Задание № 3:

Расставить на стандартной 64-клеточной шахматной доске 8 ферзей так, чтобы ни один из них не находился под боем другого». Подразумевается, что ферзь бьёт все клетки, расположенные по вертикалям, горизонталям и обеим диагоналям

Написать программу, которая находит хотя бы один способ решения задач.

# **3. Листинг программы**

import java.util.Scanner;  
public class binary {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.println("Введите размер масива:");  
 String n1 = scanner.nextLine();  
 System.out.println("Введите минимальное число масива:");  
 String min\_lim1 = scanner.nextLine();  
 System.out.println("Введите максимальное число масива:");  
 String max\_lim1 = scanner.nextLine();  
 if (n1.equals(""))  
 n1 = "50";  
 if (min\_lim1.equals(""))  
 min\_lim1 = "-250";  
 if (max\_lim1.equals(""))  
 max\_lim1 = "1013";  
 int n = Integer.parseInt(n1);  
 int min\_lim = Integer.parseInt(min\_lim1);  
 int max\_lim = Integer.parseInt(max\_lim1);  
 int[] arr = new int[n];  
 System.out.println("Исходный массив:");  
 for (int i = 0; i < n; i++) {  
 arr[i] = (int) ((Math.random() \* (max\_lim - min\_lim)) + min\_lim);  
 System.out.print(arr[i] + "\t");  
 }  
 System.out.println();  
 boolean needIteration = true;  
 while (needIteration) {  
 needIteration = false;  
 for (int i = 1; i < n; i++) {  
 if (arr[i] < arr[i - 1]) {  
 int tmp = arr[i];  
 arr[i] = arr[i - 1];  
 arr[i - 1] = tmp;  
 needIteration = true;  
 }  
 }  
 }  
 System.out.println("Отсортированный массив");  
 for (int i=0;i<n;i++){  
 System.out.print(arr[i]+" ");  
 }  
 System.out.println();  
 System.out.println("Введите элемент для поиска");  
 int item = scanner.nextInt();  
 binarySearch(arr, 0, n-1, item);  
 }  
 public static void binarySearch(int[] array, int first, int last, int item) {  
 int position;  
 int comparisonCount = 1; // для подсчета количества сравнений  
  
 // для начала найдем индекс среднего элемента массива  
 position = (first + last) / 2;  
  
 while ((array[position] != item) && (first <= last)) {  
 comparisonCount++;  
 if (array[position] > item) { // если число заданного для поиска меньше текущего  
 last = position - 1; // уменьшаем позицию на 1.  
 } else {  
 first = position + 1; // иначе увеличиваем на 1  
 }  
 position = (first + last) / 2;  
 }  
 if (first <= last) {  
 System.out.println(item + " является " + ++position + " элементом в массиве");  
 System.out.println("Метод бинарного поиска нашел число после " + comparisonCount +  
 " сравнений");  
 } else {  
 System.out.println("Элемент не найден в массиве. Метод бинарного поиска закончил работу после "  
 + comparisonCount + " сравнений");  
 }  
 }  
}

import java.util.Scanner;  
import java.util.Stack;  
  
public class Binary\_Tree {  
 public static void main(String[] args) {  
 Tree tree = new Tree();  
 Scanner scanner = new Scanner(System.in);  
 System.out.println("Введите размер масива:");  
 String n1 = scanner.nextLine();  
 System.out.println("Введите минимальное число масива:");  
 String min\_lim1 = scanner.nextLine();  
 System.out.println("Введите максимальное число масива:");  
 String max\_lim1 = scanner.nextLine();  
 if (n1.equals(""))  
 n1 = "50";  
 if (min\_lim1.equals(""))  
 min\_lim1 = "-250";  
 if (max\_lim1.equals(""))  
 max\_lim1 = "1013";  
 int n = Integer.parseInt(n1);  
 int min\_lim = Integer.parseInt(min\_lim1);  
 int max\_lim = Integer.parseInt(max\_lim1);  
 int[] arr = new int[n];  
  
 System.out.println("Исходный массив:");  
 for (int i = 0; i < n; i++) {  
 arr[i] = (int) ((Math.random() \* (max\_lim - min\_lim)) + min\_lim);  
 System.out.print(arr[i] + "\t");  
 }  
 System.out.println();  
 System.out.println("Введите элемент для поиска:");  
 int item = scanner.nextInt();  
 System.out.println();  
 for (int i=0;i<n;i++){  
 tree.insertNode(arr[i]);  
 }  
 // отображение дерева:  
 tree.printTree();  
 // находим узел по значению и выводим его в консоли  
 Node foundNode = tree.findNodeByValue(item);  
 if (foundNode==null){  
 System.out.println("Элемента нет в дереве");  
 } else {  
 foundNode.printNode();  
 }  
 }  
}  
class Tree {  
 private Node rootNode; // корневой узел  
  
 public Tree() { // Пустое дерево  
 rootNode = null;  
 }  
  
 public Node findNodeByValue(int value) { // поиск узла по значению  
 Node currentNode = rootNode; // начинаем поиск с корневого узла  
 while (currentNode.getValue() != value) { // поиск покуда не будет найден элемент или не будут перебраны все  
 if (value < currentNode.getValue()) { // движение влево?  
 currentNode = currentNode.getLeftChild();  
 } else { //движение вправо  
 currentNode = currentNode.getRightChild();  
 }  
 if (currentNode == null) { // если потомка нет,  
 return null; // возвращаем null  
 }  
 }  
 return currentNode; // возвращаем найденный элемент  
 }  
  
 public void insertNode(int value) { // метод вставки нового элемента  
 Node newNode = new Node(); // создание нового узла  
 newNode.setValue(value); // вставка данных  
 if (rootNode == null) { // если корневой узел не существует  
 rootNode = newNode;// то новый элемент и есть корневой узел  
 }  
 else { // корневой узел занят  
 Node currentNode = rootNode; // начинаем с корневого узла  
 Node parentNode;  
 while (true) // мы имеем внутренний выход из цикла  
 {  
 parentNode = currentNode;  
 if(value == currentNode.getValue()) { // если такой элемент в дереве уже есть, не сохраняем его  
 return; // просто выходим из метода  
 }  
 else if (value < currentNode.getValue()) { // движение влево?  
 currentNode = currentNode.getLeftChild();  
 if (currentNode == null){ // если был достигнут конец цепочки,  
 parentNode.setLeftChild(newNode); // то вставить слева и выйти из методы  
 return;  
 }  
 }  
 else { // Или направо?  
 currentNode = currentNode.getRightChild();  
 if (currentNode == null) { // если был достигнут конец цепочки,  
 parentNode.setRightChild(newNode); //то вставить справа  
 return; // и выйти  
 }  
 }  
 }  
 }  
 }  
  
  
  
 public void printTree() { // метод для вывода дерева в консоль  
 Stack globalStack = new Stack(); // общий стек для значений дерева  
 globalStack.push(rootNode);  
 int gaps = 32; // начальное значение расстояния между элементами  
 boolean isRowEmpty = false;  
 String separator = "-----------------------------------------------------------------";  
 System.out.println(separator);// черта для указания начала нового дерева  
 while (isRowEmpty == false) {  
 Stack localStack = new Stack(); // локальный стек для задания потомков элемента  
 isRowEmpty = true;  
  
 for (int j = 0; j < gaps; j++)  
 System.out.print(' ');  
 while (globalStack.isEmpty() == false) { // покуда в общем стеке есть элементы  
 Node temp = (Node) globalStack.pop(); // берем следующий, при этом удаляя его из стека  
 if (temp != null) {  
 System.out.print(temp.getValue()); // выводим его значение в консоли  
 localStack.push(temp.getLeftChild()); // соохраняем в локальный стек, наследники текущего элемента  
 localStack.push(temp.getRightChild());  
 if (temp.getLeftChild() != null ||  
 temp.getRightChild() != null)  
 isRowEmpty = false;  
 }  
 else {  
 System.out.print("\_\_");// - если элемент пустой  
 localStack.push(null);  
 localStack.push(null);  
 }  
 for (int j = 0; j < gaps \* 2 - 2; j++)  
 System.out.print(' ');  
 }  
 System.out.println();  
 gaps /= 2;// при переходе на следующий уровень расстояние между элементами каждый раз уменьшается  
 while (localStack.isEmpty() == false)  
 globalStack.push(localStack.pop()); // перемещаем все элементы из локального стека в глобальный  
 }  
 System.out.println(separator);// подводим черту  
 }  
}

import java.util.Scanner;  
  
public class Fibonachy {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.println("Введите размер масива:");  
 String n1 = scanner.nextLine();  
 System.out.println("Введите минимальное число масива:");  
 String min\_lim1 = scanner.nextLine();  
 System.out.println("Введите максимальное число масива:");  
 String max\_lim1 = scanner.nextLine();  
 if (n1.equals(""))  
 n1 = "50";  
 if (min\_lim1.equals(""))  
 min\_lim1 = "-250";  
 if (max\_lim1.equals(""))  
 max\_lim1 = "1013";  
 int n = Integer.parseInt(n1);  
 int min\_lim = Integer.parseInt(min\_lim1);  
 int max\_lim = Integer.parseInt(max\_lim1);  
 int[] arr = new int[n];  
 System.out.println("Исходный массив:");  
 for (int i = 0; i < n; i++) {  
 arr[i] = (int) ((Math.random() \* (max\_lim - min\_lim)) + min\_lim);  
 System.out.print(arr[i] + "\t");  
 }  
 System.out.println();  
 boolean needIteration = true;  
 while (needIteration) {  
 needIteration = false;  
 for (int i = 1; i < n; i++) {  
 if (arr[i] < arr[i - 1]) {  
 int tmp = arr[i];  
 arr[i] = arr[i - 1];  
 arr[i - 1] = tmp;  
 needIteration = true;  
 }  
 }  
 }  
 System.out.println("Отсортированный массив:");  
 for (int i = 0; i < n; i++) {  
 System.out.print(arr[i] + "\t");  
 }  
 System.out.println();  
 System.out.println("Введите элемент для поиска:");  
 int item = scanner.nextInt();  
 System.out.println();  
 FibonachySearch F = new FibonachySearch();  
 int index = F.search(arr,item);  
 System.out.println("Индекс найденного элемента:");  
 System.out.println(index);  
 }  
 public static class FibonachySearch {  
 private int i;  
 private int p;  
 private int q;  
 private boolean stop = false;  
 private void init(int[] arr){  
 stop = false;  
 int k = 0;  
 int n = arr.length;  
 for(; getFibonachyNumber(k+1) < n+1;){  
 k +=1;  
 }  
 int m = getFibonachyNumber(k+1)-(n+1);  
 i = getFibonachyNumber(k) - m;  
 p = getFibonachyNumber(k-1);  
 q = getFibonachyNumber(k-2);  
 }  
  
 public int getFibonachyNumber(int k){  
 int firstNumber = 0;  
 int secondNumber = 1;  
 for (int i = 0;i<k;i++){  
 int temp = secondNumber;  
 secondNumber += firstNumber;  
 firstNumber = temp;  
 }  
 return firstNumber;  
 }  
  
 private void upIndex(){  
 if (p==1)  
 stop = true;  
 i = i + q;  
 p = p - q;  
 q = q - p;  
 }  
  
 private void downIndex(){  
 if (q==0)  
 stop = true;  
 i = i - q;  
 int temp = q;  
 q = p - q;  
 p = temp;  
 }  
  
 public int search(int[] arr,int element){  
 init(arr);  
 int n = arr.length;  
 int resIn = -1;  
 for (; !stop;){  
 if (i < 0){  
 upIndex();  
 }  
 else if (i>=n){  
 downIndex();  
 }  
 else if (arr[i]==element){  
 resIn = i;  
 break;  
 }  
 else if (element <arr[i]){  
 downIndex();  
 }  
 else if (element > arr[i])  
 {  
 upIndex();  
 }  
 }  
 return resIn;  
 }  
 }  
  
}

import java.util.Scanner;  
  
public class interpolation {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.println("Введите размер масива:");  
 String n1 = scanner.nextLine();  
 System.out.println("Введите минимальное число масива:");  
 String min\_lim1 = scanner.nextLine();  
 System.out.println("Введите максимальное число масива:");  
 String max\_lim1 = scanner.nextLine();  
 if (n1.equals(""))  
 n1 = "50";  
 if (min\_lim1.equals(""))  
 min\_lim1 = "-250";  
 if (max\_lim1.equals(""))  
 max\_lim1 = "1013";  
 int n = Integer.parseInt(n1);  
 int min\_lim = Integer.parseInt(min\_lim1);  
 int max\_lim = Integer.parseInt(max\_lim1);  
 int[] arr = new int[n];  
  
 System.out.println("Исходный массив:");  
 for (int i = 0; i < n; i++) {  
 arr[i] = (int) ((Math.random() \* (max\_lim - min\_lim)) + min\_lim);  
 System.out.print(arr[i] + "\t");  
 }  
 boolean needIteration = true;  
 while (needIteration) {  
 needIteration = false;  
 for (int i = 1; i < n; i++) {  
 if (arr[i] < arr[i - 1]) {  
 int tmp = arr[i];  
 arr[i] = arr[i - 1];  
 arr[i - 1] = tmp;  
 needIteration = true;  
 }  
 }  
 }  
 System.out.println();  
 System.out.println("Отсортированный массив");  
 for (int i=0;i<n;i++){  
 System.out.print(arr[i]+" ");  
 }  
 System.out.println();  
 System.out.println("Введите элемент для поиска");  
 int item = scanner.nextInt();  
 System.out.println("Индекс найденного элемента:"+interpolationSearch(arr,item));  
 }  
 public static int interpolationSearch(int[] integers, int elementToSearch) {  
  
 int startIndex = 0;  
 int lastIndex = (integers.length - 1);  
  
 while ((startIndex <= lastIndex) && (elementToSearch >= integers[startIndex]) &&  
 (elementToSearch <= integers[lastIndex])) {  
 // используем формулу интерполяции для поиска возможной лучшей позиции для существующего элемента  
 int pos = startIndex + (((lastIndex-startIndex) /  
 (integers[lastIndex]-integers[startIndex]))\*  
 (elementToSearch - integers[startIndex]));  
  
 if (integers[pos] == elementToSearch)  
 return pos;  
  
 if (integers[pos] < elementToSearch)  
 startIndex = pos + 1;  
  
 else  
 lastIndex = pos - 1;  
 }  
 return -1;  
 }  
}

public class Node {  
 private int value; // ключ узла  
 private Node leftChild; // Левый узел потомок  
 private Node rightChild; // Правый узел потомок  
  
 public void printNode() { // Вывод значения узла в консоль  
 System.out.println(" Выбранный узел имеет значение :" + value);  
 }  
  
 public int getValue() {  
 return this.value;  
 }  
  
 public void setValue(final int value) {  
 this.value = value;  
 }  
  
 public Node getLeftChild() {  
 return this.leftChild;  
 }  
  
 public void setLeftChild(final Node leftChild) {  
 this.leftChild = leftChild;  
 }  
  
 public Node getRightChild() {  
 return this.rightChild;  
 }  
  
 public void setRightChild(final Node rightChild) {  
 this.rightChild = rightChild;  
 }  
  
 @Override  
 public String toString() {  
 return "Node{" +  
 "value=" + value +  
 ", leftChild=" + leftChild +  
 ", rightChild=" + rightChild +  
 '}';  
 }  
}

# **4. Результат работы программы**

Результат работы программы представлен на Рисунках 1-2.

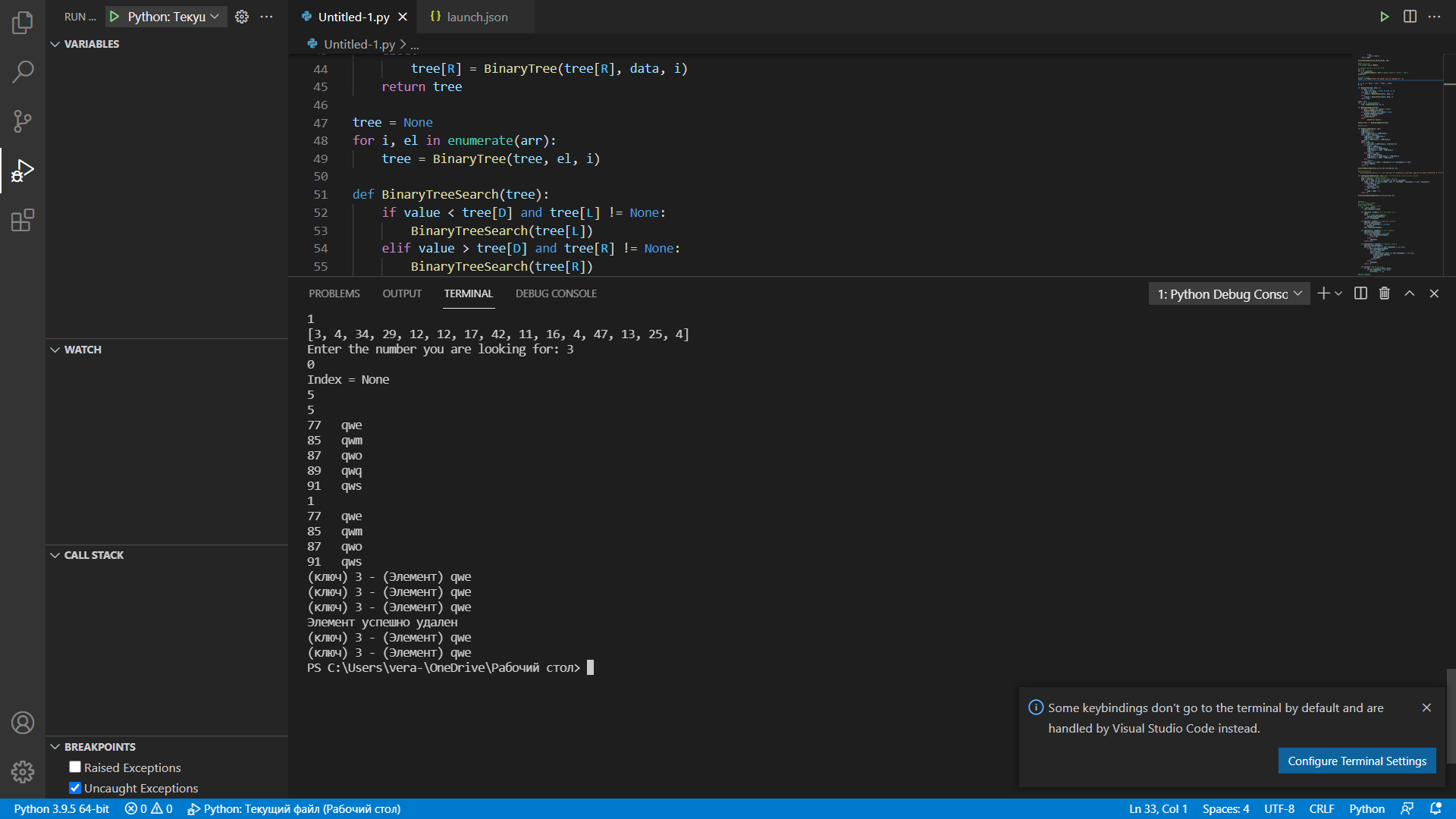


Рисунок 1 - Задание 1 и 2

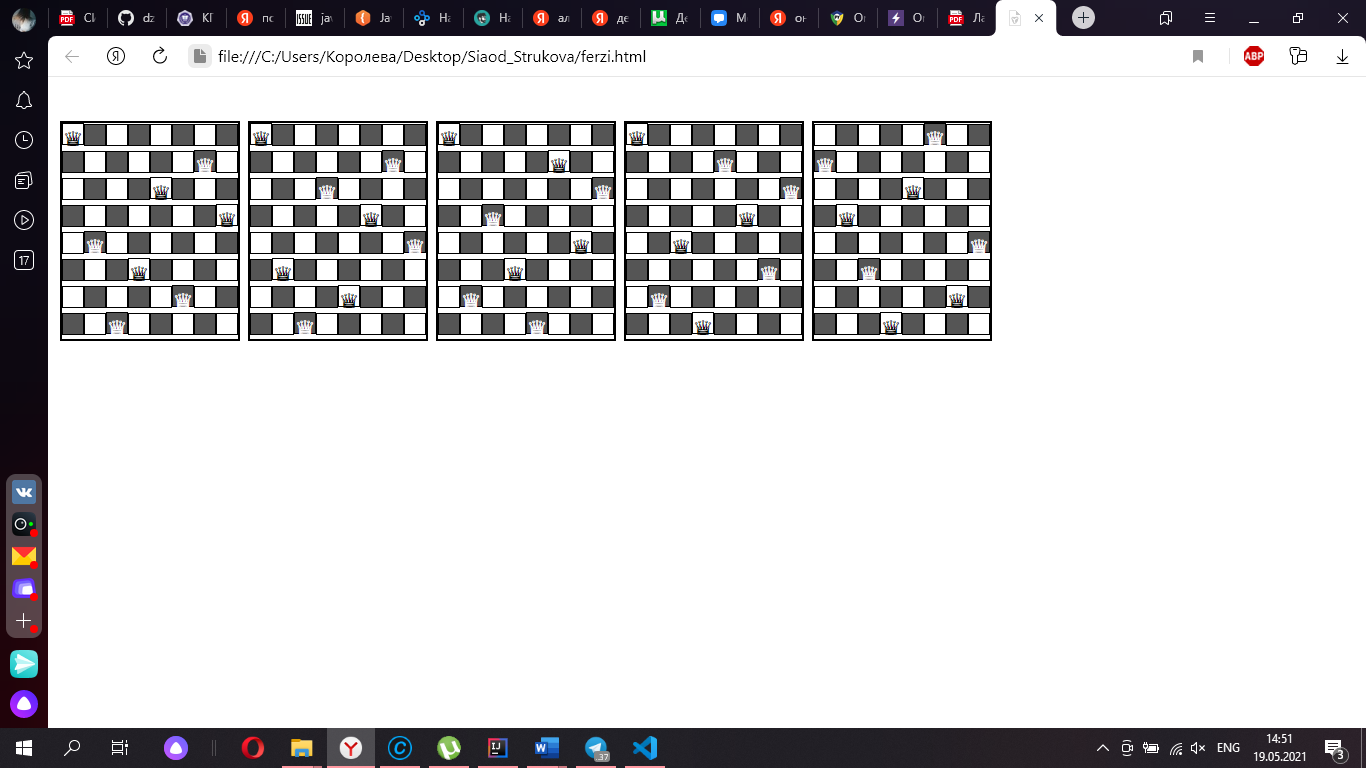


Рисунок 2 - Задание 3

**Вывод:** мы рассмотрели и изучили основные методы поиска, такие как бинарный, Фибоначчиев и тд.

# **Список использованных источников**

1) ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчёт о научно-исследовательской работе. Структура и правила оформления

2) ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления