

Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение высшего образования
“Московский технический университет связи и информатики”

Кафедра: "Математическая кибернетика и информационные
технологии"

Лабораторная работа №2 по дисциплине “Структуры и алгоритмы
обработки данных” по теме “Методы поиска”

Выполнил студент
группы БФИ1902
вариант №9
Крутиков С.С.

г. Москва, 2021

Содержание

1	Задание на лабораторную работу	3
2	Ход выполнения лабораторной работы	3
2.1	Листинги программы	3
2.2	Результаты выполнения программы.....	12
Вывод	17

1 Задание на лабораторную работу

Задание на лабораторную работу представлено на рисунке 1.

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.

Задание №1:

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

Задание №2:

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

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.

Рисунок 1 – Задание на лабораторную работу

2 Ход выполнения лабораторной работы

2.1 Листинги программы

Код в классе Main, в котором реализованы вышеуказанные методы поиска представлен на листинге 1. Также, класс Main использует классы Node, Tree и HashTable, который отвечают за реализацию двоичного дерева и метода цепочек соответственно.

Листинг 1 – Код в классе Main

```
import java.awt.image.AreaAveragingScaleFilter;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

class Main{

    public static ArrayList<Integer> list;
    public static int []arr;
    public static int n = 20;
    public static int min_limit = -250;
    public static int max_limit = 1000 + 9;

    public static void main(String[] args) {
        System.out.println("ЛАБОРАТОРНАЯ РАБОТА №2\nВыполнил студент группы
БФИ1902 Крутиков Степан Сергеевич\n");

        list_gen();

        System.out.println("Введите элемент для поиска");
        Scanner sc = new Scanner(System.in);
        int key = sc.nextInt();

        binarySearch(key);
        binaryTree(key);

        Arrays.sort(arr);
        System.out.print("\nSorted array\n");
        output_arr(arr);

        System.out.println("\nFibonacciSearch");
        FibonachySearch F = new FibonachySearch();
        int index = F.search(arr, key);
        System.out.println(index);

        System.out.println("\nInterpolationSearch");
        System.out.println(interpolationSearch(arr, key));

        HashTable table = new HashTable(7);
        try
        {
            File file = new
File(System.getProperty("user.dir")+"\\src\\input.txt");
            Scanner scanner = new Scanner(file);
            while (scanner.hasNext())
            {
                table.addElement(scanner.next());
            }
            scanner.close();
        }
        catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }

        System.out.println("Введите искомое слово: ");
        Scanner in = new Scanner(System.in);
        String word = in.nextLine();
        table.printHashTable();
    }
}
```

```

        if(table.findElement(word))
        {
            System.out.print("Такое слово есть. " + "Его Хэш: " +
table.hashFunc(word));
        }
        else
            System.out.print("Такого слова нету.");

    }

    public static void list_gen() {
        String var = "";

        Scanner sc = new Scanner(System.in);

        System.out.println("Введите количество элементов или отбейте enter
(стандартное значение = 20)");
        var = sc.nextLine();
        if (!var.equals(""))
            n = Integer.parseInt(var);

        System.out.println("Введите значение min_limit или отбейте enter
(стандартное значение = -250)");
        var = sc.nextLine();
        if (!var.equals(""))
            min_limit = Integer.parseInt(var);

        System.out.println("Введите значение max_limit или отбейте enter
(стандартное значение = 1009)");
        var = sc.nextLine();
        if (!var.equals(""))
            max_limit = Integer.parseInt(var);

        list = new ArrayList<Integer>();
        arr = new int[n];

        System.out.println("\nSource array");

        int temp = 0;

        for (int i = 0; i < n; i++) {
            temp = (int) (Math.random() * (max_limit - min_limit)) +
min_limit;
            list.add(temp);
            arr[i] = temp;
            System.out.print(list.get(i) + " ");
        }
        System.out.println("\n");
    }

    public static void output_list(ArrayList<Integer> list){
        for (int i = 0; i < n; i++) {
            System.out.print(list.get(i) + " ");
        }
    }

    public static void output_arr(int []arr){
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }
    }

```

```

    }
}

public static int findMidPoint(int min, int max){
    return (min+max)/2;
}

public static void binarySearch(int key){
    System.out.println("\nBinarySearch");

    ArrayList<Integer> sortedList = new ArrayList<Integer>(list);
    Collections.sort(sortedList);
    System.out.print("Sorted array\n");
    output_list(sortedList);
    int position;
    int first = 0;
    int last = n - 1;

    position = (first + last) / 2;

    while ((sortedList.get(position) != key) && (first <= last)) {
        if (sortedList.get(position) > key) { // если число заданного
для поиска
            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("\n" + key + " является " + ++position + "
элементом в массиве.\n");
    else
        System.out.println("\n" + "Элемент не найден в массиве.\n");
}

public static void binaryTree(int key){
    System.out.println("Tree search");
    Tree tree = new Tree();
    for (int i = 0; i < n; i++){
        tree.insertNode(list.get(i));
    }
    tree.printTree();
    Node foundNode = tree.findNodeByValue(key);
    if (foundNode == null){
        System.out.println("Такого узла в дереве нет");
    }
    else
        foundNode.printNode();
}

//Фибоначиев поиск
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;
}
}

```

```

        //Интерполяционный поиск
        public static int interpolationSearch(int[] sortedArray, int toFind) {
            // Возвращает индекс элемента со значением toFind или -1, если такого
            // элемента не существует
            int mid;
            int low = 0;
            int high = sortedArray.length - 1;

            while (sortedArray[low] < toFind && sortedArray[high] > toFind) {
                if (sortedArray[high] == sortedArray[low]) // Защита от деления
на 0
                    break;
                mid = low + ((toFind - sortedArray[low]) * (high - low)) /
(sortedArray[high] - sortedArray[low]);

                if (sortedArray[mid] < toFind)
                    low = mid + 1;
                else if (sortedArray[mid] > toFind)
                    high = mid - 1;
                else
                    return mid;
            }

            if (sortedArray[low] == toFind)
                return low;
            if (sortedArray[high] == toFind)
                return high;

            return -1;
        }
    }
}

```

Код в классе Node, который определяет узлы двоичного дерева поиска представлен на листинге 2.

Листинг 2 – Код в классе Node

```

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 +
            '}';
    }
}

```

Код в классе Tree, который отвечает за непосредственно создание двоичного дерева поиска, представлен на листинге 3.

Листинг 3 – Код в классе Tree

```

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 +

```

```

        '}'
    }
}

```

Код в классе HashTable, который отвечает за создание хэш-таблицы для метода цепочек, представлен на листинге 4.

Листинг 4 – Код в HashTable

```

import java.util.ArrayList;

public class HashTable
{
    private int size;
    private ArrayList<String>[] array;

    public HashTable(int number)
    {
        size=number;
        array= new ArrayList[size];
        for (int i=0; i<size; ++i)
            array[i]=new ArrayList<String>();
    }

    public int hashFunc(String str)
    {
        int result=0;
        for( int i=0; i<str.length(); i++)
            result+=(int)str.charAt(i);

        return result%size;
    }

    public void addElement(String str)
    {
        array[hashFunc(str)].add(str);
    }

    public boolean findElement(String str)
    {
        for (int j = 0; j < array[hashFunc(str)].size(); j++)
            if((array[hashFunc(str)].get(j)).equals(str))
                return true;
        return false;
    }

    public void printHashTable()
    {
        System.out.println("Ключ:  значение ");
        for (int i=0; i<size; ++i)
        {
            System.out.print(i + ":  ");
            for (int j = 0; j < array[i].size(); j++)
                System.out.print(array[i].get(j) + " ");
            System.out.println();
        }
    }
}

```

В классе Queen реализуется решение задачи с ферзями. Его код представлен на листинге 5.

Листинг 5 – Код в классе Queen.

```
public class Queen {
    /**
     * размерность доски
     */
    /**
     * хранит расстановку ферзей. каждый ферзь находится на отдельной линии,
на * одной линии находится не могут так как бьют друг друга.
     */
    private int[] state;
    /**
     * Порядковый номер комбинации
     */
    private int index = 1;

    /**
     * n - размерность доски и количество ферзей
     */
    public Queen(int n) {
        state = new int[n];

        for (int i = 0; i < state.length; i++) {
            state[i] = 0;
        }

        /**
         * генерирует следующую комбинацию(расстановку фигур)
         */
        public boolean next() {
            index++;
            return move(8 - 1);
        }

        /**
         * Двигает фигуру в указанной линии на одну клетку вправо и возвращает
true.
         * Если фигура находится в крайнем положении, то фигура устанавливается в
         * первое положение и двигается фигура находящаяся на линии выше и так
далее.
         * Если линий выше не осталось возвращает false.
         */
        private boolean move(int index) {
            if (state[index] < 8 - 1) {
                state[index]++;
                return true;
            }

            state[index] = 0;
            if (index == 0) {
                return false;
            } else {
                return move(index - 1);
            }
        }
    }
}
```

```

    }

    /*
     * Возвращает порядковый номер комбинации, которая в данный момент
     * установлена.
     */
    public int getIndex() {
        return index;
    }

    //проверяем бьет ли наша королева другую фигуру если да то возвращаем фолс
    public boolean isPeace() {
        for (int i = 0; i < state.length; i++) {
            for (int j = i + 1; j < state.length; j++) {
                // бьет ли по вертикали
                if (state[i] == state[j]) {
                    return false;
                }
                // бьет ли по диагонали
                if (Math.abs(i - j) == Math.abs(state[i] - state[j])) {
                    return false;
                }
            }
        }

        return true;
    }

    /*
     * Выводит доску с фигурами.
     */
    public void printState() {
        for (int i = 0; i < state.length; i++) {
            int position = state[i];
            for (int j = 0; j < 8 ; j++) {
                System.out.print(j == position ? 'X' : '_');
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        Queen c = new Queen(8);
        int counter = 0;
        do {
            if (c.isPeace()) {
                counter ++;
                c.printState();
                System.out.println("-----");
            }
        } while (c.next());

        System.out.println("Итого: " + counter);
    }
}

```

2.2 Результаты выполнения программы

После ввода всех необходимых данных для генерации массива и элемента для поиска программа выводит полученный результат поиска различными методами. Это показано на рисунке 2.

```
Run: Main x
Введите количество элементов или отбейте enter (стандартное значение = 20)
5
Введите значение min_limit или отбейте enter (стандартное значение = -250)
0
Введите значение max_limit или отбейте enter (стандартное значение = 1009)
100

Source array
97 64 61 46 66

Введите элемент для поиска
10

BinarySearch
Sorted array
46 61 64 66 97
Элемент не найден в массиве.

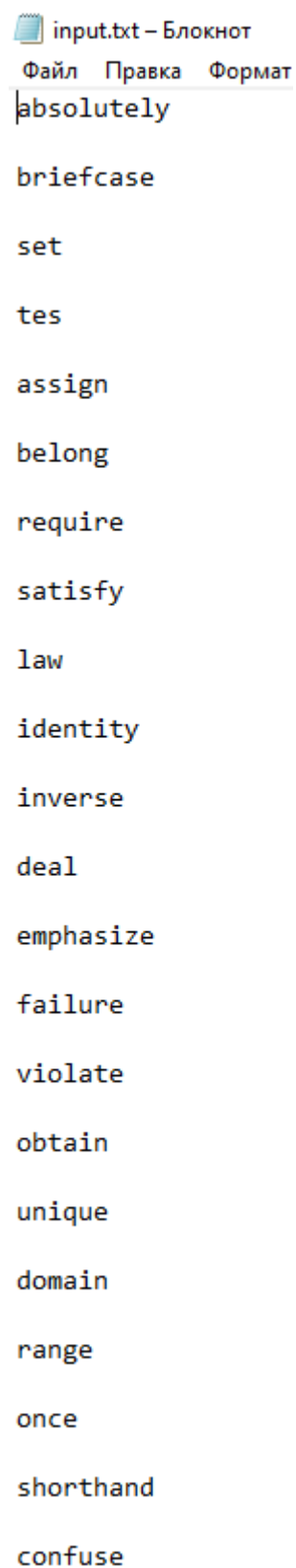
Tree search
-----
                        97
                     /  \
                  64    --
                 /  \
             61    66
            /  \
          46   --
         /  \
      --   --
-----
Такого узла в дереве нет

Sorted array
46 61 64 66 97
FibonacciSearch
-1

InterpolationSearch
-1
```

Рисунок 2 – Результат поиска элемента массива

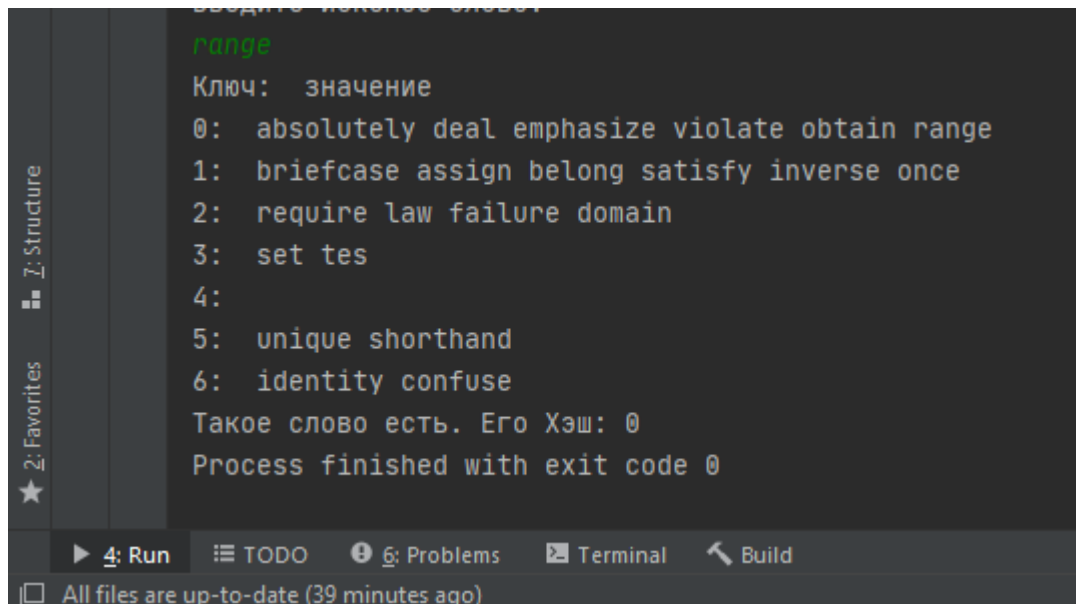
Затем происходит запрос на поиск слова из исходного файла input.txt. Содержание файла представлено на рисунке 3.



input.txt – Блокнот
Файл Правка Формат
absolutely
briefcase
set
tes
assign
belong
require
satisfy
law
identity
inverse
deal
emphasize
failure
violate
obtain
unique
domain
range
once
shorthand
confuse

Рисунок 3 – Содержимое файла input.txt

Я ввел слово range и получил ответ, что такое слово есть. Это показано на рисунке 4.



```
Введите ключевое слово:
range
Ключ: значение
0: absolutely deal emphasize violate obtain range
1: briefcase assign belong satisfy inverse once
2: require law failure domain
3: set tes
4:
5: unique shorthand
6: identity confuse
Такое слово есть. Его Хэш: 0
Process finished with exit code 0
```

Рисунок 4 – Поиск методом цепочек

Задание 3 выполнено в классе Queen. После запуска данного класса программа выводит решение. Это показано на рисунке 5.

Вывод

В результате выполнения данной лабораторной работы были получены навыки реализации основных алгоритмов поиска.

Список использованных источников

- 1 ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления
- 2 ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления