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 №2

Выполнил: студент группы БВТ1901

Кускова А. Е.

Руководитель:

Мелехин А. А.

Москва

2021

Задание:

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.

Решение:

Реализуем заданные методы поиска, используя язык программирования Java.

Задание 1

Код бинарного поиска:

```
import java.util.Arrays;
import java.util.Scanner;

public class BinarySearch {
    public static void main(String[] args) {

        int count, num, item, array[], first, last;

        Scanner input = new Scanner(System.in);
        System.out.println("Введите количество элементов: ");
        num = input.nextInt();

        array = new int[num];

        System.out.println("Введите числа: ");

        for (count = 0; count < num; count++) {
            array[count] = input.nextInt();
        }
        Arrays.sort(array);

        System.out.println("Введите элемент для бинарного поиска: ");
        item = input.nextInt();
        first = 0;
        last = num - 1;

        System.out.println();
        System.out.print("Массив: ");
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
        System.out.println();

         binarySearch(array, first, last, item);
    }

    public static void binarySearch (int[] arr, int first, int last, int
item) {
        int position = (first + last) / 2;
```

```
        while ((arr[position] != item) && (first <= last)) {
            //comparisonCount++;

            if (arr[position] > item) {
                last = position - 1;
            } else {
                first = position + 1;
            }
            position = (first + last) / 2;
        }
        if (first <= last) {
            System.out.println(item + " является " + ++position + " элементом
в массиве");
        } else {
            System.out.println("Элемент не найден в массиве");
        }
    }
}
```

Код поиска бинарным деревом:

```
import java.util.ArrayList;
import java.util.List;

public class Tree<T extends Comparable<T>> {
    private T val;
    private Tree left;
    private Tree right;
    private Tree parent;
    private List<T> listForPrint = new ArrayList<>();
    public T val() {
        return val;
    }
    public Tree left() {
        return left;
    }
    public Tree right() {
        return right;
    }
    public Tree parent() {
        return parent;
    }
    public Tree(T val, Tree parent) {
        this.val = val;
        this.parent = parent;
    }
    public void add(T...vals){
        for(T v : vals){
            add(v);
        }
    }
    public void add(T val){
        if(val.compareTo(this.val) < 0){ //если элемент меньше корня
            if(this.left==null){
                this.left = new Tree(val, this);
            }
            else if(this.left != null)
                this.left.add(val);
        }
        else{ //если элемент больше корня
            if(this.right==null){
                this.right = new Tree(val, this);
```

```
            }
            else if(this.right != null)
                this.right.add(val);
        }
    }
    private Tree<T> _search(Tree<T> tree, T val){
        if(tree == null) return null;
        switch (val.compareTo(tree.val)){
            case 1: return _search(tree.right, val);
            case -1: return _search(tree.left, val);
            case 0: return tree;
            default: return null;
        }
    }

    public Tree<T> search(T val){
        return _search(this, val);
    }
    public boolean remove(T val){
        //Проверяем, существует ли данный узел
        Tree<T> tree = search(val);
        if(tree == null){
            //Если узла не существует, вернем false
            return false;
        }
        Tree<T> curTree;
        //Если удаляем корень
        if(tree == this){
            if(tree.right!=null) {
                curTree = tree.right;
            }
            else curTree = tree.left;
            while (curTree.left != null) {
                curTree = curTree.left;
            }
            T temp = curTree.val;
            this.remove(temp);
            tree.val = temp;
            return true;
        }
        //Удаление листьев
        if(tree.left==null && tree.right==null && tree.parent != null){
            if(tree == tree.parent.left)
                tree.parent.left = null;
            else {
                tree.parent.right = null;
            }
            return true;
        }
        //Удаление узла, имеющего левое поддерево, но не имеющее правого
поддерева
        if(tree.left != null && tree.right == null){
            //Меняем родителя
            tree.left.parent = tree.parent;
            if(tree == tree.parent.left){
                tree.parent.left = tree.left;
            }
            else if(tree == tree.parent.right){
                tree.parent.right = tree.left;
            }
            return true;
        }
        //Удаление узла, имеющего правое поддерево, но не имеющее левого
поддерева
```

```
        if(tree.left == null && tree.right != null){
            //Меняем родителя
            tree.right.parent = tree.parent;
            if(tree == tree.parent.left){
                tree.parent.left = tree.right;
            }
            else if(tree == tree.parent.right){
                tree.parent.right = tree.right;
            }
            return true;
        }
        //Удаляем узел, имеющий поддеревья с обеих сторон
        if(tree.right!=null && tree.left!=null) {
            curTree = tree.right;
            while (curTree.left != null) {
                curTree = curTree.left;
            }
            //Если самый левый элемент является первым потомком
            if(curTree.parent == tree) {
                curTree.left = tree.left;
                tree.left.parent = curTree;
                curTree.parent = tree.parent;
                if (tree == tree.parent.left) {
                    tree.parent.left = curTree;
                } else if (tree == tree.parent.right) {
                    tree.parent.right = curTree;
                }
                return true;
            }
            //Если самый левый элемент НЕ является первым потомком
            else {
                if (curTree.right != null) {
                    curTree.right.parent = curTree.parent;
                }
                curTree.parent.left = curTree.right;
                curTree.right = tree.right;
                curTree.left = tree.left;
                tree.left.parent = curTree;
                tree.right.parent = curTree;
                curTree.parent = tree.parent;
                if (tree == tree.parent.left) {
                    tree.parent.left = curTree;
                } else if (tree == tree.parent.right) {
                    tree.parent.right = curTree;
                }
                return true;
            }
        }
        return false;
    }
    private void _print(Tree<T> node){
        if(node == null) return;
        _print(node.left);
        listForPrint.add(node.val);
        System.out.print(node + " ");
        if(node.right!=null)
            _print(node.right);
    }
    public void print(){
        listForPrint.clear();
        _print(this);
        System.out.println();
    }
    @Override
```

```
    public String toString() {
        return val.toString();
    }

    public static void main(String[] args) {
        //Создадим дерево с корневым элементом 33
        Tree<Integer> tree = new Tree<>(33, null);
        tree.add(5, 35, 1, 20, 4, 17, 31, 99, 18, 19);
        //Распечатаем элементы дерева
        tree.print();
        //Удалим корень
        tree.remove(33);
        tree.remove(17);
        tree.print();
        //Проверяем элементы дерева
        System.out.println(tree);
        System.out.println(tree.left());
        System.out.println(tree.left().left());
        System.out.println(tree.right().left());
        System.out.println(tree.search(18));
    }
}
```

Код Фибоначчиева поиска:

```
import java.util.Arrays;

public class FibSearch {
    private int i, p, q;
    private boolean stop = false;

    public FibSearch() {}

    private void init(int[] sequince) {
        stop = false;
        int k = 0;
        int n = sequince.length;
        for (; getFibonacciNumber(k + 1) < n + 1;) {
            k += 1;
        }
        int m = (int) (getFibonacciNumber(k + 1) - (n + 1));
        i = (int) getFibonacciNumber(k) - m;
        p = (int) getFibonacciNumber(k - 1);
        q = (int) getFibonacciNumber(k - 2);
    }

    public long getFibonacciNumber(int k) {
        long firstNumber = 0;
        long secondNumber = 1;
        for (int i = 0; i < k; i++) {
            long 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[] sequince, int element) {
        init(sequince);
        int n = sequince.length;
        int resultIndex = -1;
        for (; !stop;) {
            if (i < 0) {
                upIndex();
            } else if (i >= n) {
                downIndex();
            } else if (sequince[i] == element) {
                resultIndex = i;
                break;
            } else if (element < sequince[i]) {
                downIndex();
            } else if (element > sequince[i]) {
                upIndex();
            }
        }
        return resultIndex;
    }

    public static void main(String[] args) {
        int[] sequince = new int[] {0, 3, 5, 7, 9, 11, 15, 18, 21};
        int element = 7;

        Arrays.sort(sequince);
        System.out.println("Отсортированный массив: ");
        for (int i = 0; i < sequince.length; i++) {
            System.out.print(sequince[i] + " ");
        }
        System.out.println();
        System.out.println();

        FibSearch fs = new FibSearch();

        int index = fs.search(sequince, element);

        System.out.println("Индекс числа " + element + " равен " + index);
    }


}
```

Код интерполяционного поиска:

```
import java.util.Arrays;

public class InterpolationSearch {
    public static void main(String[] args) {
        int[] sequince = new int[] {2, 5, 1, 3, 7};
        int element = 5;

        Arrays.sort(sequince);
        System.out.println("Отсортированный массив: ");
        for (int i = 0; i < sequince.length; i++) {
            System.out.print(sequince[i] + " ");
        }
        System.out.println();
        System.out.println();
        System.out.println("Индекс числа " + element + " равен " +
interpolationSearch(sequince, element));
    }

    public static int interpolationSearch(int[] sequince, int element) {
        int left = 0;
        int right = sequince.length - 1;

        Arrays.sort(sequince);

        for (; sequince[left] < element && element < sequince[right];) {
            if (sequince[left] == sequince[right]) {
                break; //если диапазон сузился до 1 числа, заканчиваем поиск
            }
            //формула интерполяции:
            int index = (element - sequince[left]) * (left - right) /
(sequince[left] - sequince[right]) + left;

            if (sequince[index] > element) {
                right = element - 1;
            } else if (sequince[index] < element) {
                left = index + 1;
            } else return index;
        }
        if (sequince[left] == element) return left;
        if (sequince[right] == element) return right;
        return -1;
    }
}
```

Задание 2

Код простого рехэширования:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

public class HashSimple {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        System.out.print("Введите количество элементов: ");
        size = input.nextInt();
```

```
        table = new Integer[size];

        System.out.print("Введите числа: ");
        input.nextLine();

        for (int i = 0; i < size; i++) {
            int numbers = input.nextInt();
            hashSimple(numbers, numbers);
        }

        System.out.println(Arrays.toString(table));

        System.out.print("Введите число для поиска: ");
        input.nextLine();
        int number = input.nextInt();

        findHashSimple(number, number);
    }

    private static Integer[] table;
    private static int size;

    public static void hashSimple(int number, int current){
        if (table[current % size] == null){
            table[current % size] = number;
        } else {
            if (current != number + size) {
                hashSimple(number, current + 1);
            } else {
                System.out.println("Таблица заполнена");
            }
        }
    }

    public static int findHashSimple (int number, int current) {
        if (table[current % size] != null) {
            if (table[current % size] == number) {
                System.out.println("Индекс числа равен " +
Integer.toString(current % size));
                return current % size;
            } else {
                if (current != number + size) {
                    findHashSimple(number, current + 1);
                } else {
                    System.out.println("Таблица не содержит введенное
число");
                }
            }
        } else {
            System.out.println("Таблица не содержит введенное число");
        }
        return -1;
    }

}
```

Код рехэширования с помощью псевдослучайных чисел:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

public class HashRandom {

    private static Integer[] table;
    private static int size;
    private static ArrayList<Integer> random;

    public static void main(String[] args) {

        table = new Integer[size];
        random = new ArrayList<>();
        for (int i = 0; i < size; i++){
            while (true) {
                int num = (int)(Math.random() * size);
                if (!random.contains(num)){
                    random.add(num);
                    break;
                }
            }
        }

        Scanner input = new Scanner(System.in);
        System.out.print("Введите количество элементов: ");
        size = input.nextInt();

        System.out.print("Введите числа: ");
        input.nextLine();



        for (int i = 0; i < size; i++) {
            int numbers = input.nextInt();
            hashRandom(numbers, 0);
        }

        System.out.println(Arrays.toString(table));

        System.out.print("Введите число для поиска: ");
        input.nextLine();
        int number = input.nextInt();

        findHasgRandom(number, 0);
    }

    public static void hashRandom (int number, int index) {
        if (table[number % size] == null) {
            table[number % size] = number;
        } else {
            if (table[random.get(index)] == null) {
                table[random.get(index)] = number;
            } else {
                if (index + 1 < random.size() - 1) {
                    hashRandom(number, index + 1);
                } else {
                    System.out.println("Таблица заполнена");
                }
            }
        }
    }
```

```
    public static int findHasgRandom(int number, int index){
        if (table[number % size] != null){
            if (table[number % size] == number){
                System.out.println("Индекс числа равен " +
Integer.toString(number % size));
                return number % size;
            } else {
                if (table[random.get(index)] != null){
                    if (table[random.get(index)] == number){
                        System.out.println("Индекс числа равен " +
Integer.toString(random.get(index)));
                        return random.get(index);
                    } else {
                        if (index + 1 < random.size() - 1){
                            findHasgRandom(number, index + 1);
                        } else {
                            System.out.println("Таблица не содержит введенное
число");
                        }
                    }
                } else {
                    System.out.println("Таблица не содержит введенное
число");
                }
            }
        }  else {
            System.out.println("Таблица не содержит введенное число");
        }
        return -1;
    }


}
```

Код метода цепочек:

```
import java.util.ArrayList;
import java.util.Scanner;

public class HashChainingMethod {
    private static Integer[] table;
    private static int size;
    private static ArrayList<Integer> stack;
    private static ArrayList<Integer> links;

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        System.out.print("Введите количество элементов: ");
        size = input.nextInt();
        table = new Integer[size];
        stack = new ArrayList<>();
        links = new ArrayList<>();

        System.out.print("Введите числа: ");
        input.nextLine();

        for (int i = 0; i < size; i++) {
            int numbers = input.nextInt();
            chainingMethod(numbers);
```

```
        }
        System.out.println(stack.toString());

        System.out.print("Введите число для поиска: ");
        input.nextLine();
        int number = input.nextInt();

        findChainingMethod(number);
    }

    public static void chainingMethod(int number) {

        if (table[number % size] == null){

            stack.add(number);
            links.add(null);
            table[number % size] = stack.size() - 1;

        } else {

            int i = table[number % size];

            while (stack.get(i) != number || links.get(i) != null){
                if (stack.get(i) == number) {
                    System.out.println("Таблица уже содержит это значение");
                } else {
                    if (links.get(i) != null){
                        i = links.get(i);
                    } else {
                        stack.add(number);
                        links.add(null);
                        links.set(i, links.size() - 1);
                    }
                }
            }
        }
    }

    public static int findChainingMethod(int number) {
        if (table[number % size] != null){
            int i = table[number % size];
            while (true) {
                if (stack.get(i) == number){
                    System.out.println("Индекс числа равен " +
Integer.toString(i));
                    return i;
                } else {
                    if (links.get(i) != null){
                        i = links.get(i);
                    } else System.out.println("Таблица не содержит введенное
число");
                    return -1;
                }
            }
        }
        else {
            System.out.println("Таблица не содержит введенное число");
        }
        return -1;
    }
}
```

Задание 3

```
public class Ex {
    public static boolean isSafe(int[][] chessboard, int row, int col){
        int i, j;

        // Определяем, есть ли над элементом ферзь
        for (i = row - 1, j = col; i >= 0; i--) {
            if (chessboard[i][j] == 1) return false;
        }

        // Определяем, есть ли ферзь в верхнем левом углу элемента
        for(i = row - 1, j = col - 1; i >= 0 && j >= 0; i--, j--) {
            if(chessboard[i][j] == 1) {
                return false;
            }
        }

        // Определяем, есть ли ферзь в правом верхнем углу элемента
        for(i = row - 1, j = col + 1; i >= 0 && j < 8; i--, j++) {
            if(chessboard[i][j] == 1) {
                return false;
            }
        }
        return true;
    }

    public static void drawChessboard(int[][] chessboard) {
        int i;
        int j;
//        int count = 0; // Если static не добавляется, каждый раз, когда
вызывается эта функция, count будет повторно присвоено 0!
//        System.out.println("\n" + ++count);

        for(i = 0; i < 8; i++) {
            for(j = 0; j < 8; j++) {
                System.out.print(chessboard[i][j] + "  ");
            }
            System.out.println();
        }
        System.out.println();
    }

    public static void nQueen (int row, int[][] chessboard){
        int col;
        if (row == 8) {
            drawChessboard(chessboard);
        } else {
            for (col = 0; col < 8; col++){
                if (isSafe(chessboard, row, col)){
                    chessboard[row][col] = 1;
                    nQueen(row + 1, chessboard);
                }
                chessboard[row][col] = 0;
            }
        }
    }

    public static void main(String[] args) {
        int[][] chessboard = new int[8][8];
        for (int i = 0; i < chessboard.length; i++){
```

```
            for (int j = 0; j < chessboard[i].length; j++){
                chessboard[i][j] = 0;
            }
        }
        nQueen(0, chessboard);
    }

}
```

Вывод:

Реализовали методы поиска в соответствии с заданием, произвели оценку времени работы каждого алгоритма.