Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Кафедра “Математической кибернетики и информационных технологий”

дисциплина «Структуры и алгоритмы обработки данных»

Отчет по лабораторной работе №2

“Работа с методами поиска”

Выполнил: студенты группы БФИ1901

Козырев Сергей Владимирович

Проверил:

Кутейников Иван Алексеевич

Москва 2021

Оглавление

1.Задание на лабораторную работу.

Задание на лабораторную работу показано 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 работу.

2.Листинг программы (Lab2)

2.1 Листинг программы Binar.

```
package Lab2;
import java.util.Scanner;


public class Binar {
    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 = "1014";
        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;

        position = (first + last) / 2;

        while ((array[position] != item) && (first <= last)) {
            comparisonCount++;
            if (array[position] > item) {
                last = position - 1;
            } else {
                first = position + 1;
            }
            position = (first + last) / 2;
        }
        if (first <= last) {
            System.out.println(item + " является " + ++position + " элементом в м
ассиве");
            System.out.println("Метод бинарного поиска нашел число после " + comp
arisonCount +
                    " сравнений");
        } else {
            System.out.println("Элемент не найден в массиве. Метод бинарного поис
ка закончил работу после "
                    + comparisonCount + " сравнений");
        }
    }
}
```

2.2 Листинг программы BinarTree.

```
package Lab2;
import java.util.Scanner;
import java.util.Stack;

public class BinarTree {
    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 = "1014";
        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();
        for (int i=0;i<n;i++){
            tree.insertNode(arr[i]);
        }

        tree.printTree();
        tree.deleteNode(5);
        tree.printTree();

        Node foundNode = tree.findNodeByValue(3);
        if (foundNode==null){
            System.out.println("Элемента нет в дереве");
        } else {
            foundNode.printNode();
        }
    }
}
class Tree {
    private Node rootNode;

    public Tree() {
        rootNode = null;
    }

    public Node findNodeByValue(int value) {
        Node currentNode = rootNode;
        while (currentNode.getValue() != value) {
            if (value < currentNode.getValue()) {
                currentNode = currentNode.getLeftChild();
            } else {
                currentNode = currentNode.getRightChild();
            }
            if (currentNode == null) {
```

```
                return null;
            }
        }
        return currentNode;
    }

    public void insertNode(int value) {
        Node newNode = new Node();
        newNode.setValue(value);
        if (rootNode == null) {
            rootNode = newNode;
        }
        else {
            Node currentNode = rootNode;
            Node parentNode;
            while (true)
            {
                parentNode = currentNode;
                if(value == currentNode.getValue()) {
                    return;
                }
                else  if (value < currentNode.getValue()) {
                    currentNode = currentNode.getLeftChild();
                    if (currentNode == null){
                        parentNode.setLeftChild(newNode);
                        return;
                    }
                }
                else {
                    currentNode = currentNode.getRightChild();
                    if (currentNode == null) {
                        parentNode.setRightChild(newNode);
                        return;
                    }
                }
            }
        }
    }

    public boolean deleteNode(int value)
    {
        Node currentNode = rootNode;
        Node parentNode = rootNode;
        boolean isLeftChild = true;
        while (currentNode.getValue() != value) {
            parentNode = currentNode;
            if (value < currentNode.getValue()) {
                isLeftChild = true;
                currentNode = currentNode.getLeftChild();
            }
            else {
```

```
                isLeftChild = false;
                currentNode = currentNode.getRightChild();
            }
            if (currentNode == null)
                return false;
        }

        if (currentNode.getLeftChild() == null && currentNode.getRightChild() ==
null) {
            if (currentNode == rootNode)
                rootNode = null;
            else if (isLeftChild)
                parentNode.setLeftChild(null);
            else
                parentNode.setRightChild(null);
        }
        else if (currentNode.getRightChild() == null) {
            if (currentNode == rootNode)
                rootNode = currentNode.getLeftChild();
            else if (isLeftChild)
                parentNode.setLeftChild(currentNode.getLeftChild());
            else
                parentNode.setRightChild(currentNode.getLeftChild());
        }
        else if (currentNode.getLeftChild() == null) {
            if (currentNode == rootNode)
                rootNode = currentNode.getRightChild();
            else if (isLeftChild)
                parentNode.setLeftChild(currentNode.getRightChild());
            else
                parentNode.setRightChild(currentNode.getRightChild());
        }
        else {
            Node heir = receiveHeir(currentNode);
            if (currentNode == rootNode)
                rootNode = heir;
            else if (isLeftChild)
                parentNode.setLeftChild(heir);
            else
                parentNode.setRightChild(heir);
        }
        return true;
    }

    private Node receiveHeir(Node node) {
        Node parentNode = node;
        Node heirNode = node;
        Node currentNode = node.getRightChild();
        while (currentNode != null)
        {
            parentNode = heirNode;
```

```
            heirNode = currentNode;
            currentNode = currentNode.getLeftChild();
        }
        if (heirNode != node.getRightChild())
        {
            parentNode.setLeftChild(heirNode.getRightChild());
            heirNode.setRightChild(node.getRightChild());
        }
        return heirNode;
    }

    public void printTree() {
        Stack globalStack = new Stack();
        globalStack.push(rootNode);
        int gaps = 32;
        boolean isRowEmpty = false;
        String separator = "-----------------------------------------------------
------------";
        System.out.println(separator);
        while (isRowEmpty == false) {
            Stack localStack = new Stack();
            isRowEmpty = true;

            for (int j = 0; j < gaps; j++)
                System.out.print(' ');
            while (globalStack.isEmpty() == false) {
                Node temp = (Node) globalStack.pop();
                if (temp != null) {
                    System.out.print(temp.getValue());
                    localStack.push(temp.getLeftChild());
                    localStack.push(temp.getRightChild());
                    if (temp.getLeftChild() != null ||
                            temp.getRightChild() != null)
                        isRowEmpty = false;
                }
                else {
                    System.out.print("__");
                    localStack.push(null);
                    localStack.push(null);
                }
                for (int j = 0; j < gaps * 2 - 2; j++)
                    System.out.print(' ');
            }
            System.out.println();
            gaps /= 2;
            while (localStack.isEmpty() == false)
                globalStack.push(localStack.pop());
        }
        System.out.println(separator);
    }
}
```

2.3 Листинг программы Chains.

```
package Lab2;
//Ссылается на HashTable_Cains
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Chains
{
    public static void main( String args[] )
    {
        HashTable_Chains table = new HashTable_Chains(7);

        try
        {
            File file = new File("C:\\Users\\GrottiBeatz\\Test\\input.txt");
            Scanner scanner = new Scanner(file);
            while (scanner.hasNext())
            {
                table.addElement(scanner.next());
            }
            scanner.close();

        }

        catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }

        table.printHashTable();
        Scanner sc = new Scanner(System.in);
        System.out.print("Введите слово для поиска: ");
        String answer = sc.nextLine();
        if(table.findElement(answer))
        {
            System.out.print("Такое слово есть.");
        }
        else
            System.out.print("Такого слова нету.");

    }
}
```

2.4 Листинг программы Fibbonachy.

```
package Lab2;
import java.util.Scanner;

public class Fibbonachy {
    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 = "1014";
        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();
    }
    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;
```

```
    }

}
```

2.5 Листинг программы HashTable_Chains.

```
package Lab2;
import java.util.ArrayList;

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

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

    private 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();
        }
    }
}
```

2.6 Листинг программы HashTable_random.

```
package Lab2;
public class HashTable_random {

    private Item[] table;
    private int count;
    private int size;

    public HashTable_random(int size) {
        this.size = size;
        table = new Item[size];
    }
    public int hash_random(String key)
    {
        double hash=0;
        double R = 1;

        for(int i = 0; i < key.length(); i++)
           R=5*R;
           R=R%(4*size);
           hash=Math.floor(R/4);

        return (int)hash;
    }

    public void insert(String key) {
        Item item = new Item(key);
        int hash = hash_random(key);
        while (table[hash] != null) {
            hash++;
            hash %= (4*size);
        }
        table[hash] = item;
    }
    public void print()
    {
        for(int i = 0; i < size; i++)
            if(table[i] != null)
                System.out.println(i + " " + table[i].getKey());
    }
    public Item find(String key)
```

```
    {
        int hash = hash_random(key);
        while(table[hash] != null)
        {
            if(table[hash].getKey().equals(key))
                return table[hash];
            hash++;
            hash = hash % (4*size);
        }

        return null;
    }
}
```

2.7 Листинг программы HashTable.

```
package Lab2;

public class HashTable {

    private Item[] table;
    private int count;
    private int size;

    public HashTable(int size) {
        this.size = size;
        table = new Item[size];
    }
    public int hash(String key)
    {
        int hash = 0;

        for(int i = 0; i < key.length(); i++)
            hash = (31 * hash + key.charAt(i)) % size;

        return hash;
    }

    public void insert(String key) {
        Item item = new Item(key);
        int hash = hash(key);
        while (table[hash] != null) {
            hash++;
            hash %= size;
        }
        table[hash] = item;
    }
    public void print()
    {
        for(int i = 0; i < size; i++)
```

```
            if(table[i] != null)
                System.out.println(i + " " + table[i].getKey());
    }
    public Item find(String key)
    {
        int hash = hash(key);
        while(table[hash] != null)
        {
            if(table[hash].getKey().equals(key))
                return table[hash];
            hash++;
            hash = hash % size;
        }

        return null;
    }
}
```

2.8 Листинг программы Interpol.

```
package Lab2;

import java.util.Scanner;

public class Interpol {
    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[startInd
ex]) &&
                (elementToSearch <= integers[lastIndex])) {
            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;
    }
}
```

2.9 Листинг программы Koroleva.

```
package Lab2;

public class Koroleva {

    private int dimension;
    private int[] state;
    private int index = 1;

    public Koroleva(int n) {
        dimension = n;
        state = new int[n];

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

    public boolean next() {
        index++;
        return move(dimension - 1);
    }


    private boolean move(int index) {
        if (state[index] < dimension - 1) {
            state[index]++;
            return true;
        }

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

    public int getIndex() {
        return index;
    }

    public boolean isPeace() {
        for (int i = 0; i < state.length; i++) {
            for (int j = i + 1; j < state.length; j++) {
                if (state[i] == state[j]) {
                    return false;
                }
                if (Math.abs(i - j) == Math.abs(state[i] - state[j])) {
```

```
                    return false;
                }
            }
        }

        return true;
    }


    public void printState() {
        for (int i = 0; i < state.length; i++) {
            int position = state[i];
            for (int j = 0; j < dimension; j++) {
                System.out.print(j == position ? 'X' : '_');
            }
            System.out.println();
        }
    }

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

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

2.10 Листинг программы Node.

```
package Lab2;

public class Node {
    private int value;
    private Node leftChild;
    private Node rightChild;

    public void printNode() {
        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 +
                '}';
    }
}
```

2.11 Листинг программы Rehashing_random.

```
package Lab2;
import java.util.Scanner;
public class Rehashing_random {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        HashTable_random hashTable = new HashTable_random(128);
        hashTable.insert("visa");
        hashTable.insert("gang");
        hashTable.insert("beatz");
        hashTable.insert("world");
        hashTable.insert("wide");
        hashTable.print();
        System.out.println("Введите слово для поиска:");
        String word = scanner.nextLine();
        Item item = hashTable.find(word);
```

```
        if (item != null)
            System.out.println("Элемент найден, его хэш: " + hashTable.hash_rando
m(word));
        else
            System.out.println("Элемент не найден!");
    }
}
```

2.12 Листинг программы Rehashing.

```
package Lab2;
//опирается на HashTable
import java.util.Scanner;
public class Rehashing {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        HashTable hashTable = new HashTable(97);
        hashTable.insert("visa");
        hashTable.insert("gang");
        hashTable.insert("beats");
        hashTable.insert("worldwide");
        hashTable.insert("platinum");
        hashTable.insert("producer");
        hashTable.print();
        String word = scanner.nextLine();
        Item item = hashTable.find(word);
        if (item != null)
            System.out.println("Элемент найден, его хэш: " + hashTable.hash(word)
);
        else
            System.out.println("Элемент не найден!");
    }
}
 class Item{
    private String key;
    public Item(String key)
    {
        this.key = key;
    }

    public String getKey() {
        return key;
    }

    public void setKey(String key) {
        this.key = key;
    }
}
```

3.Вывод

Выполнив данную лабораторную работу, мы научились работать с методами поиска и применять их в поставленных задачах, а именно, реализовывать поиск в массивах данных.

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.