Федеральное агентство связи

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский технический университет связи и информатики»

Кафедра «Информатики»

Дисциплина «СиАОД»

Лабораторная работа №2

Выполнил: студент группы БВТ1901

Адедиха Коффи Жермен

Руководитель:

Мелехин А.

Москва 2021

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.

Задание №1

- **Бинарный поиск**

```
import java.util.Arrays;
import java.util.Scanner;
import java.util.Random;
public class BinarySearch {

    public static void main(String args[]) {
        int counter, num, item, array[], first, last;

        //Создаем объект Scanner для считывания чисел, введенных пользователем


        Scanner input = new Scanner(System.in);
        //int n;
        System.out.println("Введите размер массива:");
        num = input.nextInt();

        int[] array1 = new int[num];

        Random random = new Random();

        for (int i = 0; i < num; i++){
                array1[i] = random.nextInt(100);
            }
            System.out.println( Arrays.toString(array1));


        Arrays.sort(array1);
        System.out.println( Arrays.toString(array1));
        System.out.println("Введите элемент для бинарного поиска: ");
        item = input.nextInt();
        first = 0;
        last = num - 1;

        // метод принимает начальный и последний индекс, а также число для поиска
        long before = System.nanoTime();
        binarySearch(array1, first, last, item);
        long after = System.nanoTime();

        System.out.println("Time  in nanos: " + (after - before));
```

```
    }

    // бинарный поиск
    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
                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 + " - " + position + "ый в массиве");
            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");
        }
    }

}
```

- **Бинарное дерево**

```
import java.util.Random;
import java.util.Scanner;
import java.util.Arrays;
public class BinarySearchTree {
    public static  Node root;


    public BinarySearchTree(){
        this.root = null;
```

```
    }

    public boolean find(int id){
        Node current = root;
        while(current!=null){
            if(current.data==id){
                return true;
            }else if(current.data>id){
                current = current.left;
            }else{
                current = current.right;
            }
        }
        return false;
    }
    public boolean delete(int id){
        Node parent = root;
        Node current = root;
        boolean isLeftChild = false;
        while(current.data!=id){
            parent = current;
            if(current.data>id){
                isLeftChild = true;
                current = current.left;
            }else{
                isLeftChild = false;
                current = current.right;
            }
            if(current ==null){
                return false;
            }
        }
        //if i am here that means we have found the node
        //Case 1: if node to be deleted has no children
        if(current.left==null && current.right==null){
            if(current==root){
                root = null;
            }
            if(isLeftChild ==true){
                parent.left = null;
            }else{
                parent.right = null;
            }
        }
```

```
        //Case 2 : if node to be deleted has only one child
        else if(current.right==null){
            if(current==root){
                root = current.left;
            }else if(isLeftChild){
                parent.left = current.left;
            }else{
                parent.right = current.left;
            }
        }
        else if(current.left==null){
            if(current==root){
                root = current.right;
            }else if(isLeftChild){
                parent.left = current.right;
            }else{
                parent.right = current.right;
            }
        }else if(current.left!=null && current.right!=null){

            //now we have found the minimum element in the right sub tree
            Node successor   = getSuccessor(current);
            if(current==root){
                root = successor;
            }else if(isLeftChild){
                parent.left = successor;
            }else{
                parent.right = successor;
            }
            successor.left = current.left;
        }
        return true;
    }

    public Node getSuccessor(Node deleleNode){
        Node successsor =null;
        Node successsorParent =null;
        Node current = deleleNode.right;
        while(current!=null){
            successsorParent = successsor;
            successsor = current;
            current = current.left;
        }
```

```
        //check if successor has the right child, it cannot have left child for s
ure
        // if it does have the right child, add it to the left of successorParent
.
//      successsorParent
        if(successsor!=deleleNode.right){
            successsorParent.left = successsor.right;
            successsor.right = deleleNode.right;
        }
        return successsor;
    }
    public void insert(int id){
        Node newNode = new Node(id);
        if(root==null){
            root = newNode;
            return;
        }
        Node current = root;
        Node parent = null;
        while(true){
            parent = current;
            if(id<current.data){
                current = current.left;
                if(current==null){
                    parent.left = newNode;
                    return;
                }
            }else{
                current = current.right;
                if(current==null){
                    parent.right = newNode;
                    return;
                }
            }
        }
    }
    public void display(Node root){
        if(root!=null){
            display(root.left);
            System.out.print(" " + root.data);
            display(root.right);
        }
    }
    public static void main(String arg[]){
```

```
        BinarySearchTree b = new BinarySearchTree();

        Scanner in = new Scanner(System.in);
        int n;
        System.out.println("Введите размер массива:");
        n = in.nextInt();

        int[] array1 = new int[n];

        Random random = new Random();

        for (int i = 0; i < n; i++){
                array1[i] = random.nextInt(100);
                b.insert(random.nextInt(100));
            }

        System.out.println(Arrays.toString(array1) + "\n");


        System.out.println("Original Tree : ");
        b.display(b.root);
        System.out.println("");
        //search element
        System.out.print("Искомый Элемент : ");
        int  c = in.nextInt();
        System.out.println("Check whether Node with value " +c+" exists : " + b.f
ind(c));
        System.out.print("Удалить элемент : " );
        int  d = in.nextInt();
        //
        System.out.println("Удление (" + d+") : " + b.delete(d));
        b.display(root);
        System.out.println("");
        System.out.print("Добавить элемент : " );
        int  f = in.nextInt();
        b.insert(f);
        //System.out.println("\n Delete Node with one child (4) : " + b.delete(4)
);
        b.display(root);
        //System.out.println("\n Delete Node with Two children (10) : " + b.delet
e(10));
        //b.display(root);
    }
}
```

```
class Node{
    int data;
    Node left;
    Node right;
    public Node(int data){
        this.data = data;
        left = null;
        right = null;
    }
}
```

- **Фибоначчиев**

```
import java.util.*;
public class FibonacciSearch{
    private int i;
    private int p;
    private int q;
    private boolean stop=false;
    public FibonacciSearch(){

    }
    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=q-p;
        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  num;
FibonacciSearch fs=new FibonacciSearch();
        Scanner input = new Scanner(System.in);
```

```
        System.out.println("Введите размер массива:");
        num = input.nextInt();

        int[] array1 = new int[num];

        Random random = new Random();

        for (int i = 0; i < num; i++){
                array1[i] = random.nextInt(100);
            }
            System.out.println( Arrays.toString(array1));
            Arrays.sort(array1);
        System.out.println( Arrays.toString(array1));

     //   int [] sequince= new int []{-2,0,3,5,7,9,11,15,18,21};
        //System.out.println(Arrays.toString(sequince));
        System.out.print("Искомый элемент : ");
       int element =  input.nextInt();
       long before = System.nanoTime();
       int index= fs.search(array1,element);
       long after = System.nanoTime();
       System.out.println(" Index of the element is "+ index);
       System.out.println("Time  in nanos with Fibonacci search: " + (after - bef
ore));
          // int num = input.nextInt();
          input.close();

}
    }
```

- **Интерполяционный**

```
import java.util.Random;
import java.util.Scanner;
import java.util.Arrays;
public class interpolationSearch {
    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;
    }
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n;
        System.out.println("Введите размер массива:");
        n = in.nextInt();

        int[] array1 = new int[n];

        Random random = new Random();

        for (int i = 0; i < n; i++){
                array1[i] = random.nextInt(100);
            }

         System.out.println(Arrays.toString(array1));
        Arrays.sort(array1);
        System.out.println(Arrays.toString(array1));
        System.out.println("Введите элемент для поиска: ");
        int a = in.nextInt();
        long debut = System.nanoTime();
        int index=InterpolationSearch(array1,a);
        long fin = System.nanoTime();
        if (index==-1){
            System.out.println("Element not found in the massive");
        }
        else{
```

```
        System.out.println("Element index is: " + index);
        System.out.println("Time with interpolation searching in nanos: " + (fin
- debut));

        long debut1 = System.nanoTime();
        Arrays.binarySearch(array1,a);
        long fin1 = System.nanoTime();

        //System.out.println("Element index is: " +  Arrays.binarySearch(array1,a
));
        System.out.println("Time with java standart searching  in nanos: " + (fin
1 - debut1));

        }

      /*  long k = Math.round(Math.random()*n);
        System.out.println("K:"+k);
        String intArrayString = Arrays.toString(array1);
        System.out.println(intArrayString);
        long before = System.nanoTime();
        interpolationSearch(array1,array1[(int)k]);
        long after = System.nanoTime();
        System.out.println("Element index is: " +  interpolationSearch(array1,arr
ay1[(int)k]));
        System.out.println("Time interpolationSearch in nanos: " + (after - befor
e));
*/

/*
        before = System.nanoTime();
        Arrays.binarySearch(array1,array1[(int)k]);
        after = System.nanoTime();
        System.out.println("Element index is: " +  Arrays.binarySearch(array1,arr
ay1[(int)k]));
        System.out.println("Time sandartSearch in nanos: " + (after - before));
*/
    }
    }
```

Задание№2

- **Простое рехэширование**

```java
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);
        }
//        if (input.hasNextInt()) {
//            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);


//        System.out.println("Введите количество элементов: ");
//        int size = input.nextInt();
//        table = new Integer[size];
//
//        System.out.println("Введите числа: ");
//
//        for (int i = 0; i < size; i++){
//
```

```
//        }

    }

    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 {                                    // если индекс не пуст
            if (current != number + size) {     //проверяем на наличие свободных
мест
                hashSimple(number, current + 1); //записываем в следующую ячейку
            } 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(curre
nt % 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;
    }

}
```

- **Рехэширование с помощью псевдослучайных чисел**

```java
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(numbe
r % size));
                return number % size;
            } else {
                if (table[random.get(index)] != null){
                    if (table[random.get(index)] == number){
                        System.out.println("Индекс числа равен " + Integer.toStri
ng(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 чис
ло");

                        }
                    }
                } else {
                    System.out.println("Таблица не содержит введенное число");
                }
            }
        }  else {
```

```
            System.out.println("Таблица не содержит введенное число");
        }
        return -1;
    }


}
```

- **Метод цепочек**

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

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

    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];
        a = new ArrayList<>();
        b = new ArrayList<>();

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

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

        System.out.println(a.toString());
        System.out.println(b.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){
            a.add(number);
            b.add(null);
            table[number % size] = a.size() - 1;
        } else {

            int i = table[number % size];

            while (a.get(i) != number || b.get(i) != null){
                if (a.get(i) == number) {
                    System.out.println("Таблица уже содержит это значение");
                } else {
                    if (b.get(i) != null){
                        i = b.get(i);
                    } else {
                        a.add(number);
                        b.add(null);
                    }
                }
            }
        }
    }

    public static int findChainingMethod(int number) {
        if (table[number % size] != null){
            int i = table[number % size];
            while (true) {
                if (a.get(i) == number){
                    System.out.println("Индекс числа равен " + Integer.toString(i
));
                    return i;
                } else {
                    if (b.get(i) != null){
                        i = b.get(i);
```

```
                    } else System.out.println("Таблица не содержит введенное числ
о");
                    return -1;
                }
            }
        }
        else {
            System.out.println("Таблица не содержит введенное число");
        }
        return -1;
    }
}
```

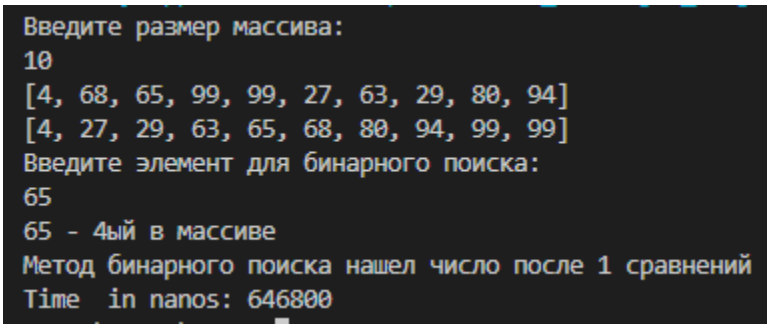
Задание 3

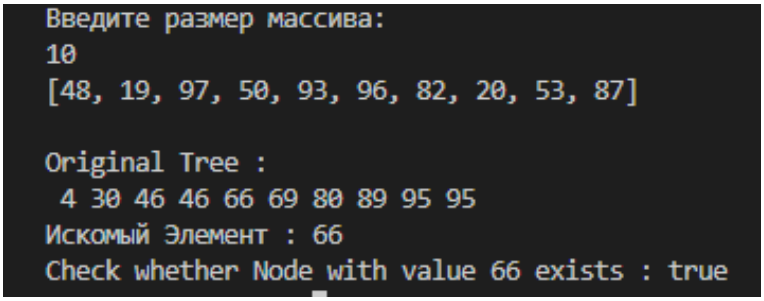
```
public class EightQueens {
    public static boolean isSafe(int[][] chessboard, int row, int col){
        int i, j;

        // 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) {
```

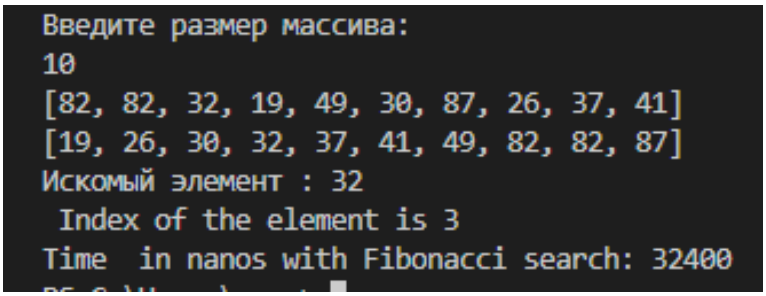
```
        int i, j;

        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);
    }

}
```

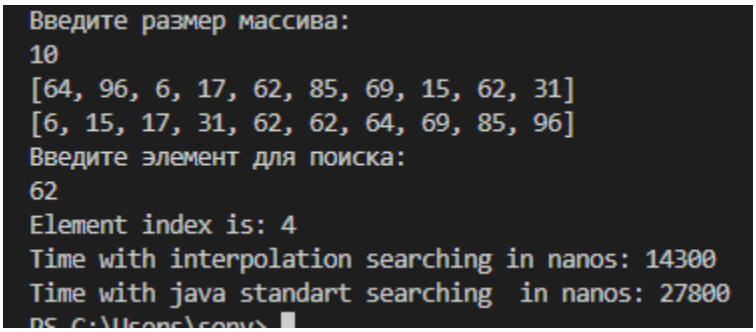
- **Работы программы**

Задание 1









Задание №2

```
[10, 11, 12, 13, 2, 45, 56, 0, 78, 5]
Введите число для поиска: 2
Индекс числа равен 4
```

```
Введите размер массива:
10
[64, 96, 6, 17, 62, 85, 69, 15, 62, 31]
[6, 15, 17, 31, 62, 62, 64, 69, 85, 96]
Введите элемент для поиска:
62
Element index is: 4
```

Задание 3

```
1  0  0  0  0  0  0  0
0  0  0  0  1  0  0  0
0  0  0  0  0  0  0  1
0  0  0  0  0  1  0  0
0  0  1  0  0  0  0  0
0  0  0  0  0  0  1  0
0  1  0  0  0  0  0  0
0  0  0  1  0  0  0  0
```

Вывод :

С этой лабораторной работы мы научились поискать элементы разными методами поиск