

Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации

Бюджетное образовательное учреждение высшего образования
Московский технический университет связи и информатики

Кафедра «Математической кибернетики и информационных технологий»

Дисциплина «Структуры и алгоритмы обработки данных»

Отчёт по лабораторной работе №2

«Методы поиска»

Выполнил студент
Группы БФИ1901
Курбатов А.О.

Проверил
Кутейников И. 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.

Задание №1:

Бинарный поиск	Бинарное дерево	<u>Фибоначчиев</u>	Интерполяционный
----------------	-----------------	--------------------	------------------

Задание №2:

Простое <u>рехэширование</u>	<u>Рехэширование с помощью псевдослучайных чисел</u>	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.

Листинг

```
import java.util.*;  
  
public class Lab2 {
```

```

public static void sort(ArrayList<Integer> sorted, int n) {
    int h = n / 2;
    while (h >= 1) {
        ShellSort(sorted, h, n);
        h /= 2;
    }
}

public static void ShellSort(ArrayList<Integer> sorted, int h, int n) {
    int swap;
    for (int j = h; j < n; j++) {
        for (int k = j; k >= h; k = k - h) {
            if (sorted.get(k) < sorted.get(k - h)) {
                swap = sorted.get(k);
                sorted.set(k, sorted.get(k - h));
                sorted.set(k - h, swap);
            }
        }
    }
}

public static String binaryTree(ArrayList<Integer> mass, int num) {
    String path = "Путь к элементу " + num + ": ";
    int root = mass.get(0);
    boolean check = false;

    for (int i = 1; i < mass.size(); i++) {
        if (num > root) {
            if (mass.get(i) > root) {
                root = mass.get(i);
                path += "R-";
            }
        }
        if (num < root) {
            if (mass.get(i) < root) {
                root = mass.get(i);
                path += "L-";
            }
        }

        if (root == num) {
            check = true;
            break;
        }
    }

    if (check) return path.substring(0, path.length() - 1);

    return "Элемент " + num + " не найден";
}

public static String binarySearch(ArrayList<Integer> mass, int num, int min,
int max) {
    int index = -1;
    if (mass.contains(num)) {
        while (min <= max) {
            int mid = (min + max) / 2;
            if (mass.get(mid) < num) {
                min = mid + 1;
            }
        }
    }
}

```

```

        } else if (mass.get(mid) > num) {
            max = mid - 1;
        } else if (mass.get(mid) == num) {
            index = mid;
            break;
        }
    }
    return "Элемент " + num + " находится на " + index + " месте";
} else return "Элемент " + num + " не найден";
}

public static String fibSearch(ArrayList<Integer> mass, int num, int n) {
    int fib1 = 0;
    int fib2 = 1;
    int fib = fib1 + fib2;

    while (fib < n) {
        fib1 = fib2;
        fib2 = fib;
        fib = fib1 + fib2;
    }

    int index = -1;

    while (fib > 1) {
        int i = Math.min(index + fib1, n - 1);
        if (mass.get(i) < num) {
            fib = fib2;
            fib2 = fib1;
            fib1 = fib - fib2;
            index = i;
        } else if (mass.get(i) > num) {
            fib = fib1;
            fib2 = fib2 - fib1;
            fib1 = fib - fib2;
        } else
            return "Элемент " + num + " находится на " + i + " месте";
    }
    if (fib2 == 1 && mass.get(n - 1) == num)
        return "Элемент " + num + " находится на " + (n - 1) + " месте";

    return "Элемент " + num + " не найден";
}

public static String interSearch(ArrayList<Integer> mass, int num) {
    int d;
    int i = 0;
    int j = mass.size() - 1;

    while (mass.get(i) < num && mass.get(j) > num) {
        if (mass.get(j) == mass.get(i))
            break;
        d = i + ((num - mass.get(i)) * (j - i)) / (mass.get(j) -
mass.get(i));

        if (mass.get(d) < num)
            i = d + 1;
        else if (mass.get(d) > num)
            j = d - 1;
    }
}

```

```

        else
            return "Элемент " + num + " находится на " + d + " месте";
    }
    if (mass.get(i) == num)
        return "Элемент " + num + " находится на " + i + " месте";
    if (mass.get(j) == num)
        return "Элемент " + num + " находится на " + j + " месте";

    return "Элемент " + num + " не найден";
}

public static String simpleRehash(ArrayList<Integer> mass, int num) {
    String[] arr = new String[mass.size()];
    for (int i = 0; i < mass.size(); i++) {
        arr[i] = Integer.toString(mass.get(i));
    }
    Map<Integer, String> map = new HashMap<>();
    boolean check;
    int isFull = 0;
    int counter = 1;
    for (int i = 0; i < arr.length; i++) {
        if (isFull == arr.length) break;
        int h = arr[i].hashCode() % 6;
        check = false;
        while (check == false) {
            if (h < 0) {
                break;
            }
            if (map.get(h) == null) {
                map.put(h, arr[i]);
                check = true;
                isFull++;
            } else {
                h = (arr[i] + counter).hashCode() % 6;
                counter++;
            }
        }
    }
    counter = 1;
    System.out.println(map.values());
    String find = Integer.toString(num);
    int hash = find.hashCode() % 6;
    int start = hash;
    do {
        if (map.get(hash) == null) {
            return "Элемент " + num + " не найден";
        } else if (map.get(hash).equals(find)) {
            return "Элемент " + num + " найден в ячейке " + hash;
        } else {
            hash = (find + counter).hashCode() % 6;
            counter++;
        }
    }
    while (hash != start);
    return "Элемент " + num + " не найден";
}

public static String chainRehash(ArrayList<Integer> mass, int num) {
    String[] arr = new String[mass.size()];
    for (int i = 0; i < mass.size(); i++) {

```

```

        arr[i] = Integer.toString(mass.get(i));
    }

    ArrayList<LinkedList> table = new ArrayList<>();

    LinkedList<String> hash0 = new LinkedList<>();
    LinkedList<String> hash1 = new LinkedList<>();
    LinkedList<String> hash2 = new LinkedList<>();
    LinkedList<String> hash3 = new LinkedList<>();
    LinkedList<String> hash4 = new LinkedList<>();
    LinkedList<String> hash5 = new LinkedList<>();

    table.add(hash0);
    table.add(hash1);
    table.add(hash2);
    table.add(hash3);
    table.add(hash4);
    table.add(hash5);

    for (int i = 0; i < arr.length; i++) {
        int h = arr[i].hashCode() % 6;
        table.get(h).add(arr[i]);
    }

    for (int i = 0; i < table.size(); i++) {
        if(!table.get(i).isEmpty())
            System.out.println("hash = " + i + ": " + table.get(i));
    }

    String find = Integer.toString(num);

    int hash = find.hashCode() % 6;

    if (table.get(hash).isEmpty() || !table.get(hash).contains(find)) {
        return "Элемент " + num + " не найден";
    }
    else {
        return "Элемент " + num + " найден в ячейке " + hash;
    }
}

static int[] board = {0,0,0,0,0,0,0,0};
static int index = 0;
static int version = 0;

public static void game() {
    do {
        if (check()) {
            if (index == 7) {
                System.out.println(version++ + " [0]=" + board[0] + " [1]="
+ board[1] + " [2]=" + board[2] + " [3]=" + board[3] + " [4]=" + board[4] + "
[5]=" + board[5] + " [6]=" + board[6] + " [7]=" + board[7]);
                board[index]++;
            } else {
                index++;
            }
        } else {
    }
}

```

```

        board[index]++;
    }
} while (board[0] < 8);
}

static boolean check() {
    int i;

    if (index == 0) {
        return true;
    }

    if (board[index]>7){
        board[index] = 0;
        index--;
        return false;
    }

    for (i=0;i<index;i++){
        if ((board[index] == board[i]) || (Math.abs(board[index] - board[i]))
== (index-i))){
            return false;
        }
    }
    return true;
}

public static void main(String[] args) {
    System.out.println("Задание 1:");
    int n = 10, min_limit = 0, max_limit = 10;
    ArrayList<Integer> mass = new ArrayList();
    Random rand = new Random();
    for (int i = 0; i < n; i++) {
        mass.add(min_limit + rand.nextInt(max_limit - min_limit + 1));
    }

    System.out.print(mass.toString());

    int num = 7;

    sort(mass, mass.size());
    System.out.println("\n" + "Отсортированный массив: ");
    System.out.println(mass.toString());

    ArrayList<Integer> tree = new ArrayList();
    tree.add(7);
    tree.add(15);
    tree.add(9);
    tree.add(1);
    tree.add(0);
    tree.add(97);
    tree.add(18);
    tree.add(19);

    long t = System.currentTimeMillis();
    System.out.println("\n" + "Поиск по бинарному дереву:");
    System.out.println(binaryTree(tree, 19));
    System.out.println("Время: " + ((double) System.currentTimeMillis() - t)
+ " ms");
}

```

```

        t = System.currentTimeMillis();
        System.out.println("\n" + "Бинарный поиск: ");
        System.out.println(binarySearch(mass, num, 0, mass.size() - 1));
        System.out.println("Время: " + ((double) System.currentTimeMillis() - t)
+ " ms");

        t = System.currentTimeMillis();
        System.out.println("\n" + "Фибоначчиев поиск:");
        System.out.println(fibSearch(mass, num, mass.size()));
        System.out.println("Время: " + ((double) System.currentTimeMillis() - t)
+ " ms");

        t = System.currentTimeMillis();
        System.out.println("\n" + "Интерполяционный поиск:");
        System.out.println(interSearch(mass, num));
        System.out.println("Время: " + ((double) System.currentTimeMillis() - t)
+ " ms");

        System.out.println("\n" + "Задание 2:");
        ArrayList<Integer> arr = new ArrayList();
        int m = 10, low = 0, high = 9;
        for (int i = 0; i < m; i++) {
            arr.add(low + rand.nextInt(high - low + 1));
        }
        System.out.print(arr.toString());

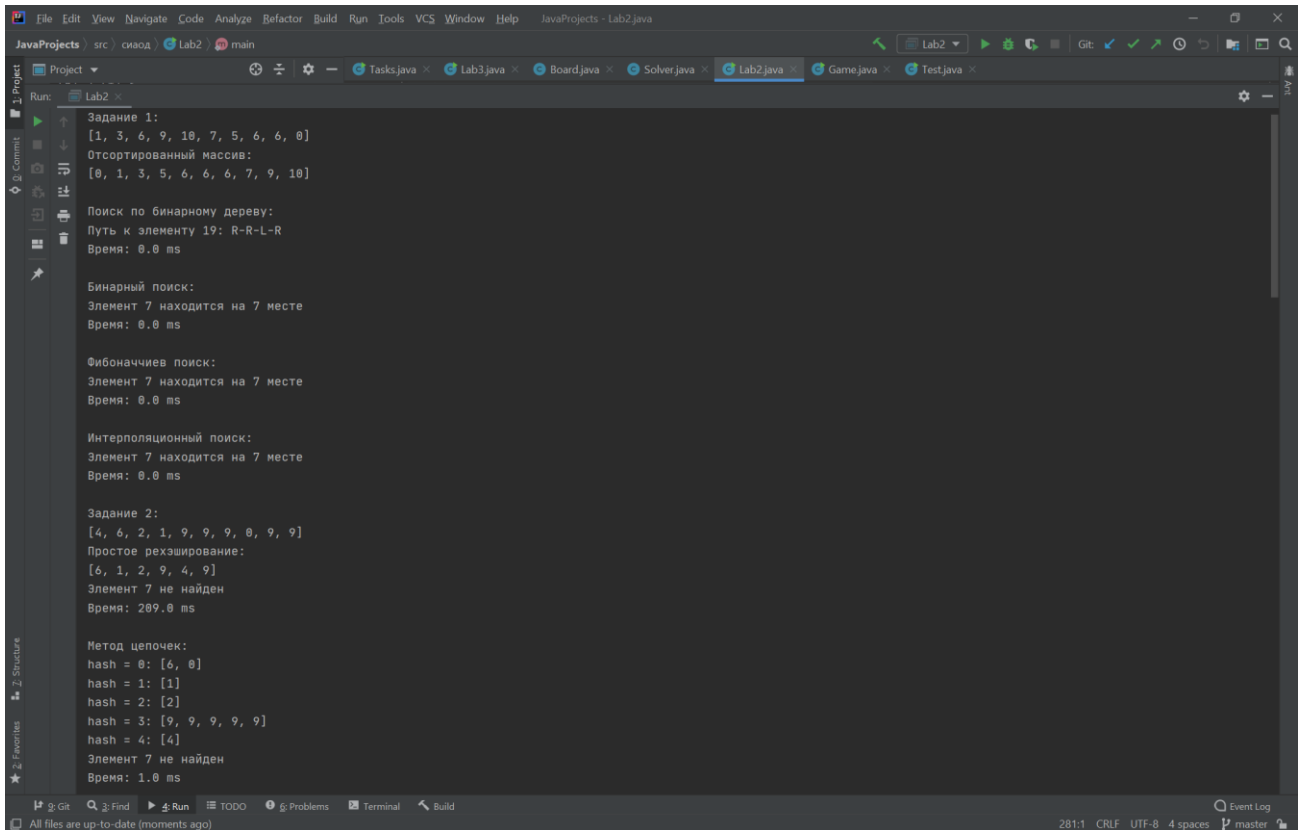
        t = System.currentTimeMillis();
        System.out.println("\n" + "Простое рехэширование: ");
        System.out.println(simpleRehash(arr, num));
        System.out.println("Время: " + ((double) System.currentTimeMillis() - t)
+ " ms");

        t = System.currentTimeMillis();
        System.out.println("\n" + "Метод цепочек: ");
        System.out.println(chainRehash(arr, 7));
        System.out.println("Время: " + ((double) System.currentTimeMillis() - t)
+ " ms");

        System.out.println("\n" + "Задание 3:");
        System.out.println("Расположить 8 ферзей: ");
        game();
    }
}

```


Результат выполнения работы



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help JavaProjects - Lab2.java
JavaProjects src \ src \ Lab2 main
Run: Lab2
Задание 1:
[1, 3, 6, 9, 10, 7, 5, 6, 6, 8]
Отсортированный массив:
[0, 1, 3, 5, 6, 6, 6, 7, 9, 10]

Поиск по бинарному дереву:
Путь к элементу 19: R-R-L-R
Время: 0.0 ms

Бинарный поиск:
Элемент 7 находится на 7 месте
Время: 0.0 ms

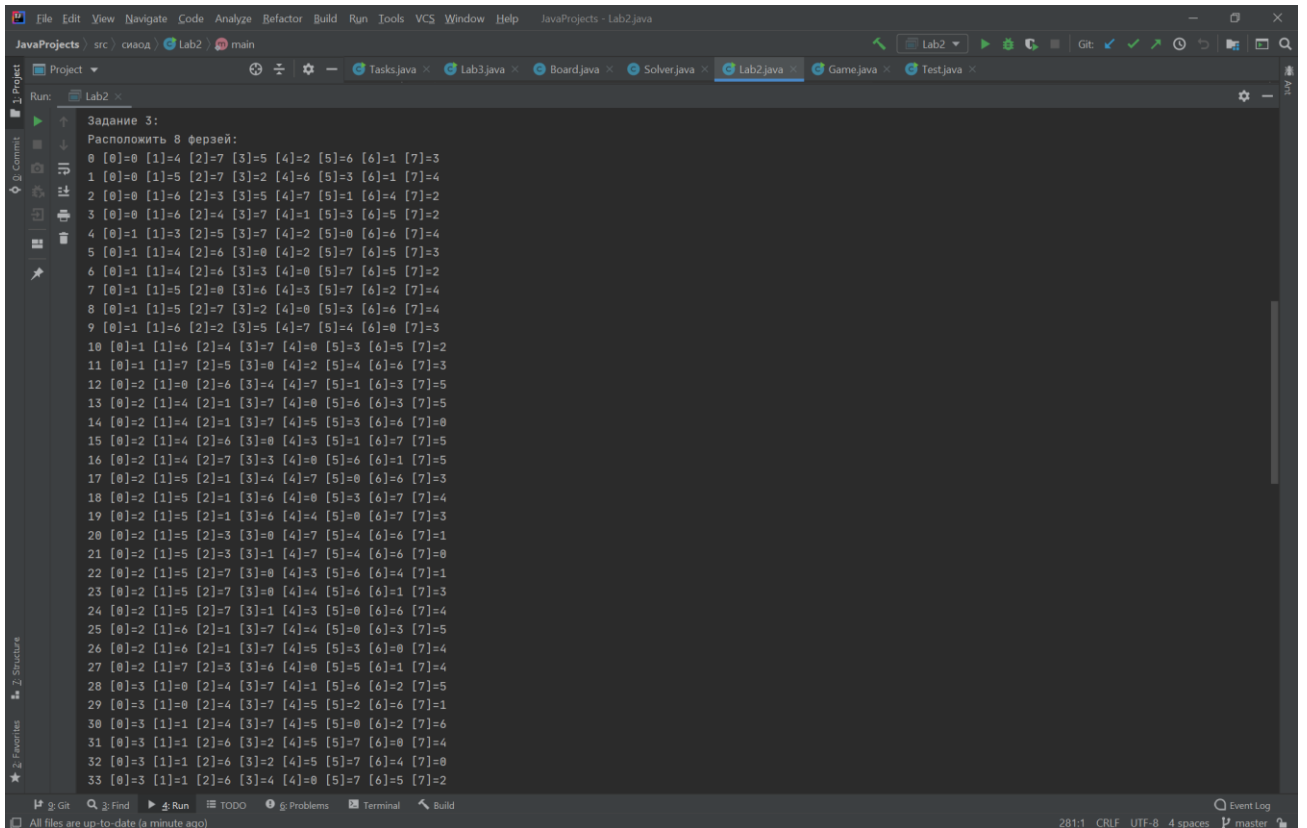
Фибоначчиев поиск:
Элемент 7 находится на 7 месте
Время: 0.0 ms

Интерполяционный поиск:
Элемент 7 находится на 7 месте
Время: 0.0 ms

Задание 2:
[4, 6, 2, 1, 9, 9, 9, 0, 9, 9]
Простое рехэширование:
[6, 1, 2, 9, 4, 9]
Элемент 7 не найден
Время: 209.0 ms

Метод цепочек:
hash = 0: [6, 0]
hash = 1: [1]
hash = 2: [2]
hash = 3: [9, 9, 9, 9, 9]
hash = 4: [4]
Элемент 7 не найден
Время: 1.0 ms

Event Log
All files are up-to-date (moments ago) 281:1 CRLF UTF-8 4 spaces master
```



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help JavaProjects - Lab2.java
JavaProjects src \ src \ Lab2 main
Run: Lab2
Задание 3:
Расположить 8 ферзей:
0 [0]=0 [1]=4 [2]=7 [3]=5 [4]=2 [5]=6 [6]=1 [7]=3
1 [0]=0 [1]=5 [2]=7 [3]=2 [4]=6 [5]=3 [6]=1 [7]=4
2 [0]=0 [1]=6 [2]=3 [3]=5 [4]=7 [5]=1 [6]=4 [7]=2
3 [0]=0 [1]=6 [2]=4 [3]=7 [4]=1 [5]=3 [6]=5 [7]=2
4 [0]=1 [1]=3 [2]=5 [3]=7 [4]=2 [5]=0 [6]=6 [7]=4
5 [0]=1 [1]=4 [2]=6 [3]=0 [4]=2 [5]=7 [6]=5 [7]=3
6 [0]=1 [1]=4 [2]=6 [3]=3 [4]=0 [5]=7 [6]=5 [7]=2
7 [0]=1 [1]=5 [2]=0 [3]=6 [4]=3 [5]=7 [6]=2 [7]=4
8 [0]=1 [1]=5 [2]=7 [3]=2 [4]=0 [5]=3 [6]=6 [7]=4
9 [0]=1 [1]=6 [2]=2 [3]=5 [4]=7 [5]=4 [6]=0 [7]=3
10 [0]=1 [1]=6 [2]=4 [3]=7 [4]=0 [5]=3 [6]=5 [7]=2
11 [0]=1 [1]=7 [2]=5 [3]=0 [4]=2 [5]=4 [6]=6 [7]=3
12 [0]=2 [1]=0 [2]=6 [3]=4 [4]=7 [5]=1 [6]=3 [7]=5
13 [0]=2 [1]=4 [2]=1 [3]=7 [4]=0 [5]=6 [6]=3 [7]=5
14 [0]=2 [1]=4 [2]=1 [3]=7 [4]=5 [5]=3 [6]=6 [7]=0
15 [0]=2 [1]=4 [2]=6 [3]=0 [4]=3 [5]=1 [6]=7 [7]=5
16 [0]=2 [1]=4 [2]=7 [3]=3 [4]=0 [5]=6 [6]=1 [7]=5
17 [0]=2 [1]=5 [2]=1 [3]=4 [4]=7 [5]=0 [6]=6 [7]=3
18 [0]=2 [1]=5 [2]=1 [3]=6 [4]=0 [5]=3 [6]=7 [7]=4
19 [0]=2 [1]=5 [2]=1 [3]=6 [4]=4 [5]=0 [6]=7 [7]=3
20 [0]=2 [1]=5 [2]=3 [3]=0 [4]=7 [5]=4 [6]=6 [7]=1
21 [0]=2 [1]=5 [2]=3 [3]=1 [4]=7 [5]=4 [6]=6 [7]=0
22 [0]=2 [1]=5 [2]=7 [3]=0 [4]=3 [5]=6 [6]=4 [7]=1
23 [0]=2 [1]=5 [2]=7 [3]=0 [4]=4 [5]=6 [6]=1 [7]=3
24 [0]=2 [1]=5 [2]=7 [3]=1 [4]=3 [5]=0 [6]=6 [7]=4
25 [0]=2 [1]=6 [2]=1 [3]=7 [4]=4 [5]=0 [6]=3 [7]=5
26 [0]=2 [1]=6 [2]=1 [3]=7 [4]=5 [5]=3 [6]=0 [7]=4
27 [0]=2 [1]=7 [2]=3 [3]=6 [4]=0 [5]=5 [6]=1 [7]=4
28 [0]=3 [1]=0 [2]=4 [3]=7 [4]=1 [5]=6 [6]=2 [7]=5
29 [0]=3 [1]=0 [2]=4 [3]=7 [4]=5 [5]=2 [6]=6 [7]=1
30 [0]=3 [1]=1 [2]=4 [3]=7 [4]=5 [5]=0 [6]=2 [7]=6
31 [0]=3 [1]=1 [2]=6 [3]=2 [4]=5 [5]=7 [6]=0 [7]=4
32 [0]=3 [1]=1 [2]=6 [3]=2 [4]=5 [5]=7 [6]=4 [7]=0
33 [0]=3 [1]=1 [2]=6 [3]=4 [4]=0 [5]=7 [6]=5 [7]=2

Event Log
All files are up-to-date (a minute ago) 281:1 CRLF UTF-8 4 spaces master
```

Вывод

В ходе данной лабораторной работы мы изучили методы поиска, реализовали их на языке программирования java и сравнили скорость этих методов.