

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 «Математической кибернетики и информационных технологий»

Лабораторная работа 2. Методы поиска.

Выполнил:

студент группы БВТ1901

Амирбеков А.Э

Проверил:

Мелехин А.

Описание

Реализовать методы поиска в соответствии с заданием. Организовать генерацию начального набора случайных данных. Для всех вариантов

Москва
2021

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.

Код

```
import java.util.*;

public class Hash1 {
    Map<Integer, Integer> dictionary = new HashMap<Integer, Integer>();

    public void insert(int a){
        int i = Hash(a);
        if(dictionary.get(i)==null) {
            dictionary.put(i,a);
        }else {
            while (i<=200){
                if(dictionary.get(i)==null) {
                    dictionary.put(i,a);
                    break;
                }else i++;
            }
            if (i==201) System.out.println("Таблица переполнена");
        }
    }

    public Hash1(int[] arr){
        for (int item: arr) this.insert(item);
    }

    public Hash1(){}

    public boolean exists(int a){
        int i = Hash(a);
        if(dictionary.get(i)==null) return false;
        if (dictionary.get(i)==a){return true;}else {
            while (i <= 200){
                if(dictionary.get(i)==a){
                    return true;
                }else i++;
            }
        }
        return false;
    }

    public void print(){

        Set<Integer> numbersSet = dictionary.keySet();

        List<Integer> numbersList = new ArrayList<Integer>(numbersSet) ;
        //set -> list

        Collections.sort(numbersList); //Sort the list
    }
}
```

```

        for (int key: numbersList) {
            System.out.println(key+" "+dictionary.get(key));
        }
    }

    public static int Hash(int digit){return digit%15;}
}
import java.util.*;

public class HashTable {

    Map<Integer, LinkedList<Integer>> dictionary = new HashMap<Integer,
LinkedList<Integer>>();

    public HashTable(int[] arr){
        for (int item: arr) this.insert(item);
    }

    public HashTable(){}

    public void insert(int a){
        LinkedList<Integer> ly= new LinkedList<Integer>();
        if (dictionary.get(Hash(a))==null){}else{
            ly = dictionary.get(Hash(a));
            ly.add(a);
            dictionary.put(Hash(a),ly);
        }

        public boolean exists(int a){
            if (dictionary.get(Hash(a))==null) return false;
            LinkedList<Integer> ly = dictionary.get(Hash(a));
            for (Integer integer : ly) {
                if ( integer == a) return true;
            }
            return false;
        }

        public void print(){Set<Integer> numbersSet = dictionary.keySet();

            List<Integer> numbersList = new ArrayList<Integer>(numbersSet) ;
//set -> list

            Collections.sort(numbersList);//Sort the list

            for (int key: numbersList) {
                System.out.println(key+" "+(dictionary.get(key)));
            }
        }

        public static int Hash(int digit){return digit%15;}
    }
}
public class Node {

    public int key;
    public Node leftChild;
    public Node rightChild;
    public void printNode(){
        System.out.println("KEY " + key);
    }
}

```

```

import com.sun.source.tree.BinaryTree;

import java.util.*;

public class Search {

    public static void main(String[] args) {
        final Random random = new Random();
        Scanner scanner = new Scanner(System.in);
        System.out.println("vvedite kol-vo elementov massiva");
        int n = scanner.nextInt();
        int[] mas = create(n);
        int item = mas[random.nextInt(n)];
        System.out.println(Arrays.toString(mas)+" ищем:"+ item);

        long t1= System.currentTimeMillis();
        System.out.println("index: " + BinarySearch(mas, item));
        long t2= System.currentTimeMillis();
        System.out.println("Binsearch: " + (t2-t1));
        System.out.println();
        t1= System.currentTimeMillis();
        System.out.println("index: " + Interpol(mas, item));
        t2= System.currentTimeMillis();
        System.out.println("Interpol: " + (t2-t1));
        System.out.println();
        BinTreeSearch(mas, item);
        System.out.println();
        t1= System.currentTimeMillis();
        System.out.println("index: " + (FibonacciSearch(mas, item)));
        t2= System.currentTimeMillis();
        System.out.println("Fibsearch: " + (t2-t1));
        System.out.println();
        System.out.println(Hashing1(mas,item));
        System.out.println();
        System.out.println(Hashing(mas,item));

    }

    public static int[] create(int n){
        final Random random = new Random();
        int[] a = new int[n];
        for (int i = 0;i<a.length;i++){
            a[i]= random.nextInt(1000);
        }

        Arrays.sort(a);

        return a;
    }

    public static int BinarySearch(int[] arr, int i){

        int start = 0;
        int end = arr.length - 1;
        return Bi(arr, start, end, i);
    }

    public static int Bi(int[] arr,int start,int end,int i){
        if (end==0) return end;
        if (i == arr[end]) return end;
        int middle = start + (end - start)/2;
        if(arr[middle]<i) {start = middle;}
        else if(arr[middle]>i){end = middle;}else return middle;
    }
}

```

```

        return Bi(arr, start, end, i);
    }

    public static int Interpol(int[] arr, int i){
        int start = 0;
        int end = arr.length - 1;
        return Inter(arr, start, end, i);
    }

    public static int Inter(int[] arr, int start, int end, int i){
        if (end==0) return end;
        if (i == arr[end]) return end;
        int middle = start + ((i-arr[start])*(end-start))/(arr[end]-
arr[start]);
        if(arr[middle]<i) {start = middle+1;}
        else if(arr[middle]>i){end = middle-1;}else return middle;
        return Inter(arr, start, end, i);
    }

    public static boolean Hashing(int [] arr, int i){
        long t1 = System.currentTimeMillis();

        HashTable HT = new HashTable(arr);

        boolean ka = HT.exists(i);
        long t2= System.currentTimeMillis();
        System.out.println("hash цепочками: " + (t2-t1));
        HT.print();
        return ka;
    }

    public static boolean Hashing1(int[] arr, int i){
        long t1 = System.currentTimeMillis();

        Hash1 HT = new Hash1(arr);

        boolean ka = HT.exists(i);
        long t2= System.currentTimeMillis();
        System.out.println("hash простой: " + (t2-t1));
        HT.print();
        return ka;
    }

    public static void BinTreeSearch(int [] arr, int i){

        long t1= System.currentTimeMillis();

        Tree tree = new Tree();

        for (int item: arr) tree.insert(item);

        Node a = tree.find(i);
        long t2= System.currentTimeMillis();
        System.out.println("bintree: " + (t2-t1));
        a.printNode();
    }

    public static int FibonacciSearch(int [] mas, int ind){
        int n = mas.length;
        int k = 0;
        int indtmp = ind;
        while (Fibonacci(k+1)<n+1) k++;
    }

```

```

        int M = Fibonacci(k+1) - (n+1);
        int i = Fibonacci(k)-M;
        int p = Fibonacci(k-1);
        int q = Fibonacci(k-2);
        return FibS(mas,i,q,p,indtmp);
    }

    public static int FibS(int[] mas, int i, int q, int p,int item){

        if (i < 0){
            if ((p != 1) ){i+=q;p-=q;q-=p; return FibS(mas,i,q,p,item);}else
{return -1;}//5
        } else {

            if (i >= mas.length) {
                if ((q != 0)) {
                    i -= q;
                    int tmp = q;
                    q = p - tmp;
                    p = tmp;
                    return FibS(mas, i, q, p, item);
                } else {
                    return -1;
                }
            } else {
                if (item < mas[i]) {
                    if ((q != 0)) {
                        i -= q;
                        int tmp = q;
                        q = p - tmp;
                        p = tmp;
                        return FibS(mas, i, q, p, item);
                    } else {
                        return -1;
                    }
                } else {
                    if (item > mas[i]) {
                        if ((p != 1)) {
                            i += q;
                            p -= q;
                            q -= p;
                            return FibS(mas, i, q, p, item);
                        } else {
                            return -1;
                        }
                    }
                }
            }

            if (item == mas[i]) return i;
        }
    }

    return FibS(mas, i ,q,p,item);
}

public static int Fibonacci(int i){
    if (i==0) return 0;
    if (i==1) return 1;
    return Fibonacci(i-1) + Fibonacci(i-2);
}

```

```

public static int[] addd(int[] a,int n){
    int[] b = new int[a.length+1];
    b = Arrays.copyOf(a, b.length);
    b[b.length-1]= n;
    Arrays.sort(b);
    return b;
}

public static int[] drop(int[] a, int n){
    int[] b = new int[a.length-1];
    System.arraycopy(a, 0,b,0,n );
    System.arraycopy(a,n+1,b,n,a.length-n-1);
    Arrays.sort(b);
    return b;
}
}
public class Tree{

    Node root;

    public Node find(int key){
        Node current = root;
        while(current.key!=key){
            if(current.key<key){
                current = current.rightChild;
            }else{
                current = current.leftChild;
            }
            if(current==null){
                return null;
            }
        }
        return current;
    }

    public void insert(int key){
        Node node = new Node();
        node.key = key;
        if(root==null){
            root = node;
        }else{
            Node current = root;
            Node prev = null;
            while (true){
                prev = current;
                if(key<prev.key){
                    current = current.leftChild;
                    if(current==null){
                        prev.leftChild = node;
                        return;
                    }
                }else{
                    current = current.rightChild;
                    if(current==null){
                        prev.rightChild = node;
                        return;
                    }
                }
            }
        }
    }
}

```

```

        public void print(Node startNode){
            if(startNode != null){//условие сработает, когда мы достигнем конца
                дерева и потомков не останется
                print(startNode.leftChild);//рекурсивно вызываем левых потомков
                startNode.printNode();//вызов метода принт
                print(startNode.rightChild);//вызов правых
            }
        }
    }
}
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Chess {
    public static void main(String[] args) {
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                board[i][j]=0;
            }
        }
        solve(0);
    }
    static int[][] board = new int[8][8];

    public static void setQueen(int i,int j){
        for (int x = 0; x < 8; x++) {
            board[x][j]+=1;
            board[i][x]+=1;
            if((0 <= i+j-x)&&(i+j-x<8)) board[i+j-x][x]+=1;
            if((0<=i-j+x)&&(i-j+x<8)) board[i-j+x][x]+=1;
        }
        board[i][j] = -1;
    }

    public static void dropQueen(int i,int j){
        for (int x = 0; x < 8; x++) {
            board[x][j]-=1;
            board[i][x]-=1;
            if((0 <= i+j-x)&&(i+j-x<8)) board[i+j-x][x]-=1;
            if((0<=i-j+x)&&(i-j+x<8)) board[i-j+x][x]-=1;
        }
        board[i][j] = 0;
    }

    public static void printPos() {
        List<String> ans = new ArrayList<String>();
        String abc = "abcdefgh";
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                if (board[i][j]==-1){
                    ans.add(abc.charAt(j)+Integer.toString(i+1));
                }
            }
        }
        System.out.println(ans);
    }
    int ch = 0;
    public static void solve(int i){
        for (int j = 0; j < 8; j++) {

```



```

        if (board[i][j]==0) {
            setQueen(i,j);
            if(i==7) {
                printPos();
            }else{
                solve(i+1);
            }
            dropQueen(i,j);
        }
    }
}

```

Вывод

Научились использовать различные методы поиска