**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Математической кибернетики и информационных технологий

**Отчет по лабораторной работе**

по предмету «СИАОД»

на тему:

«Методы поиска»

Выполнил: студент группы

Митрохин Ярослав Игоревич

Руководитель:

Кутейников Иван Александрович

Москва 2020

*Цель работы:* реализовать заданный метод поиска в соответствии с индивидуальным заданием.

*Ход работы:*

1. *Код программы*

import java.util.Arrays;  
import java.util.Random;  
  
public class lab2 {  
 public static void main(String[] args)  
 {  
 int[] array = new int[99];  
 for (int i = 0; i < array.length; i++){  
 array[i] = (int) (Math.*random*()\*100);  
 }  
 Arrays.*sort*(array);  
 for (int i = 0; i < array.length; i++){  
 System.*out*.print(array[i]+" ");  
 }  
  
 int first = 0;  
 int last = array.length-1;  
 System.*out*.println();  
 //int index = binarySear(array, 32);  
 //System.out.println(32 + " Index:"+ index);  
 double time = System.*nanoTime*();  
 //System.out.println(y + " Index:" + binarySearch(array, y, first, last));  
 *binarySearch*(array, y, first,last);  
 System.*out*.println("Time: " + (System.*nanoTime*()-time)/1000000);  
 time = 0 ;  
 System.*out*.println(y + " Index:" + *binarySearch*(array, y, first, last));  
 double time1 = System.*nanoTime*();  
// System.out.println(y + " Index:" + linearSearch(array, y));  
 *linearSearch*(array,y);  
 System.*out*.println("Time: " + (System.*nanoTime*()-time1)/1000000);  
 time1 = 0 ;  
 System.*out*.println(y + " Index:" + *linearSearch*(array, y));

1. }  
     
     
    public static int[] createArray(int size, int countRand)  
    {  
    Random random = new Random();  
    int[] arr = new int[size];  
    for (int i = 0; i < size; i++) arr[i] = i;  
    for (int i = size-1; i > 0; i--){  
    int position = random.nextInt(i+1);  
    int a = arr[position];  
    arr[position] = arr[i];  
    arr[i] = a;  
    }  
    return Arrays.*copyOf*(arr, countRand);  
    }  
     
    public static int binarySearch(int[] sortedArray,int key, int low,int high)  
    {  
    int index = -1;  
    while (low <= high)  
    {  
    int mid = (low+high)/2;  
    if (sortedArray[mid] < key)  
    {  
    low = mid + 1;  
    } else if (sortedArray[mid] > key)  
    {  
    high = mid - 1;  
    } else if (sortedArray[mid] == key) {  
    index = mid;  
    break;  
    }  
    }  
    return index;  
    }  
     
    public static int linearSearch(int arr[], int element)  
    {  
    for (int index = 0; index < arr.length; index++)  
    {  
    if (arr[index] == element)  
    return index;  
    }  
    return -1;  
    }
2. Реализовал метод добавления элемента в массив:

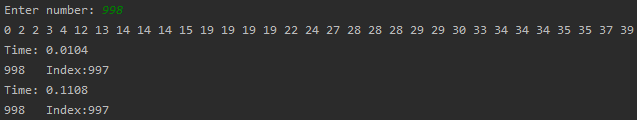
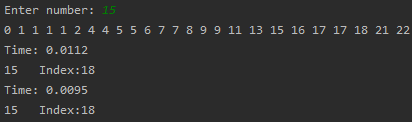
public static int[] addEl(int[] arr, int element, int index)  
{  
 int[] temp = new int[arr.length+1];  
 if (index > arr.length || index < 0)  
 {  
 System.*out*.println("Wrong element");  
 }  
 for (int i =0; i < arr.length; i++)  
 {  
 if (i < index)  
 {  
 temp[i]= arr[i];  
 } else {  
 temp[i+1] = arr[i];  
 }  
 }  
 temp[index] = element;  
 return temp;  
}

1. Реализовал метод удаления элемента из массива:

public static void deleteEl(int arr[], int element)  
{for(int i = element+1; i<arr.length-1;i++)  
 {arr[i-1] = arr[i];}}

1. *Результаты программы*

Выводится отсортированный массив(так как бинарный поиск не работает на неотсортированном массиве). Первый выполняется метод бинарного поиска, второй метод линейного поиска.

** **

*Выводы:* реализовал метод поиска в соответствии с индивидуальным заданием. В качестве стандартного метода поиска был взят метод линейного поиска. Проанализировал время выполнения обоих методов и сделал вывод. Что метод линейного поиска(0.0095мс) гораздо быстрее, чем метод бинарного поиска(0.0112мс), если массив недостаточно большой. Однако при массиве, состоящим из 1000 элементов, метод бинарного поиска(0,01мс) работает быстрее, чем метод линейного поиска(0,11мс).