**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

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

Кафедра Информатики

****

**Отчет по лабораторной работе № 2**

по дисциплине «СиАОД»

на тему:

«**Методы поиска**»

Выполнила: студентка группы БВТ1802

Лаврухина Елена Павловна

Руководитель:

Кутейников Иван Алексеевич

Москва 2020

# **Цель работы**

Реализовать заданный метод поиска в соответствии с индивидуальным 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.

Выполнение

Код программы

**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStreamReader;  
**import** java.util.ArrayList;  
**import** java.util.Collections;  
**import** java.util.List;  
**import** java.util.Random;  
  
**public class** FibbSearch {  
 **public static double** *timer*=0.0;  
 **public static** List<Integer> *array*;  
 **public static int**[] *indexes* = **new int**[100];  
 **public static void** main(String[] args) **throws** IOException {  
 *// создание и сортировка списка  
 array* = **new** ArrayList<>();  
 Random random = **new** Random();  
 **for** (**int** i = 0; i<100; i++) {  
 *array*.add(random.ints(0,(100+1)).findFirst().getAsInt());  
 }  
 System.***out***.println(*array*);  
 Collections.*sort*(*array*);  
 System.***out***.println(*array*);  
 *// ввод номера для нахождения* BufferedReader bufferedReader = **new** BufferedReader(**new** InputStreamReader(System.***in***));  
 **int** n = Integer.*parseInt*(bufferedReader.readLine());  
 *// Фибоначчиев поиск* System.***out***.println(**"Фибоначчиев:"**);  
 *time*();  
 *FibbNums*();  
 **int** indexN = *FibSearch*(0,n);  
 System.***out***.println(indexN);  
 *time*();  
 *// поиск IJ* System.***out***.println(**"Стандартный поиск:"**);  
 *time*();  
 indexN=*array*.indexOf(n);  
 System.***out***.println(indexN);  
 *time*();  
 *// удаление* System.***out***.println(**"Удаление найденного элемента:"**);  
 *array*.remove(indexN);  
 System.***out***.println(*array*);  
 }  
 **public static void** FibbNums() {  
 *indexes*[0]=1;  
 *indexes*[1]=1;  
 **for** (**int** i=2; i<*indexes*.**length**; i++) {  
 *indexes*[i] = *indexes*[i-1]+*indexes*[i-2];}  
 }  
 **public static void** time() {  
 **if** (*timer* == 0) *timer*=System.*nanoTime*();  
 **else** {  
 **double** val = (**double**)(System.*nanoTime*()-*timer*);  
 System.***out***.println(val/1000000);  
 *timer*=0;  
 }  
 }  
 **public static int** FibSearch(**int** firstint, **int** n) {  
 **int** index=-1;  
 **if** (*array*.get(firstint) == n) **return** firstint;  
 **for**(**int** i=1;i<*indexes*.**length**;i++) {  
 **if** (firstint+*indexes*[i]>*array*.size())  
 **if** (*array*.get(*array*.size()-1)<n) **return** -1;  
 **else** {  
 **return** *FibSearch*(firstint+*indexes*[i-1],n);  
 }  
 **if** (*array*.get(firstint+*indexes*[i]) == n) **return** firstint+*indexes*[i];  
 **if** (*array*.get(firstint+*indexes*[i])>n) {  
 **if** (I == 1) **return** -1;  
 **return** *FibSearch*(firstint+*indexes*[i-1],n);  
 }  
 }  
 **return** index;  
 }  
}

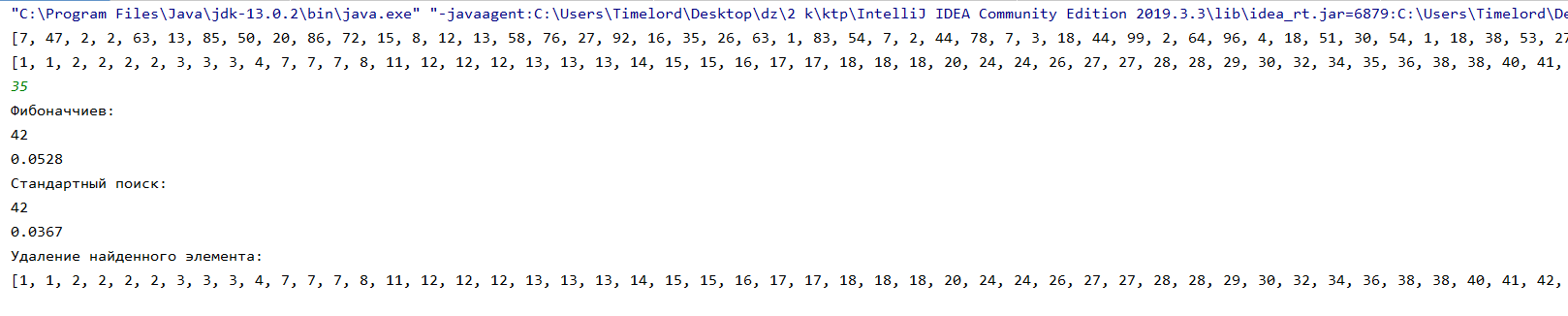
**Скриншот работы программы**

Рисунок 1 – Работа программы

Сравнение времени работы используемых функций поиска

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Индекс искомого элемента** | | 6 | 29 | 42 | 70 | 95 | **Среднее время** |
| **Время** | **Фибоначчиев** | 0,0479 | 0,0608 | 0,0528 | 0,0625 | 0,0541 | 0,05562 |
| **Стандартный** | 0,0428 | 0,0433 | 0,0367 | 0,0453 | 0,0462 | 0,04286 |

# **Вывод**

В ходе работы были изучены методы поиска элементов массива с помощью Фибоначчиева и стандартного поиска. Исходя из полученных результатов, можно сделать вывод о том, что Фибоначчиев поиск медленнее стандартного.