数据结构与算法1-线性查找法

笔记本:	我的笔记		
创建时间:	2020/9/18 21:40	**更新时间:**	2020/10/3 10:05
作者:	liuhouer		
标签:	算法		

大纲

算法与数据结构体系课程

排序算法

插入，冒泡，选择，希尔

快速，归并，堆排序

计数排序，桶排序，基数排序

查找算法

线性查找，二分查找

线性数据结构

动态数组，链表，

栈，队列，哈希表

经典树结构

二分搜索树，堆，

AVL，红黑树，B 类树

高级数据结构

线段树，并查集，Trie，

SQRT 分解

字符串算法

KMP

模式匹配

算法复杂度整理

常见算法复杂度

$$O(1) < O(logn) < O(\sqrt{n}) < O(n) < O(nlogn) < O(n^2) < O(2^n) < O(n!)$$

log指的是对数，和连成相反，是连除某个值 例如：$\log_3 27 = 3$

1.线性查找法

在一沓试卷中，找到属于自己的那张试卷
在 data 数组中查找 16

```
public class LinearSearch {


    private LinearSearch(){}


    public static <E> int search(E[] data, E target){


        for(int i = 0; i < data.length; i ++)
            if(data[i].equals(target))
                return i;
```

```
        return -1;
    }


    public static void main(String[] args){


        Integer[] data = {24, 18, 12, 9, 16, 66, 32, 4};


        int res = LinearSearch.search(data, 16);
        System.out.println(res);


        int res2 = LinearSearch.search(data, 666);
        System.out.println(res2);




        Student[] students = {new Student("Alice"),
                              new Student("Bobo"),
                              new Student("Charles")};
        Student bobo = new Student("Bobo");
        int res3 = LinearSearch.search(students, bobo);
        System.out.println(res3);
    }


    或者测试性能
public static void main(String[] args){




//        int n = 10000;
//        Integer[] data = ArrayGenerator.generateOrderedArray(n);
//
//        long start = System.currentTimeMillis();
//        for (int k = 0; k < 100; k++)
//            LinearSearch.search(data, n);
//        long time = System.currentTimeMillis() - start;
//        System.out.println("n = " + n + " , 100 runs : " + time + "ms");




        int[] dataSize = {1000000, 10000000};
        for(int n: dataSize) {
            Integer[] data = ArrayGenerator.generateOrderedArray(n);



            long startTime = System.nanoTime();
            for (int k = 0; k < 100; k++)
                LinearSearch.search(data, n);
            long endTime = System.nanoTime();



            double time = (endTime - startTime) / 1000000000.0;
            System.out.println("n = " + n + ", 100 runs : " + time + "s");
        }
    }
}
```

```
public class Student {
```

```
    private String name;


    public Student(String name){
        this.name = name;
    }


    @Override
    public boolean equals(Object student){


        if(this == student)
            return true;


        if(student == null)
            return false;


        if(this.getClass() != student.getClass())
            return false;


        Student another = (Student)student;
        return this.name.equals(another.name);
    }
}
```