二分查找/插值查找/黄金分割点查找

二分查找,插值查找和黄金分割点查找都是基于有序数组的查找

本质是选择一个index索引,将要查找的数组分解,缩小数组范围,当范围缩小到最小的时候,就找到了

二分查找是将索引定在数组的中间,每次排除一半的数组.

$$mid = \frac{low+high}{2} = low+\frac{1}{2}(high-low)$$

插值查找是根据查找值大小在数组长度的相对位置,进行动态确定索引,对于分布均匀的数组,效率很高

$$mid = low + \frac{key - a[low]}{a[high]-a[low]}(high - low)$$

黄金分割点查找,根据斐波拉契数列确定黄金分割点,然后

代码实现

二分查找

```
package search查找;

/**
 * @author jndeng
 * @create 2019-12-26 21:43
 */
public class BinarySearch {
    public static void main(String[] args) {
        int[] arr = new int[100];
        for (int i = 0; i < arr.length; i++) {
            arr[i]=i+1;
        }
        int index = binarySearch(arr, 0, arr.length - 1, 1);
        System.out.println(index);
    }

    static int binarySearch(int[] arr, int left, int right, int value) {
        System.out.println("执行了~");
        if (right >= left) {
            int mid=   (left+right)/2;
            int midValue = arr[mid];
            if (value > midValue) {
                return binarySearch(arr, mid + 1, right, value);
```

```
            }
            if (value < midValue) {
                return binarySearch(arr, left, mid - 1, value);
            } else {
                return mid;
            }
        } else {
            return -1;
        }

    }
}
```

插值查找

```
package search查找;

import sun.security.util.Length;

/**
 * @author jndeng
 * @create 2019-12-27 11:05
 */
public class InsertSearch {
    public static void main(String[] args) {
        int[] arr = new int[100];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = i + 1;
        }
        int index = insertSearch(arr, 0, arr.length - 1, 1);
        System.out.println(index);
    }

    static int insertSearch(int[] arr, int left, int right, int value) {

        System.out.println("执行了~");
        if (left > right || value < arr[0] || value > arr[arr.length - 1]) {
            return -1;
        }
        int mid = left + (value - arr[left]) * (right - left) / (arr[right] - arr[left]);
        int midValue = arr[mid];
        if (value > midValue) {
            return insertSearch(arr, mid + 1, right, value);
        }
        if (value < midValue) {
            return insertSearch(arr, left, mid - 1, value);
        } else {
            return mid;
```

```
        }


    }
}
```