二分法查找的前提是数组已经排好序了

1.适用于无重复复数的版本

```
public class Binary_search {

    public static int bi_search(int a [] , int n, int key) { //非递归
        int low = 0, high = n - 1, mid;
        while (low <= high) {
            mid = low + (high - low) / 2;
            if(a[mid] == key)
                return mid;
            else if (a[mid] > key) {
                high = mid - 1;
            }else {
                low = mid + 1;
            }
        }
        return -1;
    }

    public static int binary_search_recusion(int a [], int low, int high, int key) { //递归
        if(low > high)
            return -1;
        int mid = low + (high - low) / 2;
        if(a[mid] == key)
            return mid;
        else if(a[mid] > key)
            return binary_search_recusion(a, low, mid - 1, key);
        else
            return binary_search_recusion(a, mid + 1, high, key);
    }

    public static void main(String [] args) {
        int [] array = {1,2,3,4,5,6};
        System.out.println(bi_search(array, 6, 3));
        System.out.println(binary_search_recusion(array, 0, 6, 2));
    }
}
```

2.适用于有重复复数的版本

```
    public static int binary_search_repeat_data(int [] array, int target) { //带重复数据的二分查找，其目的是为了找到第一个相同
        int left = 0;
        int right = array.length - 1;
        while (left <= right) {
            int mid = (left + (right - left) / 2);
            if(target == array[mid]) { //如果二分查找找到一样的了，需要往前推，看第一个相同的是哪个位置
                while (mid >= 0) {
                    if(array[mid] != target) {  //不等于，证明过了一个数，即到5的部分了
                        break;
                    }
                    mid --;   //一直往前找
                }
                if(mid <= -1) {
                    return 0;
                }
                return mid + 1;    //因为多减了一次，返回的时候需要加上1
            }else if (target < array[mid]) {  //假如二分法没有找到，则分左右两半部分查找。
                right = mid - 1;
            }else {
                left = mid + 1;
            }
        }
        return -1;
    }



    public static void main(String [] args) {
        int [] array = {1,2,3,4,5,6,6};
    //  System.out.println(bi_search(array, 6, 3));
    //  System.out.println(binary_search_recusion(array, 0, 6, 2));
        System.out.println(binary_search_repeat_data(array, 6));
```