二分查找算法

二分查找，也称为折半查找，是指在**有序的**数组里找出指定的值，返回该值在数组中的索引。

查找步骤如下：

（1）从有序数组的最中间元素开始查找，如果该元素正好是指定查找的值，则查找过程结束。否则进行下一步;

（2）如果指定要查找的元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半区域查找，然后重复第一步的操作;

（3）重复以上过程，直到找到目标元素的索引，查找成功;或者直到子数组为空，查找失败。

优点是比较次数少，查找速度快，平均性能好；

缺点是要求待查表为有序表，且插入删除困难。

因此，折半查找方法适用于不经常变动而查找频繁的有序列表。

```
function binary(arr,low,high,key){
    if(low<=high){
        if(arr[low]==key) return low;
        if(arr[high]==key) return high;
        var mid=Math.ceil((high+low)/2);
        if(arr[mid]==key){
            return mid;
        }else if(arr[mid]>key){
            return binary(arr,low,mid-1,key);
        }else{
            return binary(arr,mid+1,high,key);
        }
    }
    return -1;
}

function binary_search(arr,key){
    var low=0;
    var high=arr.length-1;
    if(arr[low]==key) return low;
    if(arr[high]==key) return high;
    while(low<=high){
        var mid=Math.ceil((low+high)/2);
        if(arr[mid]==key){
            return mid;
        }else if(arr[mid]>key){
            high=mid-1;
        }else if(arr[mid]<key){
            low=mid+1;
        }else{
            return -1;
        }
    }
}

function count(arr,key){//查找重复的位置
    var countArr=[];
    var position=binary_search(arr,key);//找出值所在位置
    if(position>-1){
      for(var i=position-1;i>0;i--){
        if(arr[i]==key){
         countArr.unshift(i);
        }else{
          break;
        }
      }
      countArr.push(position);

      for(var i=position+1;i<arr.length;i++){
        if(arr[i]==key){
          countArr.push(i);
        }else{
           break;
        }
      }
    }
    return countArr;
  }
```

```
    var Arr = [3, 5, 6, 6, 9, 12, 15];
    console.log(binary( Arr, 0, Arr.length - 1,6));
    console.log(binary_search( Arr,6));
console.log(count(Arr,6));
```