先折半查找出元素待插入的位置，然后再统一的移动之后的关键字。

比较次数

减少了关键字的比较次数O(),比较次数与待排序表的初始状态无关，仅取决于关键字个数。

移动次数

移动次数没有改变，还依赖于待排序表的初始状态

时间复杂度O(n2)

代码

public static void biSearch(int[] nums){

for(int i=1;i<nums.length;i++){

//二分法查找要插入的位置

int low = 0;

int high= i-1;

while(low<=high){

int mid = (low+high)/2;

//注意这里不能单独把等于列出来，因为找的是要插入的位置，不是查找该值的索引

if(nums[i]>nums[mid])

low=mid+1;

else

high=mid-1;

}

//此时的low是要插入的位置(或者high+1)

//low之后的数据后移1位

int temp = nums[i]; //把nums[i]当前关键字赋值到temp中保存避免被覆盖

int j =i-1; //j为i之前的最后一个关键字

while(j>=low){ //low到j所有关键字后移一定要从后向前，要么覆盖数据

nums[j+1]=nums[j];

j--;

}

nums[low]=temp;//把当前关键字插入到low位置上

}

}