**数组A[1..n]中含有n个互不相同的整数元素。对A中的元素A[i]（1≤i≤n），若有A[i]<A[i-1]并且A[i]<A[i+1]，则称A[i]为A的局部最小元素，即局部最小元素是比其两个相邻元素都小的元素（注：在边界上，即i=1或i=n时，只需考虑一侧的邻居即可）。例：如果A={5,3,4,1,2}，那么A有二个局部最小元素3和1；而若A={1,2,3,4,5}，那么A就只有一个局部最小值元素1。**

**请设计一个时间复杂度为O(logn)的算法输出A中的一个局部最小元素（当有多个局部最小元素时，输出任意一个即可），给出算法的伪代码描述，并证明你的算法关于时间复杂度的结论。**

**可以使用二分法求解。**

1. 首先判断头或者尾是不是局部最小元素，如果是，则可以直接返回
2. 如果头和尾都不符合，则可以用二分法查找到局部最小元素

初始时，左边界l=0, 右边界r=n-1;

取中间位置mid, 对应的值为A[mid]

则可以根据如下规则查找：

1. 如果中间位置满足A[mid]<A[mid-1]&&A[mid]<A[mid+1]，则此时的A[mid]就是符合条件的局部最小元素, 返回这个元素即可，程序结束
2. 如果A[mid]>A[mid-1]，则可以判定在[l,mid-1]范围内一定有局部最小元素，修改右边界r=mid-1
3. 如果A[mid]>A[mid+1]，则可以判定在[mid+1,r]范围内一定有局部最小元素

修改左边界l=mid+1

**时间复杂度分析：**

1. 判断特殊情况的复杂度：2\*O(1)
2. 二分查找的过程，每次都可缩减到原规模的1/2，复杂度为log(n)

**因此，总的时间度为O(logn)**

**完整伪代码如下：**

int findPartMin(vector<int> A,int n){

if(A[0]<A[1]){//特殊情况1：头部元素就是局部最小元素

return A[0];

}

else if(A[n-1]<A[n-2]){//特殊情况2：尾部元素就是局部最小元素

return A[n-1];

}

else{//如果头尾都不是局部最小元素，则二分查找局部最小元素

int l=0;

int r=n-1;

while(l<=r){//二分查找

int mid=l+(r-l)/2;//中间元素位置

if(A[mid]<A[mid-1]&&A[mid]<A[mid+1]){

return A[mid]; //如果中间元素符合则可以直接返回这个中间元素

}

else if(A[mid]>A[mid-1]){

r=mid-1; //此时可以判定在[l,mid-1]范围内一定有局部最小元素

}

else if(A[mid]>A[mid+1]){

l=mid+1; //此时可以判定在[mid+1,r]范围内一定有局部最小元素

}

}

}

}