

Merge Sort L



Quicksort



改进插入排序算法(第三章ppt No.6)，在插入元素a(i)时使用二分查找代替顺序查找，将这个算法记做ModInsertSort，估计算法在最坏情况下的时间复杂度。

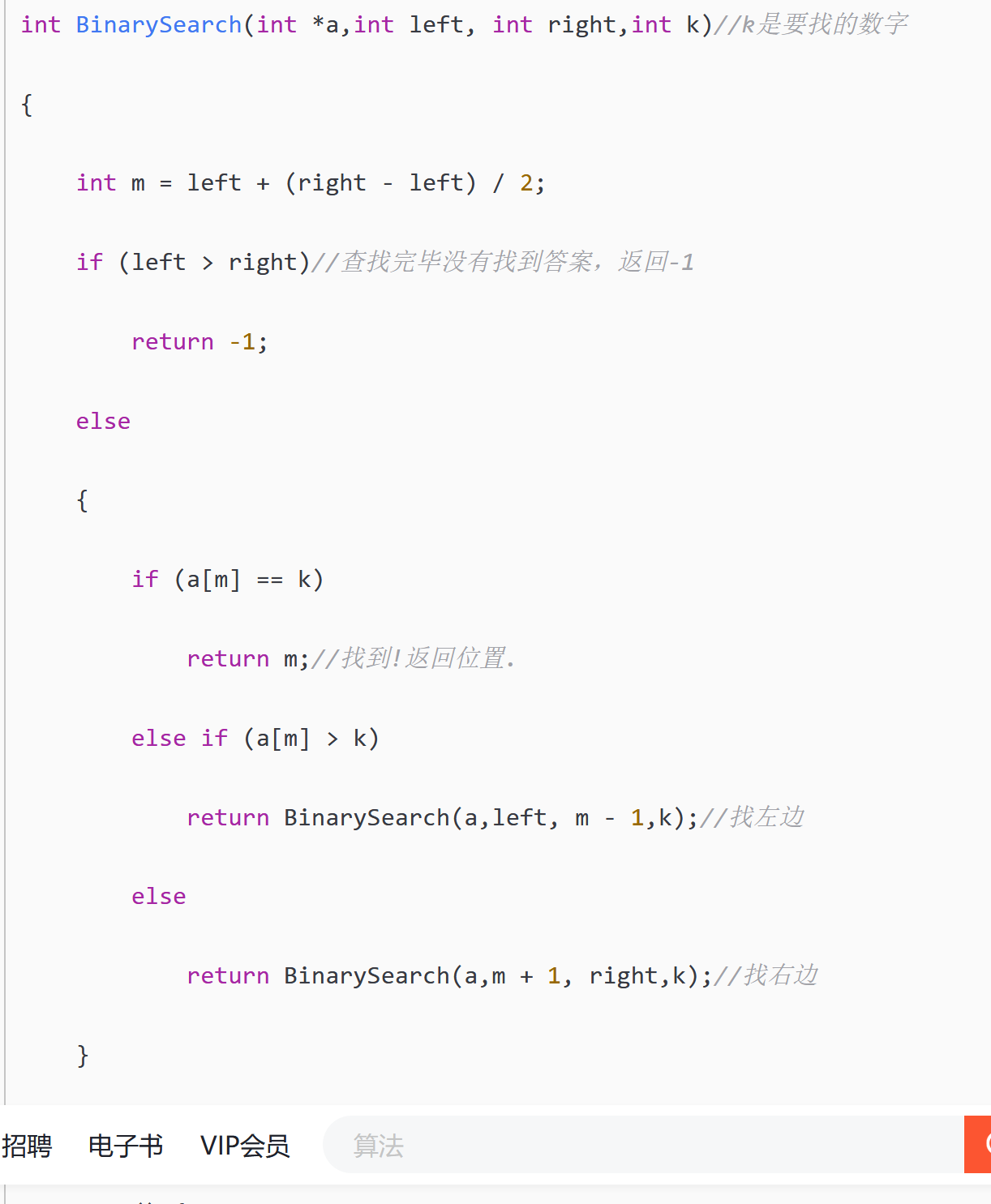
ModInsertSort(A[1..n])

for i:=2 to n do

x:=A[i]

k:=BinarySearch(A[1..i-1],x) //在A[1..i-1]查找x应该插入的位置k

A[k]:=x

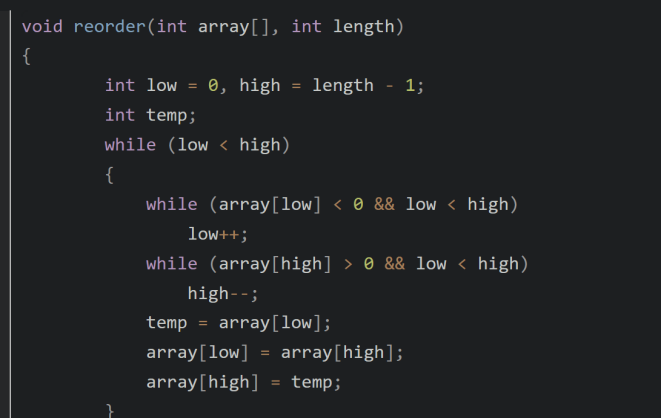


复杂度分析：外for循环n-1次，在i-1规模的数组中二分比较次数为<=log(i-1)+1，因此总比较次数为：（但移动次数没节省）

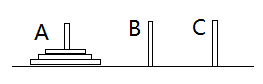


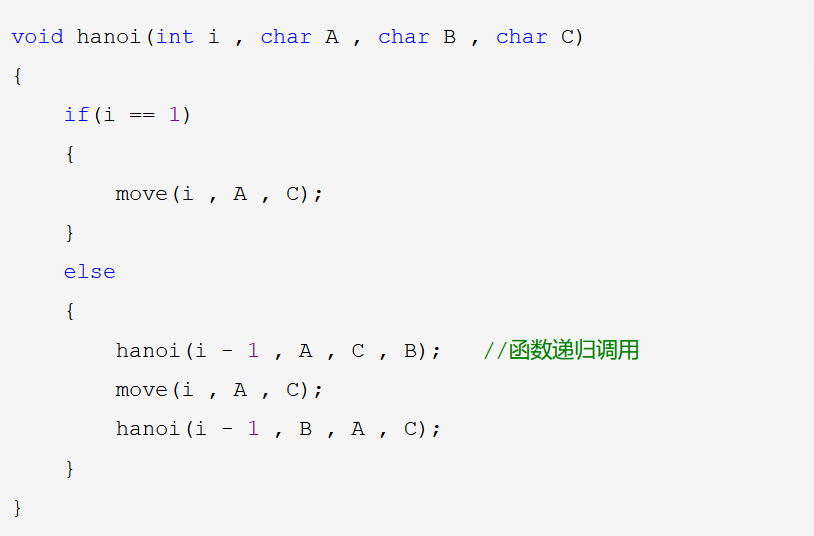
设A是n个非0实数构成的数组，设计一个算法重新排列数组中的数，使得负数都排在正数前面，要求算法复杂度为O(n)。

类似快速排序的划分过程。从后向前把每个数与0比较，找到第一个负数A[p];从前向后把每个数与0比较，找到第一个正数A[q]，如果P>q，则将A[p]与A[q]交换。交换后如果p-q=1，算法停止，否则继续这个过程。



Hanoi塔问题。图中有A、B、C三根柱子，在A柱上放着n个圆盘，其中小圆盘放在大圆盘的上边。从A柱将这些圆盘移到C柱上去，在移动和放置时允许使用B柱，但不能把大盘放到小盘的下面。设计算法解决此问题，分析算法复杂度。



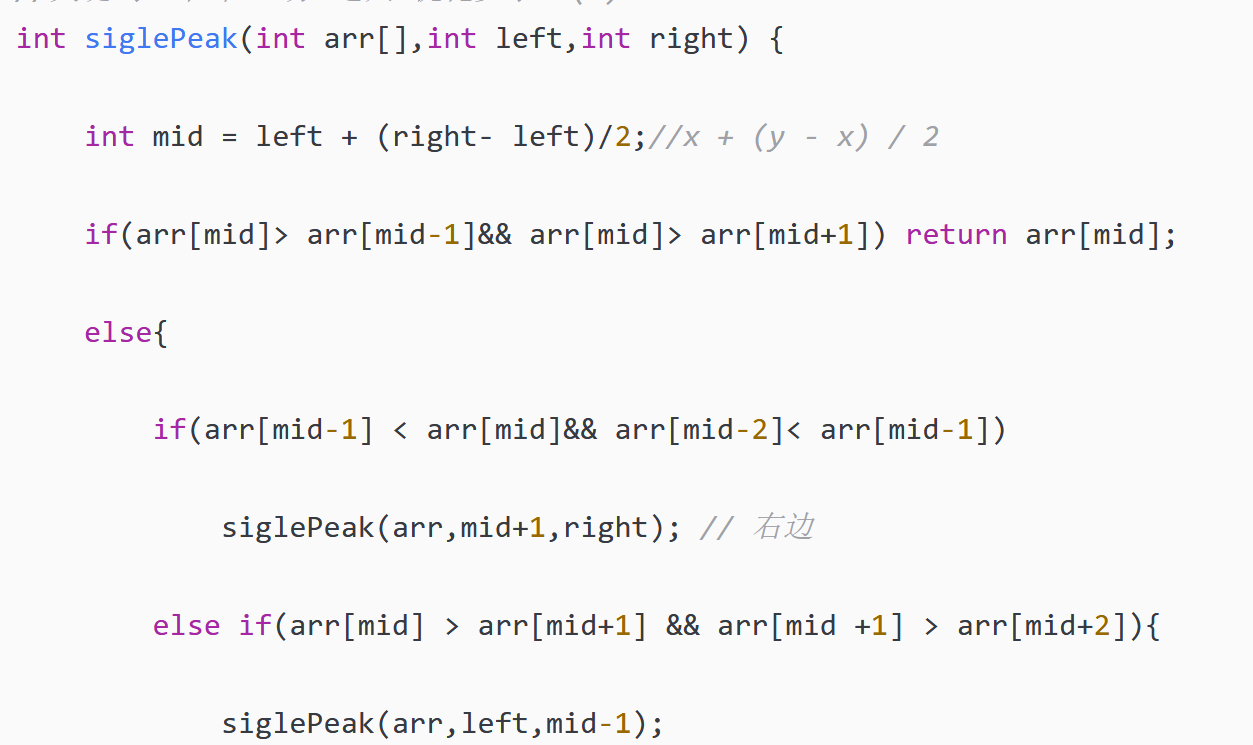


算法复杂度：T(n)=2T(n-1)+1=4T(n-2)+2+1=2n-1+2n-2+…+1=2n-1

给定含有n个不同数的数组L={x1,x2,…,xn}，如果L中存在xi，使得x1<x2<…<xi-1<xi>xi+1>…>xn，则称L是单峰的，并称xi是L的峰顶。假设L是单峰的，设计一个优于O(n)的算法找到L的峰顶。

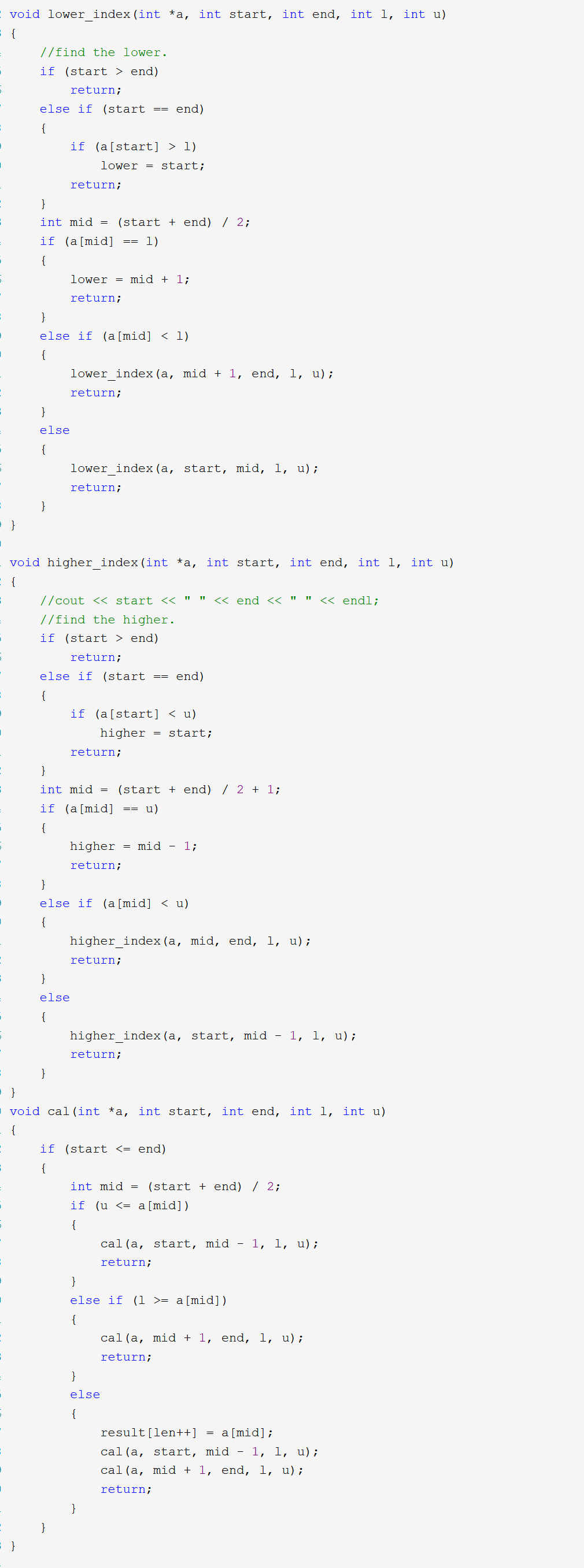
因为L中存在峰顶元素，因此|L|>=3。使用二分查找算法。如元素数等于3，则L[2]是峰顶元素，当元素数n>3时，令k=⎣n/2⎦，比较L[k]与它左边和右边相邻的项，如果L[k]>L[k-1]且L[k]>L[k+1]则L[k]为峰顶元素；否则，如果L[k-1]>L[k]>L[k+1]，则继续搜索L[1..k-1]，如果L[k-1]<L[k]<L[k+1]则继续搜索L[k+1..n]的范围。每比较两次，搜索范围减半，直到元素数小于3停止递归调用。

时间复杂度T(n)=T(n/2)+2，根据主定理，T(n)=O(logn)。

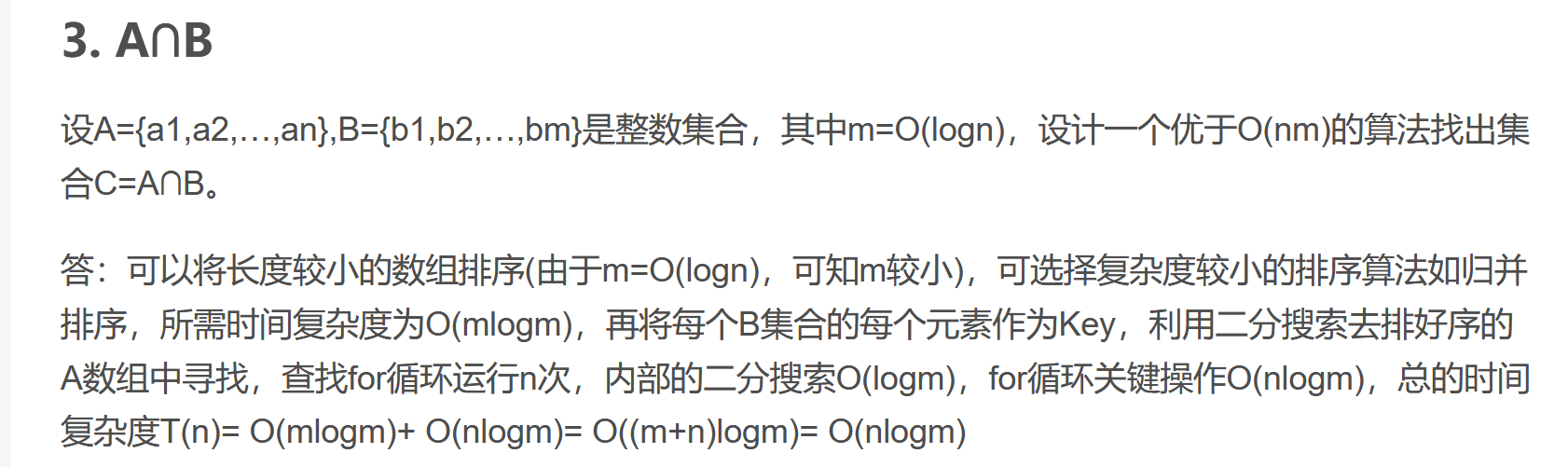


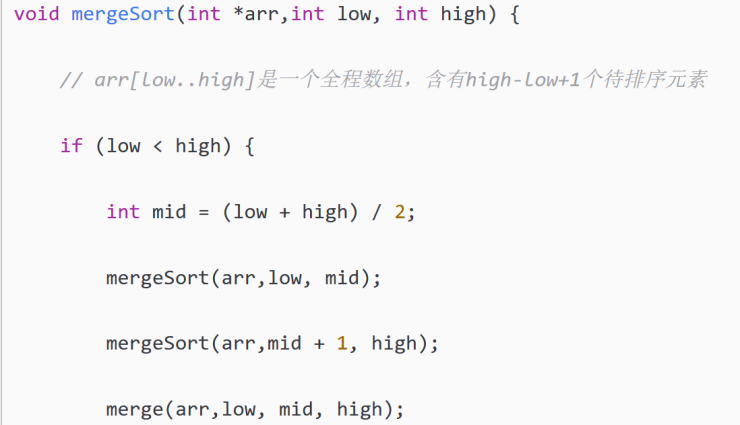
设A是n个不同的排好序的数组，给定数L和U，L<U，设计一个优于O(n)的算法，找到A中满足L<x<U的所有数x。

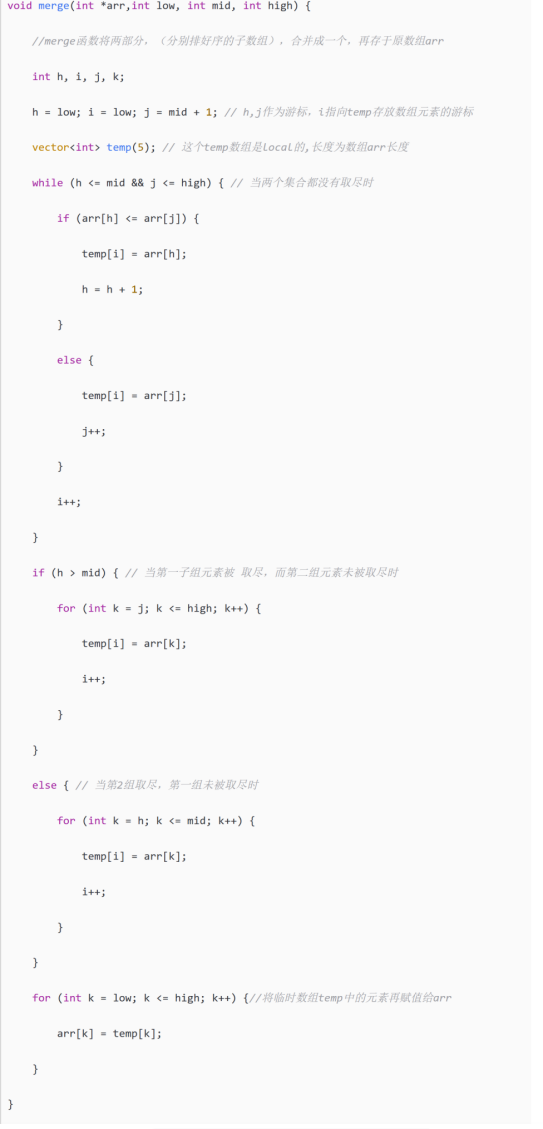
在A中使用二分查找算法找L，如果L=A[i]，找到L的位置i，然后把i加1;如果L不在A中，那么找到大于L的最小数的位置i。类似地，找到U的位置A[j]，j=j+1，或小于U的最大数A[j]。输出A中i到j的全体数。

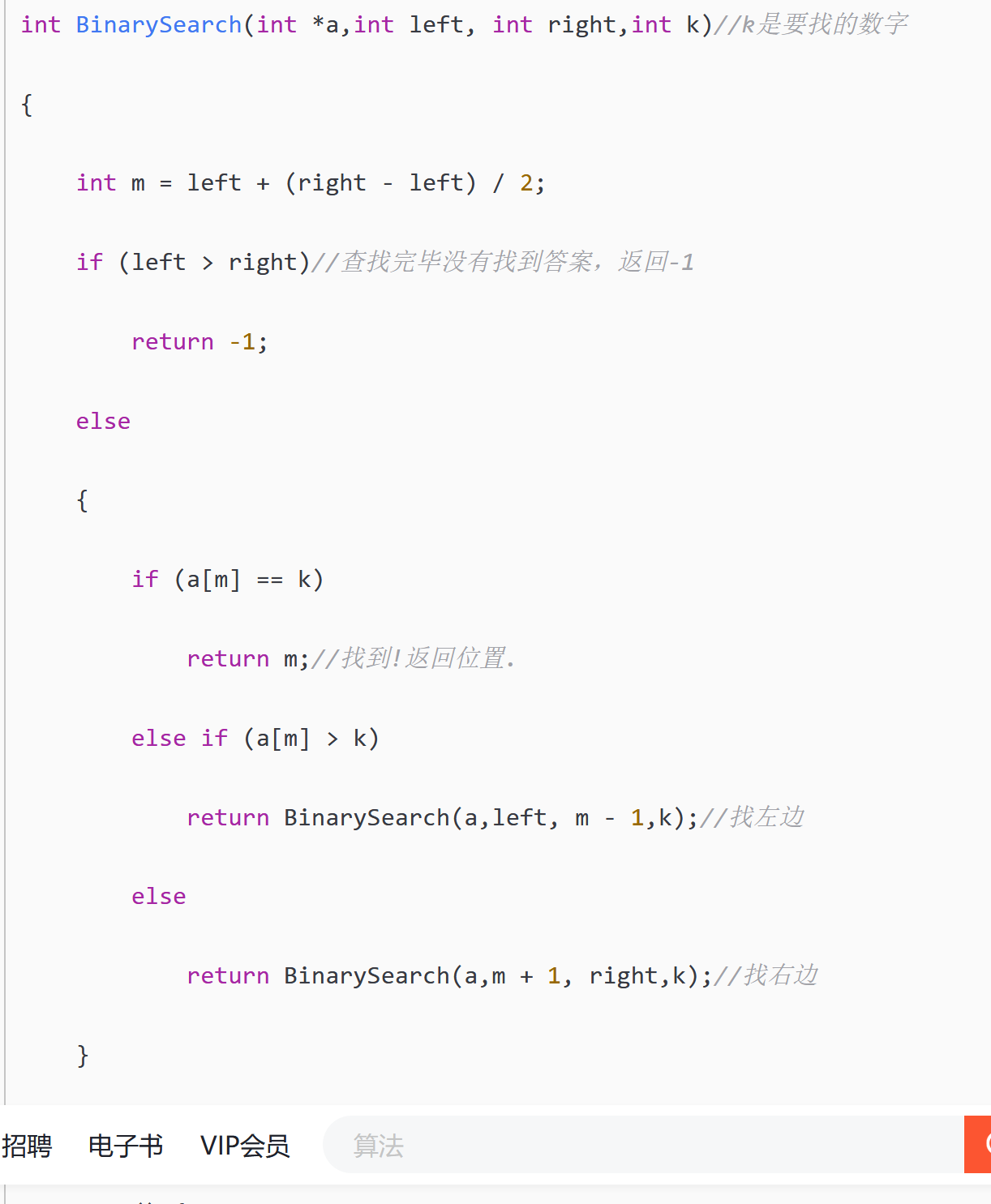


.设A={a1,a2,…,an},B={b1,b2,…,bm}是整数集合，其中m=O(logn)，设计一个优于O(nm)的算法法找出集合C=A∩B。（即找出一个优于nlogn的方法）所得方法为nlogm











设S是n个不等的正整数的集合，n为偶数，给出一个算法将S划分为子集S1和S2，使得|S1|=|S2|且  达到最大，即两个子集元素之和的差达到最大。(要求：T(n)=O(n))。

规定S的中位数x是从小到大排序的第n/2个数，用x划分S，比x小的整数属于S1，x本身也放到S1，其余的放到S2，由于n是偶数，|S1|=|S2|，易见这样的集合满足要求。

算法复杂度：找中位数和划分都是O(n)，所以T(n)=O(n)。

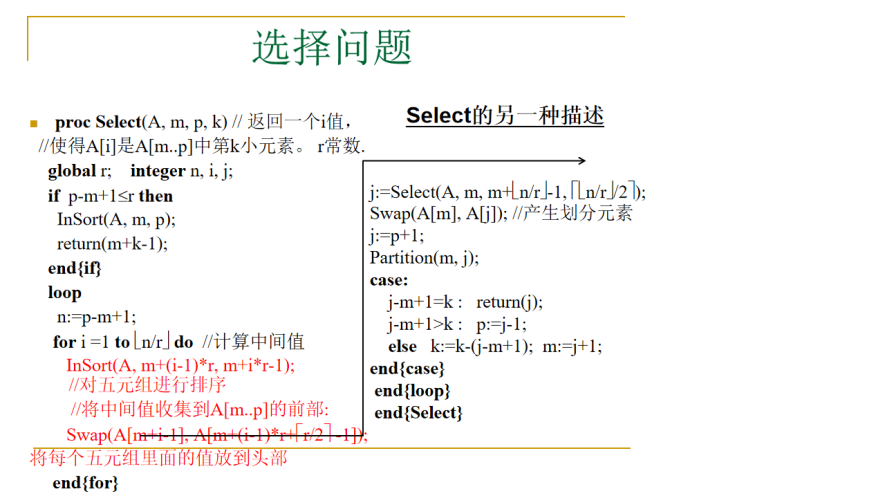
补充算法实现

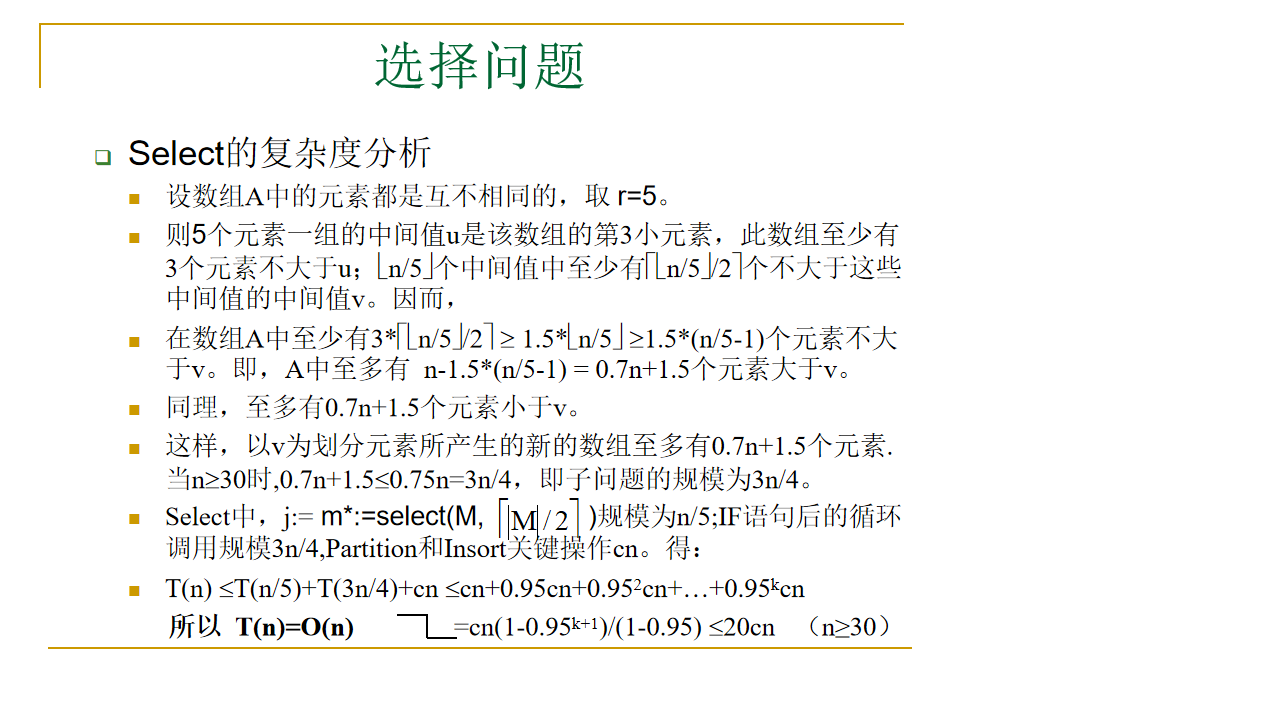
考虑第三章PPT NO.17 Select(A,k)算法：

（1）如果初始元素分组r=3，算法的时间复杂度如何？

(2) 如果初始元素分组r=7，算法的时间复杂度如何？







(1)r=3 ，不妨设n是3的倍数。每组至少2个元素不大于u，A中至少2\*⎡⎣n/3⎦/2⎤ >= ⎣n/3⎦=n/3个不大于v， 即A中至多n-⎣n/3⎦<=n-n/3=2n/3个元素大于v。同理，至多有2n/3个元素小于v。即子问题的规模小于2n/3。所以， T(n)=T(n/3)+T(2n/3)+O(n)，得T(n)=O(nlogn)。与直接排序方法的复杂度一样。

(2)问题变为4\*⎡⎣n/7⎦/2⎤ >=2 ⎣n/7⎦，子问题规模小于n-2n/7=5n/7（不妨设n是7的倍数） T(n)=T(n/7)+T(5n/7)+O(n) =n(1+6/7+(6/7)2+…+(6/7)k)+O(n)=O(n)

11.对玻璃瓶做强度试验，设地面高度为0，从0向上有n个高度，记为1,2,…,n，其中任何两个高度之间的距离都相等。如果一个玻璃瓶从高度i落到地上没有摔碎，但从高度i+1落到地上摔碎了，那么就将玻璃瓶的强度记为i。

(1)假设每种玻璃瓶只有1个测试样品，设计算法来测试出每种玻璃瓶的强度。以测试次数作为算法的时间复杂度量度，估计算法的复杂度。

(2)假设每种玻璃瓶有足够多的相同的测试样品，设计算法使用最少的测试次数来完成测试。

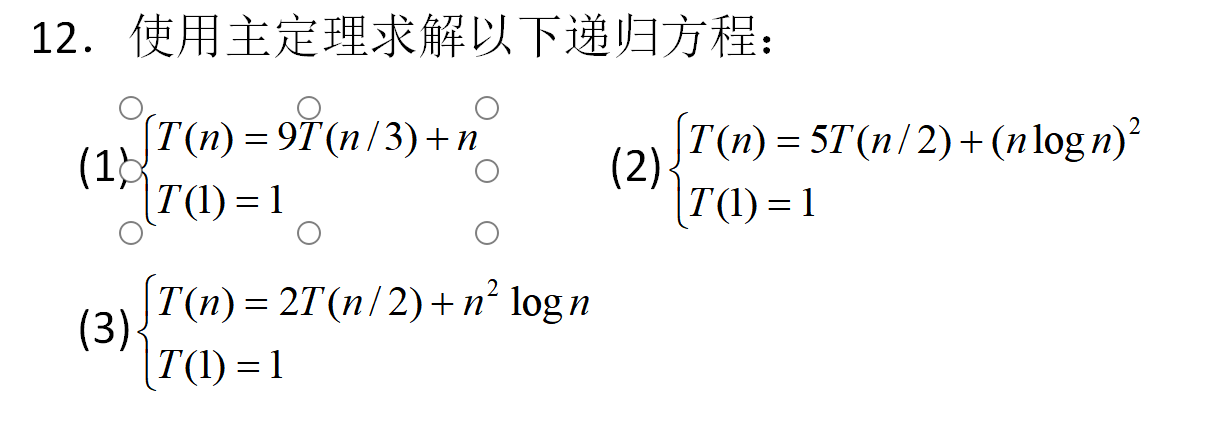
(3)假设每种玻璃瓶只有2个相同的测试样品，设计次数尽可能少的算法完成测试。

解：(1)只好顺序从下到上测试，一次一个高度，最坏T(n)=O(n)

(2)二分查找法。取n/2高度进行第一次测试，如果瓶子没有摔碎，则强度在[n/2+1,n]之间，否则在[1,n/2]之间。每次测试后可能一个瓶子的代价，测试范围减半，最坏时间复杂度T(n)=O(logn)。

(3) 为简单起见，不妨设为整数，将高度1,2,..,n分为个组，每组高度，取第一个瓶子从下到上测试每组的最大高度，即高度,2…n，如果k-1组没碎，k组碎了，那么玻璃瓶子的强度在第k组内，于是，再经至多次测试，就可以得到瓶子的强度。

T(n)=O()+O()=O()



(1)a=9,b=3,f(n)=n，log39=2,f(n)的阶低于nlogba,符合情况1，T(n)=Θ( nlogba)= Θ(n2)。

(2)a=5,b=2,f(n)=n2log2n=O(nlog25-ε)，方法T(n)= Θ(nlog25)

(3)a=2,b=2,f(n)=n2logn,取c=3/4则

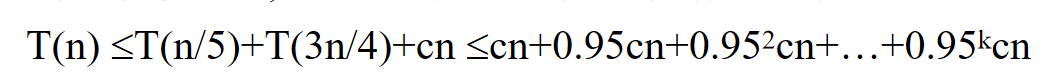
af(n/b)=2(n/2)2log(n/2)=(n2/2)(logn-1)≤(n2/2)logn≤cn2logn=cf(n)

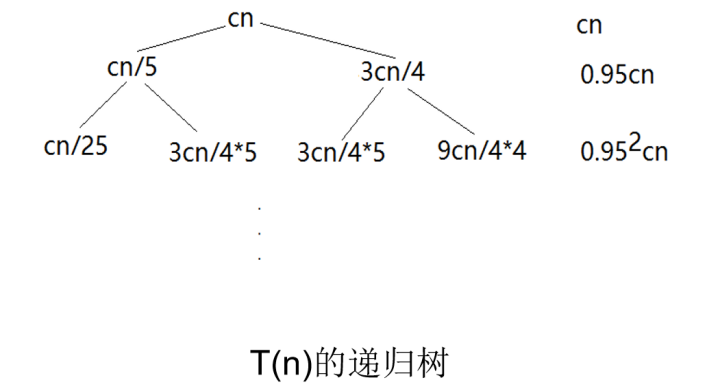
于是，符合情况3，T(n)= Θ(n2logn)

使用递归树求解：

T(n)=cn+3cn/4+(3/4)2cn+(3/4)3cn+…..=[1+3/4+(3/4)2+(3/4)3+…]cn=Θ(n)

例：





使用迭代递归法求解：

1.  (2)

(1)T(n)=nlog3+(n-1)log3+T(n-2)=log3(n+(n-1)+(n-2)+…+1)= Θ(n2)

(2)T(n)=T(n-1)+1/n=T(n-2)+1/(n-1)+1/n=…=**1/n+1/(n-1)+…+1/2+1=Θ(logn)—相等**