1、a.为一个分治算法编写伪代码，该算法同时求出一个n元数组的最大元素和最小元素的值。

b.请拿该算法与解同样问题的蛮力算法做一个比较。

2、为最近对问题的一维版本设计一个直接基于分治技术的算法,并确定它的时间复杂度。（最近点对问题定义：已知上m个点的集合，找出对接近的一对点。）

3、设计一个分治算法来计算二叉树的层数.(空树返回0,单顶点树返回1),并确定它的时间复杂度.

**1、解答：**

a.同时求出最大值和最小值，只需要将原数组一分为二，再使用相同的方法找出这两个部分中的最大值和最小值，然后经过比较就可以得到整个问题的最大值和最小值。

算法 MaxMin(A[*l..r*],Max,Min)

//该算法利用分治技术得到数组A中的最大值和最小值

//输入：数值数组A[*l..r*]

//输出：最大值Max和最小值Min

if(*r*=*l*) Max←A[*l*]；Min←A[*l*]; //只有一个元素时

else

if *r*－*l*=1 //有两个元素时

if A[*l*]≤A[r]

Max←A[r]; Min←A[*l*]

else

Max←A[*l*]; Min←A[*r*]

else //r－*l*>1

MaxMin(A[*l*,(*l*+r)/2],Max1,Min1); //递归解决前一部分

MaxMin(A[(*l*+r/)2..r],Max2,Min2); //递归解决后一部分

if Max1＜Max2 Max= Max2 //从两部分的两个最大值中选择大值

if Min2<Min1 Min=Min2; //从两部分的两个最小值中选择小值

}

b.假设n=2k,比较次数的递推关系式:

C(n)=2C(n/2)+2 for n>2

C(1)=0, C(2)=1

C(n)=C(2k)=2C(2k-1)+2

=2[2C(2k-2)+2]+2

=22C(2k-2)+22+2

=22[2C(2k-3)+2]+22+2

=23C(2k-3)+23+22+2

...

=2k-1C(2)+2k-1+2k-2+...+2 //C(2)=1

=2k-1+2k-1+2k-2+...+2 //后面部分为等比数列求和

=2k-1+2k-2 //2(k-1)=n/2,2k=n

=n/2+n-2

=3n/2－2

**b.蛮力法的算法如下：**

算法 simpleMaxMin(A[*l..r*])

//用蛮力法得到数组A的最大值和最小值

//输入：数值数组A[*l..r*]

//输出：最大值Max和最小值Min

Max=Min=A[*l*];

for i=*l*+1 to r do

if A[i]>Max Max←A[i];

else if A[i]<Min Min←A[i]

return Max,Min

}

时间复杂度t(n)=2(n-1)

**算法MaxMin的时间复杂度为3n/2-2，simpleMaxMin的时间复杂度为2n-2，都属于Θ(n)，但比较一下发现，MaxMin的速度要比simpleMaxMin的快一些。**

2、解:

1. Algorithms ClosestNumber(A[*l*..r])

//分治计算最近对问题的一维版本

//输入:升序排列的实数子数组A[*l*..r]

//输出:最近数对的距离

If r=*l* return ∞

Else if r－*l*=1 return A[r]－A[l]

Else return min{ClosestNumber(A[*l*… (*l*+r)/2 ]),

ClosestNumber(A[ (*l*+r)/2 ...r])

A[ (*l*+r)/2 +1]－A[ (*l*+r)/2 ]

}

设递归的时间效率为T(n):

对n=2k, 则: T(n)=2T(n/2)+c

利用主定理求解.T(n)=Θ(n)

3、

Algorithms Level(Tree T)

//递归计算二叉树的层数

//输入:二叉树T

//输出:二叉树T的层数

If T=NULL return 0

Else return max{Level(TL),Level(TR)}+1

算法效率类型是Θ(n)