1、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。）

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)

2、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.

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)