# 第一次作业

姓名：梁付槐　　　学号：2018Z8013261003

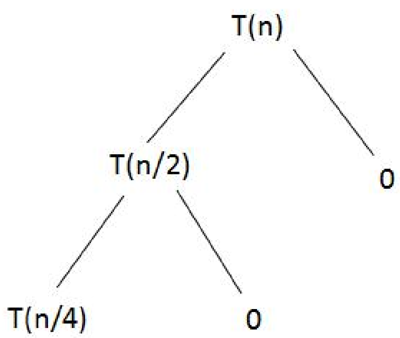
## 题目一：在两个数据库中找中位数

**算法：** 设两个数据库分别为 Db1 和 Db2，数据量均为 n。定义两个指针 p1 和 p2 分别指向 Db1 和 Db2，并设 p1 和 p2 的初值均为 n/2。然后对 p1 和 p2 进行迭代：记 p1 的位置为 m1，p2 的位置为 m2。每次迭代时刷新 $p1=p1-n/2^i$ 和 $p2=p2+n/2^i$，当不可分时迭代结束，此时经过了 $\log_2 n$ 次迭代。取 MIN 为 m1 和 m2 的最小值，则 MIN 即为所求的中位数.

**伪代码：**

```
p1=p2=n/2;
for i from 2 to log_2 n
    m1=query(Db1,p1)
    m2=query(Db2,p2)
    if m1>m2
        p1=p1-n/2^i
        p2=p2+n/2^i
    else
        p1=p1+n/2^i
        p2=p2-n/2^i
    end if
end for
MIN=min(m1,m2)
return MIN
```

图示：




**时间复杂度分析：** $T(n)=T(n/2)+2c+0$，根据 Master Theorem, $T(n)=O(\log_2 n)$.

## **题目四：**在网格图中找极小值点

**算法：**考虑六条直线$i=1; i=n; j=1; j=n; i=\lfloor (1+n)/2 \rfloor; j=\lfloor (1+n)/2 \rfloor$.将网格划分为四部分。比较这六条直线上的点对应的点的值，找出最小点。如果最小点满足局部最小，返回这个点。否则，在最小点所在区域递归查找。

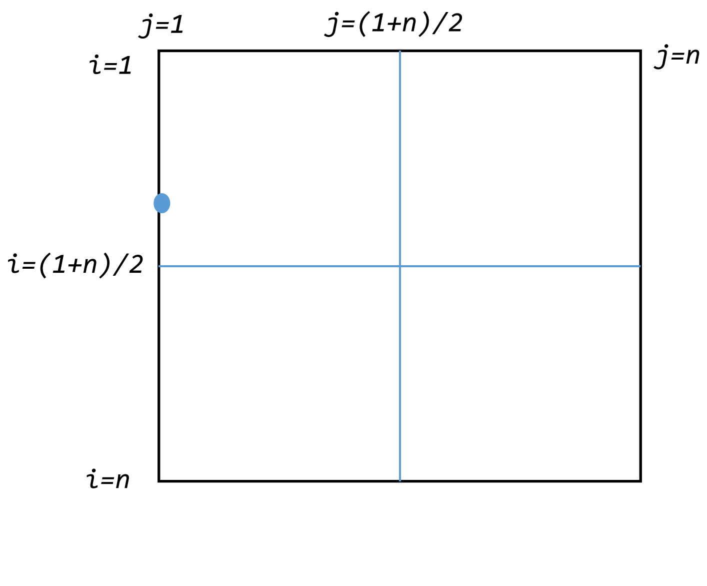
**伪代码：**

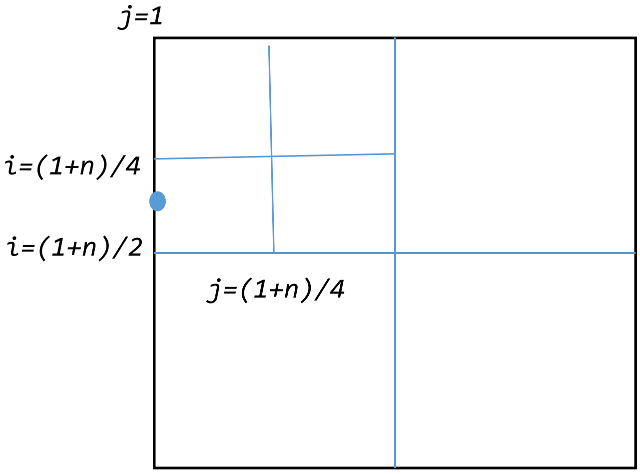
```
//Add borders to reduce consideration.
AddBorder(G,n){
    def Gadd[n+2][n+2]=infinity;
    Gadd[2:n+1][2:n+1]=G[1:n][1:n];
    return Gadd;
}

//Find local minimum
LocalMinimum(G,il,ir,jl,jr){
    im=(il+ir)/2;
    im=(il+ir)/2;
    P(imin,jmin)=min(points of givesix lines);
    (il,ir,jl,jr)=zone boundary of P(imin,jmin);

    if(P(imin,jmin)<min(G(imin-1,jmin),G(imin+1,jmin),
    G(imin,jmin-1),G(imin,jmin+1)){
        return P(imin,jmin);
    }else{
        LocalMinimum(G,il,ir,jl,jr);
    }
}
//invoking function
G=AddBorder(G,n);
LocalMinimum(G,1,n,1,n);
```

图示：






**时间复杂度分析：**T(n)=1*T(n/2)+O(n)，根据 Master Theorem, T(n)=O(n).

## **题目五：**求逆序数

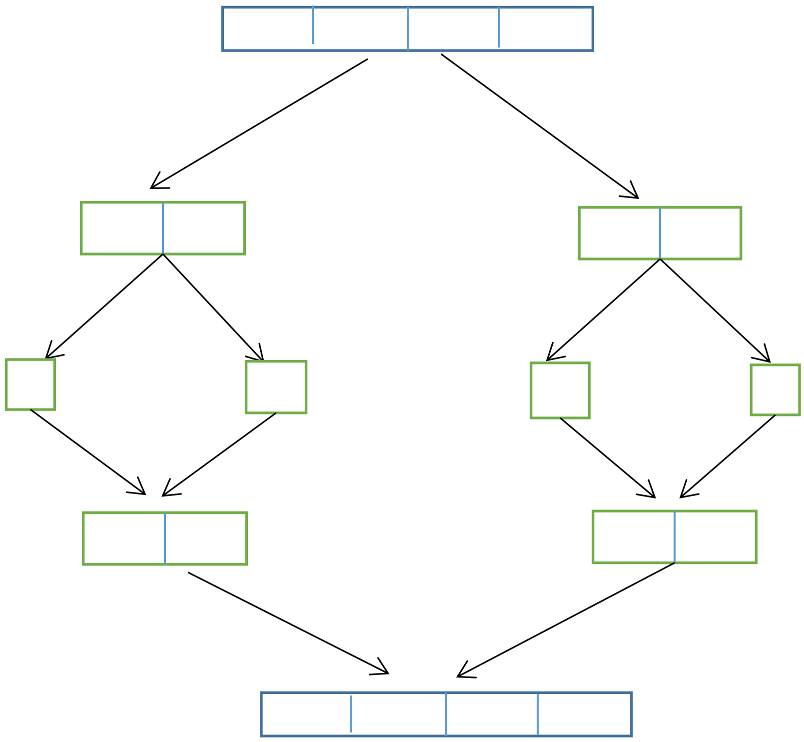
**算法：**设整个序列的逆序数为 C，每个数的逆序数为 $C_i$。将序列 $a_1, a_2, \cdots, a_n$ 分成两份：$B_0 = (a_1, a_2, \cdots, a_{n/2}), B_1 = (a_{n/2+1}, \cdots, a_n)$. 则 $C = C(B_0) + C(B_1) + M(B_0 B_1)$.

**伪代码：**

```
Merge(Arr,start,p,end){
    L=Arr[start,..,p];
    R=Arr[p+1,..,end];
    InversionCount=0;
    for(i=start:p){
        j=1;
        while(j<=end-p){
            if(L[i]>3*R[j])
            InversionCount+=1;
        }
    }
}

SignificantInversion(Arr,start,end){
    if(|Arr|==1){
        return 0;
    }else{
        P=(start+end)/2;
    N1=SignificantInversion(Arr,start,p);
    N2=SignificantInversion(Arr,p+1,end);
    N3=Merge(Arr,start,p,end);
    }
    return N1+N2+N3;
}
```

**图示：**



**时间复杂度分析：**T(n)<=2T(n/2)+cn，　每一次递归规模减半，所以 $T(n)=O(n\log_2 n)$.