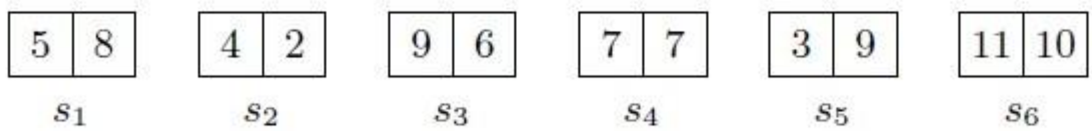
计算机算法设计与分析

朱双贺　2010E8009070012

第三次作业：

题目：现有 n 块“多米诺骨牌”S_1，S_2，……，S_n 水平放成一排，每块骨牌 S_i 包含左右两个部分，每个部分赋予一个非负整数值，如下图所示为包含 6 块骨牌的序列。骨牌可做 180 度旋转，使得原来在左边的值变到右边，而原来在右边的值移到左边，假设不论 si 如何旋转，L[i]总是存储 si 左边的值，R[i]总是存储右边的值，W[i]用于存储 si 的状态：当 L[i]<=R[i]时记为 0，否则记为 1，试采用分治法设计算法求 $\sum_{i=1}^{n-1} R[i]L[i+1]$最大值，以及当取得最大之时每个骨牌的状态。

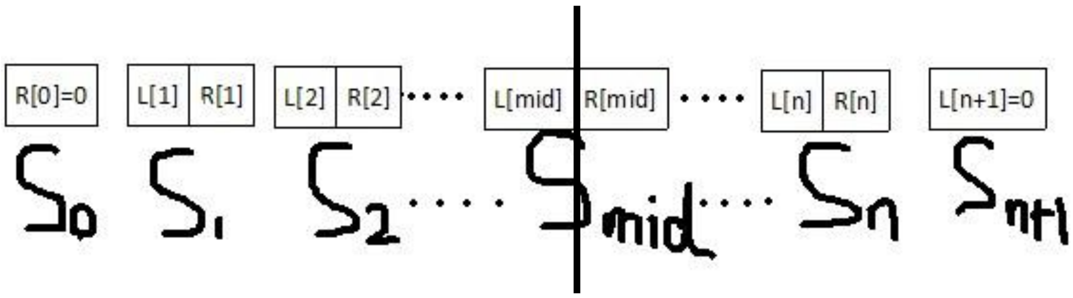


解：

输入为 n 个多米诺骨牌 S_1，S_2，……，S_n ，可将输入在中间那块骨牌的中间，即骨牌$S_{\frac{1+n}{2}}$的中间处拆分，这样原问题就被拆分为两个子问题，但是可以发现，新的子问题与原问题结构不完全一样。为解决这一问题，可对原始的输入作如下处理：

在原始输入序列左右两端各加半块骨牌，记为 S_0 和 S_{n+1} ，对应于 L、R 则为 R[0]和 L[n+1]，并且令 R[0]和 L[n+1]里面的值为 0。如下图所示：

此处拆分



则原规模为 n 的问题求 MAX（$\sum_{i=1}^{n-1} R[i]L[i+1]$）变为规模为规模为 n+2 的问题求 $\sum_{i=0}^{n} R[i]L[i+1]$，结构有所变化，但实际上由于左右两边加的半块骨牌中的元素值为 0，对原问题结果并不会产生影响。这时在按原来的拆分方法进行拆分，就可将问题拆为两个规模为原问题一半的子问题。具体算法如下：(其中数组 W 用于存放所求和最大时各个骨牌的状态，first 和 last 分别为第一块和最后一块骨牌的序号)

Algorithm Max_of_Domino(L,R,first,last,W)

```
if last-first=0 then    return;          //骨牌数不能少于两个
R[first-1]=0,L[last+1]=0;
Max=Simplify_Max_of_Domino(L,R,first-1,last+1);
return Max;
```

Algorithm Simplify_Max_of_Domino(L,R,first,last)

```
if    last-first=1 then Max=R[first]*L[last];
else  mid=<(first+last)/2>;              //这里用< >表示上取整
  |   if   L[mid]>R[mid]                 //L>R,这时W[mid]状态为1
  |   |    MaxL1=Simplify_Max_of_Domino(L,R,first,mid);
  |   |    MaxR1=Simplify_Max_of_Domino(L,R,mid,last);
  |   |
  |   |
  |   |
```

```
            Max1=MaxL1+MaxR1;
            Swap(L[mid],R[mid]);      //再交换被拆分的骨牌的左右半块，使W[mid]状态为0
            MaxL0=Simplify_Max_of_Domino(L,R,first,mid);
            MaxR0=Simplify_Max_of_Domino(L,R,mid,last);
            Max0=MaxL0+MaxR0;
            if        Max1>Max0
                      Max=Max1,W[mid]=1;
            else      Max=Max0,W[mid]=0;
        else          //L<=R,W[mid]=0
            MaxL0=Simplify_Max_of_Domino(L,R,first,mid);
            MaxR0=Simplify_Max_of_Domino(L,R,mid,last);
            Max0=MaxL0+MaxR0;
            Swap(L[mid],R[mid]);          //再交换被拆分的骨牌的左右半块，使W[mid]状态为0
            MaxL1=Simplify_Max_of_Domino(L,R,first,mid);
            MaxR1=Simplify_Max_of_Domino(L,R,mid,last);
            Max1=MaxL1+MaxR1;
            if        Max1>Max0
                      Max=Max1,W[mid]=1;
            Else      Max=Max0,W[mid]=0;
        end
    end
    return Max;
```

现在分析时间复杂度：

算法中较多的操作是乘法操作，可将乘法操作视为关键操作，除法也看作是乘法。用 T(n)表示原问题的时间复杂度，H(n)为加入两个半块，即 R[0]和 L[n+1]后的问题的时间复杂度，则有：

$$T(n)=H(n+2) \qquad ;n>=2$$

由算法可知：$H(n)=2(H(n/2)+H(n/2))+1=4H(n/2)+1$;

根据主定理：$b=4$, $c=2$; $E=\log_c^{\,b}=2$; $f(n)=1$;

$\therefore\ H(n) \in O(n^2)$; $T(n)=H(n+2) \in O(n^2)$;

原问题以乘法为关键操作的时间复杂度为： $A(n) \in O(n^2)$