计算机算法设计与分析

朱双贺 2010E8009070012

第五次作业： 多米诺骨牌的动态规划解法

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，试设计时间复杂度为O(n)的动态规划算法求$\sum_{i=1}^{n-1} R[i] * L[i+1]$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 ；即数组L[]和数组R[]。

1. 当输入规模为两块多米诺骨牌时，可直接求解
2. 输入规模为n时，可以由前n-1块骨牌的解和前n-2块骨牌的解求出n块骨牌的解

可设计自下而上的动态规划方法求解该问题

动态规划算法如下：

Algorithm Max_of_Domino(L,R,n)

```
1.  if n<2      then  break;            //输入不少于两块多米诺骨牌
2.  if L[1]>=R[1]                       //第一块肯定是L<=R（因为输入为非负）
3.      Swap( L[1],R[1]);
4.      W[1]=0;
5.  end
6.  if L[2]=R[2]
7.      W[2]=0;
8.  else if L[2]<R[2]
9.      Swap( L[2],R[2]);
10.     W[2]=1;
11. end
12. Max[1]=0;
13. Max[2]=R[1]*L[2];
14. for(i=3;n;i++)
15.     max=max'=0;
16.     if L[i]=R[i]                    //考虑最后一块左右相等的情况
17.         W[i]=0;
18.     else if L[i]<R[i]       //最后一块肯定是L>=R（因为输入为非负）
19.         Swwap(L[i],R[i]);
20.         W[i]=1;
21.     end
22.     max=Max[i-1]+R[i-1]*L[i];
23.     if L[i-1]!=R[i-1]       //倒数第二块若左右相等，则不影响结果
24.         max'=Max[i-2]+R[i-2]*R[i-1]+L[i-1]+L[i];
25.     end
26.     if max<max'                     //取较大的max
27.         Swap(L[i-1],R[i-1]);
28.         W[i-1]=1;
29.         max=max'
30.     end
31.     Max[i]=max;                     //max[n]存放最终结果；W[i]存放最终的骨牌状态
32. end
```

算法时间复杂度为：A(n)=2n∈O(n);