1、给定一个整数序列*a*1,…,*an*。相邻两个整数可以合并，合并之后的结果是两个整数之和（用和替换原来的两个整数），合并两个整数的代价是这两个整数之和。通过不断合并最终可以将整个序列合并成一个整数，整个过程的总代价是每次合并操作代价之和。试设计一个动态规划算法给出*a*1,…,*an*的一个合并方案使得该方案的总代价最大。

（1）用简明的语言表述这个问题的优化子结构；

（2）根据优化子结构写出代价方程；

（3）根据代价方程写出动态规划算法（伪代码）并分析算法的时间复杂性。

2、最长增长子序列问题

输入：由*n*个数组成的一个序列*S*：*a1,a2,…,an*

输出：子序列*S’=b1,b2,…,bk* ，满足：

(1) *b1≤b2 ≤ … ≤ bk* ，

(2) |*S’*|最大

（1）用简明的语言表述这个问题的优化子结构；

（2）根据优化子结构写出代价方程；

（3）根据代价方程写出动态规划算法（伪代码）并分析算法的时间复杂性。

优化子结构：

设序列A1~n=*a*1,…,*an的代价为wA(1~n)，我们有*

*wA(1~n)=wA(1~k)+wA(k+1~n)，1<k<n;*

*子代价方程：*

*wA(1~n)=max{wA(1~k)+wA(k+1~n)}，1<k<n;*

*wA(k~k)=ak，1<k<n;*

*伪代码：*

*for i=1 To n Do:*

*w[i,i]=ai ;*

*for l=2 To n Do:*

*for i=1 To n Do:*

*j=i+l-1;*

*for k=i To j Do:*

*temp=w[i,k]+w[k+1,i+l-1];*

*If temp>w[i,i+l-1]*

*Then w[i,i+l-1]=temp;*

*Return w[1,n];*

*时间复杂度：*

*O(n3)*

*优化子结构：*

*设dp[i]为：以ai结尾的最长子序列；*

*故原问题转换为max{dp[i],1<i<n};*

*对于dp[i]*

*若aj>ai有dp[i]=max{dp[j]}+1; 1<=j<i;*

*代价方程：*

*dp[i]=max{dp[j]}+1 (1<=j<i),if aj>ai;*

*dp[i]=1, (1<=j<i),if aj<=ai;*

*伪代码：*

*dp[1]=1;*

*for i=1 To n Do:*

*for j=1 To i Do:*

*if aj>ai AND dp[j]+1>dp[i]*

*Then dp[i]=dp[j]+1;*

*for i=1 To n Do;*

*If dp[i]>max*

*Then max=dp[i];*

*return max;*

*时间复杂度：O(n2)*