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。

**优化子结构：**

所有合并操作中的最后一次合并将两个整数和合并为一个整数，其中和一定是合并输入中的连续整数序列所得，即存在整数，使得是*a*1,…,*ak*的合并结果，而是*a*k+1,…,*an*的合并结果。由合并操作的说明可知，，，而且最后一次合并的代价为。因此，最优解的总代价是合并*a*1,…,*ak*得到的最大代价、合并*a*k+1,…,*an*得到的最大代价、以及的加和。其中包括合并*a*1,…,*ak*和*a*k+1,…,*an*对应的两个子问题的优化解代价。

**代价的递归方程：**

令表示合并的优化解代价，.

边界条件：, 如果;

当时，。

算法思路：

逐条对角线计算，首先用边界条件确定满足的，随后增加的差值，并计算满足该插值的。即位优化解的代价。

计算时，用记录最优的划分点，并通过构造优化的合并序列。

**伪代码：**略

**代价：**.

2、最长增长子序列问题定义如下：

输入：由*n*个数组成的一个序列*S*：*a1,a2,…,an*

输出：*S*的子序列*S’=b1,b2,…,bk* ，满足：

(1) *b1≤b2 ≤ … ≤ bk* ，

(2) |*S’*|最大

使用动态规划技术设计算法求解最长增长子序列问题.

**优化子结构：**

令*b1,b2,…,bk*是一个优化解，其中*bk*对应*ai*，*bk*-1对应*aj*，则*i>j*，*aiaj*而且*b1,b2,…,bk*-1必须是*a1,a2,…,ai*的最长增长子序列。

**代价的递归方程：**

令*L*[*i*]表示以*ai*结尾的增长子序列的最大长度，最长增长子序列必定以某个元素结尾，因此max{*L*[*i*]}即最长增长子序列的长度。*L*[*i*]的求解如下：

边界条件：*L*[0]=0, *L*[1]=1;

当*i*>1时，*L*[*i*]=.

**算法思路：**

从左向右计算*L*[*i*]，计算过程中*ai*的前序元素保存为使*L*[*i*]最大的*aj*;

扫描所有*L*[*i*]并取其最大值，即为最长增长子序列的长度，使用该长度计算过程中保存的前序元素构造最长增长子序列。

**伪代码：**略。

**代价：**。