参考解答

Sol1.（25 分）令 $num(l,m)$ 为长度 l 时的序列数，m 为长度 l 的正整数序列的最大值。由限制增的性质，注意到长度为 n 的整数列中元素的最大值不会超过 n。

对长度为 l 中，前 $l-1$ 个元素中是否存在最大值 m 分情况进行讨论：1. 若存在，则 a_l 可以取 $[1,m]$ 中的任意一个值；2. 若不存在，则 $a_l = m$。这意味着前 $l-1$ 个元素中的最大值只可能为 $m-1$。于是可以得到递归式：

$$num(l,m) = num(l-1,m-1) + m \cdot num(l-1,m),$$
$$(num(l,1) = 1,\ num(l,m) = 0\ if\ l < m)$$

由递归式，我们需要解决 n^2 个子问题。可以设计一个 $n \cdot n$ 的表格（l 与 m 的取值范围均为 $[1,n]$）解决每一个子问题只需要常数时间。在最后一步，对这个表格的最后一行相加即可得到解：$\sum_{i=1}^{n} num(n,i)$，而这只需要线性时间。故总的时间复杂度为 $O(n^2)$。

Sol2.（25 分）用 $x[i,j]$ 表示 x 中下标 i 至下标 j 的子字符串。定义 $OPT(i,j)$ 为对子字符串 $x[i,j]$ 进行划分的最高质量分值。则可知 $OPT(i,i) = q(x[i,i])$。如果字符串 x 不再被划分，则 $OPT(i,j) = q(x[i,j])$；否则，最优的划分会在某个节点将字符串 x 断开，并且左（右）子字符串也会是最优的划分：

$$OPT(i,j) = \max_{k=i}^{j-1} (OPT(i,k) + OPT(k+1,j))$$

由递归式，我们只需要按 $j-i$ 递增的顺序填充表格即可。注意 i,j 均在 $[1,n]$ 之间，故有 $O(n^2)$ 的格子需要填充。此外，在每个格子内，需要运行 $O(n)$ 次划分点。故总的时间复杂度为 $O(n^3)$。

Sol3.（25 分）令 $S(i,y)$ 为第 y 年选择策略 i 时，前 y 年获得的最大积蓄。则我们需要的解为 $\max_{1\le i\le n} S(i,m)$。

我们将第一年的情况单独处理：因为积蓄为 0，只能按副业 1 去赚钱；其它副业的情况赋值为 $-\infty$。由于小牛每年只能执行两次操作中的一种，则

可以得如下递归式：

$$S(i,y)=\begin{cases}p_1, & \text{if } y=1 \text{ and } i=1;\\ -\infty, & \text{if } y=1 \text{ and } i\geq 2;\\ \max\left(\max_{1<k\leq n,k\neq i} S(k,y-1)-c_i, S(i,y-1)+p_i\right), & \text{otherwise}\end{cases}$$

我们构建一个 $m\cdot n$ 的表格来进行记录。无疑，$S(1,1)=p_1$。此外，还需要一个额外的表格来存放每年的最大积蓄。这样在计算下一年的最大积蓄时，可以保证在常数时间复杂度内解决子问题。那么总的时间复杂度为 $O(mn)$。

Sol4.（25 分）我们定义 $P[i]$ 为前 i 个位置的最大期望收益（包含 i）。基于给出的限制，有如下递归式：

$$P[i]=\max\begin{cases}\max_{j<i}\{P[j]+\alpha(m_i,m_j)\cdot p_i\}\\ p_i\end{cases}$$

这里函数 α 定义为：

$$\alpha(m_i,m_j)=\begin{cases}0, & \text{if } m_i-m_j<k\\ 1, & \text{if } m_i-m_j\geq k\end{cases}$$

最大的期望收益来源于位置 j 处 $(j<i)$ 开夜宵摊的最大期望收益 $P[j]$ 以及能否在 i 处开夜宵摊。注意 $P[j]$ 虽然代表 j 处的最大期望收益，但不一定在 j 处会有夜宵摊。（如果没有，则会考虑 $k<j$ 的情况）并且可能出现 p_i 大于 $P[j]+\alpha(m_i,m_j)\cdot p_i$ 的情况。这些情况下我们都需要与 p_i 进行比较。

每次考虑 i 处的最大收益，需要将 $j<i$ 的情况都考虑一遍。由于 i 取值为 $[1,n]$，故上述算法需要 $O(n^2)$ 的时间复杂度。