动态规划与单调性

前言：动态规划与单调性是密不可分的一对兄弟。许多动态规划的题目由于其状态转移方程的特殊性而具有决策或取值上的单调。而这些特殊性质经常成了重要的突破口。现在的试题的发展趋势是多元化，发散化。单纯的动态规划很多时候受到了多方面的限制。这时候对于单调性的发掘应用也就逐渐成了一个重要的课题。下面通过一些例题来领会这种思想。（由于利用队列与斜率进行决策优化已经被分到了数据结构部分，所以这里不予赘述）

例一：服务机构设置问题（2006 年福建省选试题）

在一个按照南北方向划分成规整街区的城市里，n 个居民点分布在一条直线上的 n 个坐标点（x1<x2<x3...）处。居民们希望在城市中至少选择 1 个，但不超过 k 个居民点建立服务机构。

在每个居民点处，服务需求量为 wi（>=0），在该居民点设置服务机构的费用为 ci（>=0）。假设居民点 i 到距其最近的服务机构的距离为 di，则此居民点的服务费用为 di*wi。

建立 k 个服务机构的总费用为 A+B。A 是在 k 个居民点设置服务机构的费用的总和；B 是 n 个居民点服务费用的总和。

任务：

对于给定直线 L 上的 n 个点，编程计算在直线 L 上最多设置 k 处服务机构的最小总费用。

注：N<=1000

分析：

这个问题使我们想起了动态规划的一个经典模型：邮局问题。这里只是将此题加权。于是仿照邮局问题我们很容易的就能得到如下状态转移方程：

设 f[i,j]为在 1..i 号居民点上设立 j 个服务机构，且第 j 号服务机构设在第 i 号居民点上的最小费用。

f[i,j]=minimal(f[k,j-1]+value(k+1,j))+c[j]

其中 1<=k<=i-1，value(k+1,j)为第 k 与第 j 两个居民点上的两个服务机构相邻时，第 k+1 到第 j 这些居民点的最小服务费用。通过 O(n^2)的预处理可以算出来。

这个方程是 O(n^3)的，显然会超时。我们需要寻求一些优化方案。

我们设 s[i,j]为 f[i,j]取到最小值时的 k（即 f[i,j]的决策）。我们来观察 s[i,j]的单调情况。

I> s[i,j]<s[i+1,j]　解释：从直观上说，我们要把 j 个服务机构“平均”的分到一些居民点上，肯定要处理的居民点数越多，服务机构位置越靠后。

II> s[i,j]>s[i,j-1]　解释：从直观上说，我们要把一些服务机构“平均”的分到 i 个居民点上，肯定要处理的服务机构数越多，其位置越靠后。

以上的两个结论都可以用数学方法进行严格证明，这里略去。在考场上我们还可以将 s 数组排成方阵输出，也可以方便的通过观察规律来获得以上结论。

我们将 i> ii>合起来，得到一个很重要的式子：s[i,j-1]<s[i,j]<s[i+1,j]

这个式子告诉我们：每个决策都是有一个范围的，而不必从头扫到尾。现在的问题是我们如何在计算某个 f[i,j]之前得到其决策的范围。

其实非常方法简单：调整计算顺序。我们在外循环内从小到大计算 j，在内循环内从大到小计算 i，那么我们就可以保证在 f[i,j]被计算出以前 s[i,j-1]与 s[i+1,j]都是已知的。

从程序上来看，虽然表面上还是三重循环，实际上的运行效果非常的好。时间复杂度大约在 O(n^2)~O(n^2logn)左右。这个题目被很好的解决了。

例二：小 H 的小屋（NOI 2003）

小 H 发誓要做 21 世纪最伟大的数学家。他认为，做数学家与做歌星一样，第一步要作好包装，不然本事再大也推不出去。为此他决定先在自己的住所上下功夫，让人一看就知道里面住着一个“未来的大数学家”。

为了描述方便，我们以向东为 x 轴正方向，向北为 y 轴正方向，建立平面直角坐标系。小 H 的小屋东西长为 100Hil（Hil 是小 H 自己使用的长度单位，至于怎样折合成“m”，谁也不知道）。东墙和西墙均平行于 y 轴，北墙和南墙分别是斜率为 k1 和 k2 的直线，k1 和 k2 为正实数。北墙和南墙的墙角处有很多块草坪，每块草坪都是一个矩形，矩形的每条边都平行于坐标轴。相邻两块草坪的接触点恰好在墙上，接触点的横坐标被称为它所在墙的“分点”，这些分点必须是 1 到 99 的整数。

小 H 认为，对称与不对称性的结合才能充分体现“数学美”。因此，在北墙角要有 m 块草坪，在南墙角要有 n 块草坪，并约定 m<=n。如果记北墙和南墙的分点集合分别为 X1，X2，则应满足 X1 属于 X2，即北墙的任何一个分点一定是南墙的分点。

由于小 H 目前还没有丰厚的收入，他必须把草坪的造价降到最低，即草坪的占地总面积最小。他请你帮他求出这个面积。

注：2<=m<=n<=100

南北墙距离很远，不会出现南墙草坪和北墙草坪重叠的情况。

分析：

有一个条件是非常醒目的，就是“北墙的任何一个分点一定是南墙的分点”。这意思就是说，我们可以在确定北墙的分点的前提下，分段确定南墙的分割情况。

我们设 f[w,u,v]表示当前南北墙上的草坪长度为 w，且北墙有 u 块草坪，南墙有 v 块草坪时的最小面积。

f[w,u,v]=min{f[w-x,u-1,v-k]+area(x,k)}

其中 x 为枚举的北墙第 u 块草坪的长度，k 为枚举的南墙上对应北墙第 u 块草坪的草坪个数。area 为增加的这些草坪的最小值，可以在常数时间内被计算出来。

答案为 f[100,m,n]，时间复杂度为 O(l^2mn^2)，超时是必然的。

这样做到底慢在哪里了？原因在于我们每求出一个状态就要枚举两个值 x 与 k，这样是极不划算的。我们试图减少这个上面的时间消耗。

假如我们已经确定了 x，那么 k 与 f[w,u,v]有什么关系呢？

我们把 f[w,u,v]随着 k 变化的图像做出，我们惊讶的发现：图像是一个类似于抛物线的图像。我们把递减到递增的转折点记为 p。

由图像可知：p 就是最优的决策。我们可以相继探寻 x 与 p 的关系，发现随着 x 的增加，p 不会减少。

由于 x 与 p 都是递增的，我们可以一边枚举 x 一边把 p 的位置向前移动。这样就把 k 的枚举给省去了。

在计算每个 f[w,u,v]的时候，只枚举了一个 x，时间复杂度降为 O(l^2mn)，足够对付所有的数据了。

例三：鹰蛋（Ural 1223)

一位教授想确定一种鹰蛋的坚硬度 E。他是通过不断从一幢 N 层的楼上向下扔鹰蛋来确定 E 的。如果蛋从第 K 层楼上摔下没碎但是从第 K+1 层楼上摔下后碎了，那么就有 E=K。（0<=E<=N）

如果在一次尝试中鹰蛋没有摔碎，那么它可以继续被使用。如果在一次尝试中鹰蛋摔碎了，那么教授只能拿一个新蛋继续试验。如果所有的蛋用完了但是还是没有得出 E，那么我们认为整个实验失败了。教授现在有 M 个蛋，他想知道在试验成功的前提下确定 E 所需要的最小尝试次数。

分析：

这个题是求最值，于是我们想到了动态规划。

我们设 f[i,j]为用 i 个蛋在 j 层楼上最坏情况下确定 E 所要的次数。

显然 f[1,j]=j（j>=0），这表示说，如果我们只有一个蛋，为了使实验成功，只能从下往上一层一层的试，最坏情况自然是 j。

当 i>=2 时，如果我们在第 w 层扔下了一个蛋：

I> 蛋碎了，此时有 E<w，我们便只能用 i-1 个蛋在下面的 w-1 层中确定 E，这与原问题是一个类似的子问题，答案为 f[i-1,w-1]。

此时 f[i,j]=f[i-1,w-1]+1

II> 蛋没碎，此时有 E>=w，那么我们将在上面的 j-w 层中确定 E。如果我们把这 j-w 层抽象成一栋新楼的话，那么这个问题也与原问题类似。答案为 f[i,j-w]

此时 f[i,j]=f[i,j-w]+1

综上 f[i,j]=min{max{f[i-1,w-1],f[i,j-w]}}+1（1<=w<=j）

复杂度为 O(mn^2)=O(n^3)。我们可以通过讨论 m 与 n 的大小关系来降到 O(n^2logn)。但还是需要一些优化。

思考，如果我们能在 k 次内用 i 个蛋确定 j 层楼，那么我们可以类似的用不超过 k 次内用 i 个蛋确定 j-1 层楼，但是不一定能确定 j+1 层。表达成数学语言就是：f[i,j]>=f[i,j-1]。

那么，若 f[i-1,w-1]<f[i,j-w]，则对于 v<w，必有 f[i,j-v]>=f[i,j-w]

则 max{f[i-1,v-1],f[i,j-v]}+1>=f[i,j-v]+1>=f[i,j-w]+1=max{f[i-1,w-1],f[i,j-w]}+1

我们得出了结论：v 一定不是最优决策。

同理，若 f[i-1,w-1]>f[i,j-w]，则对于 v>w，必有 v 不是最优。

如果我们把 f[i-1,w-1]与 f[i,j-w]作在同一个坐标轴上，则可以发现最大值的图像是一个“V”型。

由以上两点，我们可以对 w 进行二分：每次在中间取一个 w，如果 f[i-1,w-1]>f[i,j-1],则去掉大于 w 的决策，反之去掉小于 w 的决策。那么最后剩下来的决策必定是最优的。

那么每次决策的复杂度降到了 O(logn)，总复杂度降到 O(nlognlogn)。我们的第一步优化取得了成功。能不能再接再厉呢？

思考，如果 f[i,j]等于 k,那么我们在这个基础上确定 j+1 层楼最多只需要在 k 的基础上加 1，因为我们可以先得到 f[i,j]，再测试一次 j+1。表达成数学语言就是：f[i,j-1]<=f[i,j]<=f[i,j-1]+1

那么如果存在一个决策 w，使得 f[i,j]=f[i,j-1]，那么 f[i,j]=f[i,j-1]，否则 f[i,j]=f[i,j-1]+1。

我们维护一个 p，使其始终满足 f[i,p]=f[i,j-1]-1 且 f[i,p+1]=f[i,j-1]（由上面的结论，这样的 p 一定存在）。在计算 f[i,j]时，我们令 w=j-p 来寻找内在联系。

I> 若 f[i,p]>=f[i-1,j-p-1]

那么 f[i,j]=max{f[i-1,w-1],f[i,j-w]}+1=max{f[i-1,j-p-1],f[i,p]}+1=f[i,p]+1=f[i,j-1]-1+1=f[i,j-1]

这说明当前的 w 可以使 f[i,j]=f[i,j-1]

II> 若 f[i,p]<f[i-1,j-p-1] 再任取一个 l

如果 l<p，则有 f[i,j]=max{f[i,l],f[i-1,j-l-1]}+1>=f[i-1,j-l-1]+1>f[i,p]+1=f[i,j-1]

得到 f[i,j]>f[i,j-1]，不可能有 f[i,j]=f[i,j-1]

同理当 $l=p$ 与 $l>p$ 时，当前的 w 都不可能使 $f[i,j]=f[i,j-1]$，由上面的结论，$f[i,j]=f[i,j-1]+1$

综上，我们得到了如下结论：

当 $f[i,p]<f[i-1,j-p-1]$时 $f[i,j]=f[i,j-1]+1$ 否则 $f[i,j]=f[i,j-1]$。

有这个结论我们可以在 $O(1)$的时间内进行状态转移，时间复杂度进一步优化成 $O(n\log n)$。这已经是一个很好的算法了。

总结：从以上 3 个题可以看出，我们在思维上都经历了一个初步思想-->不可行-->寻找单调性优化-->问题解决这样的几个步骤。我们也可以
看出，单调性在动态规划的优化中是一把锋利的刃，他让我们一路过关闯将，劈荆斩棘。但是话又说话来，如果我们不能很好的掌握这些技巧，在推理，思维以及数学上没有一定的功底，那么这把利刃只可能是鞘中之物了。