# 浅析1D/1D动态规划的优化

## Jiangde Doushi Pujizu Neirong

周尚彦

北京大学 信息科学技术学院

Mar. 12th

## Contents

## 1D/1D动态规划

**定义 0.1**

1D/1D动态规划 状态数为$O(n)$，每一个状态的决策量为$O(n)$的动态规划方程。

直接求解的时间复杂度为$O(n^2)$。但是绝大多数这样的方程通过合理的优化可以优化至$O(n \log n)$甚至$O(n)$的时间复杂度。

## 单调队列优化模型

$$f(x) = \min_{k=b[x]}^{x-1} \{g(k)\} + w[x]$$

这个就是利用单调队列进行优化的经典模型。

## 单调队列优化模型

$$f(x)=\min_{k=b[x]}^{x-1}\{g(k)\}+w[x]$$

这个就是利用单调队列进行优化的经典模型。

注意到这样一个性质：$\forall j,k$，如果$j<k$且$g[k]\leq g[j]$，那么决策$j$是毫无用处的。因为根据$b[x]$单调的特性，如果$j$可以作为合法决策，那么$k$一定可以作为合法决策，又因为$k$比$j$

所以如果把待决策表中的决策$g(i)$按照$i$排序的话，则$g(i)$必然是不降的。

所以我们可以用一个单调队列来维护决策表，对于每一个状态$f(x)$来说，计算过程分为以下几步：

- 队首元素出队，直到队首元素在给定的范围中($\geq b[i]$)。
- 此时，队首元素就是状态$f(x)$的最优决策，计算出$f(x)$。
- 计算$g(x)$，并将其插入到单调队列的尾部，同时维持队列的单调性（不断地出队，直到队列单调为止）。

时间复杂度为$O(n)$。 □

## Slide Window

POJ 2823

给定一个长度为$n$的序列，依次输出从$i$开始的长度为$m$的子序列的最小值。
$n \leq 10^6$。

## Slide Window

POJ 2823

给定一个长度为$n$的序列，依次输出从$i$开始的长度为$m$的子序列的最小值。
$n \leq 10^6$。
显然符合上述模型，不断将数列中元素插入单调队列尾部进行维护即可。

## 单调决策性优化模型

$$f(x) = \min_{i=1}^{x-1}\{f(i) + w[i, x]\}$$

## 玩具装箱

HNOI 2008

有$n$个玩具需要装箱，每个玩具的长度为$c_i$，规定装箱必须按照给定顺序进行，且同一个箱子中的两个相邻玩具之间须间隔一个单位长度。

规定一个长度为$l$的箱子的费用为$(l-L)^2$，其中$L$是给定的常数，现在要求用最小的代价将所有玩具装箱。

状态转移方程为:

$$f(x) = \min\{f(i) + w[i+1, x]\}$$

其中$w[i,j] = (j - i + \sum_{k=i}^{j} c[k] - L)^2$。
这个模型是满足上述模型$f(x) = \min_{i=1}^{x-1}\{f(i) + w[i,x]\}$的。

上述模型的优化与决策单调性相关。
令$x$的最优决策点为$k(x)$，决策单调性表述为$\forall i < j, k(i) \leq k(j)$。
具有决策单调性的充要条件是$w[i,x]$满足四边形不等式。

$$w[i,x] + w[i',x'] \leq w[i,x'] + w[i',x](i' < i < x < x')$$

。

由于时间关系，具体数学推导过程略去。

上述模型的优化与决策单调性相关。

令$x$的最优决策点为$k(x)$，决策单调性表述为$\forall i < j, k(i) \leq k(j)$。

具有决策单调性的充要条件是$w[i,x]$满足四边形不等式。

$$w[i,x] + w[i',x'] \leq w[i,x'] + w[i',x](i' < i < x < x')$$

。

由于时间关系，具体数学推导过程略去。

在实战中如何验证$w[i,x]$符合四边形不等式？一般来说不需要验证，只需要使用暴力打出决策表观察即可。

决策单调性优化的过程如下：

使用一个栈来维护数据，栈中元素保存一个决策作为最优决策的起始状态点，显然这些位置相互连接且依次递增。当插入一个新的决策时，从后到前扫描栈，对于每一个老决策来说，做这样两件事：

- 如果在老决策的起点处是新决策好，则退栈，抛弃老决策，将其区间合并至新决策中，继续扫描下一个决策。
- 如果在老决策的起点处是老决策好，则转折点必然在这个老决策的区间中；二分查找，然后新决策进栈，结束。

时间复杂度是$O(n \log n)$。

一开始，只有$f(1)$的函数值被计算出来，于是所有状态的当前最优决策都是1。

$$11111111111111111111111111111$$

现在，显然$f(2)$的值已经确定了：它的最优决策只能是1。我们用决策2来更新决策表。由于决策单调性，我们知道新的决策表会有这样的形式：

$$11111111112222222222222222222$$

使用二分法来查找“转折点”，因为如果在一个点$x$上，如果决策2更好，则所有比$x$大的状态都是决策2更好；如果$x$上决策1更好，则所有比$x$小的状态都是决策1更好。

现在决策1和决策2都已经更新完毕，则$f(3)$也已确定，现在用决策3来更新所有状态。根据决策单调性，现在的决策表只能有以下2种类型：

$$11111111112222222233333333333$$

$$11111111113333333333333333333$$

## 斜率优化模型

$$f(x)=\min_{i=1}^{x-1}\{a[x]\times f(i)+b[x]\times g(i)\}$$

其中$a[x],b[x]$是与决策无关的变量，$g(i)$可以由$f(i)$确定。

## 土地购买

USACO 2008 Mar

有$n$块土地需要购买，每块土地都是长方形的，有特定的长与宽。你可以一次性购买一组土地，价格是这组土地中长的最大值乘以宽的最大值。

比方说一块$5 \times 3$的土地和一块$2 \times 9$的土地在一起购买的价格就是$9 \times 3$。显然，怎样分组购买土地是一门学问，你的任务就是设计一种方案用最少的钱买下所有的土地。输出最小花费。(宽度记为$w[i]$，长度为$l[i]$)

$\forall i,j$，如果$w[i] \leq w[j] \& l[i] \leq l[j]$那么土地$i$可以和$j$一起买下来而不产生花费。

所以我们可以按$w$从小到大排序然后去除多余元素得到一个$l$递减的新序列。我们只需要将新序列划为若干段即可得到答案。

$\forall i,j$，如果$w[i] \le w[j] \& l[i] \le l[j]$那么土地$i$可以和$j$一起买下来而不产生花费。

所以我们可以按$w$从小到大排序然后去除多余元素得到一个$l$递减的新序列。我们只需要将新序列划为若干段即可得到答案。

状态转移方程为$f(x) = w[x] \times l[k+1] + f(k)$

$\forall i,j$，如果$w[i] \le w[j] \& l[i] \le l[j]$那么土地$i$可以和$j$一起买下来而不产生花费。

所以我们可以按$w$从小到大排序然后去除多余元素得到一个$l$递减的新序列。我们只需要将新序列划为若干段即可得到答案。

状态转移方程为$f(x) = w[x] \times l[k+1] + f(k)$

可以验证该方程满足决策单调性，可以用决策单调性优化来做。

$$f(x) = w[x] \times l[k+1] + f(k)$$

将其中只与$k$有关的部分看作$y$，将只与$x$有关的部分看作$b$，将既与$x$又与$k$相关的部分看作$k \times x$（其中与$x$相关的看作$k$，与$k$相关的看作$x$）
原方程就可以写成$y = kx + b$的形式：

$$y(f(k)) = k(-w[x]) \times x(l[k+1]) + b(f(x))$$

把每一个$k$看作平面上的一个点，在求解$f(x)$的过程中，我们要利用已知的斜率$k$来求得一个最小的截距$b$，实际上就是在平面上的线性规划：
把一条斜率确定的直线从负无穷大向上平移，碰到的平面上的第一个点就是最优决策点。
显然，所有的最有决策点都在平面的下凸壳上。

回顾一下求凸包的算法：

- 先将所有点按照$x$坐标排序。
- 使用一个栈来维护当前凸包中的点，每加入一个点时将不在凸壳上的点依次弹出栈。

回顾一下求凸包的算法：

- 先将所有点按照$x$坐标排序。
- 使用一个栈来维护当前凸包中的点，每加入一个点时将不在凸壳上的点依次弹出栈。

在动态规划过程中需要向图中动态加点，也就是需要动态维护凸壳，这显然比较困难。

回顾一下求凸包的算法：

- 先将所有点按照$x$坐标排序。
- 使用一个栈来维护当前凸包中的点，每加入一个点时将不在凸壳上的点依次弹出栈。

在动态规划过程中需要向图中动态加点，也就是需要动态维护凸壳，这显然比较困难。
但在这道题中加入点的$x$坐标是单调的！同时，这道题中的斜率也是单调的（确保其决策单调）！

回顾一下求凸包的算法：

- 先将所有点按照$x$坐标排序。
- 使用一个栈来维护当前凸包中的点，每加入一个点时将不在凸壳上的点依次弹出栈。

在动态规划过程中需要向图中动态加点，也就是需要动态维护凸壳，这显然比较困难。
但在这道题中加入点的$x$坐标是单调的！同时，这道题中的斜率也是单调的（确保其决策单调）！
所以只需要在状态转移时动态维护凸壳并在凸壳上寻找最优决策点即可。

## 货币兑换

NOI 2007

现在有A、B两种金券。

对于第$i$天，A金券价格为$A[i]$元/单位，B金券价格为$B[i]$元/单位。

买入：对于在第$i$天买入的顾客，假设顾客买入$G$元金券，将得到A、B两种金券，它们的总价值为$G$元，且满足A金券比B金券的比例正好为$Rate[i]$。

卖出：顾客提供一个$[0, 100]$的实数$P$，则将$P\%$的A金券和$P\%$的B金券以当天的价格卖出。

现已知未来$n$天每天的$A[i], B[i], Rate[i]$，初始金钱$S$元，求$n$天之后最大获利是多少。

一个贪心的结论是：为了获得最大获利，每次交易时应将手中的钱或金券全部进行交易。
所以任一时刻手中要不全是钱，要不全是金券。
令$f(x)$代表第$x$天结束手里全是钱的最大获利。

$$f(x) = \max\left(f(x-1), \max\left\{\frac{f(i) \times Rate(i) \times A[i] + f(i) \times B[i]}{A[i] \times Rate[i] + B[i]}\right\}\right)$$

$$f(x) = \max\left(f(x-1), \max\left\{\frac{f(i) \times Rate(i) \times A[x] + f(i) \times B[x]}{A[i] \times Rate[i] + B[i]}\right\}\right)$$

max右边的部分不难将其写成$y = kx + b$的形式。

$$\frac{f(i)}{A[i] \times Rate[i] + B[i]} = \frac{-f(i) \times Rate(i) \times A[x]}{(A[i] \times Rate[i] + B[i]) \times B[x]} + \frac{f(x)}{B[x]}$$

然而$k$和$x$都不单调。

## Solution 1

使用平衡树维护凸壳。

## Solution 1

使用平衡树维护凸壳。

- 以$x$坐标作为关键字，维护一个从左到右斜率单调的点集
- 每插入一个点时查询前驱和后继是否任满足凸性，删除至满足凸性为止。
- 注意到斜率为$k$的直线第一个碰到的点$x$满足：$k_{xl} \leq k \leq k_{xr}$。

最终算法复杂度为$O(n \log n)$。

## Solution 2

分治。

## Solution 2

分治。

定义过程$solve(l,r)$：使用$[l,i-1]$中的决策点来更新$f(i)$。则目标过程是$solve(1,n)$。

对于$solve(l,r)$，先递归$solve(l,mid)$，可以得到$f[l\dots mid]$；接着用$f[l\dots mid]$来更新$f[mid+1\dots r]$；然后再递归$solve(mid+1,r)$。

## Solution 2

分治。

定义过程$solve(l,r)$：使用$[l,i-1]$中的决策点来更新$f(i)$。则目标过程是$solve(1,n)$。

对于$solve(l,r)$，先递归$solve(l,mid)$，可以得到$f[l\dots mid]$；接着用$f[l\dots mid]$来更新$f[mid+1\dots r]$；然后再递归$solve(mid+1,r)$。

如何用$f[l\dots mid]$来更新$f[mid+1\dots r]$?

可以排序预处理使得$f[mid+1\dots r]$按照斜率单调排序。

求凸壳的过程在递归两边结束后可以$O(n)$合并。（类似归并排序）

总复杂度为$O(n\log n)$。

## Exercises

- POI 2011 Lightning Conductor
- ZJOI 2007 仓库建设
- POJ 1180 Batch Scheduling
- APIO 2010 特别行动队
- APIO 2014 序列分割
- ICPC World Final 2011 F Machine Works

## Fin.

The end.
Thanks.