动态规划
决策单调性

Robert_JYH
冀昱昊
2024年8月

动态规划(DP)

- 1.状态类型/结构：
  (1)背包、区间、序列、坐标、集合、环形
  (2)树、基环树、图、虚树、笛卡尔树、自动机、仙人掌
  (3)概率期望、数位、计数、容斥、数学模型
- 2.转移方式：递推、记忆化搜索
- 3.优化：
  (1)预处理、前缀和、部分和、费用提前计算
  (2)状态量：跳过无用状态、改进状态表示、状压、倍增
  (3)决策量：利用单调性/凸性
  （四边形不等式、单调队列、斜率优化、分治、WQS二分、SMAWK等）
  (4)加速转移：矩阵（快速幂）、数据结构、FFT、引入组合模型
  (5)空间：滚动数组、bitset
- 4.与其它知识综合：图论、数学、数据结构、字符串

1D/1D问题

- 1D/1D 动态规划，指的是状态数为 $O(n)$，每一个状态决策量为 $O(n)$的动态规划方程，大多关注连续区间的划分问题，形如 $dp[i]=\min\limits_{L(i)\le j\le R(i)}\left(dp[j]+w(j,i)\right)$。
- 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组织与优化都是可以优化到 $O(nlogn)$乃至 $O(n)$的时间复杂度的。
- 优化手段（针对决策量）：四边形不等式、单调队列、斜率优化、SMAWK……

引例 [HAOI2008]玩具装箱

- P 教授有n件玩具，第 i件玩具长度为 C_i,它可以将任意件编号连续的玩具装入一个箱子。
- 如果一个箱子中有多个玩具，那么两件玩具之间要加入一个单位长度的填充物。形式地说，如果将第 i 件玩具到第 j 个玩具放到一个容器中，那么容器的长度将为 $x=j-i+\sum_{k=i}^{j} C_k$ 。
- 如果箱子长度为 x，其制作费用为$(x-L)^2$ 。其中 L 是一个常量。P 教授不关心容器的数目，他可以制作出任意长度的容器。但他希望所有容器的总费用最小。
- $1 \leq n \leq 50000$

引例 [HAOI2008]玩具装箱

- $dp[i] = \min\limits_{1 \le j < i}\left(dp[j] + w(j,i)\right)$,

其中$w(j,i)$= $(i - j - 1 + \sum_{k=1}^{i} C_k - \sum_{k=1}^{j} C_k - L)^2$

一、单调队列优化DP

$dp[i]=\min_{b(i)\le j<i} g[j]+w(i)$，其中$b[i]$随$i$不降，$w(i)$和$j$无关

具体实现

常用结论

对于函数$dp[i]=\min\limits_{b(i)\le j<i} g[j]+w(i)$，其中$b[i]$随$i$不降，如果$\forall j\le k\wedge g[k]\le g[j]$，那么决策$j$可删去。

- 我们可以使用一个单调队列来维护决策表。对于每一个状态 $f[i]$来说，计算过程分为以下三步：

1、 队首元素出队，直到队首元素在给定的范围中。

2、 此时，队首元素就是状态 $f[i]$的最优决策。

3、 维持队列的单调性（不断地出队，直到队列单调为止），计算 $g[i]$，并将其插入到单调队列的尾部。（如果i本身也是决策点，3需要提前）

- 重复上述步骤直到所有的函数值均被计算出来。这样的算法均摊时间复杂度是$O(1)$的。

引例 最大子序和

- 输入一个长度为n的整数序列，从中找出一段不超过m的连续子序列，使得整个序列的和最大。（$1\le n,m\le 300000$）
- $dp[i]=-\min_{b(i)\le j<i} sum(j)+sum(i)$，其中$b(i)=i-m$,随$i$不降
- 计算过程分为以下三步：

1、 队首元素出队，直到队首元素不超出m的范围中。

2、 队首元素就是右端点为i的最优决策。

3、 不断地出队，直到队尾的值小于$sum(i)$，决策i入队。

```
int l = 1, r = 1;
q[1] = 0;
for (int i = 1; i <= n; i++) {
    while (l <= r & q[l] < i - m)
        l++;
    ans = max(ans, sum[i] - sum[q[l]]);
    while (l <= r & sum[q[r]] >= sum[i])
        r--;
    q[++r] = i;
}
```

例1 [POI2014] PTA-Little Bird

- 有 n 棵树排成一排，第 i 棵树的高度是 d_i。
- 有 q 只鸟要从第 1 棵树到第 n 棵树。
- 当第 i 只鸟在第 j 棵树时，它可以飞到第 $j+1, j+2, \cdots, j+k_i$ 棵树。
- 如果一只鸟飞到一颗高度大于等于当前树的树，那么它的劳累值会增加 1，否则不会。
- 由于这些鸟已经体力不支，所以它们想要最小化劳累值。
- $1 \le n \le 10^6$
- $f_i = \min_{i-j \le k} f_j + [d_i \ge d_j]$

例2 CF922E Birds

- 一条直线上有 n 棵树，第 i 棵树上有 c_i 只鸟。
- 在第 i 棵树底下召唤一只鸟的魔法代价是 $cost_i$。每召唤一只鸟，魔法上限会增加 B。从一棵树走到另一棵树，会增加魔法 X。一开始的魔法和魔法上限都是 W。
- 问最多能够召唤的鸟的个数。
- $1 \le n \le 1000, 1 \le B, X, W \le 10^9, 1 \le \sum_{\{i=1\}}^{n} c_i \le 10000$

例2 CF922E Birds

- 设$f[i][j]$表示到第i棵树，召唤了j只鸟所剩下的能量最大值
- $f[i][j] = \max_{\{j-c[i]\le k\le j\}}(f[i-1][k]-(j-k)*cost[i])+X$
- 单调队列优化即可
- 负数不能转移，超上限赋为上限
- 实际上是完全背包

例3 [SCOI2010] 股票交易

- 一共 t 天，第 i 天股票买入价 ap_i，最多可买 as_i 股；卖出价 bp_i，最多可卖 bs_i 股。并且在任何时候你最多持有 $maxp$ 股股票，相邻两次交易（买入或卖出）要间隔 $w+1$ 天。默认初始你有无限的钱，问 t 天后你最多能赚多少钱。
- $1 \leq w < t \leq 2000, 1 \leq maxp \leq 2000$

首先默认 $w = w + 1$。

设 $f_{i,j}$ 表示第 i 天持有 j 张股票时，获得的最大收益。

考虑买入，$f_{i,j} = \max\limits_{j-k \leq as_i} f_{i-w,k} + (k-j) \times ap_i = -j \times ap_i + \max\limits_{j-k \leq as_i} f_{i-w,k} + k \times ap_i$。发现 j 这一维可以单调队列优化。

卖出同理，综上复杂度为 $O(t \times maxp)$。

例4 [NOI 2005]瑰丽华尔兹

- 不妨认为舞厅是一个 N 行 M 列的矩阵，矩阵中的某些方格上堆放了一些家具，其他的则是空地。钢琴可以在空地上滑动，但不能撞上家具或滑出舞厅，否则会损坏钢琴和家具，引来难缠的船长。每个时刻，钢琴都会随着船体倾斜的方向向相邻的方格滑动一格，相邻的方格可以是向东、向西、向南或向北的。而艾米丽可以选择施魔法或不施魔法：如果不施魔法，则钢琴会滑动；如果施魔法，则钢琴会原地不动。
- 艾米丽是个天使，她知道K段时间每段时间的船体的倾斜情况。她想使钢琴在舞厅里滑行的路程尽量长
- $1 \le N, M \le 200, K \le 200, T \le 40000$

例4 [NOI 2005]瑰丽华尔兹

- $\boldsymbol{f(w,i,j)}$ 表示第 w 段时间，走到 $(\boldsymbol{i},\boldsymbol{j})$ 这个格子时的最长路长度。
- 向南走：$f(k,i,j) = \max\limits_{i-len\leqslant pos\leqslant i}\{f(k-1,pos,j)+i-pos\}$
- 单调队列优化即可

二、决策单调性

$$dp[i] = \min_{l[i] \le j < r[i]} \left(g[j] + w(j,i)\right)$$

决策单调性

- 决策单调性：设 $p[i]$ 表示 $dp[i]$ 转移的最优决策点，那么决策单调性可描述为 $\forall i \leqslant j, p[i] \leqslant p[j]$。也就是说随着 i 的增大，所找到的最优决策点是递增态（非严格递增）。
- 完全单调性：任意两个决策点j_1, j_2之间都存在k使得在k之前j_1更优，在k之后j_2更优。
- 方法：二分队列（栈）/分治（整体二分/CDQ）
- 时间复杂度：$O(nlog\, n)$

具体实现（1）——二分队列

建立一个存储三元组 (i, l, r) 的队列，表示 $l \sim r$ 的最优决策点是 i。

（1）判断队尾元素表示区间左端点的决策是否比决策i更优。如果是，则转（2）。否则退队，继续执行（1）。

（2）二分查找在队尾的决策区间里，决策i最优的最左位置pos，把队尾的决策结束位置设为$pos-1$，并把决策i压入队，决策起始位置为pos，结束位置为n。

（3）检查当前决策元素所在区间是否已全部作为决策元素过，若是，删去该区间。

时间复杂度$O(nlogn)$

若决策点单调向左移动，可以用单调栈维护这个过程。

该方法要求单次求w复杂度较低。

主要代码参考

```
    stk[top] = node(0, 1, n);
    for (int i = 1; i <= n; i++) {
        f[i] = f[stk[noww].x] + w(stk[noww].x, i);
        while (i < stk[top].l &&
               f[i] + w(i, stk[top].l) < f[stk[top].x] + w(stk[top].x, stk[top].l))
            top--;
        int k = find(i);
        stk[top].r = k - 1;
        if (k <= n) stk[++top] = node(i, k, n);
        if (i == stk[noww].r) noww++;
    }
```

具体实现（2）——分治

- 考虑我们需要得到 $[L,R]$ 这个区间的 DP 值，并且备选的决策区间是 $[l,r]$。若对于 MID 这个位置的 DP 值取最优时的决策点是 loc，那么由决策单调性，$[L,MID-1]$ 的备选区间就是 $[l,loc]$，而 $[MID+1,R]$的备选区间是 $[loc,r]$ 。这样把问题分成两个子问题分治递归下去。
- 而如何找到 loc，方法是直接暴力扫 $[l,r]$中所有点作为决策点去转移到 MID 上，找到其中的最优值。整个算法时间复杂度是 $O(nlogn)$ 的。
- 适用于w不能以较低复杂度算出，但单个f的计算与其它未计算出f无关的情形

例5 [ICPC2017 WF]Money for Nothing

- 坐标平面上有 m 个红点，n 个蓝点。你需要找到一个边平行于坐标轴的矩形，使得它以一个红点为左下角，蓝点为右上角，且面积最大。
- $m, n \le 5 \times 10^5$
- ***决策单调性：对于每个红点，其围成最大矩形的蓝点具有单调性。***
- ***整体二分***

具体实现（2）——整体二分(示例代码)

```
void divide(int l, int r, int nl, int nr) {
    if (l > r || nl > nr)
        return;
    int mid = (l + r) >> 1;
    ll res = -1e18;
    int Mid = 0;
    for (int i = nl; i <= nr; i++)
        if (a[mid].x < b[i].x || a[mid].y < b[i].y)
            if (a[mid] * b[i] > res)
                res = a[mid] * b[i], Mid = i;
    if (Mid) {
        divide(l, mid - 1, nl, Mid);
        divide(mid + 1, r, Mid, nr);
        ans = max(ans, res);
    }
}
```

三、1D/1D的四边形不等式

$$dp[i] = \min_{1 \le j < i}\left(dp[j] + w(j,i)\right)$$

四边形不等式与凸完全单调性

- 定义：设$w(x,y)$ 为定义在整数集合上的一个二元函数，若对于定义域上的 $\forall a \leqslant b \leqslant c \leqslant d$, 有$w(a,c)+w(b,d) \leqslant w(a,d)+w(b,c)$，那么函数 w 满足四边形不等式。
- 定理：设$w(x,y)$ 为定义在整数集合上的一个二元函数，若 $\forall a < b, w(a,b)+w(a+1,b+1) \leqslant w(a+1,b)+w(a,b+1)$，那么函数 w 满足四边形不等式。
- 凸完全单调性：若一个二元函数$w(x,y)$ 满足四边形不等式，我们称其是凸完全单调的。

一维线性DP的决策单调性

定理（决策单调性）

当函数 w 满足四边形不等式时函数$dp[i]=\min\limits_{1\le j<i}\left(dp[j]+w(j,i)\right)$ 具有决策单调性。

证明： $\because$ $p[i]$ 在 $dp[i]$ 的决策点中最优

$\therefore$ $\forall i\in[1,N],\forall j\in[0,p[i]-1],dp[p[i]]+w(p[i],i)\leqslant dp[j]+w(j,i)$

易知 $\forall i'\in[i+1,N]$，均满足 $j<p[i]<i<i'$

又 $\because$ 函数 w 满足四边形不等式$\therefore$ $w(j,i)+w(p[i],i')\leqslant w(j,i')+w(p[i],i)$

移项得： $w(p[i],i')-w(p[i],i)\leqslant w(j,i')-w(j,i)$

与第一个式子相加，有： $dp[p[i]]+w(p[i],i')\leqslant dp[j]+w(j,i')$

最后的式子含义是：把 $p[i]$ 作为 $dp[i']$ 的决策点，一定比小于 $p[i]$ 的任意一个 j 都要更好。也就是说，$dp[i']$ 的最优决策点不可能小于 $p[i]$，即 $p[i']\geqslant p[i]$，所以方程具有决策单调性。

证明决策单调性

- 尽管由上面的定理，我们可以通过证明函数 w 满足四边形不等式来证明DP方程满足决策单调性，但通常情况证明函数 w 满足四边形不等式较为复杂（但通过展开也是可以证明的）。
- 我们一般通过打出函数 w 的变化表来观察其是否满足四边形不等式。
- 或者，更直接地，我们直接观察DP方程的最优决策点是否满足决策单调性。

引例 [HAOI2008]玩具装箱(决策单调)

- P 教授有n件玩具，第 i件玩具长度为 C_i,它可以将任意件编号连续的玩具装入一个箱子。
- 如果一个箱子中有多个玩具，那么两件玩具之间要加入一个单位长度的填充物。形式地说，如果将第 i 件玩具到第 j 个玩具放到一个容器中，那么容器的长度将为 $x = j - i + \sum_{k=i}^{j} C_k$ 。
- 如果箱子长度为 x，其制作费用为$(x - L)^2$ 。其中 L 是一个常量。P 教授不关心容器的数目，他可以制作出任意长度的容器。但他希望所有容器的总费用最小。
- $1 \le n \le 50000$

引例 [HAOI2008]玩具装箱

- $dp[i] = \min\limits_{1 \le j < i}\left(dp[j] + w(j,i)\right)$，

其中$w(j,i)$= $(i - j - 1 + \sum_{k=1}^{i} C_k - \sum_{k=1}^{j} C_k - L)^2$

- 决策表如图所示，验证可得$w(j,i)$满足四边形不等式，直接套模板优化即可。（证明方法：带入作差）

	$j = 1$	$j = 2$	$j = 3$	$j = 4$
$i = 2$	0			
$i = 3$	9	4		
$i = 4$	25	0	9	
$i = 5$	100	25	4	0

例6 [NOI2009]诗人小G

- 小 G 是一个出色的诗人，经常作诗自娱自乐。但是，他一直被一件事情所困扰，那就是诗的排版问题。
- 一首诗包含了若干个句子，对于一些连续的短句，可以将它们用空格隔开并放在一行中，注意一行中可以放的句子数目是没有限制的。小 G 给每首诗定义了一个行标准长度（行的长度为一行中符号的总个数），他希望排版后每行的长度都和行标准长度相差不远。显然排版时，不应改变原有的句子顺序，并且小 G 不允许把一个句子分在两行或者更多的行内。在满足上面两个条件的情况下，小 G 对于排版中的每行定义了一个不协调度, 为这行的实际长度与行标准长度差值绝对值的 *P* 次方，而一个排版的不协调度为所有行不协调度的总和。
- 小 G 最近又作了几首诗，现在请你对这首诗进行排版，使得排版后的诗尽量协调（即不协调度尽量小），并把排版的结果告诉他。
- $w(j,i) = |sum_i - sum_j - 1 - l|^p$,满足四边形不等式

例7 CF868F Yet Another Minimization Problem

- 给定一个序列 a，要把它分成 k 个子段。每个子段的费用是其中相同元素的对数。求所有子段的费用之和的最小值。
- $n \leq 10^5, k \leq 20$

例7 CF868F Yet Another Minimization Problem

- 设前i个数，分了j段
- $f[i][j] = \min\limits_{1 \le k \le i} f[k-1][j-1] + w(k,i)$
- $w(k,i)$满足四边形不等式，可以通过设区间内某数个数证明。
- w直接计算复杂度较高。但每一层的DP值计算与当前层无关，使用分治，用类似莫队方法移动端点更新答案
- 这是一个“假二维”

例8 [POI 2011]Lightning Conductor

给定一个长度为 n 的序列 $\{a_n\}$，对于每个 $i \in [1, n]$，求出一个最小的非负整数 p，使得 $\forall j \in [1, n]$，都有 $a_j \le a_i + p - \sqrt{|i-j|}$

$1 \le n \le 5 \times 10^5$，$0 \le a_i \le 10^9$。

四、区间DP的四边形不等式

$$f_{\{l,r\}} = min\ f_{\{l,k\}} + f_{\{k+1,r\}} + w(l,r)$$

四边形不等式与区间单调性

- 四边形不等式：设$w(x,y)$为定义在整数集合上的一个二元函数，若对于定义域上的 $\forall a \leqslant b \leqslant c \leqslant d$，有$w(a,c)+w(b,d) \leqslant w(a,d)+w(b,c)$，那么函数 w 满足四边形不等式。
- 区间单调性： $\forall a \leqslant b \leqslant c \leqslant d$，有$w(b,c) \leqslant w(a,d)$，那么函数 w 存在区间单调性。
- 若w同时满足四边形不等式和区间单调性，且两者不等号方向一直，则$f_{\{l,r\}} = min\ f_{\{l,k\}} + f_{\{k+1,r\}} + w(l,r)$ 具有决策单调性。
- 单调性引理：设$s_{l,r}$是最优决策点，则$s_{i,j-1} \leq s_{i,j} \leq s_{i+1,j}$
- 利用引理可以将区间DP优化到$O(n^2)$

例9 [NOI1995] 石子合并

- 在一个圆形操场的四周摆放 N 堆石子，现要将石子有次序地合并成一堆，规定每次只能选相邻的 2 堆合并成新的一堆，并将新的一堆的石子数，记为该次合并的得分。
- 试设计出一个算法,计算出将 N 堆石子合并成 1 堆的最小得分和最大得分。

五、双线性的斜率优化DP

$f[i] = \min f[j] + g]i] + g[j] + s[i]s[j]$

双线性函数

- 线性函数：若$f(k_1a_1+k_2a_2)=k_1f(a_1)+k_2f(a_2)$成立，我们称其为线性函数。
  - 一次函数，常函数。
- 双线性函数：双线性函数在一个变元固定时,是另一个变元的线性函数。
- 双线性函数一定是凸完全单调的。
- $O(n)$斜率优化的条件是对应函数为双线性函数。

例10 [HAOI2008]玩具装箱(斜率优化)

- P 教授有n件玩具，第 i件玩具长度为 C_i,它可以将任意件编号连续的玩具装入一个箱子。
- 如果将第 i 件玩具到第 j 个玩具放到一个容器中，那么容器的长度将为 $x = j - i + \sum_{k=i}^{j} C_k$，其制作费用为$(x - L)^2$。他希望所有容器的总费用最小。
- 设 $S[n] = \sum_{\mathrm{i=1}}^{\mathrm{n}}(C[i] + 1)$，将 L 提前加 1，用 $dp[i]$ 表示装前 i 个的最小花费，转移方程为：$dp[i] = \min(dp[j] + (S[i] - S[j] - L)^2)$。
- 把 min 去掉，化简得：$dp[i] = S[i]^2 - 2S[i]L + dp[j] + (S[j] + L)^2 - 2S[i]S[j]$
- 把同类型的项用括号括起来，即：$dp[i] - (S[i]^2 - 2S[i]L)$

$= (dp[j] + (S[j] + L)^2) + (-2S[i]S[j])$ ——双线性函数

本质：线性规划

- 目标函数：$z = dp[i]$
- 函数：$\underbrace{(dp[j] + (S[j] + L)^2)}_{y} = \underbrace{2S[i]}_{k}\,\underbrace{S[j]}_{x} + \underbrace{dp[i] - (S[i]^2 - 2S[i]L)}_{b}$
- 可行域：(x, y)点集
- 最优解：一根斜率固定的直线从下往上移动到第一个切到的点时停止时，对应的总截距

思考方向：斜率法（斜率表示）

- 设 $j_1,j_2(0\leqslant j_1<j_2<i)$为 i 的两个决策点，满足决策点 j_2优于 j_1，
- 有：$(-2S[i]S[j_2])+(dp[j_2]+(S[j_2]+L)^2)\leqslant(-2S[i]S[j_1])+(dp[j_1]+(S[j_1]+L)^2)$
- 参变分离,用 $Function(j)$ 来表示出 $Function(i)$ 。

> 移项得：$-2S[i](S[j_2]-S[j_1])\leqslant(dp[j_1]+(S[j_1]+L)^2)-(dp[j_2]+(S[j_2]+L)^2)$
>
> $\because C[j]\geqslant 1$
>
> $\therefore S[j+1]>S[j]$
>
> 又 $\because j_2>j_1$
>
> $\therefore S[j_2]-S[j_1]>0$
>
> $\therefore 2S[i]\geqslant\frac{(dp[j_2]+(S[j_2]+L)^2)-(dp[j_1]+(S[j_1]+L)^2)}{S[j_2]-S[j_1]}$
>
> 设 $Y(j)=dp[j]+(S[j]+L)^2,X(j)=S[j]$,
>
> 即 $2S[i]\geqslant\frac{Y(j_2)-Y(j_1)}{X(j_2)-X(j_1)}$

思考方向：斜率法（比较方法）

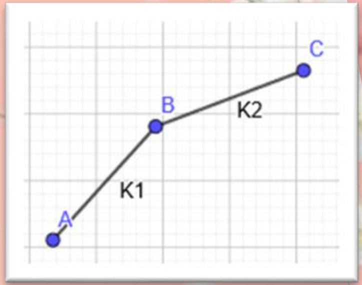
- 等式右边是一个关于点 $P(j_2)$和 $P(j_1)$的斜率式，其中 $P(j)=\big(X(j),Y(j)\big)=(S[j],dp[j]+(S[j]+L)^2)$。

常用结论（具体问题需具体分析）

对于本题中方程，$K=2S[i]$，如果存在两个决策点 j_1,j_2满足 $(0\leqslant j_1<j_2<i)$，使得不等式 $\frac{Y(j_2)-Y(j_1)}{X(j_2)-X(j_1)}\leqslant K$成立，或者说使得 $P(j_2),P(j_1)$两点所形成直线的斜率小于等于 K，那么决策点 j_2优于 j_1。

思考方向：斜率法（比较方法）

- 假设有三个点 $P(j1), P(j2), P(j3)$，$k1, k2$ 为斜率，如下图所示情况：
- 显然有 $k2 < k1$。设 $k0 = 2S[i]$，由上述结论可知：



(a). 若 $k1 \leqslant k0$，则 $j2$ 优于 $j1$ 。反之，则 $j1$ 优于 $j2$ 。

(b). 若 $k2 \leqslant k0$，则 $j3$ 优于 $j2$ 。反之，则 $j2$ 优于 $j3$ 。

- 于是这里可以分三种情况来讨论：

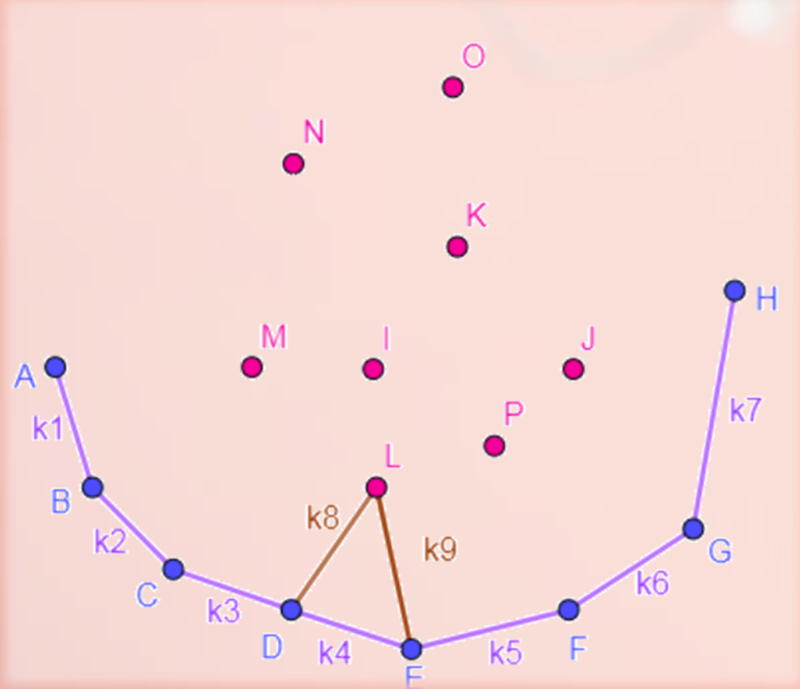
(1). $k0 < k2 < k1$。由 (a),(b) 可知：$j1$ 优于 $j2$ 优于 $j3$ 。

(2). $k2 \leqslant k0 < k1$。由 (a),(b) 可知：$j1$ 和 $j3$ 均优于 $j2$。

(3). $k2 < k1 \leqslant k0$。由 (a),(b) 可知：$j3$ 优于 $j2$ 优于 $j1$ 。

思考方向：斜率法（凸包）

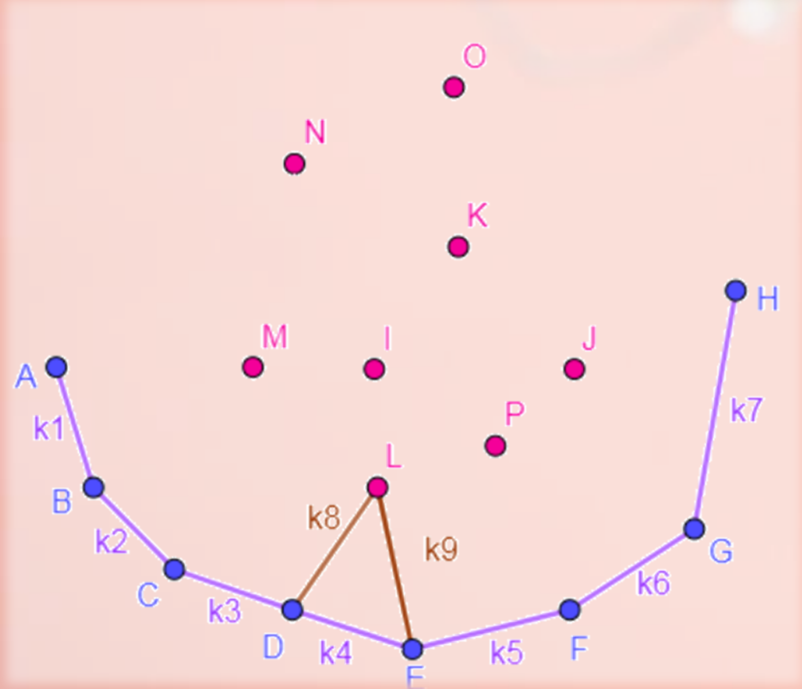
- 对于这三种情况，$j2$ 始终不是最优解，于是我们可以将 $j2$ 从候选决策点中删除。
- 实际上在图中选取候选决策点相连，它们形成了一个下凸壳。



- 凸包：不严谨的讲，给定二维平面上的点集，凸包就是将最外层的点连接起来构成的凸多边形，它能包含点集中所有的点。
- 下凸壳：凸包中所连线段斜率不断变大的点集。

思考方向：斜率法（凸包）

- 对于这三种情况，$j2$ 始终不是最优解，我们可以将 $j2$ 从候选决策点中删除。
- 实际上在图中选取候选决策点相连，它们形成了一个下凸壳。



- 由以上方法可知：我们要找到凸包中斜率小于K_0的斜率最大线段的右端点，即斜率大于K_0的斜率第一个线段的左端点。

本质：线性规划

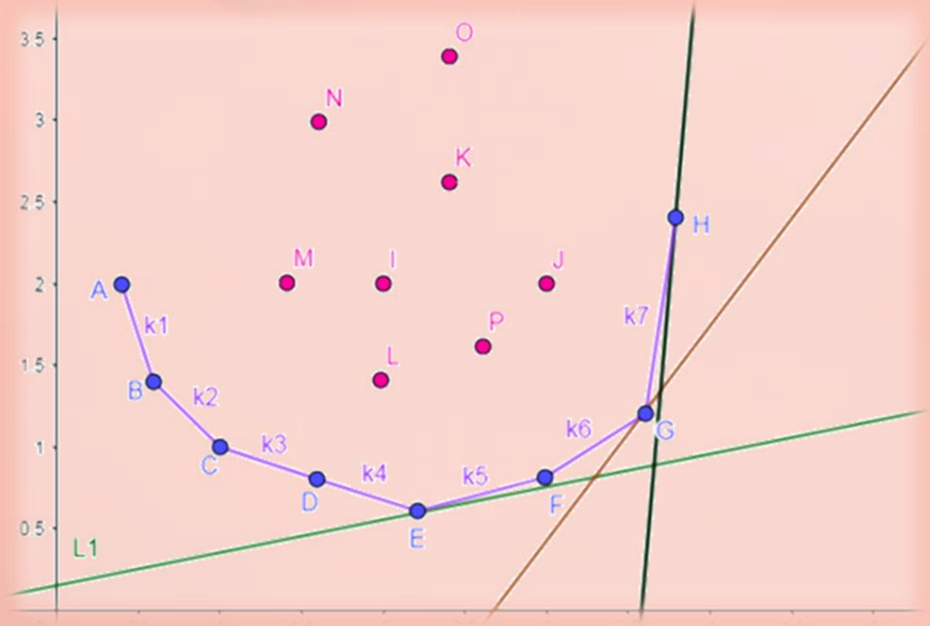
- 目标函数：$z = dp[i]$
- 函数：$\underbrace{(dp[j] + (S[j] + L)^2)}_{y} = \underbrace{2S[i]}_{k}\,\underbrace{S[j]}_{x} + \underbrace{dp[i] - (S[i]^2 - 2S[i]L)}_{b}$
- 可行域：下凸壳上可行的点集
- 约束条件：凸壳的斜率单调限制，枚举范围限制等
- 最优解：凸包中斜率小于K_0的斜率最大线段的右端点（过该点的斜率为$2S[i]$的直线的纵截距最小）

本质：线性规划

- 目标函数：$z = dp[i]$
- 函数：$\underbrace{(dp[j] + (S[j] + L)^2)}_{y} = \underbrace{2S[i]}_{k}\,\underbrace{S[j]}_{x} + \underbrace{dp[i] - (S[i]^2 - 2S[i]L)}_{b}$
- 本题函数满足四边形不等式，用单调队列维护凸包点集：
- (1)判断当队首的第一根线段斜率小于等于 $k_0[i]$ 时就出队，直至斜率大于 $k_0[i]$，在凸包上找到最优决策点 j 。
- (2)用最优决策点 j 更新 $dp[i]$ 。
- (3)将 i 作为一个决策点加入图形并更新凸包（如果点 i 也是 $dp[i]$ 的决策点之一，则需要将 (3) 换到最前面）。

决策单调性的本质

- 条件：$k=2S[i]$单调递增，$x=S[j]$单调递增
- 代数：k递增就说明我们找到的第一个斜率大于k_0的线段在不断地向后移，也就是说，如果我们找到了某一个最优决策点 j，那么在下一次决策中，最优决策点 j' 必定在 j 的后面。
- 几何：



主要代码参考

```
inline long double slope(int i, int j) {
    return (long double)(Y(j) - Y(i)) / (X(j) - X(i));
}
    q[++r] = 0;
    for (int i = 1; i <= n; i++) {
        while (l < r && slope(q[l], q[l + 1]) <= K(i))
            l++;
        j = q[l];
        dp[i] = dp[j] + (s[i] - s[j] - L) * (s[i] - s[j] - L);
        while (l < r && slope(q[r - 1], q[r]) >= slope(q[r], i))
            r--;
        q[++r] = i;
    }
```

例11 [Usaco2008 Mar] 土地购买

- 农夫John准备扩大他的农场,他正在考虑n块长方形的土地。
- 每块土地的价格是它的面积,但他可以同时购买多块土地. 这些土地的价格是它们最大的长a乘以它们最大的宽b。
- 他希望买下所有的土地,但是他发现分组来买这些土地可以节省经费。 他需要你帮助他找到最小的经费。
- $1 \le n \le 50000$

- 按长度降序排列 $dp[i] = \min\limits_{1 \le j < i}\left(dp[j] + l[j+1] \times \max\limits_{j+1 \le k \le i} w[k]\right)$
- 对于一块土地，如果存在另一块土地，其长、宽都大于等于它的长、宽，那么这块土地可不考虑。
- $dp[i] = \min\limits_{1 \le j < i}(dp[j] + l[j+1] \times w[i])$，满足四边形不等式，满足双线性，可以斜率优化

例12 [APIO 2010]特别行动队

你有一支由 n 名预备役士兵组成的部队，士兵从 1 到 n 编号，你要将他们拆分成若干特别行动队调入战场。出于默契的考虑，同一支特别行动队中队员的编号**应该连续**，即为形如 $(i, i+1, \cdots i+k)$的序列。所有的队员都应该属于且仅属于一支特别行动队。

编号为 i 的士兵的初始战斗力为 x_i ，一支特别行动队的初始战斗力 X 为队内士兵初始战斗力之和，即 $X = x_i + x_{i+1} + \cdots + x_{i+k}$。

通过长期的观察，你总结出对于一支初始战斗力为 X 的特别行动队，其修正战斗力 $X' = aX^2 + bX + c$，其中 a，b，c 是已知的系数（$a < 0$）。作为部队统帅，现在你要为这支部队进行编队，使得所有特别行动队的修正战斗力之和最大。试求出这个最大和。

例13 [CEOI2004] 锯木厂选址

- 从山顶上到山底下沿着一条直线种植了 n 棵树。当地的政府决定把他们砍下来。为了不浪费任 何一棵木材，树被砍倒后要运送到锯木厂。木材只能按照一个方向运输：朝山下运。山脚下有一个锯木厂。另外两个锯木厂将新修建在山路上。你必须决定在哪里修建两个锯木厂，使得传输的费用 总和最小。假定运输每公斤木材每米需要一分钱。

例14 丝之割

形式化题意：有 m 条弦，一条弦可以抽象为一个二元组 (u,v)，你可以进行任意次切割操作，一次切割操作你将选择两个下标 i 和 j 满足 $i,j\in[1,n]$，然后所有满足 $u>i,v<j$ 的弦 (u,v) 都将被破坏，同时你将付出 $a_i\times b_j$ 的代价。求破坏所有弦的最小代价和。

对于所有测试点，保证 $1\le n,m\le 3\times 10^5$，$2\le u\le n$，$1\le v\le n-1$，$1\le a_i,b_i\le 10^6$。

六、一般性的斜率优化DP

$f[i] = \min f[j] + g]i] + g[j] + s[i]s[j]$，$s[i]/s[j]$不一定单调

例15 [SDOI2012] 任务安排

机器上有 n 个需要处理的任务，它们构成了一个序列。这些任务被标号为 1 到 n，因此序列的排列为 $1, 2, 3 \cdots n$。这 n 个任务被分成若干批，每批包含相邻的若干任务。从时刻 0 开始，这些任务被分批加工，第 i 个任务单独完成所需的时间是 T_i。在每批任务开始前，机器需要启动时间 s，而完成这批任务所需的时间是各个任务需要时间的总和。

注意，同一批任务将在同一时刻完成。每个任务的费用是它的完成时刻乘以一个费用系数 C_i。

请确定一个分组方案，使得总费用最小。

对于 100% 数据，$1 \le n \le 3 \times 10^5$，$1 \le s \le 2^8$，$|T_i| \le 2^8$，$0 \le C_i \le 2^8$

- T_i为负数意味着斜率不单调，不能轻易出队，需要二分斜率

例16 [NOI2014]购票

- 今年夏天，NOI 在 SZ 市迎来了她 30 周岁的生日。来自全国 n 个城市的 OIer 们都会从各地出发，到 SZ 市参加这次盛会。
- 全国的城市构成了一棵以 SZ 市为根的边有边权s_v的有根树。
- 从城市 v 前往SZ市的方法为：选择城市 v 的一个祖先 a，支付购票的费用，乘坐交通工具到达 a。再选择城市 a 的一个祖先 b，支付费用并到达 b。以此类推，直至到达 SZ 市。
- 对于任意一个城市 v，我们会给出一个交通工具的距离限制 l_v。对于城市 v 的祖先 a，只有当它们之间所有道路的总长度不超过 l_v 时，从城市 v 才可以通过一次购票到达城市 a。对于每个城市 v，我们还会给出两个非负整数 p_v, q_v作为票价参数。若城市 v 到城市 a 所有道路的总长度为 d，那么从城市 v 到城市 a 购买的票价为 $d \cdot p_v + q_v$。
- 每个城市的 OIer 都希望自己到达 SZ 市时，用于购票的总资金最少。你的任务就是，告诉每个城市的 OIer 他们所花的最少资金是多少。
- $n \leq 2 \times 10^5$

例16 [NOI2014]购票

- $O(n^2)$：$f_i = \min\limits_{j \in anc(i), D_i - D_j \le l_i} \{f_j + p_i(D_i - D_j) + q_i\}$
- 链：
- 1.考虑斜率优化
- 设$f_k + p_i(D_i - D_k) + q_i < f_j + p_i(D_i - D_j) + q_i$，则有$\frac{f_k - f_j}{D_k - D_j} < p_i$
- 由于p_i不单调，我们需要在凸包上二分斜率，而不能直接维护队列
- 2.考虑$D_i - D_j \le l_i$的限制，即$D_i - l_i \le D_j$
- 这个限制不具有单调性，我们可以按$D_i - l_i$由大到小排序后的结果将j加入凸包

例16 [NOI2014]购票

- 3.如果转移的时候f_j还没有求出怎么办？
- CDQ分治
- 如何搬到树上？
- 点分治
- 先处理重心及根的一块，再递归处理子树

例17 [NOI2007] 货币兑换

小 Y 最近在一家金券交易所工作。该金券交易所只发行交易两种金券：A 纪念券（以下简称 A 券）和 B 纪念券（以下简称 B 券）。每个持有金券的顾客都有一个自己的帐户。金券的数目可以是一个实数。

每天随着市场的起伏波动，两种金券都有自己当时的价值，即每一单位金券当天可以兑换的人民币数目。我们记录第 K 天中 A 券和 B 券的价值分别为 A_K 和 B_K（元/单位金券）。

为了方便顾客，金券交易所提供了一种非常方便的交易方式：比例交易法。

比例交易法分为两个方面：

a) 卖出金券：顾客提供一个 $[0, 100]$ 内的实数 OP 作为卖出比例，其意义为：将 $OP\%$ 的 A 券和 $OP\%$ 的 B 券以当时的价值兑换为人民币；

b) 买入金券：顾客支付 IP 元人民币，交易所将会兑换给用户总价值为 IP 的金券，并且，满足提供给顾客的 A 券和 B 券的比例在第 K 天恰好为 Rate_K；

例17 [NOI2007] 货币兑换

注意到，同一天内可以进行多次操作。

小 Y 是一个很有经济头脑的员工，通过较长时间的运作和行情测算，他已经知道了未来 N 天内的 A 券和 B 券的价值以及 Rate。他还希望能够计算出来，如果开始时拥有 S 元钱，那么 N 天后最多能够获得多少元钱。

$0 < A_K \le 10$，$0 < B_K \le 10$，$0 < \mathrm{Rate}_K \le 100$，$\mathrm{MaxProfit} \le 10^9$。

- x坐标不单调开平衡树/cdq分治/李超树

例17 [NOI2007] 货币兑换

设在第 i 天用 d 元能买到 cR_i 数量的 A 券和 c 数量的 B 券，有 $cR_iA_i + cB_i = d$，解得 $c = \dfrac{d}{R_iA_i + B_i}$。于是设第 i 天最多能获得多少钱，有：

$$f_i = \max_{1 \le j < i} \frac{f_jR_j}{R_jA_j + B_j}A_i + \frac{f_j}{R_jA_j + B_j}B_i$$

稍微化简可得 $f_i = \max_{1 \le j < i} cR_jA_i + cB_i$，其中 c 只与 j 有关。注意到有两个和 i, j 同时有关的项，所以用不起来斜率优化了吗？Nope！由于没有只与 j 有关的项，所以我们可以进行一些移项：将其看作直线 $ax + by = c$ 化简得到 $y = \dfrac{c - ax}{b}$，本题中即 $c = \dfrac{f_i}{B_i} - cR_j \times \dfrac{A_i}{B_i}$。然后使用斜率优化即可。

注意到斜率和插入点横坐标都不单调，所以需要使用李超线段树。虽然查询位置不是整数，但是注意到我们已经知道了所有查询位置 $\dfrac{A_i}{B_i}$，故将 $\dfrac{A_i}{B_i}$ 离散化即可。

时间复杂度 $\mathcal{O}(n \log n)$。

总结

做题步骤：

（1）由线性DP列出线性规划方程 $y(j)=k(i)x(j)+b(i,others)$

（若$x(j)$单减，最好将$x(j)$设置为单增）

（2）由min/max，增减性分析 $\geq k/\leq k$，上凸/下凸

（3）按三步骤打代码（若不满足都单调则不能$O(n)$实现）

斜率单调暴力移指针　斜率不单调二分找答案

x坐标单调开单调队列　x坐标不单调开平衡树/cdq分治/李超树

七、决策单调性的矩阵表示

蒙日矩阵

- 一个矩阵A是蒙日矩阵，当且仅当对于所有的2×2的子矩阵，都满足四边形不等式：$A(a,b)+A(a+1,b+1) \leqslant A(a+1,b)+A(a,b+1)$.
- $f[i]=\min g[j]+w(j,i)=\min A[j][i]$
- $f[i]$的最优决策点即为矩阵A^T每一行取最小值的位置。
- 在矩阵上我们可以以$O(n+m)$的复杂度完成这件事情，这种算法称为SMAWK算法。
- 推荐阅读：2017集训队论文《浅谈决策单调性动态规划的线性解法》
- 2022集训队论文《浅谈一类蒙日矩阵》
- Lawrence L. Larmore《The SMAWK Algorithm》

题单

<决策单调性： 单调队列>

- 例1 [POI2014] PTA-Little Bird
- 例2 CF922E Birds
- 例3 [SCOI2010] 股票交易
- 例4 [NOI 2005]瑰丽华尔兹

题单

<决策单调性：四边形不等式>

- 例5 [ICPC2017 WF]Money for Nothing
- 例6 [NOI2009]诗人小G
- 例7 CF868F Yet Another Minimization Problem
- 例8 [POI 2011]Lightning Conductor
- 例9 [NOI1995] 石子合并

题单

<决策单调性：斜率优化>

- 例10 [HAOI2008]玩具装箱
- 例11 [Usaco2008 Mar] 土地购买
- 例12 [APIO 2010]特别行动队
- 例13 [CEOI2004] 锯木厂选址
- 例14 丝之割
- 例15 [SDOI2012] 任务安排
- 例16 [NOI2014]购票
- 例17 [NOI2007] 货币兑换

参考资料

[1]辰星凌《斜率优化》

[2]DP optimization - Knuth Optimization

[3]Lawrence L. Larmore《The SMAWK Algorithm》

[4]WQS 二分学习笔记 - Sshwy's Notes

[5] [总结]一些 DP 优化方法 - NaVi_Awson - 博客园 (cnblogs.com)

[6] DP 优化方法大杂烩 I. - qAlex_Weiq - 博客园 (cnblogs.com)

Thanks