XJOI 进阶讲义（七）

决策单调性优化 DP

Mr_Spade

2021.10

决策单调性优化是一个对求最优化的动态规划进行转移加速的一种方法，通常可以起到很明显的优化效果。

决策单调性优化具体的实现方式很多，在题目中应用方式有限但比较灵活，大家要熟练掌握各种方式的适用范围。同时，也有一些转移不满足决策单调性但也有相似之处，对它们的加速方法也会在本讲中介绍。

决策单调性优化的问题一般难度不大，关键在于了解常用的方法及其适用范围。

决策单调性优化在早年的 NOI 中是较为常见的考点，但近年显得冷门一些，使许多同学做完常用方法的模板题之后就不再深究，对于方法具体的应用范围了解得也非常模糊，使得这一知识点成为了很多人的盲区。但作为一种应用较为广泛的算法，决策单调性优化仍可能在未来的比赛中出现，因此掌握常用的应用方式是非常重要的。

对于决策单调性优化 DP 的讨论主要集中于两种类型的转移方程中：前缀转移与区间转移。转移类型不同，则决策单调性的适用范围和应用方法也不同。

形如

$$f[i] = \min_{j<i} g[j] + w(j, i)$$

的方程被称为前缀转移方程，其特点是对每个前缀设计状态。如果 $f = g$，这样的转移被称为自转移，否则称为他转移。

对于 i，若 $j<i$ 满足对任意 $j'<i$，有 $g[j]+w(j,i) \le g[j']+w(j',i)$，则称 j 是 i 的决策点。为了方便，我们通常将 i 称为被决策点。

对于同一个 i，可能存在多个决策点。

对于 i，若 $j < i$ 满足对任意 $j' < i$，有 $g[j] + w(j, i) \le g[j'] + w(j', i)$，则称 j 是 i 的决策点。为了方便，我们通常将 i 称为被决策点。

对于同一个 i，可能存在多个决策点。

对于前缀转移，决策单调性的定义如下：

若 $i_1 \le i_2$，则对于**任意** i_1 的决策点 j_1，存在 $j_1 \le j_2$ 使得 j_2 是 i_2 的决策点；对于**任意** i_2 的决策点 j_2，存在 $j_1 \le j_2$ 使得 j_1 是 i_1 的决策点。

对于 i，若 $j < i$ 满足对任意 $j' < i$，有 $g[j] + w(j, i) \le g[j'] + w(j', i)$，则称 j 是 i 的决策点。为了方便，我们通常将 i 称为被决策点。

对于同一个 i，可能存在多个决策点。

对于前缀转移，决策单调性的定义如下：

若 $i_1 \le i_2$，则对于**任意** i_1 的决策点 j_1，存在 $j_1 \le j_2$ 使得 j_2 是 i_2 的决策点；对于**任意** i_2 的决策点 j_2，存在 $j_1 \le j_2$ 使得 j_1 是 i_1 的决策点。

在这一定义中，有两点需要注意：由于第一个决策点的任意性，因此不必担心选出不合适的决策点使得决策单调性不再成立；另一方面，由于第二个决策点的存在性，因此对于一列特定的决策点，如最左端的决策点，由定义并不一定满足决策单调性。

分治策略

如果一个前缀转移具有决策单调性，并且是他转移，那么可以直接使用分治策略求解 f。

由于是他转移，故直接取 $mid = \lfloor \frac{n+1}{2} \rfloor$，求解 $f[mid]$ 并求出其任一决策点 k。递归求解 f 在 $[1, mid-1]$ 和 $[mid+1, n]$ 的部分，由于决策单调性，左半部分可以在 $1 \sim k$ 的范围内扫描，右半部分可以在 $k \sim n$ 的范围内扫描。进一步进行分治，能够发现每一层扫描的总长度都为 $O(n)$，于是总复杂度为 $O(n \log n)$。

分治策略

如果一个前缀转移具有决策单调性，并且是他转移，那么可以直接使用分治策略求解 f。

由于是他转移，故直接取 $mid = \lfloor \frac{n+1}{2} \rfloor$，求解 $f[mid]$ 并求出其任一决策点 k。递归求解 f 在 $[1, mid-1]$ 和 $[mid+1, n]$ 的部分，由于决策单调性，左半部分可以在 $1 \sim k$ 的范围内扫描，右半部分可以在 $k \sim n$ 的范围内扫描。进一步进行分治，能够发现每一层扫描的总长度都为 $O(n)$，于是总复杂度为 $O(n \log n)$。

如果转移为自转移，显然不能先求解 $f[mid]$，于是分治策略失效。

分治策略只能应用到他转移中。对于自转移，决策单调性目前无法发挥作用对转移进行优化，我们需要进行更多的假设，使得转移具有更良好的性质。

若对任意 $a \le b \le c \le d$，总有 $w(a,c)+w(b,d) \le w(b,c)+w(a,d)$ 成立，则称 w 满足四边形不等式。

若对任意 $a \le b \le c \le d$，总有 $w(a, c) + w(b, d) \le w(b, c) + w(a, d)$ 成立，则称 w 满足四边形不等式。

Theorem 1

若 w 满足四边形不等式，则 f 具有决策单调性。

若对任意 $a \le b \le c \le d$，总有 $w(a,c) + w(b,d) \le w(b,c) + w(a,d)$ 成立，则称 w 满足四边形不等式。

Theorem 1

若 w 满足四边形不等式，则 f 具有决策单调性。

Proof.

对任意 $i_1 < i_2$，假设 j_1 是 i_1 的某个决策点，则显然对任意 $j' < j_1$，有 $g[j_1] + w(j_1, i_1) \le g[j'] + w(j', i_1)$，在四边形不等式中代入
$a = j', b = j_1, c = i_1, d = i_2$，得到
$w(j', i_1) + w(j_1, i_2) \le w(j_1, i_1) + w(j', i_2)$，两式结合，得到
$g[j_1] + w(j_1, i_2) \le g[j'] + w(j', i_2)$，于是 i_2 必然存在不小于 j_1 的决策点 j_2。另一方向的证明是类似的。 □

除了决策单调性之外，四边形不等式还能带来一些更好的性质。

考虑改写四边形不等式为一个等价形式：
$w(a, d) - w(a, c) \geq w(b, d) - w(b, c)$，这也等价于
$(g[a] + w(a, d)) - (g[a] + w(a, c)) \geq (g[b] + w(b, d)) - (g[b] + w(b, c))$。

上式就是说，如果被决策点从 c 增大到 d，那么 a 转移到被决策点的代价的增量不小于 b 转移到被决策点的代价的增量。我们将这一与四边形不等式等价的不等式称为“逐渐变劣”，这意味着更前方的决策点每次增加的代价不小于更后方的决策点，从而随着被决策点的增大越来越处于劣势。

“逐渐变劣”暗示了四边形不等式能导出比决策单调性更强的性质：对于任意 $j_1 < j_2$，一旦对于某个被决策点 i，有从 j_2 转移优于从 j_1 转移，那么由于“逐渐变劣”，对于任意 $i' > i$，都有从 j_2 转移优于从 j_1 转移。这意味着 $j_1 < j_2$ 后的点存在某个分界点 x，对于 $< x$ 的被决策点，j_1 比 j_2 更优，对于 $\geq x$ 的被决策点，j_2 比 j_1 更优。

“逐渐变劣”暗示了四边形不等式能导出比决策单调性更强的性质：对于任意 $j_1 < j_2$，一旦对于某个被决策点 i，有从 j_2 转移优于从 j_1 转移，那么由于“逐渐变劣”，对于任意 $i' > i$，都有从 j_2 转移优于从 j_1 转移。这意味着 $j_1 < j_2$ 后的点存在某个分界点 x，对于 $< x$ 的被决策点，j_1 比 j_2 更优，对于 $\geq x$ 的被决策点，j_2 比 j_1 更优。

上述结论能推导出决策单调性吗？

“逐渐变劣”暗示了四边形不等式能导出比决策单调性更强的性质：对于任意 $j_1 < j_2$，一旦对于某个被决策点 i，有从 j_2 转移优于从 j_1 转移，那么由于“逐渐变劣”，对于任意 $i' > i$，都有从 j_2 转移优于从 j_1 转移。这意味着 $j_1 < j_2$ 后的点存在某个分界点 x，对于 $< x$ 的被决策点，j_1 比 j_2 更优，对于 $\geq x$ 的被决策点，j_2 比 j_1 更优。

上述结论能推导出决策单调性吗？

满足决策单调性的转移方程一定满足上述结论吗？

Problem 1 「POI2011」Lightning Conductor

给定序列 $a_1, a_2, \ldots, a_n$，设 $p_i = \max_j a_j - a_i + \sqrt{|i-j|}$，求 $p_1, p_2, \ldots, p_n$。

$n \leq 5 \times 10^5$

Problem 1

绝对值的处理较为麻烦，考虑对 $j < i$ 和 $j > i$ 的部分分别求解，两者是类似的，就以 $j < i$ 为例。

令 $w(j, i) = a_j - a_i + \sqrt{|i - j|}$，此时转移方程变为 $p_i = \max_{j<i} 0 + w(j, i)$。容易看出，$w$ 满足“逐渐变劣”（由于本题为求解 max，不等号需要反向），因此直接推导出决策单调性。该问题中的转移为他转移，故采用分治策略即可求解所有 p_i，复杂度 $O(n \log n)$。

Problem 2 CF321E Ciel and Gondolas

你需要将 $1, 2, \ldots, n$ 这 n 个数划分为 k 个连续的区间。对于一个区间 $[l, r]$，其贡献为 $\sum_{i=l}^{r} \sum_{j=l}^{r} u_{i,j}$，其中 $u_{i,j}$ 是给定的。求最小总贡献。

$1 \le k \le n \le 3000, 0 \le u_{i,j} \le 9$

Problem 2

令 $w(j,i)=\sum_{k_1=j+1}^{i}\sum_{k_2=j+1}^{i}u_{k_1,k_2}$，并设 $f[k][i]$ 表示将前 i 个数划分为 k 段的最小总贡献，就有：

$$f[k][i]=\min_{j<i}f[k-1][j]+w(j,i)$$

容易发现 w 满足四边形不等式，从而转移具有决策单调性。此处的转移为他转移，直接使用分治对每个 k 求解，复杂度为 $O(n^2\log n)$。

虽然四边形不等式是决策单调性的充分不必要条件，但实际应用中决策单调性几乎总是被四边形不等式推导出来的。因此我们在提及“决策单调性”时，几乎总认为转移满足“逐渐变劣”。

“逐渐变劣”是否能被用于加速自转移呢？

对于 i，考虑它所有可能的决策点 $1, 2, \ldots, i-1$。

“逐渐变劣”的假设意味着对每个可能的决策点 j，它总是在某个分界点 x_j 以后劣于 $j+1$。从而，若 $x_{j-1} \geq x_j$，则在 j 变得比 $j-1$ 更优（x_{j-1}）之前，$j+1$ 就已经比 j 优（x_j）。于是 j 不会成为决策点，可以不予考虑。

不断如上述过程一样删除不可能的决策点，我们就得到了一系列可能的决策点 $j_1, j_2, \ldots, j_m$，且假设 j_k 在 x_k 时刻变得劣于 j_{k+1}，就总有 $x_1 < x_2 < \cdots < x_{m-1}$。

此外，若我们目前希望决策 i，且 j_1 劣于 j_2，根据“逐渐变劣”，j_1 也不会再成为决策点，因此还可以不断将 j_1 删除。容易发现，一旦 j_1 优于 j_2，那么 j_1 就是目前最优的，也就是决策点。

以上推导启发我们始终维护分界点递增且首项最优的决策点 $j_1, j_2, \ldots, j_m$。考虑我们现在已决策 $i-1$ 并维护完对于 $i-1$ 而言的 $j_1, j_2, \ldots, j_m$，而现在想决策 i。当我们在末尾插入一个新的可能的决策点 $i-1$，我们首先计算它和 j_m 的分界点 x_m。若 $x_{m-1} \geq x_m$，则 j_m 不会成为决策点，可以将其删除；不断重复这个过程，最后加入 $i-1$，就维护了分界点递增的决策点。

对于目前的被决策点 i，若 j_1 劣于 j_2，则显然以后都将劣于 j_2，于是直接删除 j_1。不断重复这个过程直到 j_1 优于 j_2，此时 j_1 根据之前的推导，即为 i 的决策点。

以上推导启发我们始终维护分界点递增且首项最优的决策点 $j_1, j_2, \dots, j_m$。考虑我们现在已决策 $i-1$ 并维护完对于 $i-1$ 而言的 $j_1, j_2, \dots, j_m$，而现在想决策 i。当我们在末尾插入一个新的可能的决策点 $i-1$，我们首先计算它和 j_m 的分界点 x_m。若 $x_{m-1} \geq x_m$，则 j_m 不会成为决策点，可以将其删除；不断重复这个过程，最后加入 $i-1$，就维护了分界点递增的决策点。

对于目前的被决策点 i，若 j_1 劣于 j_2，则显然以后都将劣于 j_2，于是直接删除 j_1。不断重复这个过程直到 j_1 优于 j_2，此时 j_1 根据之前的推导，即为 i 的决策点。

以上过程可以很方便的利用双端队列进行操作。每个决策点只会进队和出队一次，这部分的贡献是线性的。关键在于如何求解分界点 x，根据“逐渐变劣”，x 具有可二分性。使用二分求解 x, 用以上方法可以将自转移/他转移的过程加速到 $O(n \log n)$。

这一方法就叫做二分队列算法。

Problem 3 「NOI2009」诗人小 P

给定一个序列 $a_1, a_2, \ldots, a_n$，你需要将 $1, 2, \ldots, n$ 这 n 个数划分为若干连续的区间。对于一个区间 $[l, r]$，其贡献为 $|\sum_{i=l}^{r} a_i - L|^P$，其中 L, P 是常数。求最小总贡献。

$a_i \le 30, n \le 10^5, L \le 3 \times 10^6, P \le 5$。

Problem 3

设 S_i 是 a_i 的前缀和，则设 f_i 表示将 $1, 2, \ldots, i$ 划分的最小总贡献，易得转移方程：

$$f_i = \min_{j<i} f_j + |S_i - S_j - L|^P$$

令 $w(j, i) = |S_i - S_j - L|^P$，容易得到 w 满足“逐渐变劣”，因此直接推导出决策单调性。

本问题中的转移为自转移，于是使用二分队列算法，可以 $O(n \log n)$ 解决问题。

Problem 4 Ciel and Gondolas • 改

你需要将 $1, 2, \ldots, n$ 这 n 个数划分为 k 个连续的区间。对于一个区间 $[l, r]$，其贡献为 $\sum_{i=l}^{r}\sum_{j=l}^{r} u_{i,j}$，其中 $u_{i,j}$ 是给定的。求最小总贡献。

$1 \le k \le n \le 10^5, 0 \le u_{i,j} \le 9$

Problem 4

对于确定划分为 k 段的题目而言，n 的范围太大了。于是直接猜测 $f[k][n]$ 的取值关于 k 具有下凸性。

使用凸优化，对每一次划分加入额外的代价 w'，转移方程改写为：

$$f[i] = \min_{j<i} f[j] + w(j, i) + w'$$

仍然满足四边形不等式。但经过凸优化后，他转移变为自转移，需要改为使用二分队列而不是分治求解 f 的值。复杂度 $O(n \log n \log W)$。

此外为了避免凸包三点共线的情况干扰求解，需要保证转移时 k 最小，实现代码需要注意细节。

在一些转移方程中，$w(j,i)$ 可以被写作 $a(i)+b(j)+c(i)d(j)$ 的形式。这样的形式在一定程度上分离了 i 和 j 的贡献，从而便于进行处理。

首先来探究，在 a,b,c,d 满足什么条件下，w 能满足四边形不等式。

满足四边形不等式，等价于满足“逐渐变劣”，即对 $j_1 \le j_2 \le i_1 \le i_2$ 满足 $w(j_1,i_2)-w(j_1,i_1) \ge w(j_2,i_2)-w(j_2,i_1)$，代入得 $c(i_2)d(j_1)-c(i_1)d(j_1) \ge c(i_2)d(j_2)-c(i_1)d(j_2)$，整理得 $(c(i_2)-c(i_1))(d(i_2)-d(i_1)) \le 0$，也就是说，当 c 和 d 具有相反的单调性时，w 满足四边形不等式。

由于只需要知道 $c(i)d(j)$ 的值，我们总可以令 c 不降而 d 不增，减少分类讨论。即：当 c 不降、d 不增时，w 满足四边形不等式。

在以上条件下，对于某个被决策点 i，由于 $a(i)$ 部分相同，只需要比较 $b(j)+c(i)d(j)$ 的值，而这形如一条直线 $d(j)x+b(j)$ 在 $c(i)$ 处的点值。

利用直线求值的特殊性，对于决策点 j_1,j_2，我们可以避开二分搜索分界点，直接利用直线方程计算分界点，就将二分队列的复杂度从 $O(n\log n)$ 进一步优化到 $O(n)$。实际上，这就是维护凸包的斜率优化算法。也就是说，斜率优化是二分队列在特定情形下的进一步优化。

Problem 5 「HNOI2008」玩具装箱

给定一个序列 $a_1, a_2, \ldots, a_n$，你需要将 $1, 2, \ldots, n$ 这 n 个数划分为若干连续的区间。对于一个区间 $[l, r]$，其贡献为 $|\sum_{i=l}^{r} a_i - L|^2$，其中 L 是常数。求最小总贡献。

$n \leq 10^5, 0 \leq a_i, L \leq 10^4$。

Problem 5

仍然设 S_i 是 a_i 的前缀和，则设 f_i 表示将 $1,2,\ldots,i$ 划分的最小总贡献，易得转移方程：

$$f_i=\min_{j<i} f_j+|S_i-S_j-L|^2$$

将 $|S_i-S_j-L|^2$ 改写为 $(S_i-L)^2+(S_j)^2+(S_i-L)(-2S_j)$，就能满足 $a(i)+b(j)+c(i)d(j)$ 的形式，同时 S_i-L 不降，$-2S_j$ 不升，故使用斜率优化即可 $O(n)$ 解决问题。

Problem 6 玩具装箱・改

给定一个序列 $a_1, a_2, \ldots, a_n$，你需要将 $1, 2, \ldots, n$ 这 n 个数划分为若干连续的区间。对于一个区间 $[l, r]$，其贡献为 $|\sum_{i=l}^{r} a_i - L|^2$，其中 L 是常数。求最小总贡献。

$n \le 10^5, -10^4 \le a_i, L \le 10^4$。

在 $a(i)+b(j)+c(i)d(j)$ 并不满足当 c 不降、d 不增时，转移通常不具有决策单调性。但这类问题和斜率优化有类似之处，在此一并进行讲解。

处理这类问题的方法很多。若 d 仍满足不降性，通过单调队列维护凸包并在凸包上二分，即可加速转移到 $O(n\log n)$；若均不满足，还可以使用平衡树维护凸包……然而，前者适用范围仍然有限，而后者过于复杂，在考场上实现价值不大。因此本讲主要介绍一种适用于 c,d 不满足任何性质，且便于实现的 $O(n\log n)$ 李超树算法。

李超树能够支持两种操作：插入一条线段；查询某个点处所有已插入线段的最小值。主要使用线段树实现。

首先，如果根据线段树的划分只在线段完全包含的区间插入线段，那么线段就在该区间转化为直线。

李超树的核心方法是，在线段树的 $[l, r]$ 节点上维护一条“优势直线” L，这条直线满足其在 mid 处的点值是所有直线中最小的。“优势直线”能带来一个关键的性质：对于一条非优势直线 L'，其能小于 L 的区域或在 $[l, mid]$ 中，或在 $(mid, r]$ 中，或总是不小于 L（此时该直线可以直接删除），从而可以根据这两类情形将剩余线段分类到左右子树中。从而，如果希望关心 x 处的最小值，假设 $x \leq mid$，则被分类到 $(mid, r]$ 节点的所有直线在 x 处不会比优势直线更优，只要递归 $[l, mid]$ 的部分查询即可。于是查询已经可以做到 $O(\log n)$，关键在于如何解决插入。

插入

需要插入一条线段时，我们先将线段对应的 x 区间在线段树上进行拆分，这样，就将原问题转化为 $O(\log n)$ 个插入直线的问题。

现想要在 $[l, r]$ 区间上插入一条新的直线 L'。若 $[l, r]$ 节点还没有优势直线（即还没有直线），则直接将 L' 作为优势直线并返回。否则，设优势直线为 L，若 $L(mid) \le L'(mid)$，则根据之前的讨论，将 L' 插入到左子树或右子树中；若 $L(mid) > L'(mid)$，则交换 L 与 L'，对 L 进行同样的讨论即可。

插入一条线段需要拆分为 $O(\log n)$ 个插入直线的问题，每个问题是 $O(\log n)$ 的，因此总复杂度 $O(n \log^2 n)$。然而，如果只插入直线，则复杂度是 $O(n \log n)$ 的。

于是，在进行满足 $w(j,i)=a(i)+b(j)+c(i)d(j)$ 的前缀转移时，每添加一个决策点 j 都可以视为加入一条形如 $d(j)x+b(j)$ 的直线，而每次决策则可以视为查询 $c(i)$ 处所有已插入直线的最小值。直接使用李超树实现上述过程，复杂度就优化为 $O(n\log n)$。

Problem 6

同样，设 S_i 是 a_i 的前缀和，则设 f_i 表示将 $1, 2, \dots, i$ 划分的最小总贡献，易得转移方程：

$$f_i = \min_{j<i} f_j + |S_i - S_j - L|^2$$

将 $|S_i - S_j - L|^2$ 改写为 $(S_i - L)^2 + (S_j)^2 + (S_i - L)(-2S_j)$，就能满足 $a(i) + b(j) + c(i)d(j)$ 的形式，但由于 a_i 可能为负，c, d 没有单调性。使用李超树，即可 $O(n \log n)$ 解决问题。

Problem 7 Lightning Conductor • 改

给定序列 $a_1, a_2, \ldots, a_n$，设 $p_i = \min_j a_j - a_i + \sqrt{|i-j|}$，求 $p_1, p_2, \ldots, p_n$。

$n \le 5 \times 10^5$

Problem 7

依然可以分为 $j < i$ 和 $j > i$ 考虑，然而，“逐渐变劣”的性质不再满足了，根据根号函数的性质，现在转移似乎满足“逐渐变优”——即更前方的决策点每次增加的代价不大于更后方的决策点，从而随着被决策点的增大越来越处于优势。

“逐渐变优”能够导出决策单调性吗？即使可以，自转移又应该采用何种方式加速？

若对任意 $a \le b \le c \le d$，总有 $w(a,c) + w(b,d) \ge w(b,c) + w(a,d)$ 成立，则称 w 满足反四边形不等式。

改写后，可以得到反四边形不等式为一个等价形式：$w(a,d) - w(a,c) \le w(b,d) - w(b,c)$，这也等价于 $(g[a] + w(a,d)) - (g[a] + w(a,c)) \le (g[b] + w(b,d)) - (g[b] + w(b,c))$，也就是“逐渐变优”。

若对任意 $a \le b \le c \le d$，总有 $w(a, c) + w(b, d) \ge w(b, c) + w(a, d)$ 成立，则称 w 满足反四边形不等式。

改写后，可以得到反四边形不等式为一个等价形式：$w(a, d) - w(a, c) \le w(b, d) - w(b, c)$，这也等价于 $(g[a] + w(a, d)) - (g[a] + w(a, c)) \le (g[b] + w(b, d)) - (g[b] + w(b, c))$，也就是“逐渐变优”。

Theorem 2

若 w 满足反四边形不等式，则 f不具有决策单调性。

若对任意 $a \le b \le c \le d$，总有 $w(a, c) + w(b, d) \ge w(b, c) + w(a, d)$ 成立，则称 w 满足反四边形不等式。

改写后，可以得到反四边形不等式为一个等价形式：$w(a, d) - w(a, c) \le w(b, d) - w(b, c)$，这也等价于 $(g[a] + w(a, d)) - (g[a] + w(a, c)) \le (g[b] + w(b, d)) - (g[b] + w(b, c))$，也就是“逐渐变优”。

Theorem 2

若 w 满足反四边形不等式，则 f不具有决策单调性。

Proof.

在“逐渐变优”的条件下，某个决策点可能被后面的新决策点替代，也可能被前面的决策点在“逐渐变优”作用下替代，因此不具有决策单调性。 □

对于 i，考虑它所有可能的决策点 $1, 2, \ldots, i-1$。

“逐渐变优”的假设意味着对每个可能的决策点 j，它总是在某个分界点 x_j 以后优于 $j+1$。从而，若 $x_{j-1} \le x_j$，则在 j 变得比 $j+1$ 更优（x_j）之前，$j-1$ 就已经比 j 优（x_{j-1}）。于是 j 不会成为决策点，可以不予考虑。

不断如上述过程一样删除不可能的决策点，我们就得到了一系列可能的决策点 $j_1, j_2, \ldots, j_m$，且假设 j_k 在 x_k 时刻变得优于 j_{k+1}，就总有 $x_1 > x_2 > \cdots > x_{m-1}$。

此外，若我们目前希望决策 i，且 j_m 劣于 j_{m-1}，根据“逐渐变优”，j_m 也不会再成为决策点，因此还可以不断将 j_m 删除。容易发现，一旦 j_m 优于 j_{m-1}，那么 j_m 就是目前最优的，也就是决策点。

以上推导启发我们始终维护分界点递增且末项最优的决策点 $j_1, j_2, \ldots, j_m$。考虑我们现在已决策 $i-1$ 并维护完对于 $i-1$ 而言的 $j_1, j_2, \ldots, j_m$，而现在想决策 i。当我们在末尾插入一个新的可能的决策点 $i-1$，我们首先计算它和 j_m 的分界点 x_m。若 $x_{m-1} \leq x_m$，则 j_m 不会成为决策点，可以将其删除；不断重复这个过程，最后加入 $i-1$，就维护了分界点递增的决策点。

对于目前的被决策点 i，若 j_m 劣于 j_{m-1}，则显然以后都将劣于 j_{m-1}，于是直接删除 j_m。不断重复这个过程直到 j_m 优于 j_{m-1}，此时 j_m 根据之前的推导，即为 i 的决策点。

以上推导启发我们始终维护分界点递增且末项最优的决策点 $j_1, j_2, \ldots, j_m$。考虑我们现在已决策 $i-1$ 并维护完对于 $i-1$ 而言的 $j_1, j_2, \ldots, j_m$，而现在想决策 i。当我们在末尾插入一个新的可能的决策点 $i-1$，我们首先计算它和 j_m 的分界点 x_m。若 $x_{m-1} \le x_m$，则 j_m 不会成为决策点，可以将其删除；不断重复这个过程，最后加入 $i-1$，就维护了分界点递增的决策点。

对于目前的被决策点 i，若 j_m 劣于 j_{m-1}，则显然以后都将劣于 j_{m-1}，于是直接删除 j_m。不断重复这个过程直到 j_m 优于 j_{m-1}，此时 j_m 根据之前的推导，即为 i 的决策点。

以上过程可以很方便的利用栈进行操作。每个决策点只会进栈和出栈一次，这部分的贡献是线性的。关键在于如何求解分界点 x，根据“逐渐变优”，x 具有可二分性。使用二分求解 x, 用以上方法可以将自转移/他转移的过程加速到 $O(n \log n)$。

这一方法就叫做二分栈算法。

Problem 7

虽然该题的转移为他转移，但由于反四边形不等式条件下不满足决策单调性，故无法使用分治策略。利用二分栈算法，即可 $O(n\log n)$ 解决问题。

形如

$$f[i][j]=\min_{i\le k<j} f[i][k]+f[k+1][j]+w(i,j)$$

的方程被称为区间转移方程，其特点是对每个区间设计状态。一般区间转移只考虑自转移的情形，否则可以归纳到前缀转移。

对于 $\langle i,j\rangle$，若 $k\in[i,j)$ 满足对任意 $k'\in[i,j)$，有 $f[i][k]+f[k+1][j]\le f[i][k']+f[k'+1][j]$，则称 k 是 $\langle i,j\rangle$ 的决策点。为了方便，我们通常将 $\langle i,j\rangle$ 称为被决策点。

对于同一个 $\langle i,j\rangle$，可能存在多个决策点。

对于 $\langle i,j\rangle$，若 $k\in[i,j)$ 满足对任意 $k'\in[i,j)$，有 $f[i][k]+f[k+1][j]\le f[i][k']+f[k'+1][j]$，则称 k 是 $\langle i,j\rangle$ 的决策点。为了方便，我们通常将 $\langle i,j\rangle$ 称为被决策点。

对于同一个 $\langle i,j\rangle$，可能存在多个决策点。

对于区间转移，决策单调性的定义如下：

若 $i_1\le i_2\le i_3, j_1\le j_2\le j_3$，则对于**任意** $\langle i_1,j_1\rangle$ 的决策点 k_1，**任意** $\langle i_3,j_3\rangle$ 的决策点 k_3，存在 $k_1\le k_2\le k_3$ 使得 k_2 是 $\langle i_2,j_2\rangle$ 的决策点。

“四边形不等式”优化

对于区间转移，不同于前缀转移的分治策略，有一个简单的只依赖于决策单调性的算法可以将区间转移的复杂度从 $O(n^3)$ 加速到 $O(n^2)$，由于历史原因，这个方法被称作“四边形不等式”优化，但其本身实际上不依赖四边形不等式。

我们记 $pos[i][j]$ 代表 $\langle i,j\rangle$ 的某个决策点，由决策单调性，若 $j-i>2$，则有 $pos[i][j-1]\le pos[i][j]\le pos[i+1][j]$，注意到 $\langle i,j-1\rangle$ 和 $\langle i+1,j\rangle$ 均先于 $\langle i,j\rangle$ 被求解，因此只需在该范围内枚举 k 即可。考虑 $j-i$ 为定值的枚举范围的和为 $O(n)$，故总复杂度为 $O(n^2)$。

若对任意 $a \le b \le c \le d$，总有 $w(a,c) + w(b,d) \le w(b,c) + w(a,d)$ 成立，则称 w 满足四边形不等式。

若对任意 $a \le b \le c \le d$，总有 $w(b,c) \le w(a,d)$，则称 w 满足包含单调。

若对任意 $a \le b \le c \le d$，总有 $w(a,c) + w(b,d) \le w(b,c) + w(a,d)$ 成立，则称 w 满足四边形不等式。

若对任意 $a \le b \le c \le d$，总有 $w(b,c) \le w(a,d)$，则称 w 满足包含单调。

Theorem 3

若 w 满足四边形不等式和包含单调，则 f 具有决策单调性。

证明的一个重要部分是证明 f 也满足四边形不等式，较复杂，见手写部分。满足四边形不等式后，直接利用前缀转移的方法即可证明。

若对任意 $a \le b \le c \le d$，总有 $w(a,c) + w(b,d) \le w(b,c) + w(a,d)$ 成立，则称 w 满足四边形不等式。

若对任意 $a \le b \le c \le d$，总有 $w(b,c) \le w(a,d)$，则称 w 满足包含单调。

Theorem 3

若 w 满足四边形不等式和包含单调，则 f 具有决策单调性。

证明的一个重要部分是证明 f 也满足四边形不等式，较复杂，见手写部分。满足四边形不等式后，直接利用前缀转移的方法即可证明。同时有一点需要注意：若不是求解 min 而是求解 max，则包含单调的不等号也要反向。

Problem 8 Just a Data Structure Problem • 改

给定一棵 m 个点的带非负边权的树，以及一个序列 $a_1, a_2, \ldots, a_n (1 \le a_i \le m)$，现在希望对区间 $[1, n]$ 建立一棵线段树状结构，但分界点可以随意选取。线段树中的 $[l, r]$ 节点的贡献为 $\sum_{i=l}^{r} \sum_{j=l}^{r} dist(a_i, a_j)$，其中 $dist(x, y)$ 表示 x, y 两点在树上的距离。求最小贡献和。

$n, m \le 1500$

Poblem 8

“四边形不等式”优化模板题。考虑令 $w(l,r)=\sum_{i=l}^{r}\sum_{j=l}^{r}dist(a_i,a_j)$，可以用矩形前缀和 $O(n^2)$ 预处理，并设 $f[i][j]$ 表示对区间 $[i,j]$ 建立线段树的最小贡献和，就符合区间转移的形式。

容易验证 w 满足四边形不等式和包含单调，于是利用“四边形不等式”优化，即可 $O(n^2)$ 解决问题。

Problem 9 「NOI1995」石子合并

在一个圆形操场的四周摆放 N 堆石子，第 i 堆有 a_i 个，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。

$n \le 100, a_i \le 20$

Problem 9

为了破环，考虑总有一对相邻石子从不合并，于是可以将 a 重复两次。

令 $f[i][j]$ 表示合并第 i 堆到第 j 堆的最小/最大得分，$w(i,j)$ 为第 i 堆到第 j 堆的石子数，就符合区间转移的形式。

由于 n 很小，直接 $O(n^3)$ 即可解决，最后取 $f[1][n], f[2][n+1], \ldots, f[n][2n-1]$ 的最值，即可还原到环上的问题。

石子合并的转移是否满足决策单调？

由于此时对任意 $a \le b \le c \le d$，总有 $w(a,c) + w(b,d) = w(b,c) + w(a,d)$ 成立，故 w 既满足对于 min 的四边形不等式，也满足对于 max 的四边形不等式，直接利用决策单调优化？

石子合并的转移是否满足决策单调？

由于此时对任意 $a \le b \le c \le d$，总有 $w(a,c)+w(b,d)=w(b,c)+w(a,d)$ 成立，故 w 既满足对于 min 的四边形不等式，也满足对于 max 的四边形不等式，直接利用决策单调优化？

不要忽略包含单调：石子合并的 w 仅对 min 满足包含单调，故对最小得分有决策单调性，但最大得分则没有。

石子合并究竟是否能 $O(n^2)$ 解决呢？

经过打表验证，发现求最大得分时，$dp[i][j]$ 的决策点总是 i 或 $j-1$，从而复杂度也优化至 $O(n^2)$。

事实上，大量构造的随机数据表明，若 w 满足反四边形不等式和反包含单调，则上述性质总是成立的。

但是我暂时并没有证明出这一点，请谨慎使用该结论。
~~证明可以留作思考题。~~另外，也要注意 f 并不满足反四边形不等式。

Problem 10 「百度之星 2021 复赛」Just a Data Structure Problem

给定一棵 m 个点的带非负边权的树，以及一个序列 $a_1, a_2, \ldots, a_n (1 \le a_i \le m)$，现在希望对区间 $[1, n]$ 建立一棵线段树状结构，但分界点可以随意选取。线段树中的 $[l, r]$ 节点的贡献为 $\sum_{i=l}^{r} \sum_{j=l}^{r} dist(a_i, a_j)$，其中 $dist(x, y)$ 表示 x, y 两点在树上的距离。求最小贡献和。

此外，还会对 a 进行 r 次修改。具体来说，称原本的 a 为第 0 个版本，则第 i 个版本的 a 是在第 q_i 个版本的 a 末尾添加一个元素 $x(1 \le x \le m)$，求所有 $r+1$ 个版本的最小贡献和。

$n, m \le 1500, r \le 300$

Problem 10

首先建立版本的依赖树，遍历版本依赖树，修改就变成了末尾添加和撤销，不需要可持久化。

修改后当然满足决策单调性，仍可在 $pos[i][j-1]$ 到 $pos[i+1][j]$ 的范围内枚举 k。

然而，从修改的角度看，“四边形不等式”的优化是均摊的。可能出现个别 $pos[i][j-1]$ 到 $pos[i+1][j]$ 范围很大的情形。若在该情况下反复进行撤销-添加，则原本的复杂度分析就失效。

由于反四边形不等式不具有转递到 f 的性质，二分栈算法自然在区间转移上失效了。

但是四边形不等式传递到 f 的性质使得二分队列仍然有效。固定某个区间右端点 r，所有 $f[i][r]$ 的决策点随 i 递减单调不增，且由 f 满足四边形不等式，仍然有“逐渐变劣”的性质。

Problem 10

将前缀转移的二分队列算法应用到该题中即可加速添加操作。注意到添加相对较少，在初始状态使用“四边形不等式”优化可以进一步降低复杂度。

最终复杂度为 $O(m^2 + n^2 + nr\log n)$。

决策单调性优化多数情况下还是比较简单，关键在于掌握常用应用方法和适用范围。

此外，对于决策单调、四边形不等式等性质，如果证明不容易，可以先做猜想，使用暴力代码进行一定范围的验证。一旦得到验证也可以使用对应的算法。