斜率优化动态规划

李新年

2025 年 1 月 23 日

目录

直线和一次函数

斜率：表示一个直线倾斜程度。定义为和正方向水平轴的夹角的正切值。
经过两个点 (x_1, y_1) 和 (x_2, y_2) 的直线的斜率为 $\frac{y_2-y_1}{x_2-x_1}$。
可看出过两个竖直的点的直线斜率无定义。需特别关注。

直线和一次函数

斜率：表示一个直线倾斜程度。定义为和正方向水平轴的夹角的正切值。
经过两个点 (x_1, y_1) 和 (x_2, y_2) 的直线的斜率为 $\frac{y_2-y_1}{x_2-x_1}$。
可看出过两个竖直的点的直线斜率无定义。需特别关注。
非竖直的直线可用一次函数表示。
一次函数形如 $y = kx + b$，其中 k 表示直线的斜率，b 表示直线在纵轴的截距，即一定经过 $(0, b)$。
给定一个点 (x_0, y_0)，经过此点，且斜率为 k 的直线解析式为 $y = k(x - x_0) + y_0$
也可看作是 $\frac{y-y_0}{x-x_0} = k$ 这个限制的变形。

一本通打印文章

输入长度为 n 的非负序列，输入权值 M，把序列划分为若干连续段，每段的价值为这段权值和的平方加 M，求一种划分方式，使得每段价值之和最小。$n \leq 500000$。

一本通打印文章

设 dp_i 表示前 i 个数的最佳答案。考虑枚举倒数第二段的右端点 j，写出转移式为：

$$dp_i = \min_{j=0}^{i-1}\{dp_j + sum[j+1:i]^2 + M\}$$

一本通打印文章

设 dp_i 表示前 i 个数的最佳答案。考虑枚举倒数第二段的右端点 j，写出转移式为：

$$dp_i = \min_{j=0}^{i-1}\{dp_j + sum[j+1:i]^2 + M\}$$

尝试优化，令 s_i 表示前缀和，则转移式为

$$dp_i = M + \min_{j=0}^{i-1}\{dp_j + (s_i - s_j)^2\}$$

$$dp_i = M + s_i^2 + \min_{j=0}^{i-1}\{dp_j + s_j^2 - 2s_is_j\}$$

一本通打印文章

设 dp_i 表示前 i 个数的最佳答案。考虑枚举倒数第二段的右端点 j，写出转移式为：

$$dp_i=\min_{j=0}^{i-1}\{dp_j+sum[j+1:i]^2+M\}$$

尝试优化，令 s_i 表示前缀和，则转移式为

$$dp_i=M+\min_{j=0}^{i-1}\{dp_j+(s_i-s_j)^2\}$$

$$dp_i=M+s_i^2+\min_{j=0}^{i-1}\{dp_j+s_j^2-2s_is_j\}$$

有 $2s_is_j$ 让含 j 项和含 i 项无法分开，无法直接数据结构优化。

一本通打印文章

设 $res_j = M + s_i^2 + dp_j + s_j^2 - 2s_is_j$，表示从 j 转移到 i 的值。显然 $dp_i = \min_{j=0}^{i-1} res_j$。

一本通打印文章

设 $res_j = M + s_i^2 + dp_j + s_j^2 - 2s_is_j$，表示从 j 转移到 i 的值。显然 $dp_i = \min\limits_{j=0}^{i-1} res_j$。

$dp_j + s_j^2 = (2s_i) \times s_j + (res_j - M - s_i^2)$

一本通打印文章

设 $res_j = M + s_i^2 + dp_j + s_j^2 - 2s_is_j$，表示从 j 转移到 i 的值。显然 $dp_i = \min_{j=0}^{i-1} res_j$。

$dp_j + s_j^2 = (2s_i) \times s_j + (res_j - M - s_i^2)$

若把 $P_j(s_j, dp_j + s_j^2)$ 看作二维平面上的一个点?

则直线 $y = (2s_i)x + (res_j - M - s_i^2)$ 一定过 P_j。且此直线斜率固定为 $2s_i$，纵截距为 $res_j - M - s_i^2$。

对于固定的 i，求 res_j 中的最小值，转化为：给定若干个 P_j，求过每个点斜率为 $2s_i$ 的直线的截距最小值。

斜率优化

直观理解：只有所有的 P_j 里“最靠下”的点有可能成为答案。如何定义“最靠下”？

斜率优化

直观理解：只有所有的 P_j 里 “最靠下” 的点有可能成为答案。如何定义 “最靠下”？
我们定义 “不靠下” 的点：

- 点集中两个点 $A(x_A, y_A), B(x_B, y_B)$，若满足 $x_A = x_B$，且 $y_A < y_B$，则 B 是 “不靠下” 的点。
- 点集中三个点 $A(x_A, y_A), B(x_B, y_B), C(x_C, y_C)$，若满足 $x_A < x_B < x_C$，且 $k_{AB} > k_{BC}$，则 B 是 “不靠下” 的点。

称剩余不是 “不靠下” 的点形成的点集为：下凸壳。

斜率优化

只有下凸壳上的点可能成为最优转移点 res_j。

斜率优化

只有下凸壳上的点可能成为最优转移点 res_j。
证明：一个“不靠下”的点，第一种情况显然不可能。

斜率优化

只有下凸壳上的点可能成为最优转移点 res_j。
证明：一个“不靠下”的点，第一种情况显然不可能。
第二种情况，即这个电是 P_A, P_B, P_C，其中 $s_A < s_B < s_C$，且 $k_{AB} > k_{BC}$ 中的 P_B 点。直接放缩可证明 $k_{AB} > k_{AC} > k_{BC}$（三弦引理）。

斜率优化

只有下凸壳上的点可能成为最优转移点 res_j。
证明：一个“不靠下”的点，第一种情况显然不可能。
第二种情况，即这个电是 P_A, P_B, P_C，其中 $s_A < s_B < s_C$，且 $k_{AB} > k_{BC}$ 中的 P_B 点。直接放缩可证明 $k_{AB} > k_{AC} > k_{BC}$（三弦引理）。
先考虑两个转移点 P, Q。当 $2s_i < k_{PQ}$ 时，P 比 Q 优。当 $2s_i \geq k_{PQ}$ 时，Q 比 P 优。

斜率优化

只有下凸壳上的点可能成为最优转移点 res_j。
证明：一个“不靠下”的点，第一种情况显然不可能。
第二种情况，即这个电是 P_A, P_B, P_C，其中 $s_A < s_B < s_C$，且 $k_{AB} > k_{BC}$ 中的 P_B 点。直接放缩可证明 $k_{AB} > k_{AC} > k_{BC}$（三弦引理）。
先考虑两个转移点 P, Q。当 $2s_i < k_{PQ}$ 时，P 比 Q 优。当 $2s_i \geq k_{PQ}$ 时，Q 比 P 优。
当 $2s_i < k_{AC} < k_{AB}$ 时，A 比 B 优。
当 $2s_i \geq k_{AC} > k_{BC}$ 时，C 比 B 优。
所以无论何时，B 一定不是最优转移点。

斜率优化

实现方式：用单调队列维护下凸壳。每在右边（s_i 单调不降）加一个新点，按照斜率单调递增的原则维护单调队列。每次求 dp_i 时，在左侧弹掉斜率小于 $2s_i$ 的点，则队头为切点。时间复杂度为 $O(n)$。

斜率优化的一般形式

转移式为 $1D1D$ 形式，形如 $dp_i = \min\limits_{j}\{F(i) + G(j) + H(i) \times R(j)\}$：

斜率优化的一般形式

转移式为 $1D1D$ 形式，形如 $dp_i = \min_{j}\{F(i) + G(j) + H(i) \times R(j)\}$：

转化为 $G(j) = (-H(i)) \times R(j) + (res_i - F(i))$

斜率优化的一般形式

转移式为 $1D1D$ 形式，形如 $dp_i = \min_j\{F(i) + G(j) + H(i) \times R(j)\}$：
转化为 $G(j) = (-H(i)) \times R(j) + (res_j - F(i))$
再转化为，二维平面上有若干点 $P_j(R(j), G(j))$，用斜率为 $-H(i)$ 的直线去过这些点。最小（大）化纵截距，
维护下（上）凸壳解决。

斜率优化的一般形式

转移式为 $1D1D$ 形式，形如 $dp_i = \min_{j}\{F(i) + G(j) + H(i) \times R(j)\}$:
转化为 $G(j) = (-H(i)) \times R(j) + (res_j - F(i))$
再转化为，二维平面上有若干点 $P_j(R(j), G(j))$，用斜率为 $-H(i)$ 的直线去过这些点。最小（大）化纵截距，
维护下（上）凸壳解决。
上一道题满足了 R 函数是单调不降函数，这让我们可以直接用队列维护下凸壳。而且满足了 (-H) 函数是单调不降函数，这让我们可以用单调队列直接把队头弹掉。

一本通 Cats Transport

小 S 是农场主，他养了 M 只猫，雇了 P 位饲养员。农场中有一条笔直的路，路边有 N 座山，从 1 到 N 编号。第 i 座山与第 $i-1$ 座山之间的距离是 D_i。饲养员都住在 1 号山上。
有一天，猫出去玩。第 i 只猫去 H_i 号山玩，玩到时刻 T_i 停止，然后在原地等饲养员来接。饲养员们必须回收所有的猫。每个饲养员沿着路从 1 号山走到 N 号山，把各座山上已经在等待的猫全部接走。饲养员在路上行走需要时间，速度为 1 米每单位时间。饲养员在每座山上接猫的时间可以忽略，可以携带的猫的数量为无穷大。
例如有两座相距为 1 的山，一只猫在 2 号山玩，玩到时刻 3 开始等待。如果饲养员从 1 号山在时刻 2 或 3 出发，那么他可以接到猫，猫的等待时间为 0 或 1。而如果他于时刻 1 出发，那么他将于时刻 2 经过 2 号山，不能接到当时仍在玩的猫。
你的任务是规划每个饲养员从 1 号山出发的时间，使得所有猫等待时间的总和尽量小。饲养员出发的时间可以为负。
$N, M \le 10^5, P \le 100, D_i \le 10^4, 1 \le H_i \le N, T_i \le 10^9$

一本通 Cats Transport

设 sd_i 为 i 号山到 1 号山的距离。
某饲养员出发时间为 s，则他能接到的猫需要满足 $s + sd_{H_i} \geq T_i$，等待时间为 $s + sd_{H_i} - T_i$。

一本通 Cats Transport

设 sd_i 为 i 号山到 1 号山的距离。
某饲养员出发时间为 s，则他能接到的猫需要满足 $s + sd_{H_i} \geq T_i$，等待时间为 $s + sd_{H_i} - T_i$。
那么令 $a_i = T_i - sd_{H_i}$，问题转化为在数轴上放置 P 个存档点，最小化每个 a_i 和右侧最近的存档点距离之和。

一本通 Cats Transport

设 sd_i 为 i 号山到 1 号山的距离。
某饲养员出发时间为 s，则他能接到的猫需要满足 $s + sd_{H_i} \geq T_i$，等待时间为 $s + sd_{H_i} - T_i$。
那么令 $a_i = T_i - sd_{H_i}$，问题转化为在数轴上放置 P 个存档点，最小化每个 a_i 和右侧最近的存档点距离之和。
很明显所有饲养员只会在某个 a_i 时刻出发，则问题转化为离散问题。

一本通 Cats Transport

设 sd_i 为 i 号山到 1 号山的距离。
某饲养员出发时间为 s，则他能接到的猫需要满足 $s + sd_{H_i} \geq T_i$，等待时间为 $s + sd_{H_i} - T_i$。
那么令 $a_i = T_i - sd_{H_i}$，问题转化为在数轴上放置 P 个存档点，最小化每个 a_i 和右侧最近的存档点距离之和。
很明显所有饲养员只会在某个 a_i 时刻出发，则问题转化为离散问题。
把 a_i 从小到大排序。$dp_{i,j}$ 表示前 i 个猫，用 j 个饲养员接，等待时间之和最小值。

一本通 Cats Transport

转移为 $dp_{i,j}=\min\limits_{k=0}^{i-1} dp_{k,j-1}+a_i\times(i-k)-(s_i-s_k)$。$s_i$ 为 a_i 的前缀和。

一本通 Cats Transport

转移为 $dp_{i,j} = \min_{k=0}^{i-1} dp_{k,j-1} + a_i \times (i-k) - (s_i - s_k)$。$s_i$ 为 a_i 的前缀和。
第二维可滚动，套用斜率优化一般形式，化为

$$dp_{k,j-1} + s_k = a_i \times k + (res_k - a_i \times i + s_i)$$

一本通 Cats Transport

转移为 $dp_{i,j}=\min\limits_{k=0}^{i-1} dp_{k,j-1}+a_i\times(i-k)-(s_i-s_k)$。$s_i$ 为 a_i 的前缀和。
第二维可滚动，套用斜率优化一般形式，化为

$$dp_{k,j-1}+s_k=a_i\times k+(res_k-a_i\times i+s_i)$$

平面上的点为 $P_k(k,dp_{k,j-1}+s_k)$，斜率为 a_i
横坐标单调递增，斜率单调递增，可直接单调队列维护。时间复杂度 $O(NP)$。

一本通任务安排

有 N 个任务排成一个序列在一台机器上等待执行，它们的顺序不得改变。机器会把这 N 个任务分成若干批，每一批包含连续的若干个任务。从时刻 0 开始，任务被分批加工，执行第 i 个任务所需的时间是 T_i。另外，在每批任务开始前，机器需要 S 的启动时间，故执行一批任务所需的时间是启动时间 S 加上每个任务所需时间之和。一个任务执行后，将在机器中稍作等待，直至该批任务全部执行完毕。也就是说，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 Ci。请为机器规划一个分组方案，使得总费用最小。

$n \leq 300000, 1 \leq S \leq 256, |T_i| \leq 256, 0 \leq C_i \leq 256$

费用需要差分计算，将每个任务的费用拆到每组任务的花费时间上。

费用需要差分计算，将每个任务的费用拆到每组任务的花费时间上。
设 dp_i 表示前 i 个任务做完后，对总费用的贡献最小值。

费用需要差分计算，将每个任务的费用拆到每组任务的花费时间上。
设 dp_i 表示前 i 个任务做完后，对总费用的贡献最小值。
设 st 为 T 的前缀和，sc 为 c 的前缀和。枚举当前上一批任务的右端点 j，有

$$dp_i = \min_{j=0}^{i-1}\{dp_j + (sc_n - sc_j)(S + st_i - st_j)\}$$

费用需要差分计算，将每个任务的费用拆到每组任务的花费时间上。
设 dp_i 表示前 i 个任务做完后，对总费用的贡献最小值。
设 st 为 T 的前缀和，sc 为 c 的前缀和。枚举当前上一批任务的右端点 j，有
$dp_i = \min_{j=0}^{i-1}\{dp_j + (sc_n - sc_j)(S + st_i - st_j)\}$
按照惯例，整理为

$$dp_j - sc_n \times st_j - sc_j \times S + sc_j \times st_j = st_i \times sc_j + (res_j - sc_n \times S - sc_n \times st_i)$$

二维平面上的点为 $P_j(sc_j, dp_j - sc_n \times st_j - sc_j \times S + sc_j \times st_j)$，查询斜率为 st_i。

费用需要差分计算，将每个任务的费用拆到每组任务的花费时间上。
设 dp_i 表示前 i 个任务做完后，对总费用的贡献最小值。
设 st 为 T 的前缀和，sc 为 c 的前缀和。枚举当前上一批任务的右端点 j，有
$dp_i = \min_{j=0}^{i-1}\{dp_j + (sc_n - sc_j)(S + st_i - st_j)\}$
按照惯例，整理为

$$dp_j - sc_n \times st_j - sc_j \times S + sc_j \times st_j = st_i \times sc_j + (res_j - sc_n \times S - sc_n \times st_i)$$

二维平面上的点为 $P_j(sc_j, dp_j - sc_n \times st_j - sc_j \times S + sc_j \times st_j)$，查询斜率为 st_i。
横坐标依然单调递增，可直接维护下凸壳。但是查询斜率不递增了，不能用单调队列维护。

费用需要差分计算，将每个任务的费用拆到每组任务的花费时间上。
设 dp_i 表示前 i 个任务做完后，对总费用的贡献最小值。
设 st 为 T 的前缀和，sc 为 c 的前缀和。枚举当前上一批任务的右端点 j，有
$dp_i = \min_{j=0}^{i-1}\{dp_j + (sc_n - sc_j)(S + st_i - st_j)\}$
按照惯例，整理为

$$dp_j - sc_n \times st_j - sc_j \times S + sc_j \times st_j = st_i \times sc_j + (res_j - sc_n \times S - sc_n \times st_i)$$

二维平面上的点为 $P_j(sc_j, dp_j - sc_n \times st_j - sc_j \times S + sc_j \times st_j)$，查询斜率为 st_i。横坐标依然单调递增，可直接维护下凸壳。但是查询斜率不递增了，不能用单调队列维护。
用单调栈维护，查询时在单调栈上二分即可。时间复杂度 $O(n \log n)$。

NOI2007 货币兑换

有两种货物。每天这两种货物的市场价格都会改变，第 k 天，单价分别为 A_k 和 B_k。
有两种交易方式，一种是买，支付 P 元，获得数量比例为 r_k 的 A 货物和 B 货物。
另一种是卖，将当前手里占比为 p 的 A 货物和 B 货物卖成现金。$p \in [0, 1]$。
每天可以自由买卖，初始没有货物，有 S 元钱，求 N 天后最多多少现金。
$N \le 10^5, 0 < A_k, B_k \le 10, 0 < r_k \le 100$

NOI2007 货币兑换

必然存在一种最优的方案满足：每次买都花光所有现金，每次卖都卖光所有货物。

NOI2007 货币兑换

必然存在一种最优的方案满足：每次买都花光所有现金，每次卖都卖光所有货物。那么在同一天内，一定是先卖光（或者不卖），再全买进（或者不买）。

NOI2007 货币兑换

必然存在一种最优的方案满足：每次买都花光所有现金，每次卖都卖光所有货物。
那么在同一天内，一定是先卖光（或者不卖），再全买进（或者不买）。
设 dp_i 表示第 i 天卖光 (或者不卖) 后最多有多少钱。枚举上次是第 j 天买的，得

$$dp_i = \max(dp_{i-1}, \max_{j=0}^{i-1}\{dp_j \times (a_j \times A_i + b_j \times B_i)\})$$

其中 $a_j = \frac{r_j}{A_j \times r_j + B_j}, b_j = \frac{1}{A_j \times r_j + B_j}$，表示当天 1 元钱会买进 A 货物和 B 货物的数量。

NOI2007 货币兑换

必然存在一种最优的方案满足：每次买都花光所有现金，每次卖都卖光所有货物。
那么在同一天内，一定是先卖光（或者不卖），再全买进（或者不买）。
设 dp_i 表示第 i 天卖光 (或者不卖) 后最多有多少钱。枚举上次是第 j 天买的，得
$dp_i = \max(dp_{i-1}, \max_{j=0}^{i-1}\{dp_j \times (a_j \times A_i + b_j \times B_i)\})$
其中 $a_j = \frac{r_j}{A_j \times r_j + B_j}, b_j = \frac{1}{A_j \times r_j + B_j}$，表示当天 1 元钱会买进 A 货物和 B 货物的数量。
我们姑且忽略和 dp_{i-1} 取 max 的那部分，按照惯例定义
$res_j = dp_j \times (a_j \times A_i + b_j \times B_i)$，推式子可得

$$dp_j \cdot b_j = -\frac{A_i}{B_i} \times (dp_j \cdot a_j) + \frac{res_j}{B_i}$$

NOI2007 货币兑换

必然存在一种最优的方案满足：每次买都花光所有现金，每次卖都卖光所有货物。

那么在同一天内，一定是先卖光（或者不卖），再全买进（或者不买）。

设 dp_i 表示第 i 天卖光 (或者不卖) 后最多有多少钱。枚举上次是第 j 天买的，得

$dp_i = \max(dp_{i-1}, \max_{j=0}^{i-1}\{dp_j \times (a_j \times A_i + b_j \times B_i)\})$

其中 $a_j = \frac{r_j}{A_j \times r_j + B_j}, b_j = \frac{1}{A_j \times r_j + B_j}$，表示当天 1 元钱会买进 A 货物和 B 货物的数量。

我们姑且忽略和 dp_{i-1} 取 max 的那部分，按照惯例定义

$res_j = dp_j \times (a_j \times A_i + b_j \times B_i)$，推式子可得

$$dp_j \cdot b_j = -\frac{A_i}{B_i} \times (dp_j \cdot a_j) + \frac{res_j}{B_i}$$

此时二维平面上的点为 $P_j(dp_j \cdot a_j, dp_j \cdot b_j)$，用斜率为 $-\frac{A_i}{B_i}$ 的直线去过这些点，最大化纵截距。

NOI2007 货币兑换

必然存在一种最优的方案满足：每次买都花光所有现金，每次卖都卖光所有货物。
那么在同一天内，一定是先卖光（或者不卖），再全买进（或者不买）。
设 dp_i 表示第 i 天卖光 (或者不卖) 后最多有多少钱。枚举上次是第 j 天买的，得
$dp_i = \max(dp_{i-1}, \max_{j=0}^{i-1}\{dp_j \times (a_j \times A_i + b_j \times B_i)\})$
其中 $a_j = \frac{r_j}{A_j \times r_j + B_j}, b_j = \frac{1}{A_j \times r_j + B_j}$，表示当天 1 元钱会买进 A 货物和 B 货物的数量。
我们姑且忽略和 dp_{i-1} 取 max 的那部分，按照惯例定义
$res_j = dp_j \times (a_j \times A_i + b_j \times B_i)$，推式子可得

$$dp_j \cdot b_j = -\frac{A_i}{B_i} \times (dp_j \cdot a_j) + \frac{res_j}{B_i}$$

此时二维平面上的点为 $P_j(dp_j \cdot a_j, dp_j \cdot b_j)$，用斜率为 $-\frac{A_i}{B_i}$ 的直线去过这些点，最大化纵截距。
无论是点的横坐标还是斜率都不单调了。可以用李超线段树来做，但斜率优化也可以做。

一般形式其他优化

回顾形式 $G(j) = (-H(i)) \times R(j) + (res_j - F(i))$。用斜率为 $-H(i)$ 的直线过 $P_j(R(j), G(j)))$
现在我们考虑，如果横坐标 $R(j)$ 不单调递增，怎么维护凸壳呢?

一般形式其他优化

回顾形式 $G(j) = (-H(i)) \times R(j) + (res_j - F(i))$。用斜率为 $-H(i)$ 的直线过 $P_j(R(j), G(j)))$

现在我们考虑，如果横坐标 $R(j)$ 不单调递增，怎么维护凸壳呢?

序列上的斜率优化可考虑分治。

一般形式其他优化

回顾形式 $G(j) = (-H(i)) \times R(j) + (res_j - F(i))$。用斜率为 $-H(i)$ 的直线过 $P_j(R(j), G(j)))$

现在我们考虑，如果横坐标 $R(j)$ 不单调递增，怎么维护凸壳呢?

序列上的斜率优化可考虑分治。

对于一个分治区间 $[l, mid]$ 和 $[mid+1, r]$，我们考虑转移点在 $[l, mid]$ 范围内，对 $i \in [mid+1, r]$ 的所有 dp_i 的贡献。这样转移点内部就离线下来了，可以做排序，变成了横坐标单调的情况。对于每个 $i \in [mid+1, r]$，在固定的凸壳上做二分。

一般形式其他优化

回顾形式 $G(j) = (-H(i)) \times R(j) + (res_j - F(i))$。用斜率为 $-H(i)$ 的直线过 $P_j(R(j), G(j)))$

现在我们考虑，如果横坐标 $R(j)$ 不单调递增，怎么维护凸壳呢?

序列上的斜率优化可考虑分治。

对于一个分治区间 $[l, mid]$ 和 $[mid + 1, r]$，我们考虑转移点在 $[l, mid]$ 范围内，对 $i \in [mid + 1, r]$ 的所有 dp_i 的贡献。这样转移点内部就离线下来了，可以做排序，变成了横坐标单调的情况。对于每个 $i \in [mid + 1, r]$，在固定的凸壳上做二分。

先递归到左子区间，把 $[l, mid]$ 的所有 dp 最终值求出来。然后考虑 $[l, mid]$ 对 $[mid + 1, r]$ 的转移。最后递归到 $[mid + 1, r]$，只需考虑内部转移。

一般形式其他优化

回顾形式 $G(j) = (-H(i)) \times R(j) + (res_j - F(i))$。用斜率为 $-H(i)$ 的直线过 $P_j(R(j), G(j)))$
现在我们考虑，如果横坐标 $R(j)$ 不单调递增，怎么维护凸壳呢?
序列上的斜率优化可考虑分治。
对于一个分治区间 $[l, mid]$ 和 $[mid+1, r]$，我们考虑转移点在 $[l, mid]$ 范围内，对 $i \in [mid+1, r]$ 的所有 dp_i 的贡献。这样转移点内部就离线下来了，可以做排序，变成了横坐标单调的情况。对于每个 $i \in [mid+1, r]$，在固定的凸壳上做二分。
先递归到左子区间，把 $[l, mid]$ 的所有 dp 最终值求出来。然后考虑 $[l, mid]$ 对 $[mid+1, r]$ 的转移。最后递归到 $[mid+1, r]$，只需考虑内部转移。
相当于一个分治树上的中序遍历。

NOI2007 货币兑换

这个题就利用分治加斜率优化去做。直接做时间复杂度为 $O(n\log^2 n)$。
但是分治本身结构就是归并排序的结构。利用归并排序，可将时间复杂度降为 $O(n\log n)$。

NOI2019 回家路线

一个铁路系统可看作一个图，我在 0 时刻从 1 号点出发，要到 n 号点。一共有 m 条边，第 i 条边从 p_i 时刻从 x_i 出发，在 q_i 时刻到达 y_i。
只能在某个点换乘，需要满足上一辆车的 $q_j \le p_i$。
对于一次换乘，我在某个点如果等待了 t 个时刻，我的烦躁值会增加 $At^2 + Bt + C$。
最终我在 q_k 时刻到了 n 号点，我的烦躁值再加 q_k。
求最小的烦躁值。
$n \le 10^5, m \le 2 \times 10^5$

NOI2019 回家路线

虽然自然的想法是以到达了哪个点为状态。但是这里边的信息定的特别死板，把点的信息，下一个点的信息，上下车时刻都已经定死了，可以考虑以边为状态。

NOI2019 回家路线

虽然自然的想法是以到达了哪个点为状态。但是这里边的信息定的特别死板，把点的信息，下一个点的信息，上下车时刻都已经定死了，可以考虑以边为状态。
设 f_i 表示上了第 i 条边之后最小的烦躁值之和。转移是枚举前一条经过的边是什么。
具体转移为

$$f_i = \min_{q_j \le p_i, y_j = x_i} f_j + A(p_i - q_j)^2 + B(p_i - q_j) + C$$

NOI2019 回家路线

虽然自然的想法是以到达了哪个点为状态。但是这里边的信息定的特别死板，把点的信息，下一个点的信息，上下车时刻都已经定死了，可以考虑以边为状态。
设 f_i 表示上了第 i 条边之后最小的烦躁值之和。转移是枚举前一条经过的边是什么。
具体转移为

$$f_i = \min_{q_j \le p_i, y_j = x_i} f_j + A(p_i - q_j)^2 + B(p_i - q_j) + C$$

求 f_i 时把边按照 p_i 排序，转移点按照 q_i 排序，这样的话转移点是一段前缀，每次 p_i 增大增加一部分转移点即可。
而且转移点可以挂到 y_i 上，求 f_i 时从 x_i 上挂着的转移点去最优化。
转移式是经典斜率优化，推式子即可。时间复杂度 $O(n+m)$。

NOI2014 购票

给一棵 n 个点的有根有边权树，根节点为 1 号。i 号点父节点为 f_i，和父亲的边权为 s_i。
对于一个点 v，想要走到根，只能每步选择一个祖先，支付费用并走到这个祖先。
需要满足从 v 到这个祖先的路径长度不超过 l_v。设路径长度为 d，则费用为 $p_v \times d + q_v$。
对每个节点求最小路费。
$n \leq 2 \times 10^5$

NOI2014 购票

设 f_i 为从 i 出发最小路费。转移显然

$$f_i = \min_{j \in anc(i), dis_i - dis_j \le l_i} f_j + (dis_i - dis_j) \times p_i + q_i$$

NOI2014 购票

设 f_i 为从 i 出发最小路费。转移显然

$$f_i = \min_{j \in anc(i), dis_i - dis_j \le l_i} f_j + (dis_i - dis_j) \times p_i + q_i$$

一眼斜率优化，$f_j = p_i \times dis_j + f_i - dis_i p_i - q_i$。二维点坐标是 (dis_j, f_j)，斜率为 p_i，最小化纵截距，维护下凸壳。但是转移顺序迷惑。

NOI2014 购票

设 f_i 为从 i 出发最小路费。转移显然

$$f_i = \min_{j \in anc(i), dis_i - dis_j \le l_i} f_j + (dis_i - dis_j) \times p_i + q_i$$

一眼斜率优化，$f_j = p_i \times dis_j + f_i - dis_i p_i - q_i$。二维点坐标是 (dis_j, f_j)，斜率为 p_i，最小化纵截距，维护下凸壳。但是转移顺序迷惑。

无论如何一定是求出来靠近根的 dp 值之后再求靠近叶子的 dp 值。dfs 的话不方便动态维护凸包。

NOI2014 购票

设 f_i 为从 i 出发最小路费。转移显然

$$f_i = \min_{j \in anc(i), dis_i - dis_j \le l_i} f_j + (dis_i - dis_j) \times p_i + q_i$$

一眼斜率优化，$f_j = p_i \times dis_j + f_i - dis_i p_i - q_i$。二维点坐标是 (dis_j, f_j)，斜率为 p_i，最小化纵截距，维护下凸壳。但是转移顺序迷惑。

无论如何一定是求出来靠近根的 dp 值之后再求靠近叶子的 dp 值。dfs 的话不方便动态维护凸包。

那就离线，cdq 分治启动！

树上就点分治。

NOI2014 购票

对于某一个点分治的连通块，由于是有根树，从连通块重心处分开，有一个子连通块是重心父节点那边，其余子连通块都是子树内。

NOI2014 购票

对于某一个点分治的连通块，由于是有根树，从连通块重心处分开，有一个子连通块是重心父节点那边，其余子连通块都是子树内。
先递归到父节点那边的连通块进行点分治，求出所有这个连通块里的 f 值。然后考虑这部分对于分治重心和子连通块的贡献。这时重心的 f 值已经是正确的了，在考虑重心对子连通块的贡献。这之后只需要考虑每个子连通块内部的贡献即可，递归进去 cdq 分治。

NOI2014 购票

对于某一个点分治的连通块，由于是有根树，从连通块重心处分开，有一个子连通块是重心父节点那边，其余子连通块都是子树内。
先递归到父节点那边的连通块进行点分治，求出所有这个连通块里的 f 值。然后考虑这部分对于分治重心和子连通块的贡献。这时重心的 f 值已经是正确的了，在考虑重心对子连通块的贡献。这之后只需要考虑每个子连通块内部的贡献即可，递归进去 cdq 分治。
分治的优势就是在确定转移点和被贡献的点之后，就完全无需考虑两边内部关系了，直接打乱就好。
把父连通块和子连通块各自都按照 dis_i 排序，依次用双指针做斜率优化 dp 即可。
由于 p_i 无单调性，在凸壳上二分。时间复杂度 $O(n\log^2 n)$。

感谢聆听!