

最小矩形覆盖 — 算法对话记录

洛谷 P3187 [HNOI2007] XXII 赛前模板训练赛

2026-05-05

背景

题目链接：[洛谷 P3187 \[HNOI2007\] 最小矩形覆盖](#)。

题面要点：给定平面上 n 个点，求一个 **覆盖所有点的最小面积矩形**（矩形可以任意旋转，不求轴对齐），输出面积和按逆时针的四个顶点。约束 $3 \leq n \leq 5 \times 10^4$ ，坐标 $\in [0, 10]$ ，最小矩形面积至少 0.1，误差容忍 10^{-4} 。

下面按对话方式逐步推进。**蓝色** 标注首次出现的关键术语 / 符号。

第 1 步：理解题目

Claude

先用样例热身。 $n = 6$ 个点

$(1, 3), (1, 4), (2, 1), (3, 0), (3, 6), (6, 3),$

标程给的最小矩形顶点是 $(3, 0), (6, 3), (3, 6), (0, 3)$ ，面积 18。

画一下图（或在脑里走一遍）：这 4 个矩形顶点和原始 6 个点放在一起。**原始 6 个点里，有几个落在矩形边界上？分别在哪条边上？**（不在边上的就是严格落在矩形内部）这一步只是想让你看清楚“最小矩形”长什么样——为后面想算法做铺垫。

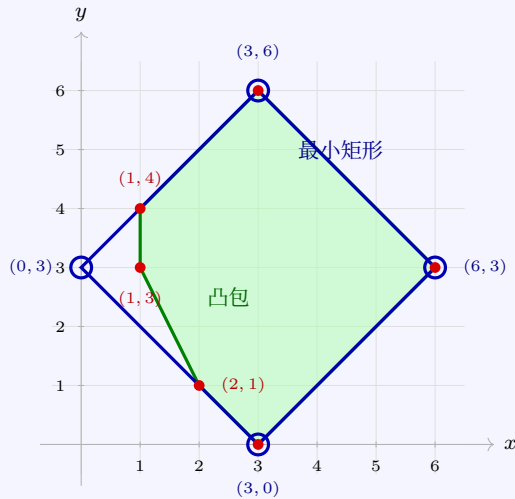
zzy

这个样例数据范围比较小，不过 $O(n)$ 的算法应该可以做到。然后这个理解题目不是很重要，毕竟这个题目是一道模板题，描述非常简单。再说了，我也没法告诉你，我也不想画这个图，你自己把这个图画出来，用 LaTeX 画出来。

Claude

好——我把图补上，复杂度修正一下，然后跳到下一步。

样例可视化：绿色填充多边形 = 输入 6 点的凸包（CCW 顺序 $(3, 0) \rightarrow (6, 3) \rightarrow (3, 6) \rightarrow (1, 4) \rightarrow (1, 3) \rightarrow (2, 1)$ ，6 边形）；蓝色 outline = 标程给的最小覆盖矩形；蓝环 = 矩形 4 顶点；红实心点 = 输入 6 点。



对照：6 输入点全部在凸包上（凸包就是这 6 个点）。其中 3 个 $(3,0)$, $(3,6)$, $(6,3)$ 同时也是矩形顶点； $(1,4)$ 落在矩形顶边 $(3,6)-(0,3)$ 上； $(2,1)$ 落在左边 $(0,3)-(3,0)$ 上； $(1,3)$ 是凸包顶点但严格在矩形内部。**凸包和矩形的偏差区** = 矩形里没被凸包覆盖到的两块小三角，分别贴在 $(0,3)$ 那个“空角”两侧（绿色和蓝色 outline 不重合的部分肉眼可见）。

复杂度修正： $n \leq 5 \times 10^4$ ，本题 $O(n)$ 严格地说做不到——求凸包要 sort 是 $O(n \log n)$ 。凸包给定后的旋转卡壳那部分确实是 $O(h)$ 。所以全流程 $O(n \log n)$ 。这里 h 表示凸包顶点数。

第 2 步：核心引理 + 暴力骨架

Claude

跳到算法骨架。模板的根本依赖是把矩形 **方向**（连续的 $[0, \pi/2)$ 角度）离散化到有限多个候选——这一步靠的是一条几何引理。

我先把它抛出来给你确认：

核心引理：覆盖凸包的最小面积矩形，至少有一条边与凸包的某条边共线。

先问一个问题：接受这条引理后，你能直接说出一个暴力算法吗——枚举什么、对每个枚举对象做什么、整体复杂度多少？

zzy

不是啊，哪有你这么讲题，让我先接受一个引理啊，这个引理怎么来的？我接受了吗？你代替我接受了是吧？我们也不用严格的证明对吧？但是你不能直接跳过这个引理是怎么来的吧，你这也太搞笑了吧。

zzy

不是啊，你就算这个证明，也需要引导我一次性写出太多东西也是禁止行为。任何回答都必须有引导性，不能一次性吐出太多内容。一次性吐出太多内容是绝对禁止的，懂吗。

Claude

懂了，回单步引导。

先做一个具体观察，不抛任何理论。在你刚才那张样例图上数：

最小矩形的 **4 条边**，每条边上各自经过几个凸包顶点？（提醒：这个样例的凸包顶点 = 全部 6 个输入点——之前在图下方写过——所以你只需要数 6 个红点分别落在哪条矩形边/角上。）

zzy

我们也都知道，我们需要解决的问题是什么，就是我们想要证明的就是至少有一条边是和凸包的边是平行的，或者说其实是重合的吧，其实是重合的。就是我们的这个最小覆盖矩形...

Claude

对，重合 (collinear / 同一条直线上) 比“平行”强——平行只锁方向，重合还锁位置。这就是引理要证的精确陈述。

回到我刚问那个图的问题，因为它直通证明：在图上数，最小矩形的 4 条边，每条边各自经过几个凸包顶点？

zzy

每条边至少都经过了至少两个红点对，每条边都经过了两个红点。

Claude

对，每条边恰好 2 个凸包顶点——这是个很硬的结构信号。

下一个微观察：刚才数出来的“矩形某条边上的 2 个凸包顶点”，比如最小矩形底边上的 $(3, 0)$ 和 $(6, 3)$ ——它们在凸包上是相邻的吗？也就是说，凸包的顶点序列里，它们俩之间是不是直接由一条凸包边连接、中间没有插入别的凸包顶点？

zzy

对，当然是相邻的，当然是相邻的。

Claude

好。底边 $(3, 0)-(6, 3)$ 上的 2 个凸包顶点相邻 \Rightarrow 底边完整包含凸包边 $(3, 0) \rightarrow (6, 3) \Rightarrow$ 底边和该凸包边重合 (共线)。引理在这个样例里被超额满足——4 条边全部重合于某条凸包边。

切换到一般情形思考 (这个样例太特殊)。把凸包想成一个一般、不对称的凸多边形，外接最小覆盖矩形。

反证 setup 一步：假设最小覆盖矩形没有任何一条边和凸包某条边重合。那么矩形每条边和凸包的接触应该是几个点？

zzy

如果假设最小覆盖矩形没有任何一条边和凸包某条边重合，那么矩形每条边和凸包的接触点就只有一个点嘛，每条边和凸包的接触点最多只有一个点。

Claude

对——准确说是恰好 1 个点：

- “ ≤ 1 ” 是反证假设直接给的 (不重合 \Rightarrow 不会经过 2 个相邻凸包顶点；非相邻 2 个顶点更不可能在同一条支撑切线上)。
- “ ≥ 1 ” 是因为矩形必须紧贴凸包：如果某条边一个凸包顶点都不接触，那条边就能往凸包内推一点，矩形面积变小——它就不是最小的了，矛盾。

所以 4 条矩形边各自接触凸包恰好 1 个顶点，命名为 **支撑顶点** L, R, T, B (左/右/上/下边对应)。

下一步问题：在 L, R, T, B 这 4 个支撑顶点固定不变的前提下，矩形“覆盖凸包、 L, R, T, B 各

自落在对应矩形边上”——还剩几个自由度？也就是说，可以怎样扰动这个矩形让它仍然满足这两个条件？

zzy

LRTB 各自落在对应矩形边上，此时还剩几个自由度。嗯，什么叫几个自由度？不太懂你意思，这几个自由度是什么意思啊？不太懂。

Claude

好，“自由度”说人话——

自由度 = 保持约束（这里 = “ L, R, T, B 各自落在对应矩形边上”）的前提下，矩形还能怎么动。能独立动几个方向就是几个自由度。

举例：

- 一个无约束的矩形在平面上有 5 个自由度——中心 2D (2) + 旋转角 (1) + 宽 (1) + 高 (1)。
- 加约束“中心必须在原点” \Rightarrow 砍掉 2 个 \Rightarrow 剩 3 个自由度。每加 1 个独立约束就砍 1 个自由度。

重新具体问：4 支撑顶点 L, R, T, B 固定。如果我把矩形逆时针旋转一个小角度 ϵ ，同时重新调整它的位置/宽/高（不局限于原参数），让 L, R, T, B 仍然分别落在新矩形对应的 4 条边上——这样的新矩形**存在**吗？（提示：每条新矩形边应该是过对应支撑点、方向沿着新旋转角的那条切线。）

zzy

应该是存在的，就是旋转一个小角度，肯定是可以的。

Claude

对——所以 4 支撑固定时，矩形只剩 **1 个自由度**：旋转角 θ 。每个 θ 唯一确定一个矩形（4 条边各自就是过 L, R, T, B 沿对应方向的切线），矩形的宽、高、面积都成了 θ 的（光滑）函数： $\text{width}(\theta)$ 、 $\text{height}(\theta)$ 、 $A(\theta)$ 。当前矩形对应 $\theta = \theta_0$ 。

关键问题：反证假设“当前矩形是最小的” $\Rightarrow A(\theta)$ 在 θ_0 处取局部极小。这给一阶导数 $A'(\theta_0)$ 一个什么**必要条件**？

zzy

那这个一阶导数 $A'(\theta_0)$ 理论上应该等于 0 对吧，因为这个极值点嘛，极值点它肯定这个导数值为 0 嘛。

Claude

对， $A'(\theta_0) = 0$ 是必要条件。但单凭这个还推不出矛盾—— $A' = 0$ 既可以是极小、也可以是极大或鞍点。要把矛盾构造出来，得继续做二阶或结构性分析。

为了往下走，先把 $A(\theta)$ 显式写出来。

记号：

- $\mathbf{u} = R - L$ (“左右”支撑顶点之差，固定向量)，
- $\mathbf{v} = T - B$ (“上下”支撑顶点之差，固定向量)，
- $\hat{n}_\theta = (\cos \theta, \sin \theta)$ (矩形“水平”单位方向)，

- $\hat{n}_\theta^\perp = (-\sin \theta, \cos \theta)$ (“垂直”方向，与 \hat{n}_θ 正交)。

下一步问题：在这个记号下，矩形的宽 $\text{width}(\theta)$ 等于什么内积？高 $\text{height}(\theta)$ 等于什么内积？（提示：宽就是把所有凸包顶点投影到 \hat{n}_θ 方向后的“最右”减“最左”——但因为左右支撑就是 L, R ，“最右”投影 = $R \cdot \hat{n}_\theta$ 、“最左” = $L \cdot \hat{n}_\theta$ ）

zzy

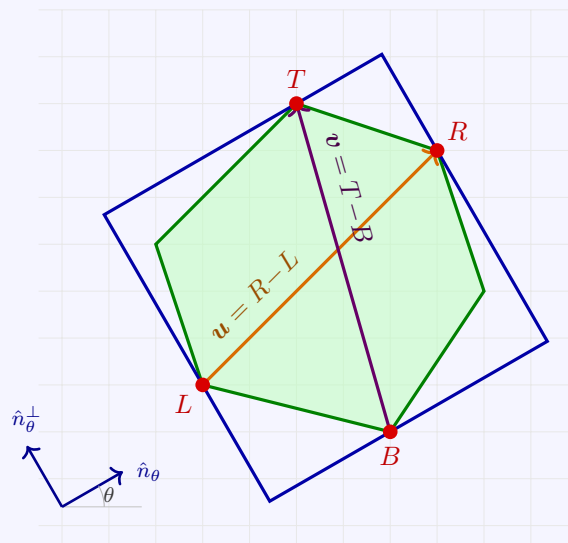
建议你还是把图画出来比较好。你画一个图，我能更清楚地看到这些量。注意画图要检查啊，要检查。然后还有推荐绘图方式好吧。

Claude

好，画一张一般情形的示意图。

图设定：

- 凸包：6 顶点不对称凸六边形，CCW $H_1 = (3, 3), H_2 = (7, 2), H_3 = (9, 5), H_4 = (8, 8), H_5 = (5, 9), H_6 = (2, 6)$ 。
- 取 $\theta = 30^\circ$ ：算出 $L = H_1, R = H_4, T = H_5, B = H_2$ 。选 30° 而非更小的角度是为了把旋转视觉上拉够大——任何 $\theta \in (18.43^\circ, 71.57^\circ)$ 都属于同一段平滑段（4 支撑不变），但 θ 接近段端点时矩形看起来会和凸包某条边伪 flush，误导读者以为“它们已经重合”； 30° 离两端均够远。
- 矩形：通过 4 条支撑切线方程算出 4 顶点。



图注：以 $\theta = 30^\circ$ 、不对称凸六边形为例。绿色多边形 = 凸包；蓝色矩形 = 旋转 θ 后的覆盖矩形；红点 $L, R, T, B = 4$ 支撑顶点（每条矩形边只接触凸包一个顶点，没有任何一条边与凸包的某条边重合——这正是反证 setup 假设的状态）； $u = R - L$ （橙色箭头）和 $v = T - B$ （紫色箭头）是反证 setup 里两个固定向量；左下角“指南针” $\hat{n}_\theta = (\cos \theta, \sin \theta)$ 、 $\hat{n}_\theta^\perp = (-\sin \theta, \cos \theta)$ 。

下一步问题：在这张图上， $\text{width}(\theta)$ 等于哪个内积？ $\text{height}(\theta)$ 等于哪个内积？（即把宽和高分别写成两个向量的点积形式。）

zzy

我在想一共就两个向量，而且点积是符合交换率的吧，一共就两个向量点积、而且符合交换率的话，这哪能凑得出来两个内积啊？而且我也有一点忘记这个内积的定义了。

Claude

点积: $\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 = |\mathbf{a}| |\mathbf{b}| \cos \angle(\mathbf{a}, \mathbf{b})$ 。

向量数其实 4 个, 不是 2 个: 位移 $\mathbf{u} = R - L$ 、 $\mathbf{v} = T - B$, 加上单位方向 $\hat{n}_\theta = (\cos \theta, \sin \theta)$ 、 $\hat{n}_\theta^\perp = (-\sin \theta, \cos \theta)$ 。“宽 / 高”各 = 一个位移 · 一个方向, 组合两对。

重问: $\text{width}(\theta)$ 配哪对? $\text{height}(\theta)$ 配哪对?

zzy

我在想一个问题。我当然记得点积是指投影的长度, 但记得这个投影的长度和 B 肯定有关系, 不止和 A 有关系, 和 B 也有关系。但你的 \hat{n} 是一个单位向量, 但是你在阐述其几何意义的时候, 怎么能够用单位向量呢?

或者我可以这么理解吗? 我就把 B 视为一个单位向量, 虽然这个 B 它可能长度不唯一, 但是我可以把这个 B 视为一个单位向量。是这个意思吗?

Claude

你的“定义 $|B|$ 为长度单位”是换单位制的描述: 以 $|B|$ 为新单位时 $|B| = 1$, 点积值 = 投影长度 (按新单位测的数值); 换回绝对单位再乘 $|B|$ 。本质和“投影长度 $\times |B|$ ”是同一件事, 描述不同。

原问题再问: $\text{width}(\theta)$ 配哪对? $\text{height}(\theta)$ 配哪对?

zzy

$\mathbf{u} \cdot \hat{n}_\theta$ 、 $\mathbf{v} \cdot \hat{n}_\theta^\perp$ 。

Claude

对。所以

$$A(\theta) = (\mathbf{u} \cdot \hat{n}_\theta)(\mathbf{v} \cdot \hat{n}_\theta^\perp).$$

这就是反证 setup 下 A 的解析式 (4 支撑 L, R, T, B 固定 $\Rightarrow \mathbf{u}, \mathbf{v}$ 固定; 只有 θ 在变)。

下一步: 把两个内积分量展开。设 $\mathbf{u} = (u_1, u_2)$ 、 $\mathbf{v} = (v_1, v_2)$ 。 $\mathbf{u} \cdot \hat{n}_\theta$ 用 $\cos \theta, \sin \theta$ 怎么写? $\mathbf{v} \cdot \hat{n}_\theta^\perp$ 呢?

zzy

$\mathbf{u} \cdot \hat{n}_\theta = u_1 \cos \theta + u_2 \sin \theta$; \mathbf{v} 同法展开 $\mathbf{v} \cdot \hat{n}_\theta^\perp = -v_1 \sin \theta + v_2 \cos \theta$ 。

Claude

对, 所以

$$A(\theta) = (u_1 \cos \theta + u_2 \sin \theta)(-v_1 \sin \theta + v_2 \cos \theta).$$

下一步: 把这个乘积展开 + 化简 (提示: 用 $\cos^2 \theta = \frac{1+\cos 2\theta}{2}$ 、 $\sin^2 \theta = \frac{1-\cos 2\theta}{2}$ 、 $\sin \theta \cos \theta = \frac{\sin 2\theta}{2}$ 把所有 θ 都升到 2θ)。 $A(\theta)$ 化简后只剩什么形式?

zzy

设 $a = u_1 \cos \theta$, $b = u_2 \sin \theta$, $c = v_1 \sin \theta$, $d = v_2 \cos \theta$, 则

$$\begin{aligned} A(\theta) &= (a + b)(-c + d) = -ca - cb + ad + bd \\ &= -u_1 v_1 \cos \theta \sin \theta - u_2 v_1 \sin^2 \theta + u_1 v_2 \cos^2 \theta + u_2 v_2 \sin \theta \cos \theta \\ &= (u_2 v_2 - u_1 v_1) \cos \theta \sin \theta - u_2 v_1 \cdot \frac{1 - \cos 2\theta}{2} + u_1 v_2 \cdot \frac{1 + \cos 2\theta}{2}. \end{aligned}$$

全部升到 2θ 后再合并：“这有什么规律？”——展开后得到混着 $\cos 2\theta$ 、 $\sin 2\theta$ 、常数项的一坨，结构性的东西不显然。

Claude

把 $/2$ 因子都收齐后：

$$A(\theta) = \underbrace{\frac{u_1 v_2 - u_2 v_1}{2}}_{C_0} + \underbrace{\frac{u_2 v_1 + u_1 v_2}{2}}_{C_1} \cos 2\theta + \underbrace{\frac{u_2 v_2 - u_1 v_1}{2}}_{C_2} \sin 2\theta.$$

规律就在这里： $C_0 + C_1 \cos 2\theta + C_2 \sin 2\theta$ 这种形式可以合并成单一正弦——

$$A(\theta) = C_0 + R \sin(2\theta + \varphi), \quad R = \sqrt{C_1^2 + C_2^2}.$$

——周期为 π 的简单正弦波（频率 2θ 的简谐振荡），绕常数 C_0 上下浮动，振幅 R 。最初看起来杂乱（ $\cos \theta, \sin \theta, \cos^2 \theta, \sin^2 \theta$ 全混着），最后塌缩成“常数 + 一个正弦”这个最简结构。

下一步： $A(\theta)$ 既然是 2θ 的简单正弦波，它的极小值取在 θ 为何值时？（提示： $\sin(\cdot)$ 函数的极小在哪里？）

zzy

极小值在 $2\theta + \varphi = \frac{3\pi}{2} + 2k\pi$ 时取到。

Claude

对。等价地 $\sin(2\theta + \varphi) = -1$ ，所以 $A_{\min} = C_0 - R$ ，是孤立的极小点（只要 $R > 0$ ）。

但等一下——我们这一路推下来 $A(\theta) = C_0 + R \sin(2\theta + \varphi)$ 是在反证 setup 下：4 支撑 L, R, T, B 固定 $\Rightarrow u, v$ 固定 \Rightarrow 系数 C_0, R, φ 固定。问题是：

这个公式在 θ 取哪个范围内有效？换句话说， θ 转到什么时候 L, R, T, B 中的某一个会从一个 hull 顶点切换到下一个 hull 顶点？

zzy

这个比较难确定吧。

Claude

对——精确长度依赖 hull 几何（每次支撑切换发生在 hull 某个顶点处），不需要算。

反证其实只关心一个性质：smooth 区间 I 是开区间——

- θ 在 I 内部时，4 支撑保持 L, R, T, B （4 单点）；
- θ 触到 I 端点时，某个支撑切换 \Leftrightarrow 矩形某条边和 hull 某条边重合（flush 状态）。

反证假设 $\Leftrightarrow \theta_0$ 不 flush $\Leftrightarrow \theta_0$ 是开区间 I 的内点。

下一步： θ_0 是 $A(\theta)$ 在开区间 I 上的极小，且在 I 的内点。这给出 A 在 θ_0 处什么一阶必要条件？写出该等式（用 C_1, C_2 或 u, v 都行）。

zzy

什么叫一阶必要条件????

Claude

术语澄清——**一阶必要条件** (first-order necessary condition, 又叫 **费马引理**):

若 $f(x)$ 在定义域内点 x_0 处取局部极值 (极大或极小), 且 f 在 x_0 处可导, 则必有

$$f'(x_0) = 0.$$

1D 直观: 极值点处函数曲线切线水平 (左升右降 = 上凸极大; 左降右升 = 下凸极小; 切线斜率都是 0)。

为什么强调“内点”: 在区间端点取极值时, f' 不需要为 0 (端点处只能往一边走, 函数可以是单调的)。例: $f(x) = x$ 在 $[0, 1]$ 上极小是 $x = 0$, 但 $f'(0) = 1 \neq 0$ 。

为什么是“必要”不是“充分”: $f'(x_0) = 0$ 不一定保证 x_0 是极值——也可能是鞍点 / 拐点 (如 $f(x) = x^3$ 在 $x = 0$)。所以一阶条件只筛掉了非极值的候选, 是否真极值还要看二阶导。

回到题目: $A(\theta)$ 是 θ 的光滑函数 (sinusoid), θ_0 是 A 在开区间 I 内部的极小 \Rightarrow 直接套用费马引理 $\Rightarrow A'(\theta_0) = 0$ 。

回到原题: 把 $A(\theta) = C_0 + R \sin(2\theta + \varphi)$ 求导, 写出 $A'(\theta_0) = 0$ 的具体形式。

zzy

继续吧, 跳过引理证明部分, 我们直接来看这个算法证明写。

Claude

好——**核心引理** 就此 buy-in: 覆盖凸包的最小面积矩形, 至少有一条边与凸包某条边共线。sinusoid + 费马引理那条严格化路径打住, 先把算法跑通; 要回填证明随时再开。

已建立的产物保留作直觉: 4 支撑 L, R, T, B 固定的平滑段内

$$A(\theta) = C_0 + R \sin(2\theta + \varphi)$$

是 2θ 的简单正弦——后面写算法时“ θ 取段端点 (也就是某支撑切线 flush 到凸包某条边)”这个结构感会直接复用。

第 3 步: 算法骨架 (暴力)

Claude

接受引理后, 最直接的暴力算法枚举什么对象?

zzy

暴力枚举矩形的一条边和另外两个点呗。

Claude

方向对——枚举一条**凸包边** e 当矩形“底边”（引理保证最优解里至少有这么一条 e 存在），然后再确定剩下的支撑点。准确计数是 3 个支撑点不是 2 ：

- **底边方向**由 e 给定，矩形**方向锁死**。设 $\hat{n}_e =$ 沿 e 的单位向量， $\hat{n}_e^\perp =$ 垂直 e 朝凸包内侧的单位向量。
- **T** (top)：凸包顶点中 \hat{n}_e^\perp 投影最大的那个 \Rightarrow 决定高。
- **L, R** (left, right)：凸包顶点中 \hat{n}_e 投影最小 / 最大的两个 \Rightarrow 共同决定宽 = R 投影 - L 投影。

所以是 1 条边 + 3 个点 (T, L, R)。 L, R 在同一根投影轴上一次扫描就能同时拿到，可能这就是你脑子里把它们合并成“一个点”的来源。

下一问：固定 e 之后，逐个扫描凸包 h 个顶点找 T, L, R 是 $O(h)$ ；外层枚举所有 h 条凸包边。这个暴力的总复杂度是多少？

zzy

$O(n^2)$ 的这个级别吧。

Claude

对， $O(h^2) \leq O(n^2)$ （凸包顶点数 $h \leq n$ ）。 $n \leq 5 \times 10^4 \Rightarrow n^2 = 2.5 \times 10^9$ ，2s 时限过不去，要再砍一个数量级。

优化的入口观察：把凸包边按 CCW 顺序记作 e_0, e_1, \dots, e_{h-1} 。当外层枚举从 e_i 推进到 e_{i+1} （底边方向连续微转一点点）时， T, L, R 这三个支撑顶点在凸包上是怎么动的？（提示：从几何上想象底边逆时针偏转一个小角度，三条支撑切线随之偏转，各自的接触顶点会怎么变。）

zzy

我大概理解了，就是维护三个指针就可以解决这个问题。

Claude

对——这就是 **旋转卡壳 (rotating calipers)**。结构性理由（不要跳）：

单调性 / monotonicity：底边方向 θ 沿 CCW 连续增加时， T, L, R 三个支撑顶点**也只沿凸包 CCW 推进，从不回退**。原因——每个支撑顶点都是凸包上某个“投影极值”顶点（ $T = \hat{n}_e^\perp$ 投影最大； $R = \hat{n}_e$ 投影最大； $L = \hat{n}_e$ 投影最小 = $-\hat{n}_e$ 投影最大），凸函数的极值点随方向 CCW 旋转只会 CCW 漂移（投影函数在凸包顶点序列上单峰）。

摊还代价：外层枚举凸包 h 条边一圈，每个指针在凸包上最多走一整圈 h 步。三个指针合计 $O(h)$ 步，加外层 $O(h)$ 次面积更新 \Rightarrow 卡壳部分 $O(h)$ 摊还。

全流程复杂度：求凸包 $O(n \log n)$ + 旋转卡壳 $O(h) \leq O(n) \Rightarrow$ 总 $O(n \log n)$ 。 $5 \times 10^4 \log(5 \times 10^4) \approx 8 \times 10^5$ ，2s 绰绰有余。

每个指针的推进判定（具体写代码会用）：以 T 为例——当前底边 $e_i = H_i \rightarrow H_{i+1}$ ， T 当前指 H_t 。问“ T 该不该 advance 到 H_{t+1} ？”等价于问“ H_{t+1} 离底边更远吗？”——用叉积表达：

$$(H_{i+1} - H_i) \times (H_{t+1} - H_i) \geq (H_{i+1} - H_i) \times (H_t - H_i),$$

也即 $(H_{i+1} - H_i) \times (H_{t+1} - H_t) \geq 0$ 。 L, R 类似但用点积比较投影。

zzy

应该是需要维护 4 个指针吧？一个是边，另外 3 个是这个点，也就是代表这个切线。

Claude

对，更准确——4 个指针，分两类：

- **驱动指针** (for 循环)：底边索引 i ，每轮 $i \rightarrow i + 1$ 强制 advance 1 步，跑遍 h 条凸包边作为“当前底边”候选。
- **跟随指针** (while 循环)： T, L, R 三个支撑点，每轮在自己的 advance 条件不再满足前一直 $++ \pmod{h}$ 。

摊还修正：4 个指针都沿凸包 CCW、各自最多走一整圈 h 步，合计 $4h = O(h)$ 次操作 \Rightarrow 卡壳部分仍然 $O(h)$ 。驱动 / 跟随的区别只是循环类型不同（确定步进 vs 条件步进），单调不后退这一点都满足。

第 4 步：实现

Claude

现在去把代码写了——`src/main.cpp`（或你已有的源文件），用旋转卡壳走一遍。流程：

- Andrew 求凸包，CCW 顺序输出 h 个顶点。
- 初始化 T, L, R 三指针：以 e_0 为底边**独立扫一遍**找到三个初始位置。
- 外层 $i = 0 \rightarrow h - 1$ 枚举底边 e_i ；内层各自 while 推进指针到本轮的最优位置；算面积更新答案。
- 还原最优时的 4 个矩形顶点（底边线 \cap 左切线、 \cap 右切线；同理上面）。

写完告诉我。

zzy

你直接写一个这个函数参考的实现吧，毕竟我们是在学习嘛。

Claude

好，我把参考实现写到 `src/P3187_HNOI2007_minimum_rectangle_cover.cpp` 里 `minimum_rectangle_cover`。

相对原 stub 的两处签名调整：

- 返回类型：`array<DB, 5>` \rightarrow `pair<DB, array<Point, 4>>`（5 个 DB 装不下面积 + 4 顶点 = 9 个数；Point 数组语义更清晰）。
- *Real* 前置：函数内部 `getintersect` 需要浮点除法，必须 `Real = DB`（模板顶部 `using Real = ll`；改成 `using Real = DB`；）。本题输入也是浮点，本来就要切。

三个写代码的细节（mentor 推导没明说、写时会卡的）：

1. **初始化不能简单赋 0**。 `up = r = l = 0` + 推进 while 严格 “>” 时，`cross/dot` 都从 0 起步，不会被严格 “>” 触发推进 \Rightarrow 卡死在 0。要在外层 for 之前对 `edge = 0` 这条底边独立扫一遍找各自最优起点。

2. 推进条件用双线性化简。“ $\text{cross}(e, A_{u+1} - A_e) > \text{cross}(e, A_u - A_e)$ ” 减一下 = “ $\text{cross}(e, A_{u+1} - A_u) > 0$ ” = “ $\text{cross}(e, e_of(up)) > 0$ ”。三个支撑点的判定都能写成“当前 *polygon* 边 $\cdot e$ ” 或 “当前 *polygon* 边 $\times e$ ” 形式，比展开式干净一档。
3. 面积比较不要除 $\sqrt{|e|^2}$ 。直接乘出 $\text{Area} \cdot |e|^2 = \text{dot}(e, A_r - A_l) \cdot \text{cross}(e, A_{up} - A_e)$ 比较；除法和开根都只在最后输出最小答案前做一次。

参考代码（你自己在 `cpp` 里照着练）：

```

1 // 最小矩形覆盖：返回 {面积, 4 个矩形顶点 (以底边 BL 起、CCW) }
2 // 前置: Real = DB (reconstruct 走 getintersect 必须浮点)
3 pair<DB, array<Point, 4> > minimum_rectangle_cover(vector<Point> A) {
4     A = make_convex_hull(A);
5     int n = SZ(A);
6     auto e_of = [&](int i) { return A[(i + 1) % n] - A[i]; }; // 第 i 条凸包边的方向向量
7
8     // 初始化 up / r / l: 在 edge = 0 这条底边上独立扫一遍，各自找最优点。
9     // 不能直接 up = r = l = 0 —推进 while 用严格 ">", 从 0 开始时 cross/dot 都是 0, 会卡住。
10    int up = 0, r = 0, l = 0;
11    {
12        Point e0 = e_of(0);
13        for (int j = 1; j < n; ++j) {
14            if (cross(e0, A[j] - A[0]) > cross(e0, A[up] - A[0])) up = j;
15            if (dot(e0, A[j] - A[0]) > dot(e0, A[r] - A[0])) r = j;
16            if (dot(e0, A[j] - A[0]) < dot(e0, A[l] - A[0])) l = j;
17        }
18    }
19
20    DB best_area = numeric_limits<DB>::infinity();
21    int best_edge = 0, best_up = up, best_r = r, best_l = l;
22
23    // 4 指针: edge 是 for 驱动指针 (确定步进), up/r/l 是 while 跟随指针 (条件步进)
24    for (int edge = 0; edge < n; ++edge) {
25        Point e = e_of(edge);
26        // 推进条件用 "polygon edge  $\times e$ " 化简:  $\text{cross}(e, A[up+1]-A[edge]) - \text{cross}(e, A[up]-A[edge]) = \text{cross}(e, e\_of(up)) > 0$ 
27        while (cross(e, e_of(up)) > 0) up = (up + 1) % n; // A[up+1] 离边更远 cross(e, e_of(up)) > 0
28        while (dot(e, e_of(r)) > 0) r = (r + 1) % n; // A[r+1] 沿 e 投影更大
29        while (dot(e, e_of(l)) < 0) l = (l + 1) % n; // A[l+1] 沿 e 投影更小
30
31        // 面积  $\times |e|^2 = (\text{宽} \times |e|) \times (\text{高} \times |e|)$ 
32        //          \_____/ \_____/
33        //          dot(e, A[r]-A[l]) cross(e, A[up]-A[edge])
34        DB w_e = (DB) dot(e, A[r] - A[l]);
35        DB h_e = (DB) cross(e, A[up] - A[edge]);
36        DB area = w_e * h_e / (DB) e.norm();
37
38        if (area < best_area) {
39            best_area = area;
40            best_edge = edge; best_up = up; best_r = r; best_l = l;
41        }
42    }
43
44    // 还原 4 顶点 = 4 条切线两两相交 (4 条线方向: 底/顶 e; 左/右 e~)
45    Point e = e_of(best_edge);
46    Point e_perp(-e.y, e.x); // CCW 90° 旋转
47    Line bottom(A[best_edge], A[best_edge] + e);
48    Line top_(A[best_up], A[best_up] + e);
49    Line right_(A[best_r], A[best_r] + e_perp);
50    Line left_(A[best_l], A[best_l] + e_perp);
51
52    array<Point, 4> corners = {
53        getintersect(bottom, left_), // BL
54        getintersect(bottom, right_), // BR
55        getintersect(top_, right_), // TR

```

```
56         getintersect(top_, left_), // TL
57     };
58     return {best_area, corners};
59 }
```

下一步：把这份代码塞进 `Solve()` 里跑题面样例，AC 之后再做 stress test（cyaron-stress skill 的非均匀曲率形状）。