

Bridge Across Islands ——两凸多边形最小距离

algorithm-mentor session

2026-05-08

第 1 步: 理解题目

zzy

跳过这一步, 直奔算法。

Claude

快速对一下数据范围: $N, M \leq 10^4$, 坐标 $\in [-10^4, 10^4]$, 多组数据, 误差 $\leq 10^{-3}$ 。两个**已经给出顶点序的凸多边形**, 求边界间最小距离——边到边、点到点、点到边都可能取到。

第 2 步: 暴力解法思路

Claude

先把暴力理清楚再考虑优化。

最直接的写法: 对两个多边形的所有「特征对」逐个算距离取 \min 。问题是——**两凸多边形之间的最小距离**最终一定取在哪种「特征对」上? 点-点、点-边、边-边, 哪些是真的可能成为答案, 哪些其实可以不用单独算?

zzy

应该只有点-边可以取到吧? 但如果两个凸包相交可能不大行。

Claude

方向对。补完三件事:

(1) 三类特征对的归并。对两个不相交的凸多边形, 最小距离的实现位置只有三种:

- **顶点 - 边内部**: 从某顶点向另一多边形的某条边作垂线, 垂足落在边内部。
- **顶点 - 顶点**: 两个多边形上各取一个顶点。这其实是上一种的退化——拿 $\text{dist}(v_A, \text{线段} e_B)$ (线段距离, 不是直线距离) 算时, 如果垂足落在 e_B 外面, 点-线段距离就退成点-点距离; 所以它包含在「顶点对线段」里, 不用单独写。
- **边 - 边**: 仅当两边平行时才可能, 且此时两条边的端点也一定在同一最小距离上——所以这种情形也被「顶点对线段」覆盖。

结论:**枚举所有 (顶点, 对面边) 对, 算点-线段距离取 \min** , 三种情形一网打尽。

(2) **对称性**。单边枚举不够——必须枚举「 A 的顶点对 B 的边」和「 B 的顶点对 A 的边」两个方向。反例: 细长 A 戳向粗 B 的边中间, A 的端点垂足落在 B 的边内部, 这种情形「 A 顶点 - B 边」抓得到; 反过来「 B 顶点 - A 边」抓不到——但加上对称的另一个方向就能 cover 所有非对称几何。

(3) **相交怎么办**。本题语境两座岛屿在海里、天然分离; 真相交答案就是 0。更稳妥的写法是先做个凸包-凸包相交判定 (可以用 SAT 或半平面), 相交直接返 0。不过实务上, 如果你只在乎题目数据, 可以省掉这步直接算最小点-线段距离——数据保证分离时不会出错。

zzy

跳过暴力, 直接想优化。

第 4 步: 优化思路——旋转卡壳

Claude

暴力 $O(NM)$ 在 $N = M = 10^4$ 多组数据下走不动。先记一个简短事实再问问题:
暴力之所以浪费, 是因为对 A 的每个顶点 p , 我们都把 B 的全部 M 条边都拿来算了一遍点-线段距离。但凸性下, p 看向 B 时, B 上「真正可能成为 p 最近边」的边其实非常少。
问题: 固定 A 上一个顶点 p , 把 B 想象成站在远处的一个凸多边形。 B 上有多少条边有资格成为「离 p 最近的那条»? 给一个数 + 一句话直觉解释为什么。

zzy

只有一条。这不是废话吗?

Claude

对, 但这条「废话级 fact」是旋转卡壳的全部根基。把它换一个说法:
每个 $p \in A$, 对应 B 上唯一一条「facing 边」(facing edge)。 形象点: 从 p 看过去, B 朝向 p 的那一面只有一条边「正对」着 p , 其它边要么被它挡住, 要么转到另一侧去了。
下一步要观察的是 **动态** 行为: 把顶点 p 沿 A 的边界走一圈 (CCW), 那条「facing 边」会在 B 上怎么变化?
更具体一点: 它在 B 上的位置 (顶点编号) 是乱跳、往返、还是有某种规律?
写一句你的直觉就够, 不用证明。

zzy

肯定是有规律的, 这个规律类似于往返。

Claude

对, 「短程单调、长程往返」——关键观察是:
 B 上能当 facing 边的, 只是 B 朝向 A 的那半弧上的边。 B 背向 A 那半弧的所有边都不参与候选——没有任何 $p \in A$ 会和那些边构成最近对。

把它跑动起来:

- **短程 (p 走在 A 的 facing 弧上):** p 沿 A 朝 B 那半 CCW 走时, B 的 facing 边在 B 的 facing 弧上单调推进。这是「单调」的来源。
- **长程 (p 越过 A 的背面):** A 的 facing 弧走完后, p 进入 A 背向 B 那半。此时 B 没有新边可选—— B 的 facing 边只能沿同一段 facing 弧**退回来**。 p 走完整圈, B 的 facing 边轨迹 = 「推过去 \rightarrow 退回来」, 就是**往返**。

这个观察对算法有直接好处: 旋转卡壳跑两凸多边形最近距离时, **两个指针不需要走完一整圈**, 各自只走完自己的 facing 弧就够了——总步数 $O(N + M)$ 。

下一题: facing 弧的两个端点 (单调推进的起止点 = 「开始往返」的转折点), 在 A / B 上对应几何意义上的什么特殊点?

zzy

跳过这些细问, 直接给算法骨架。

Claude

好, 直接给。

Setup(确定起始 facing 对):

- 把 A 、 B 都按 CCW 整理 (读完先判向、CW 就 reverse)。
- 取 A 上 y 最小的点 p , B 上 y 最大的点 q 。 p 看上去就是 q , q 看下去就是 p ——这一对是初始 *facing pair*, 也是 A/B 各自 facing 弧的一个端点。

核心循环 (双指针走 facing 弧):

- 当前状态: $p = A_i, q = B_j$ 。 两条出边: $\vec{a} = A_{i+1} - A_i$ 、 $\vec{b} = B_j - B_{j+1}$ (B 的方向 **反向**, 因为 facing 弧逆 B 的 CCW 序前进)。
- 比较 \vec{a} 与 \vec{b} 的极角: 谁角更小就先推谁。具体: 看 $\text{cross}(\vec{a}, \vec{b})$ ——正则推 i , 负则推 j , 0 任选。几何意义: 角度跑得慢的那条边可以多容纳一格, 所以让它的指针先动。
- 每步, 顺手算两个点-线段距离更新答案: $\text{dist}(A_i, \text{seg}[B_j, B_{j+1}])$ 和 $\text{dist}(B_j, \text{seg}[A_i, A_{i+1}])$ 。

终止: 两个指针各自把 facing 弧走完时退出——安全做法是循环 $N + M$ 步 (或: 让 i 走完 A 一圈、 j 同步推, 反正一定够)。

复杂度: $O(N + M)$ per testcase。

要点提示 (写代码时容易翻车的几个):

- 输入 CCW 不能假设, 先 *signed area* 判向, 负就 reverse。
- 点-线段距离记得分三段 (垂足在内 / 在两端外)。
- cross 用整数运算最稳; 坐标 $\leq 10^4$, $N, M \leq 10^4$, 中间结果远在 long long 内。

去 `src/Bridge_Across_Islands.cpp` 实现这个, 写完告诉我。

第 5 步: cross product 判定的几何意义

Claude

第一版用「 $\text{dis}_2 < \text{dis}_1$ 就推 B 」的贪心, 在 OJ 上 TLE ——因为 A 走到背面弧时, 贪心不再单调, j 想往回走但只能前推, 陷在 B 上转圈。

正解换成 cross product 判定:

$$\text{cross}(A_{i+1} - A_i, B_j - B_{j+1}) \begin{cases} > 0 & \Rightarrow ++i \\ < 0 & \Rightarrow ++j \\ = 0 & \text{任选 (推荐两个都试一下取 min)} \end{cases}$$

几何意义. 设

$$\vec{a} = A_{i+1} - A_i, \quad \vec{b}' = B_j - B_{j+1}.$$

\vec{a} 是 A 在 A_i 的 **CCW 出边** 方向。 \vec{b}' 是 B 在 B_j 的 **CCW 出边反向**——反向是因为 B 的 facing 弧逆 B 的 CCW 序前进, 从「站在 facing 弧上往前走」的视角看, B 边方向就是 $-(B_{j+1} - B_j)$ 。初始 facing pair (A 最低点 p 、 B 最高点 q) 处:

- \vec{a} 大约朝 $+x$ (A 最低处的 CCW 切线水平向右)
- \vec{b}' 也大约朝 $+x$ (B 最高处 CCW 切线朝左, 反向后朝右)

两者初始几乎平行同向, 且两者沿各自多边形 CCW 推进时都向 CCW 方向旋转 (凸性)。 $\text{cross}(\vec{a}, \vec{b}')$ 衡量两者角度差:

- > 0 : \vec{b}' 在 \vec{a} 的 CCW 一侧 $\Rightarrow \vec{a}$ 角度落后 \Rightarrow 让 A 推一步追上 $\Rightarrow ++i$
- < 0 : 反之 $\Rightarrow ++j$

为什么这样不会 TLE. 每步刚好推一个指针、各自只能 forward。 i 走完 A 一圈、 j 走完 B 一圈就停, 总步数 $\leq N + M$ 。背面弧问题自动消失:cross 一直按角度差驱动, 根本不依赖「dis 谁更小」这种局部贪心。

可视化见同目录 `cross_product_demo.html` ——把两个凸多边形、 \vec{a} / \vec{b} 两条向量、cross product 符号、当前应推哪个指针, 逐步展开。