RC 拆解：经典 `convex_diameter` 为什么对 / 改造哪里破了

zzy & Claude

2026-05-07

本文回答三件事：

1. 经典 RC 实现 `src/M_Computational_Geometry.cpp:587--621`（`convex_diameter`）为什么**对**。
2. 改造版 `src/M_Computational_Geometry.cpp:745--763`（外层切法 + 内层 RC 嫁接子弧增长）哪里**破**了 §1 的覆盖保证。
3. 具体到对拍反例 `n=6 spike-3` 凸包，为什么 dp[3][1] 锚住 $(A_0, A_4) = 103120$ 而漏 $(A_3, A_0) = 117770$。

上下文与对拍反例数据见同目录 `debug.tex` §「对拍反例：n=6 spike-3 凸包」。

1 经典 RC 为什么对

反极对定理（陈述）

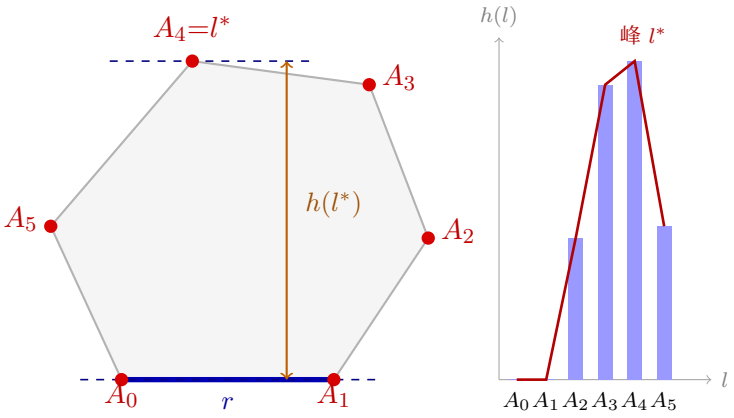
凸多边形 $P = A_0 A_1 \cdots A_{n-1}$（CCW 顺序）的直径必由**某条边** $r^* = (A_{r^*}, A_{r^*+1})$ **与到该边所在直线的垂直距离最大的顶点** l^* 实现，且

$$\operatorname{diam}(P) = \max\big(d(A_{l^*}, A_{r^*}),\ d(A_{l^*}, A_{r^*+1}) \big).$$

推论：暴力做法是对每条 r 找 $l^*(r)$，每个 $O(n)$，总 $O(n^2)$。RC 把内层降到摊还 $O(1)$。

单峰性（per-edge）

固定一条边 r，沿凸多边形顶点序 $A_0, A_1, \ldots, A_{n-1}$ 遍历，每个顶点到 $\operatorname{line}(r)$ 的垂直距离 $h(l) = d(A_l, \operatorname{line}(r))$ 是一个**单峰函数**（沿一侧严格递增到峰 $l^*(r)$，再严格递减）。单峰来源是凸性：凸多边形在 $\operatorname{line}(r)$ 一侧的边界是一条凸链，凸链上点到一条直线的距离沿链是单峰的。



图注：固定边 $r = (A_0, A_1)$（蓝粗）后，沿 CCW 序遍历其它顶点，距离 $h(l) = d(A_l, \operatorname{line}(r))$ 是单峰（右图柱状）：$h(A_2) = 2.4 \to h(A_3) = 5.0 \to h(A_4) = 5.4$（峰）$\to h(A_5) = 2.6$。$l^*(r) = A_4$ 是反极顶点；定理说 $\operatorname{diam}(P) = \max(d(A_4, A_0), d(A_4, A_1))$。

跨 edge 的单调推进

当 r 沿 CCW 旋转一步（$r \to r+1$，即 r 的两端点都往前挪一格），其反极顶点 $l^*(r+1)$ 只会**不回退地往前推** 0 步、1 步或多步——不会跳回 CW 方向。几何来源：$\mathrm{line}(r+1)$ 是 $\mathrm{line}(r)$ 同向小幅旋转的结果，新的距离单峰函数也只是「平移」而非反向；峰位 l^* 因此只前进或不动。

实现上 `cpp:605--610` 的 `need_l_move(l, r)` 用面积比较代替距离比较：

$$\mathrm{area}(l,r) = |A_r A_{r+1} \times A_l| = |\mathrm{base}(r)| \cdot h(l).$$

因为 $|\mathrm{base}(r)|$ 对左右两端 l 与 $l+1$ 是同一条底，所以 $\mathrm{area}(l,r) \le \mathrm{area}(l+1,r) \iff h(l) \le h(l+1)$。这一不等式表示「$l$ 还没爬到当前 r 的单峰顶点，应当继续推」。r 转一圈推 n 步、l 全程也只推 $\le n$ 步，摊还 $O(n)$。

覆盖完全的两条保证

1. **每条 r 都被外循环遍历到** → 反极对定理里的「那条 r^*」必被到访。
2. **r 转一圈、l 同步走 $\le n$ 步** → 当 $r = r^*$ 时 l 必正好停在 l^*（单调不回退 + 单峰位置 cyclic 平移）。

两条合起来：直径对 (A_{l^*}, A_{r^*}) 或 (A_{l^*}, A_{r^*+1}) 必被 `cpp:616--617` 的两行 `max` collect。

2 改造版想做什么

子弧 + chord = 闭子多边形

枚举切法 (i,j) 的内层任务是：求子弧 $\mathrm{arc}(i, i+1, \ldots, j)$ 上所有顶点对距离的最大值。

这个任务等价于「对**闭子多边形** $R' = \mathrm{arc}(i,\ldots,j) \cup \mathrm{chord}(A_j, A_i)$ 跑一次 `convex_diameter`」——因为 chord 把子弧封闭成一个新的凸多边形，而其顶点集恰好就是子弧上的顶点（chord 不引入新点）。任意凸多边形上 §1 的反极对定理都成立，所以对 R' 跑经典 RC **是对的**。

改造想偷一手：嫁接单调性到「子弧增长」维度

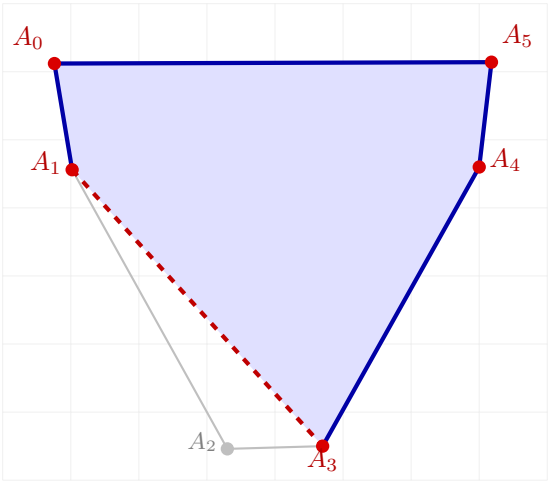
对每个 (i,j) 都重跑完整 RC 的代价是 $O(n^3)$。改造版想偷懒：**固定外层 i，让 j 沿 CCW 增长——子弧扩张**；希望把子弧 $j-1$ 的 RC 状态**延续**给 j，只为新加入的顶点 A_j 增量更新，而不是从头跑。

具体在 `cpp:745--763`：

- 外层 $i \in [0, 2n)$，子弧起点 $i \bmod n$ 固定。
- 内层 j 从 $i+2$ 起步（即子弧第一个非平凡右端是 A_{i+2}，子弧已含 A_i, A_{i+1}, A_{i+2}），每加一步把新顶点 $\mathrm{up} = A_{j \bmod n}$ 接入 RC。
- 维护一根 `edge` 指针从 $i \bmod n$ 起步，靠 `need_edge_move(edge, up)` 单调推进——意图把它当作「$\mathrm{up} = A_j$ 的反极边」。
- 每步 collect 三种 candidate（`cpp:759--761`）：

$$d^2(A_{\mathrm{edge}}, A_{\mathrm{edge}+1}), \quad d^2(A_{\mathrm{up}}, A_{\mathrm{edge}}), \quad d^2(A_{\mathrm{up}}, A_{\mathrm{edge}+1}).$$

注意角色相比经典版被翻转了：经典 RC 是「外 r / 内 l」，改造是「外 up（vert）/ 内 edge」。经典 RC 角色对调本身不破坏正确性（双指针对称）——但前提是**对同一个固定凸多边形完成完整一圈**。改造里的子多边形 R' 每次 j 推进时都在变。



图注：以对拍反例 $n = 6$ spike-3 hull、切法 $(i_1 = 1, j_1 = 3)$ 为例。**蓝实** = 子弧 $\mathrm{arc}(3, 4, 5, 0, 1)$ 上的 4 条原多边形边；**红虚** = chord (A_1, A_3)。两者合起来构成闭子多边形 $R' = \mathrm{arc}(3..1) \cup \mathrm{chord}$（蓝色填充区域），它本身是凸的，对它跑经典 RC 会得到 R' 的正确直径。改造版希望嫁接「j 增长」的单调性到对 R' 的 RC 上。A_2（灰）不在 R' 内。

3 改造在哪破了 §1 的覆盖保证

经典 RC 的两条覆盖保证（**每条边都当过 r + 每个顶点都被 l 扫过**）在改造里**两个都破了**。

破点 1：up 不取 arc 起点 A_i / A_{i+1}

改造内层 `cpp:748 for (int j = i + 2; ...)`，所以 up $= j \bmod n$ 永远不取 $i \bmod n$、也不取 $(i+1) \bmod n$——子弧的前两个顶点**从来没当过 vert**。

经典 RC 在闭子多边形 R' 上跑一遍时，每个顶点都会作为某条 r 的反极 l^* 被到访（覆盖保证 2）；而改造把 A_i / A_{i+1} 这两个 vert 角色的访问**完全跳过**。后果：若直径对 (A_a, A_b) 必须有一端当过 up 才能 collect，且这一端恰好是 A_i 或 A_{i+1}，则漏。对拍反例 dp[3][1] 漏的 (A_3, A_0) 正是这种：$A_3 = A_i$，永远不进 up。

破点 2：edge 单调推进永不绕到 chord (A_j, A_i)

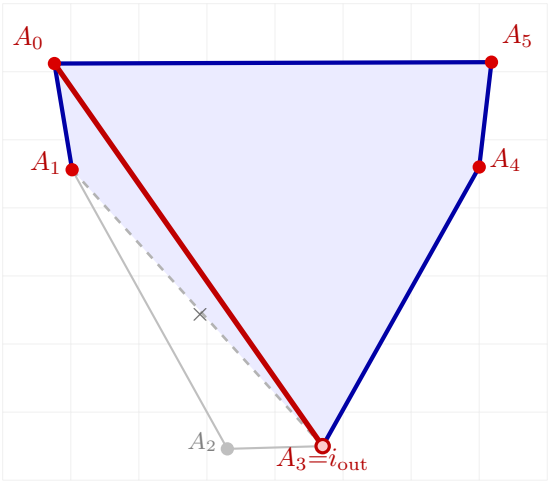
闭子多边形 R' 的边集 = 子弧上的原边 + **一条 chord** (A_j, A_i)。经典 RC 在 R' 上跑一圈时会把 chord 也作为某次 r 处理（覆盖保证 1）；而改造里 edge 起步于 $i \bmod n$，靠 need_edge_move 单调**递增**，永远停留在原多边形的相邻顶点对 $(A_{\text{edge}}, A_{\text{edge}+1})$ 上——chord 是非相邻的 (A_j, A_i)，这种数值永远不会在 edge 变量里出现。后果：若直径对的实现方式是「以 chord 为 r、某顶点为 l^*」，则漏。

两个破点合起来

直径对 (A_a, A_b) 在改造里能被 collect 的形式只有三种（`cpp:759--761`）：

1. a, b 同时是某条原边的两端点（$d^2(A_{\text{edge}}, A_{\text{edge}+1})$）。
2. 一端 $\in \{A_{\text{edge}}, A_{\text{edge}+1}\}$，另一端 $= A_{\text{up}}$，且这一对在某轮 j 时 edge 恰好处于配对位置。

(A_3, A_0) 对在 dp[3][1] 的求解过程里满足不了 (1)（不是相邻边端点），也满足不了 (2)（A_3 不进 up；A_0 进 up 时 edge 已被推过 3，见 §4 trace）。



图注：同一闭子多边形 R'。**蓝实** = edge 单调推进能扫到的原多边形子弧边（4 条）；**灰虚** × = chord (A_1, A_3)，edge 数值永远不会取到（破点 2）；红圈顶点 $A_3 = i_{\text{outer}}$ 被强调，因为内层 j 起步 $i+2$，A_3 与 $A_4 = A_{i+1}$ 都永远不进 up（破点 1）。**红粗** = R' 真直径 (A_3, A_0)，$d^2 = 117770$——一端在 forbidden up 上、另一端不与它构成原边端点对，三种 candidate 形式都接不到，故漏。

4 具体到 dp[3][1] = 103120

按 `cpp:745--763` 实际跑一遍外层 $i = 3$（dp[3][*] 的写入路径），edge 起步 $= 3$，依次走 $j = 5, 6, 7$。

j	up $= j \bmod n$	edge（推进后）	本轮新 candidate（d^2）	ans 累积
5	A_5	3（不动）	(A_3, A_4) 55250; (A_5, A_3) 94900; (A_5, A_4) 6010	94900
6	A_0	$3 \to 4$	(A_4, A_5) 6010; **(A_0, A_4) 103120**; (A_0, A_5) 103042	**103120**
7	A_1	4（不动）	(A_4, A_5) 6010; (A_1, A_4) 89405; (A_1, A_5) 101105	103120

图注：外层 $i = 3$ 时 `dp[3][*]` 的写入 trace。**橙** 行 $j = 6$：`need_edge_move(3, A_0)` 为 true（$\text{dist}(A_0, \text{line}(A_3A_4)) = 309 < \text{dist}(A_0, \text{line}(A_4A_5)) = 319$），edge 从 3 推到 4；再 check `need_edge_move(4, A_0)` 为 `false`（A_0 在 line(A_5A_0) 上 dist $= 0$），停。本轮 collect 锁死为 (A_0, A_4) / (A_0, A_5) / (A_4, A_5)，ans 顶到 **103120**（来自 (A_0, A_4)）。$j = 7$ 后 ans 不再被刷新，dp[3][1] 落盘 $= 103120$。

为什么是 (A_0, A_4) 而不是真直径 $(A_3, A_0) = 117770$：

- A_3 当 up：要等到 $j = i + 6 = 9$ 时 $j \bmod n = 3$。但 dp[3][1] 落盘在 $j = 7$，早就结束了（**破点 1**：$A_3 = A_i$ 永远不进 up，更别说 dp[3][1] 在 $j = 7$ 落盘）。
- A_3 当 edge 端点：$j = 6$ (up $= A_0$) 那一轮 `need_edge_move` 把 edge 从 3 推到 4，candidate 锁死为 (A_0, A_4) / (A_0, A_5) / (A_4, A_5)，A_3 已不在 edge 两端（**破点 2** 的具体表现：edge 单调向前不回头，绕不回 A_3 那条边）。

所以改造 RC 在 dp[3][1] 落盘时段内能够到的 R' 内最大 d^2 就是 103120（$j = 6$ 锁住），它看起来像反极意义下的 max，实际只是「在改造的部分覆盖下、于 $j \le 7$ 时段可见的 max」——**不**是 R' 的真直径。

5 结构性结论

经典 RC 把「凸多边形直径」翻译成「edge × 反极 vertex 的 1D 单调扫描」，前提是**对同一个凸多边形完成完整一圈**（覆盖保证 1 + 2）。

改造把这个 1D 扫描嫁接到「子弧增长」的另一维（外层 j 推进），但子弧每次扩张并**没有**补完 §1 的两条保证：

- up 不取 arc 起点 → 顶点遍历不完整。
- edge 不绕到 chord → 边遍历不完整。

所以单调扫描覆盖到的 candidate 集合是不完整的——这**不是** off-by-one、**不是**边界条件漏判，是**算法假设和单调嫁接方式不兼容**。

修法走 `debug.tex` §「修复路径」已写的方案 B（区间 DP 根治）或方案 C（chord 对踵 + sub-arc 缓存）。