有效的简单多边形三角化算法的认识和实现

合作者：陈小雕 彭宇 刘玉身

摘 要 *Godfried Toussaint* 提出了一个有效的简单多边形三角化的算法。我们分析了文中提出的一种新的三角化复杂性的度量标准。并在此标准下实现了算法。在算法的实现过程中还针对一些特殊情形给予了分析。

关键字 三角化 复杂性 对角线 computational complexity output complexity

引言 简单(also Jordan)多边形的三角剖分是多年来人们热衷的问题。早在 1911 年，Lennes[1]就提出了插入内对角线将多边形分为两半的递归算法，复杂度为 O（n^2）。后来 Meisters[2]提出了双耳定理，提出了一个用割耳的方法实现剖分的算法，其复杂度依然为 O(n^2)。第一个打破 O(n^2)界限的是 Garey[3]等人，他们提出一个复杂度为 O(nlogn)的算法，这个算法先要把多边形分解为单调多边形，然后用一个线性算法，剖分单调多边形。Chazelle[4]用 divide-and-conquer 的方法也实现了 O(nlogn)的复杂度。在经过了 O(nlognlongn)的算法，O(nlog*n)的算法，以及一些特殊形状多边形的 O(n)算法实践之后，人们终于大胆地向前迈了一步，1990 年 Bernard Chazelle 发表文章认为一个 n 个顶点的简单多边形可以在 O(n)的时间内进行三角剖分。当然，到目前为止，这个算法只是理论上的很好的突破，还从来没有人将它实现，因此，它的具体的实用价值还没有体现出来。

所以，人们注重寻找简单有效并且也容易实现的算法。这里值得一提的 是 Toussiant 和 Kong 等人提出了基于 Graham Scan 方法的一种算法，其复杂度也是 O(nlogn)，但是它是非常容易实现。在这篇文章中，*Godfried Toussaint* 仅仅利用了一些几何知识，对多边形的一些特征进行分析，设计出一个简单且也容易实现的算法，它不需要排序和运用平衡二叉树这样的复杂数据结构。这个算法的复杂度是 O($n(1+t_0)$ ，其中 $t_0< n$，t_0 度量了由算法得到的三角化的形状复杂性，更精确地说，**t_0 表示所有三角剖分后完全由内对角线组成的三角形的个数**。这个算法的最坏情况为 O(n^2)。但是，它不仅是第一个计算复杂性是输出复杂性的函数（即输出敏感）的三角剖分算法（这具有很好的理论意义），还有另外一个贡献

是，它引进了一种新的度量三角形的形状复杂性的方法。这在以后有人发展了这个概念，用α来表示多边形的形状复杂度，并且应用到模式识别和图象分析中。

在这篇文章中，我们主要实现了由 *Godfried Toussaint* 提出的一个算法，并学习和分析了文中的三角形的形状复杂性的一种度量标准。

1．一种新的三角化复杂性度量的分析

2． 1 基本概念(key word)

一个简单多边形的三角剖分的对偶图是一棵度不超过 3 的树。就是这棵树中度为 3 的节点数目（如果针对多边形本身，则它是三角剖分中和原来多边形没有公共边的三角形的数目）。（这是本文的一个很重要的概念。）

1）a *principal* vertex（主要点）：

如果三角形[X i-1 ， X i ， X i+1]中不含有多边形 P 中的点，则说 X i 是主要点。

2）Ear（耳）：

对简单多边形 P 的主要点 X i，如果对角线[X i-1 ， X i+1]完全位于 P 中（对角线[X i-1 ， X i+1]是内对角线），则说 X i 是一个耳。

3）Mouth（嘴）：

对角线[X i-1 ， X i+1]是外对角线的简单多边形 P 的主要点 X i 是一个嘴。

4）Anthropomorphic：

如果简单多边形 P 恰好包含两个耳和一个嘴，则说 P 是 *Anthropomorphic*。

5）Sleeve（袖）：

如果一个被三角剖分的多边形的对偶树是一个链，则说这个被三角剖分的多边形是一个袖。

6）楔：以v_i为中心,顺时针从v_j到v_r的无界扇形区域.

2． 2 理论框架

下面具体列举了本文用到的几个主要引理。

引理 1：每一个 n(n > 3)边形多可以在 O(n)时间内被分成两个子多边形。

注释：任何一个非三角形的多边形必定存在一条内对角线。这可以有一个基本是构造性的证明。在算法实现中我们借助了这个引理。因为我们不能用分治的

方法递归地简单寻找内对角线，因为这样复杂度是 O(nlogn)。下面的接连的几个引理是用来帮助快速地查找内对角线。

引理 2：如果 X_i 简单多边形 P 的一个凹点，那么可以在 $H(X_{i-1}, X_i)$ 范围内必然存在多边形的一条内对角线$[X_i, X_j]$，并且可以在 O(n)时间内找到。$H(X_{i-1}, X_i)$ 表示射线(X_{i-1}, X_i) 的左半边。

引理 3：如果 X_i，X_{i-1} 是简单多边形 P 的两个相邻的凹点，那么必然存在一个顶点 $X_k(k!=i, i-1)$， 使得$[X_i, X_k, X_{i-1}]$是多边形内部的三角形，而且 k 可以在 O(n)时间内找到。

引理 4：如果 d_{ij} 是多边形 P 的两个顶点 X_i，X_j 的连线，如果以下 A 或 B 两个条件有一个不满足，那么 d_{ij} 不是 P 的内对角线。

(A) $[X_i , X_{i+1}] \in Wedge[X_i X_j X_{i-1}]$,

(B) $[X_j , X_{j-1}] \in Wedge[X_j X_{j+1} X_i]$,

其中 $Wedge[X_i X_j X_{i-1}]$表示从 X_i 点引出一条射线，从和 $X_i X_j$ 重合的线顺时针扫描到和 $X_i X_{i-1}$ 的重合的线的无界扇形区域。

引理 5：如果在由多边形 P 的三个顶点 X_r, X_s, X_t 组成的三角形中，有其他的顶点在这个三角形内部，X_k 是这些顶点中使得$\angle X_k X_r X_s$ 最小的一个。那么 $X_k X_r$ 以及 $X_k X_s$ 都是 P 的内对角线。这个 X_k 可以在 O(n)时间内找到。

引理 6：若$[X_i , X_{i+1}] \in Wedge[X_i X_j X_{i-1}]$，$[X_i , X_{i+1}] \in Wedge[X_j X_{j+1} X_i]$，而且 X_{j-1} ，X_j， X_i 是一个 left turn，同时 X_{i+1}， X_i， X_j 是一个 right turn，那么以下两种情况同时满足：

(a) X_{j-1} 在$\Delta X_j X_i X_{i+1}$ 内部

(b) X_{i+1} 在$\Delta X_j X_i X_{j-1}$ 内部。

引理 7：若$[X_i , X_{i+1}] \in Wedge[X_i X_j X_{i-1}]$，$[X_i , X_{i+1}] \in Wedge[X_j X_{j+1} X_i]$，而且 X_{j-1} ，X_j， X_i 是一个 left turn，或者 X_{i+1}， X_i， X_j 是一个 right turn，那么在 (X_{i+1}, X_{j-1})中必然有一点和 d_{ij} 组成一个内三角形。这个点可以在 O(1)时间内找到。

注释：这个引理很有用。将给我们分析一个袖的三角剖分可以在 O(n)内完成提供了有力的理论基础。这是一个关键。

2．3 关于上面理论的一些分析和解释

要摆脱和前人分治算法的不同性，就必须要建立或引用一些别人还没有用到的概念。本文就是在这种思想的牵引下，运用了一个叫袖（sleeve）的概念，将原来多边形转化为若干个（两两不相交的）袖，而不是转化为单调多边形。因为，经过分析，发现如果多边形是袖的话，我们就可以在一个常数时间内完成内对角线的的查找。这样，我们的注意力（工作中心）就可以转移到如何将一个简单多边形转化到若干个袖。至此，这个算法的复杂度也就很清楚了。我们认为这是本文的一个成功之处。当然，建立了一个有效的判断多边形的形状复杂性的衡量标准也是本文的又一个贡献。

3．算法的实现

3．0 总的算法如下：

Input： 一个按逆时针排序的简单多边形 P。

Output： P 以及 n-3 条内对角线。

Begin

Step1： 用 PROCEDURE DIAGONAL 算法找到一条内对角线[X_i， X_j]

Step2： 用 PROCEDURE TRIANGULATION 算法剖分 CH[X_i，…，X_j]

Step3： 用 PROCEDURE TRIANGULATION 算法剖分 CH[X_j，…，X_i]

End

3.1 找到简单多边形的内对角线

具体算法步骤

PROCEDURE DIAGONAL 算法(参见图 1)：

具体算法如下：

Begin

Step1： 找到 P 的一个凸点 X_i。

Step2： 构造$\angle X_{i-1}X_iX_{i+1}$的平分线。

Step3： 找到射线 Ray(X_i)和各条边的交点，令和 X_i 距离最近的点为 y；如果 y 是一个顶点，那么退出，[X_i，y]作为内对角线。

Step4： 构造三角形[X_i，y，X_{j+1}]。

Step5： 对所有的 $j+1<k<i$，如果 X_k 位于三角形[X_i，y，X_{j+1}]内，则把 X_k 记

为 X_k^*。如果没有位于三角形内的点，则退出，[X_i，X_{j+1}]作为内对角线。

Step6： 找到三角形内使得角∠yX_i X_k^*最小的那个顶点 z，如果 z≠X_{i-1}，则退出，[X_i， z]作为内对角线。

Step7： 构造三角形[X_j，y，X_i]。

Step8： 对所有的 $j > k > i$，如果 X_k 位于三角形[X_j，y，X_i]内，则把 X_k 记为 X_k^*。如果没有位于三角形内的点，则退出，[X_i，X_j]作为内对角线。

Step9： 找到三角形内使得点∠yX_iX_k最小的那个顶点 w，如果 w≠X_{i-1}，则退出，[X_i， w]作为内对角线。

Step10： 退出，[X_{i-1}，X_{i+1}]作为内对角线。

End

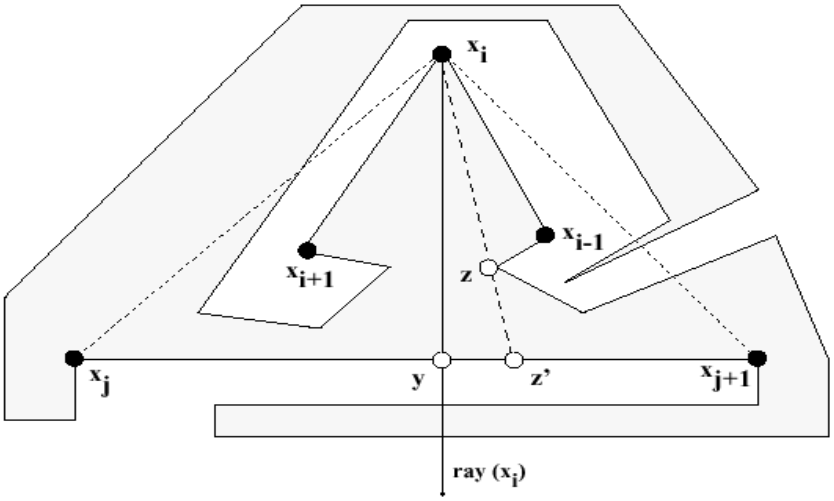


图1：Illustrating Procedure DIAGONAL

3．2 剖分子多边形的算法极其具体步骤：

PROCEDURE TRIANGULATION 算法剖分：

输入：带一个内对角线 $d_{ab}=[x_a,x_b]$，且呈逆时针排序的 n 边简单多边形 P。

输出：经过 CH[X_a， … ， X_b]三角剖分后所得的多边形 P。

D 是用来保存袖子入口对角线的列表，TS 是用来保存过程中的对角线的列表，TP 是一个保存结果的列表。

Begin

初始化：D← d_{ab}（d_{ab}是一个内对角线，袖子入口）

While D 不为空 **Do**：

1. 从 D 中取出一条对角线 $d_{ab}=[x_a,x_b]$，并且从列表中删除这对角线。

2. i←a；j←b。

While i≠j **Do** {对由 d_{ab} 确立的袖子进行三角剖分}

1.1 检测是否 $[x_i,x_{i\oplus 1}]\in \text{Wedge}[x_i x_j x_{j-1}]$ 或者 $[x_j,x_{j-1}]\in \text{Wedge}[x_j x_{j\oplus 1} x_i]$

1。2 **if** 上面的两种检测都不满足(Case 1 in Fig。 10)，**then**（a）在 TS[d_{ab},d_{ij}] 中

确定最深的三角形 $\Delta x_r x_s x_l$ (见 Fig。 9)，其中 $\Delta x_r x_s x_l$ 包含了 CH[$x_{i\oplus 1},...,x_{j-1}$] 中的顶点； (b) 确定 CH[$x_{i\oplus 1},...,x_{j-1}$]中的顶点 x_k，使 x_k 相对于 x_r 和 x_s 可见；（c）将对角线 $d_{sk}=[x_s,x_k]$ 和 $d_{kr}=[x_k,x_r]$ 加入到 T(P) 和 D 中；（d）删除 TS[d_{ab},d_{ij}]中从 d_{ij}（包括 d_{ij}）到 d_{lr}（包括 d_{lr}）的所有对角线，将 TS[d_{ab},d_{ij}] 中所有保留下来的对角线插入到 T(P)，然后 **Exit**。

1.3 **Else** (Case 2 in Fig。10) **If** x_{j-1},x_j,x_i 是一个右旋 **AND** $x_{i\oplus 1},x_i,x_j$ 是一个右旋(Case 2。1 in Fig。 10) **then** 添加 [$x_{i\oplus 1},x_j$] 到 TS[d_{ab},d_{ij}]来产生 TS[$d_{ab},d_{i\oplus 1,j}$]；i ←i+1 然后 **go to** 1。1

1.4 **If** x_{j-1},x_j,x_i 是一个左旋 **AND** $x_{i\oplus 1},x_i,x_j$ 是一个左旋 (Case 2。2 in Fig。 10)

then 添加 [x_{j-1},x_i] 到 TS[d_{ab},d_{ij}]来产生 TS[$d_{ab},d_{i,j-1}$]；j←j-1

然后 **go to** 1。1

1.5 **If** x_{j-1},x_j,x_i 是一个左旋 **AND** $x_{i\oplus 1},x_i,x_j$ 是一个右旋 (Case 2。3 in Fig。 10)

then：

1。6 **I f** $x_{j-1}\in int\ \Delta x_{i\oplus 1}x_i x_j$ (Case 2。3。1 in Fig。 10) **then** 添加 [x_{j-1},x_i] 到 TS[d_{ab},d_{ij}]来产生TS[$d_{ab},d_{i,j-1}$]；j←j-1 然后 **go to** 1。1

1。7 **If** $x_{i⊞1} \in int\,\Delta x_i x_j x_{j-1}$ (Case 2。3。2 in Fig。 10) **then** 添加 $[x_j, x_{i⊞1}]$ 到 TS[d_{ab}, d_{ij}]来产生TS[$d_{ab}, d_{i⊞1,j}$]；i←i+1 然后 **go to** 1。1

1。8 **If** $x_{i⊞1} \notin \text{int}\,\Delta x_i x_j x_{j-1}$ **AND** $x_{j-1} \in \text{int}\ \Delta x_{i⊞1} x_i x_j$ (Case 2。3。3 in Fig。 10)

then 从中任意选择一个，添加(a) $[x_j, x_{i⊞1}]$ 或(b)$[x_{j-1}, x_i]$到 TS[d_{ab}, d_{ij}]。 **If** (a) 被选择 **then** i←i+1；**If** (b) 被选择 **then** j←j-1。 **Go to** 1。1。

1.9 **If** x_{j-1}, x_j, x_i 是一个右旋 **AND** $x_{i⊞1}, x_i, x_j$ 是一个左旋 (Case 2。4 in Fig。 10)

then：

1.10 **If** CH[$x_{i⊞1},...,x_{j-1}$]与 d_{ij} 相交(Case 2。4。1 in Fig。 10) **then** 在 TS[d_{ab}, d_{ij}]

中确定最深的三角形 $\Delta x_r x_s x_l$ (see Fig。 11)，其中 $\Delta x_r x_s x_l$ 包含了 CH[$x_{i⊞1},...,x_{j-1}$] 中的顶点； (b) 确定 CH[$x_{i⊞1},...,x_{j-1}$]中的顶点 x_k，使 x_k 相对于 x_r 和 x_s 可见；（c）将对角线 $d_{sk} = [x_s, x_k]$ 和 $d_{kr} = [x_k, x_r]$ 插入到 T(P) 和 D 中；（d）删除 TS[d_{ab}, d_{ij}]中从 d_{ij}（包括 d_{ij}）到 d_{lr}（包括 d_{lr}）的所有对角线，将 TS[d_{ab}, d_{ij}]中所有保留下来的对角线插入到 T(P)，然后 **Exit**。

1.11 **Else** (Case 2。4。2 in Fig。 10) (a) 从CH[$x_{i⊞1},...,x_{j-1}$]中找出一个顶点 x_k，使

它关于 x_i 可见且位于通过 x_i和x_j 的有向直线的右部(见 Fig。12) (b) **If** x_k 也关于 x_j 可见 **then** 将对角线 $d_{ik} = [x_i, x_k]$ 和 $d_{kj} = [x_k, x_j]$ 插入到 T(P) 和 D 中，然后 **Exit**。 **Else** (c) 找到一个顶点 $x_{k^*} \in$CH[$x_k,...,x_j$]，这个顶点位于 $\Delta x_i x_k x_j$ 的内部且使$\angle x_{k^*} x_j x_i$ 的角度取到最小值；将对角

线 $d_{ik^*}=[x_i,x_{k^*}]$ 和 $d_{k^*j}=[x_{k^*},x_j]$ 插入到 T(P) 和 D 中。

End While

End While

End

3．3 一些奇异情况（特殊情况）处理

比如，输入的合法性，包括输入是一个线性的点集。输入是一个自交的点集。输入是只有两个点的集合，输入中有重合的点等等。

计算的简化，比如判断一个点是否在一个简单多边形内的射线象限法，等等。这将在具体的实际编程中得到体现。

3．4 一些说明：

(1) 已经有人（Thomas Shermer）在发展本文所提出的多边形形状复杂度的衡量标准，在此基础上定义了 $T\min, T\max$，Tmin=min{ t_0 }，Tmax=max{ t_0 }。并在理论上证明了 Tmax 能够在 O(T(n))内计算出来，其中 T(n)是任何一个三角剖分的计算复杂度。他还证明了 Tmin 能够在 $O(n^3)$时间,$O(n^2)$的空间内完成. 这样，我们要是能够实现求 Tmin 的算法，我们就能在模式识别和其他领域内推广他的应用。

(2) 我们能不能暂时不考虑复杂度，但使得多边形的三角化的形状最好。要解决这个问题，首先得解决什么叫三角化的形状最好的的问题。我们初步引出这个概念：三角化形状因子。

定义 1：三角形的形状因子。

一个三角形的形状因子 α，就是内角两两之差的平方的总和。即

$$\alpha=\sum_{i=1}^{3}\sum_{j=i+1}^{3}(\vartheta_i-\vartheta_j)^2$$

其中，θ_i,三角形的三个内角,$i=1,2,3$

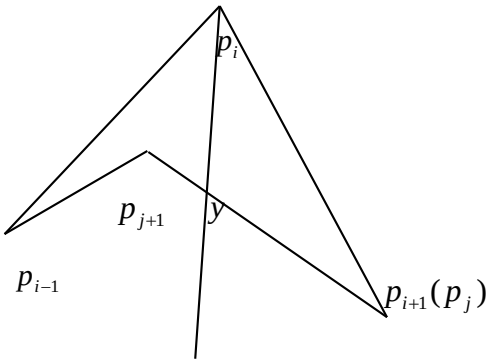
定义 2：一个简单多边形的某个三角化的形状因子，就是对于该三角化得到的所有的三角形的形状因子的总和。

定义 3：一个简单多边形的三角化形状因子，就是这个简单多边形的所有的三

角化的形状因子的最小的一个。我们称达到三角化形状因子的（可以多个）那个或那些三角为这个简单多边形的最优三角化。

4 一点遗憾：

Godfried Toussaint 在证明引理之前，曾经指出了一些别人的错误的证明，但是，我们在这里遗憾地指出，他在这篇文章引用的“正确”证明还是同样的有遗漏。我们举出反例如下。



这个反例说明，根据这个方法得到的对角线在这种情况下不是真正的内对角线，而是原来的多边形的一条边。

不过，依据这种方法找对角线，并不能影响后面程序的执行，算法复杂度也不受到本质的影响。

但是，这个结论确实是正确的，我们在附录中给出我们的证明。

6 结束语：

一般说来，每个方法都有其局限性。但是，他们还是可以进行比较的。本文的方法在作者看来是比较快的 ，并且在好几个类型的多边形剖分中能达到线性。我们的基本目的就是通过自己实现这个算法，并从中感受这个算法的特点，进一步加强自己对这方面的了解。

我们当时还考虑这个问题，如何将多边形三角化的好看些，一个应用背景就是在做计算机视觉的过程中，将人脸的三角化的好坏直接影响了整个 morphing 的过程。当我们想搞三

角化的形状最好的问题并查资料时，我们发现，这个问题已经是被人讨论和解决的问题。所以我们就没有做下去。因为，我们也没有继续做下去的条件。

由于各方面的因素，我们很遗憾地说，我们现在还没能实现求 Tmin 的算法。

最后，真诚的感谢我们的任课老师邓俊辉教授。

附录：

引理 1 的重新证明：

(1)任何简单 n $(n \geq 4)$ 多边形都存在一个凸点。否则，多边形的每个内角都不小于 180 度。则内 角和不小于 180*n 度。矛盾。

(2)选取一个凸点。记它的序号为 I， 和它相邻的点的序号分别是 I-1，I+1。j 我们将这三个点分别记做 p_i, p_{i-1}, p_{i+1}。由这三个点组成的三角形记为 $\Delta p_i p_{i-1} p_{i+1}$。

(3)情况 1：如果 $\Delta p_i p_{i-1} p_{i+1}$ 内部(包括边界)没有这个多边形的其他顶点，那么结论 A:：线段 $Lp_{i-1} p_{i+1}$ 就是多边形的一条内对角线。这样，我们就找到了一条内对角线。

(4)情况 2：如果 $\Delta p_i p_{i-1} p_{i+1}$ 内部还有这个多边形的其他顶点，记它们为 $\{p_{k_i}\}_{i=1}^{l}$，$k_i \neq i-1, i, i+1.$ 那么，我们可以 将它们按到直线 $p_{i-1} p_{i+1}$ 的距离从大到小排序；

如果取到最大值的点只有一个，记这个点为 p_k，则结论 B：

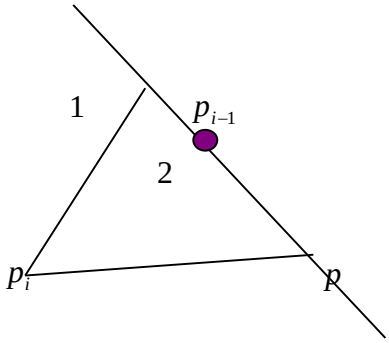
线段 $p_i p_k$ 就是一条内对角线；

否则，我们取到直线 $p_{i-1} p_i$ 距离最小的那个点，同样的记这个点为 p_k，则结论 C：

线段 $p_i p_k$ 就是一条内对角线；

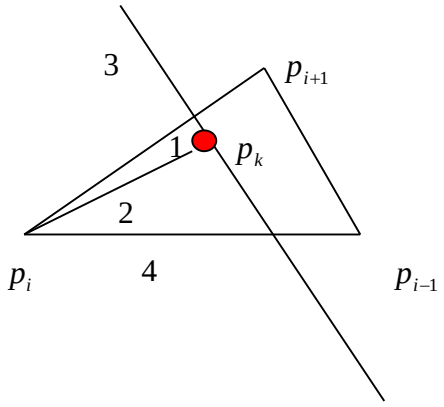
下面我们分别证明结论 A，B，C 的正确性。我们采用的都是反证法：

结论 A 的证明：



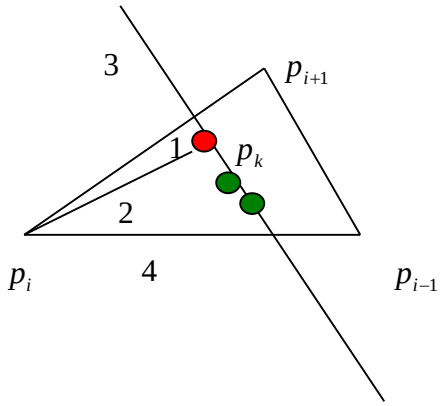
如上图，假如还有其他的边 L 和线段 $p_{i-1}p_{i+1}$ 相交，不妨设交点是 Q，则必有这条边的一个端点落在线段的一侧，在标明 1 的区域内，否则，两个端点如果落在同一侧，则不可能有交点，如果落在直线 $p_{i-1}p_{i+1}$ 上，则边必定过某个顶点 p_{i-1}或者p_{i+1}，这样，多边形不是简单多边形。这样，这条边必定是从点 Q 进入三角形内部，但是不停留在三角形内部，而进入区域 1。这意味着这条边 L 和三角形的某条边 $p_{i-1}p_i$ 或者边 $p_i p_{i+1}$ 相交，这样，多边形业不是简单多边形，矛盾。

结论 B 的证明：



如上图，假设存在某条边 L 和线段 $p_i p_k$ 相交，则必定有一个端点在区域 1，2，3，4 的并内。否则，L 的两个端点就落在线段 $p_i p_k$ 的同侧，L 不可能和线段 $p_i p_k$ 有交点。如果一个端点 Q 落在区域 1 或者 2 内，则和 p_k 是三角形内部离直线 $p_{i-1}p_{i+1}$ 最远的点矛盾。那么，点 Q 必定落在区域 3 或者 4 内。如果交点是 p_i，则这个多边形自交，不是简单多边形，矛盾。所以，Q 必定是从区域 3 或者 4 进入三角形内部。这样，边 L 必定和边 $p_{i-1}p_i$ 或者边 $p_i p_{i+1}$ 相交，这样，多边形业不是简单多边形，矛盾。所以，假设不成立。

结论 C 的证明：



我们的证明和结论 B 的证明基本相同。这里要增加的就是讨论边 L 和线段 $p_i p_k$ 可能的交点是 p_k 的时候，如果边 L 的端点在区域 3 或者 4 时，和上面的证明同样。但是，当边 L 的两个端点都在三角形内部，那么，我们根据点 p_k 是距离直线 $p_{i-1}p_i$ 最近并且和离直线 $p_{i-1}p_{i+1}$ 最远的点，不可能出现两个端点都在三角形内部的边 L 过点 p_k 。

这样，我们就证明了整个引理。