Delaunay 三角化演示程序

1 Randomized Incremental 算法

1.1 主要数据结构

1.1.1 DCEL 结构

采用 DCEL 结构保存 Delaunay 三角化过程中顶点，面，半边结构。

1.1.1.1 顶点 Vertex

保存顶点坐标、顶点对应半边以及顶点是否为无穷远点。

1.1.1.2 面 Face

保存面的对应半边。

1.1.1.3 半边 Halfedge

保存当前半边的 twinEdge，prevEdge 和 nextEdge，以及半边的源顶点、半边所对应的面。

1.1.2 Bucket

每个 bucket 放置当前已三角化的三角形内部的未三角化顶点。

1.2 算法主要流程

1.2.1 先构造 3 个无穷远点作为初始三角形，保证所有顶点都在这个三角形内部。

这 3 个无穷远点在程序中设为(0,-10000),(-10000,10000),(10000,10000)

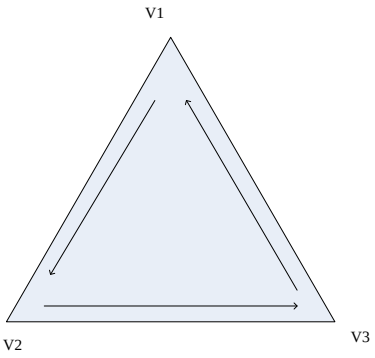
1.2.2 按顶点顺序 p1,…,pk 添加顶点。根据顶点落在哪个 bucket 中确定顶点落在哪个三角形中。加入顶点需要修改 bucket 和 Delaunay 三角化的 DCEL 结构。

1.2.2.1 修改 bucket

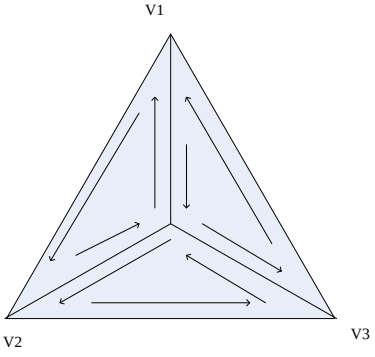
将当前顶点落入的 bucket 中所有顶点取出重新判断落在新生成的 3 个三角形的哪个中。删去原先的 bucket 加入新的 3 个 bucket。

1.2.2.2 修改 DCEL 结构

增加 1 个顶点和 3 个面，并去掉原先面



变为



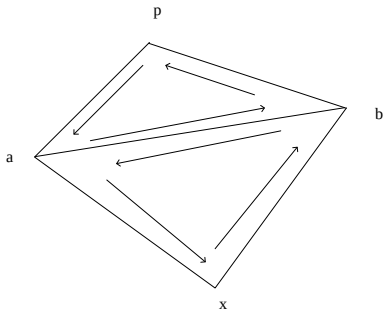
1.2.2.3 SwapTest(p,a,b,c)

判断是否需要修改边，测试主要包括 sTest(p,a,b)；sTest(p,b,c)；sTest(p,c,a)

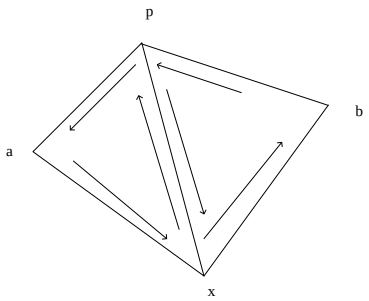
1.2.2.4 sTest(p,a,b)子程序

获得 a,b 右侧的点 x；如果 x 不存在（只有在 a,b 同为无穷远点时才会出现）则返回；判断 x 是否违反空圆条件，若违反则需要 filpEdge，并且修改 bucket 和测试 sTest(p,a,x)；sTest(p,x,b)

filpEdge 需要修改 DCEL 结构



变成



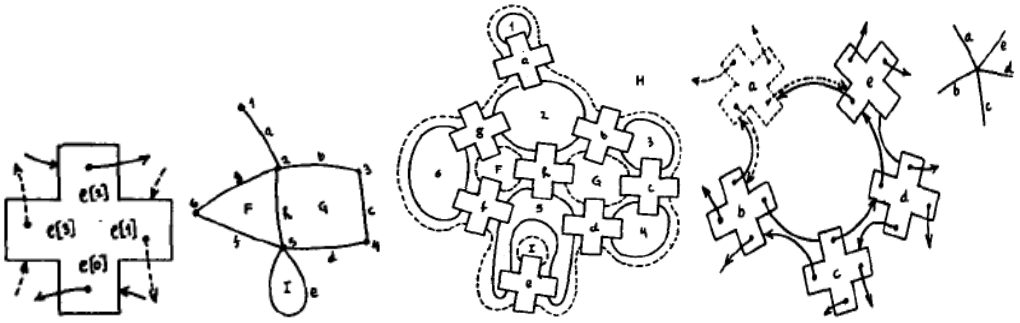
2 分治算法

2.1 算法概述

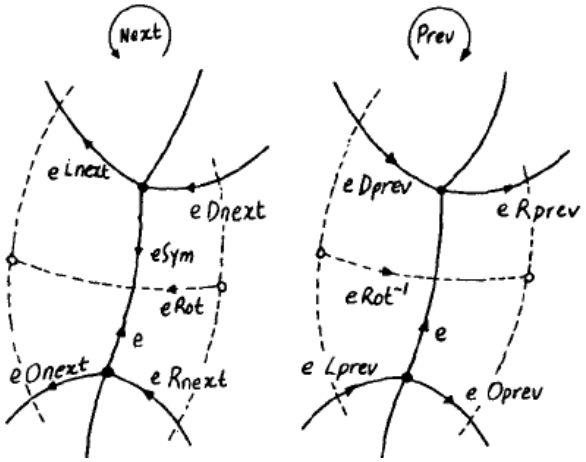
本算法来自 Leo J. Guibas and Jorge Stolfi, “Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams”, ACM Transactions on Graphics 4(2):74-123, April 1985。它提供了一个时间复杂度为 O(nlog(n))，空间复杂度为 O(n)的分治算法来解决平面点集的 Delaunay 三角化问题。

2.2 数据结构

本算法的最大特点是采用了一种独特的数据结构——QuadEdge 来表示三角化的边。在这种数据结构下，一个 QuadEdge 代表平面的图中的一条有向边，同时，此结构还记录了这条边的两个顶点的连接信息和在这条边两侧的面的连接信息。事实上，一个 QuadEdge 可以看作是如下结构：



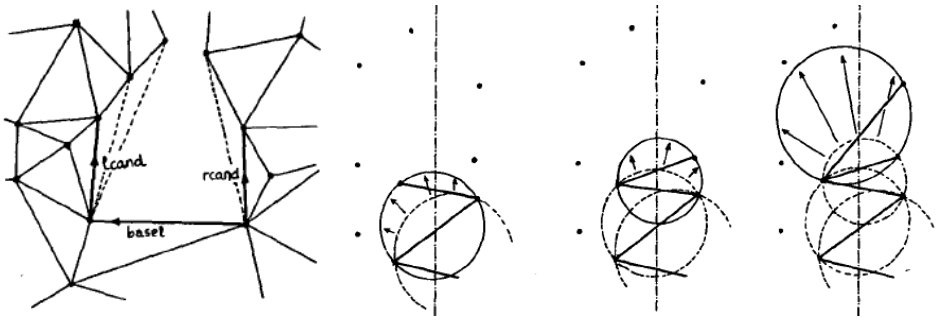
采用这样的数据组织后，与一条边相关的各种边都可以在常数时间内访问到。



实际的数据结构设计时可以只存储边的起点、边的 Rot 边，边在其起点的连接边链表中的下一条边就可以在常数时间内访问上述的全部相关边了。

2.3　算法流程

- 把给定的全部顶点按 x 坐标从小到大（相同时按 y 坐标从小到大）排序。
- 递归调用分治法处理程序，
- 如果当前要处理的顶点数目为 2，生成相应的边，返回。
- 如果当前要处理的顶点数目为 3，生成相应三角型，返回。
- 如果当前要处理的顶点数目大于等于 4，把当前顶点序列按顺序平均分为两半，分别调用分治法处理程序求解。
- 在对两半顶点序列分别 Delaunay 三角化后，寻找其底部的公切线边 e，作为第一条连接边。
- 依次测试 e 的左右邻边，删除不满足空圆条件的边。
- 从最终满足条件的左右邻边中选择满足空圆条件的边作为新的连接边 e’，另 e 为 e’。
- 如果 e’是左右两个区域的上公切线，返回。
- 否则重复此过程，生成两个区域的下一条连接边



此算法思想如下：

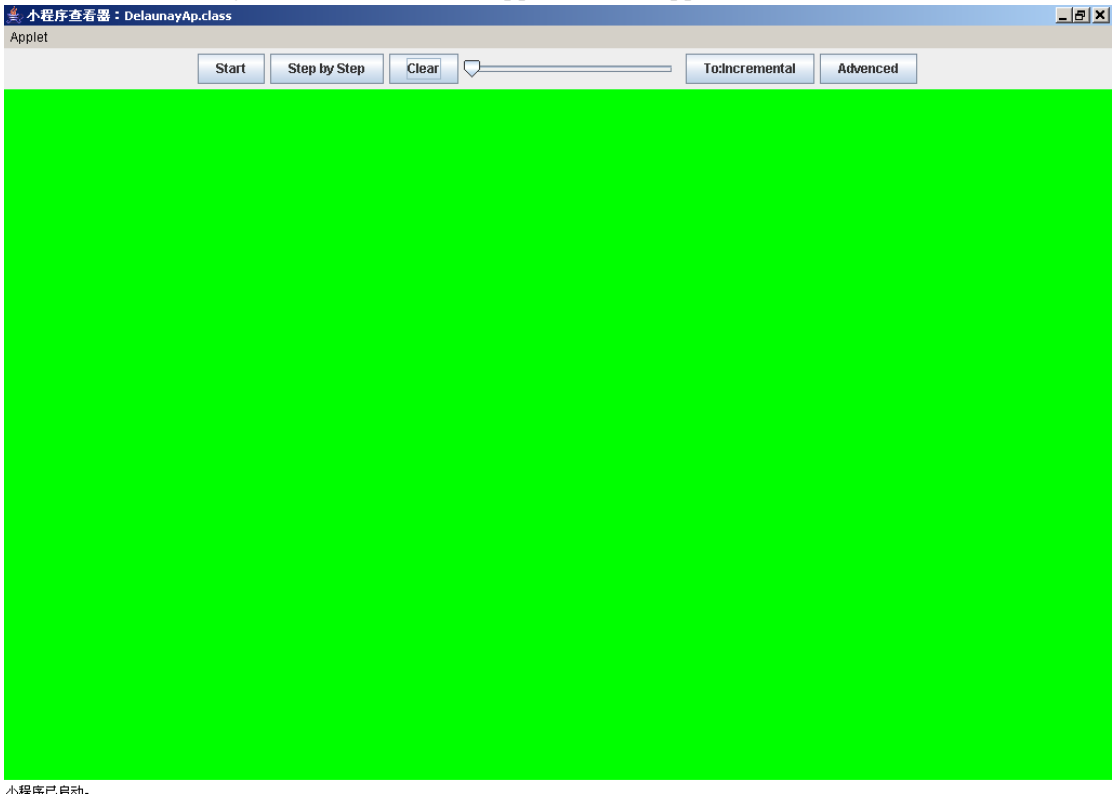
- 左右两个 Delaunay 三角化后的子区域的并的 Delaunay 过程不会再在一个子区域内添加新的边，增加的边都是跨区域的。
- 两个区域底部的公切线是一条 Delaunay 三角化中的边
- 合并左右两个区域时增加的边与两个区域的分割线的交点按其连接顺序排列
- 固定一条连接边后，招下一个连接边的过程可以看作一个圆向上膨胀直到被某个顶点限制的过程。（文中称为 buble rising）

2.4 效率分析

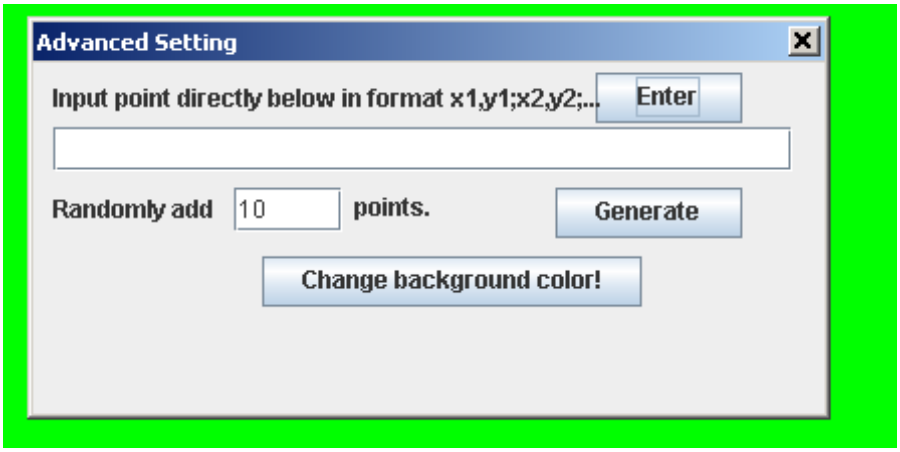
上述算法是标准的分治算法，其时间复杂度取决于其合并步骤的复杂度。上述的合并步骤中，采用 QuadEdge 结构后，在实现时采用一些技巧可以使寻找两个子区域底部的公切线过程与后续的处理过程独立且其边访问次数为 O(n)。而后面的删除边和添加边的数目也是 O(n)，而算法的初始化排序的过程其复杂度也为 O(n)，所以整个过程的时间复杂度为 O(nlogn)。而此算法的计算过程中只生成和存储了顶点和 QuadEdge 的信息，顶点和 QuadEdge 需要的存储空间都是常数，而整个平面图的顶点数目和边数目都是 O(n)，所以此算法的空间复杂度为 O(n)。

3 程序使用说明

本演示程序用 java 编写，可以作为 application 或 applet 运行。启动后本程序界面如下：



直接在绿色部分点击鼠标就可以输入需要剖分的顶点，也可以点击“Advanced”按钮，在弹出的对话框中以“x1,y1;x2,y2;…”的格式输入多个点的坐标（坐标只接受整数值），或者随机生成指定数目的顶点。此对话框还可以设置界面的背景颜色。



输入完需要的点后，可以点击“To:Incremental”/“To:Divide&Conquer”按钮切换演示的算法，按钮显示“To:Incremental”时使用的是分治算法，否则就使用增量算法。

确定算法后可以点击“Start”按钮让程序自动演示三角化的过程，也可以点击“Step by Step”按钮手工观看这个过程。拉动按钮旁的滚动条就可以控制演示动画的速度。在自动演示过程中可以点击“Pause”按钮暂停，再按“Continue”按钮就可以继续执行。执行过程中可以按“Clear”按钮直接停止执行，并清除输入的顶点。执行结束后，点击“Restart”按钮就可以重新开始执行过程。在增量法执行结束后，再在界面上输入顶点还可以自动继续增量执行下去。