计算几何课程结题报告

-Streaming computation of Delaunay Triangulations

李勇　　张楠　　武严严

1 选题背景

随着技术的进步，出现了很多大规模的网格模型，比如在有的地形模拟系统中，我们需要对几 GB 的数据进行 Delaunay Triangulation,这时由于普通计算机内存的限制我们就不能采用普通的算法来进行 Delaunay Triangulation。类似于这种问题不仅出现在 Delaunay Triangulation 中而且还出现在很多其他的计算几何和图形学问题中，尤其是针对大型三维网格模型的相关操作，一般称之为 out-of-core 算法。

此类算法有三种分类，一种是采用分治的办法，但是分治往往需要对于分割的边界进行复杂的处理。第二种方法是基于使用外存的办法。第三种办法是对原始数据进行预处理。我们选择的大作业是基于文献[1]，就方法而言应归属于第三种方法，根据文献所述应是现在对于大规模数据最有效的 Delaunay Triangulation 方法。

2 实验目标

实现普通的递增的二维 Delaunay Trinagulation 算法
实现用于大规模数据处理的 Streaming Dealaunay Triangulation 算法

3 数据结构及算法

算法部分描述请参看 proposal 文档。

点数据结构

```
class vertex2d
{
public:
    double x;
    double y;
    int index;//在该程序中没有大的用处实际上可以保存为该点所处文件的行的行号
};
```

边数据结构

```
class triangle
{
public:
    list<vertex2d>::iterator  i;
    list<vertex2d>::iterator  j;
```

```
    list<vertex2d>::iterator  k;
    list<triangle>::iterator  ti; //与该三角形公用节点i对应的边的三角形
    list<triangle>::iterator  tj;
    list<triangle>::iterator  tk;
};
list<vertex2d> ver;//内存中的点列表
list<triangle> tri; //内存中的三角形链表
```

批注 [J1]:
如此保存的三角划分，只能用于此类流显示或流绘制的场合

递增的二维 Delaunay Trinagulation 算法：

1 设立 3 个辅助点
2 从文件读入点 d
3 点定位，定位到所属三角形 abc
4 Connect(a,d) Connect(b,d) Connect(c,d) //对应程序 split3()函数
5 Swaptest(Δabd) //对应程序中的 swaptest 函数
Swaptest(Δacd)
Swaptest(Δbcd)
如果检测发现边需要翻转则调用的是 split2()函数,翻转完毕继续对新的关于 d 的三角形进行 swaptest.
6 插入调整完所有点显示出来结果

Streaming Delaunay Trinagulation 算法

1 读输入文件扫描出点集的包围矩形，并对该矩形进行划分
2 读输入文件确定每个划分区域中点的数目
3 读输入文件并重现调整格式以输出，输出格式如下：

```
-98 100 99 -98 3    //包围矩形划分相关信息
100
i -60 -78        //一个普通的点
i -89 -38
i -45 -43
i -64 -60
i -88 -97
i -85 -71
f 7                    //区域 7 所有的点已输出
```

4 设立 3 个辅助点
5 从输入文件读入矩形相关数据
6 读入标志如果是点 d
点定位，定位到所属三角形 abc
Connect(a,d) Connect(b,d) Connect(c,d) //对应程序 split3()函数
Swaptest(Δabd) //对应程序中的 swaptest 函数
Swaptest(Δacd)
Swaptest(Δbcd)
如果检测发现边需要翻转则调用的是 split2()函数,翻转完毕继续对新的关于 d 的三角形进行 swaptest.

7　读入标志如果是 f 区域终结标志
扫描内存中的三角形判断其外接圆是否于未输入区域相交，如相交在保存在内存中不输出，如不相交则从内存中删除该三角形并输出到屏幕。　//对应程序 crtest()
8 读完则处理完毕

4 效果及效率分析与测试

首先 Streaming Delaunay Trinagulation 和普通的递增算法得到的最终结果是一样的，其次从单区域执行的显示结果上看来，跨未输入区域的三角形并没有被输出出来说明结果也是正确的。

该种方法比普通的方法具有两个优点：
1 内存中保存的三角形数目不会很多，因为可输出的三角形已经都被输出了，内存中保存的只是和划分区域相交且该区域还没有被读入到内存中的那些三角形。这个特点对于大规模数据集的 Delaunay triangulation 有很大的好处，因为这情况下普通的 Delaunay triangulation 往往由于内存的不够而不能很好的工作。
2 可以提前看到部分剖分结果

5　用户手册：

普通递增的 Delaunay triangulation 算法，直接点击任务栏操作一项里的 Delaunay triangulation,然后选择要输入的原始点文件，可以采用 i.100 或 i.10,屏幕上就会显示出剖分完的结果。
如果要观看一步步显示，就选择 Delaunay triangulation 单步，这样就会得到一个单步显示的过程，每一步结束都会有对话框您可以选择确定那样就是继续单步也可以选择取消那样就可以看到最终结果。

Streaming Delaunay triangulation 算法，首要进行预处理，选择任务栏的预处理选项，然后选择要输入的原始点文件，经过处理后的文件保存在 process.txt 中，然后点击 stream DT, 就会得到一个流处理的剖分过程，其中每输入一个区域我们会弹出一个对话框终止程序一下，最终得到一个所有的处理结果。如果要观看一步一步地显示可以点击 stream DT 单步，这里面可输出的三角形我们用黄色表示，还要保留在内存中的三角形我们用白色表示。

[1]Martin Isenburg, Yuanxin Liu, Jonathan Shewchuk, and Jack Snoeyink, Streaming Computation of Delaunay Triangulations, ACM Transactions on Graphics 25(3):1049-1056, July 2006. Special issue on Proceedings of ACM SIGGRAPH 2006.

[2]AMENTA, N., CHOI, S., AND ROTE, G. 2003. Incremental constructions con BRIO. In Proceedings of the Nineteenth Annual Symposium on Computational Geometry, Association for Computing Machinery, San Diego, California, 211–219.

批注 [J2]: 提交的测试数据规模太小，无法验证吞吐能力

批注 [J3]: 提交的是 Debug 版本，最好用 Release 版本针对时间做一测试

批注 [J4]: 系统流程控制不甚方便，中途无法切换操作或中止

批注 [J5]: 相对于首次检查时，更加直观