Point Location：Trapezoidal Map

吴家兴	夏春鸿	张穗云
2013213506	2013213510	2013311925
软件学院	软件学院	软件学院
wu_jiaxing@foxmail.com	xiachunhong7@gmail.com	zhsuiy@gmail.com

1. 简介

点定位是计算几何中的一个经典问题，它被广泛地运用在计算几何、地学信息系统（GIS）、计算机辅助图形设计（CAD）等领域[1]。其中，最常见的点定位应用是，在计算机屏幕上，用户鼠标点击了一个点，那么要进行接下来的处理就一定要先知道所点的位置。总的来说，点定位的问题可以归结为，给定一个查询点，要返回这个查询点是否在某区域的内部，外部还是在区域边界上。

关于点定位的方法很多，本文主要介绍的是通过构建梯形图来实现点定位的算法。输入一系列互不相交的线段，构造一个叶子节点为梯形的有向无环图(DAG)，最后给定一个查询点，算法会返回该查询点所在的梯形，从而完成定位过程。

2. 功能描述

2.1 构造过程

我们实现了一个点定位算法[2]，可以根据用户输入的一组线段，完成梯形图的构造，同时支持构造过程中的回退、前进、清空功能。同时，再构建图的过程中，我们也同时生成了梯形图所对应的可视化图形，可以通过“显示 DAG”按钮进行查看。

2.2 查询过程

将程序切换为查询状态后，我们能够根据用户给定的查询点，返回该点所在的梯形，并将对应的梯形进行高亮显示。勾选复选框“显示动画”，还可以查看查询 DAG 的过程。

2.3 其他功能

状态栏记录了当前的边数、放大倍数、构建梯形图的时间和搜索时间的信息，可以方便用户随时查看。此外，为了方便测试，我们还支持放缩加速/正常的切换。

3. 数据结构

3.1 梯形

在实现过程中，梯形的数据结构用上边、下边、左点和右点四个量来唯一标识一个梯形，而不是用梯形的四个点来唯一标识，这样便于梯形的拆分和合并，不需要进行额外的计算，避免浮点数的计算。

3.2 查找树

查找树采用双向链表的结构，树节点有点节点、边节点和梯形节点三种类型，这三种类型的节点都继承了 Node 类。同时，节点类保存他的父节点列表，因为一个梯形节点可能被共享，便于进行更新替换。

3.3 平面划分

平面区域由梯形组成，在实现过程中我们将平面区域的所有梯形用 List 存储，List 中保存每个梯形的指针。

4. 算法

4.1 算法实现

4.1.1 梯形的拆分

梯形的拆分分两种情况：线段的端点落在梯形内部和线段穿过梯形。

对于第一种情况，如果线段的一个端点落在某个梯形内部，如图 2 所示，我们将图 1 的梯形拆分成图 2 中的 1、2、3 三个小梯形。通过使用我们上述的梯形数据结构，图 2 中的梯形 1 可以通过将原始梯形的右顶点改成当前的端点得到，梯形 2 可以通过将原始梯形的下边改成新插入的边、将左点改成当前的端点得到，梯形 3 可以通过将原始梯形的上边改成新插入的边、将左点改成当前的端点得到。

对于第二种情况，如果线段直接穿过梯形，我们只要改变原来梯形的下边和上边就可以得到两个新的梯形。

在上述过程中，并不需要进行浮点数的计算，从而可以保证精确度。

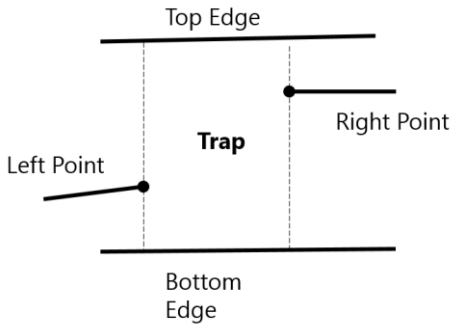


图 1

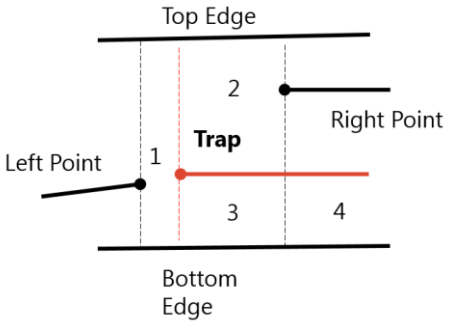


图 2

4.1.2 梯形的合并

在算法执行过程中，我们先计算出线段穿过的所有梯形，对第一个和最后一个梯形按照

上述拆分做法将其拆成三个梯形，对被拆分出第一个梯形和最后一个梯形（从左到右）单独处理，其他的梯形从左到右依次进行合并。

如下图所示，我们分别尝试合并新插入的边上方的梯形和下方的梯形。如果两个梯形的上边和下边相同，且前一个梯形的右点和下一个梯形的做点相同，则进行合并操作，如图 3 中的梯形 3 和梯形 4 即可合并成一个梯形；如果上述条件不满足，那么将上一个梯形加入到平面区域的划分中，如图 3 中的梯形 2 和梯形 5 即不能合并，继续往下尝试合并。

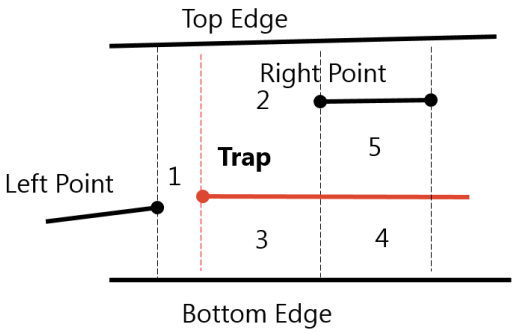


图 3

4.1.3 搜索树的更新

新插入线段和需要更新的梯形之间的关系具体包括以下三种情况：线段在梯形内部、线段从一边穿过梯形、线段穿过整个梯形，我们分情况进行了处理，具体操作参看邓老师课件[06.pl.e.TrapMap，14-16, 2013]。

4.2 复杂度分析

梯形图算法的空间复杂度为 O(n)，查询的时间复杂度的期望值为 O(logn)，具体的证明过程可以参看邓老师课件[06.pl.e.TrapMap，19-31, 2013]。

由于我们在搜索树中使用了双向链表，每个子节点保存他的父亲节点列表，由于每个梯形最多只有两个父亲节点，所以梯形图的空间复杂度还是 O(n)。具体的证明如下：

反证法：假设一个梯形被三个或三个以上的节点共享，由于被共享的梯形的父节点一定是边节点，假设即变为一个梯形被三个或三个以上的边节点共享，由于边节点的父节点一定是点节点，每条边只有两个端点，则说明该边节点端点对应的点节点至少出现了两次，与这棵树中所有点节点只出现一次相矛盾，所以梯形的父节点列表最多只有两个节点，而其他的关系都是对梯形进行替换产生的，所以所有节点的父节点最多只有两个。

5. DAG 的生成

我们使用 TrapzoidalMap 实现 PointLocation 的过程中，使用 GraphViz 类库对梯形划分画出了相应的树形结构。

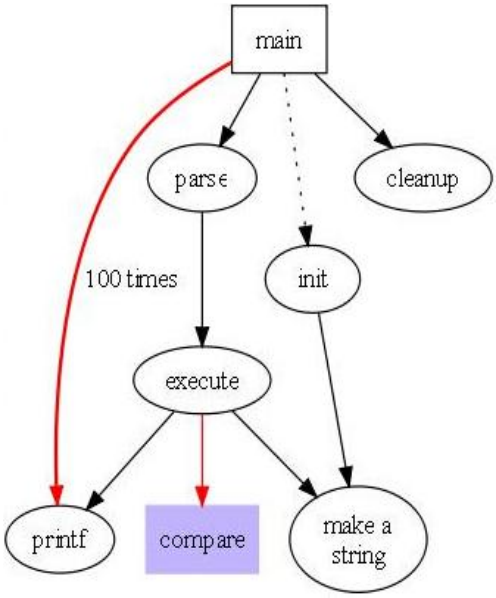
5.1 GraphViz 介绍

GraphViz 是一个由 AT&T 实验室启动的开源工具包，用于绘制 DOT 语言脚本描述的图形[3]。它是将结构化的信息转化为抽象的图形和网络的一种方式，在网络，生物信息，软件工程，数据库等方面都有很重要的应用。

GraphViz 由 DOT 语言和一组可以生成和/或处理 DOT 文件的工具组成。所谓的 DOT 语言，就是一种简单的图形描述语言，可以描述出图形的结构信息。DOT 文件处理工具可以将生成的图形转换成多种输出格式，如 PostScript ,PDF ,SVG ,PNG 等，另外对于生成的 concrete diagrams 还可以定义节点的颜色，形状，线的样式等信息。

下面为一个示例：

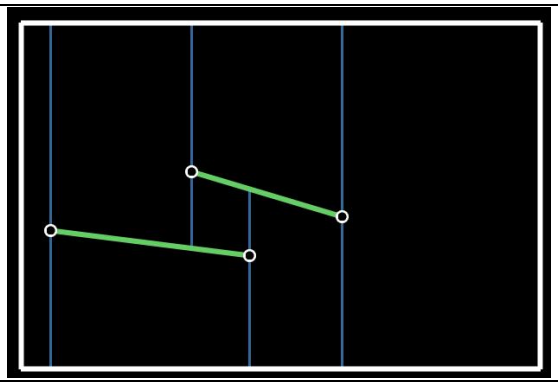
DOT 脚本	digraph G{
	size = "4, 4";
	main[shape=box];
	main->parse;
	parse->execute;
	main->init[style = dotted];
	main->cleanup;
	execute->{make_string; printf}
	init->make_string;
	edge[color = red];
	main->printf[style=bold, label="100 times"];
	make_string[label = "make a\nstring"]
	node[shape = box, style = filled, color = ".7.3 1.0"];
	execute->compare;
	}

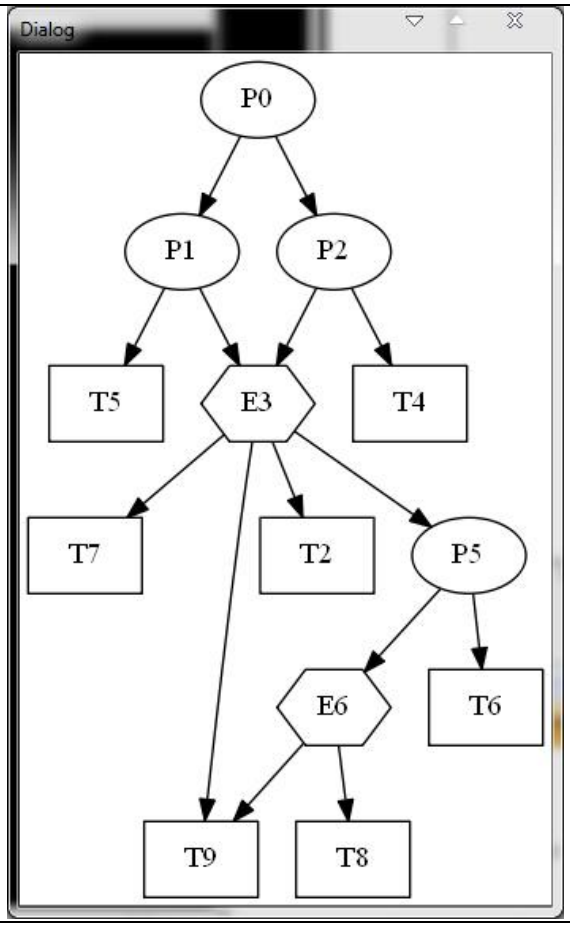
图形	

5.2 具体实现

在我们的实现中 ,我们在树形结构中定义了三种节点 ,分别为 PointNode，EdgeNode，和 TrapNode. PointNode 在图中表示为椭圆，EdgeNode 表示为正六边形，TrapNode 表示为正方形。

在我们的实现中相应的文件为：

梯形图	
DOT 脚本	*digraph G{* *P0->P1* *P0->P2* *P1->T5* *T5[shape=box]* *P1->E3* *E3[shape=polygon,sides=6]* *E3[shape=polygon,sides=6]* *E3->T7* *T7[shape=box]* *E3->T9* *T9[shape=box]* *P2->E3* *E3[shape=polygon,sides=6]* *P2->T4* *T4[shape=box]* *E3[shape=polygon,sides=6]* *E3->T2* *T2[shape=box]* *E3->P5* *P5->E6* *E6[shape=polygon,sides=6]* *P5->T6* *T6[shape=box]* *E6[shape=polygon,sides=6]* *E6->T8* *T8[shape=box]* *E6->T9* *T9[shape=box]* *}*

树结构	

6. 展望

6.1 DAG 的布局

利用 Graphviz 画出的图形，虽然连接关系上与梯形图是一一对应的，但是上下级关系却不能很好地呈现。DAG 每个节点的左右分支有特别的含义，对于顶点来说，左分支代表顶点左侧的部分 ,右分支代表顶点右侧的部分 ;对于线段来说 ,左分支代表线段上面的部分，右分支代表线段下面的部分。但是在生成的梯形图上，由于受布局的限制，同一级的兄弟节点可能并不在同一水平线上，而是分布在高低不同的位置，不能很直观地判断出节点之间的关系。

针对这一问题，可以有两种改进的方法：1.详细查看 Graphviz 的调用方法，尝试将兄弟节点安排在同一高度；2. 寻找其他的 DAG 生成工具。

6.2 节点的交互

目前实现的程序中，DAG 和主界面中的点、线、梯形是独立的，没有建立直观的对应关系。理想的结果应该是在主界面点击某点、线段、梯形，在 DAG 中对应的节点应该可以作出响应，如闪动、变色等。

7. 总结

在本次课程作业中，我们实现了一种基于梯形图的点定位算法。根据用户给定的一系列不相交的线段，程序可以生成对应的梯形图；接下来，根据用户给定的查询点，可以通过查询 DAG 返回查询点所在的梯形；最后，算法中的 DAG 也通过使用 Graphviz 工具可视化地显示了出来。

当然，算法也存在不足之处，我们可以针对 DAG 图形的生成展开进一步的工作。总的来说，通过本学期计算几何的学习，我们对计算几何的一些经典算法有了比较深刻的理解，掌握了一些无论是理论或是实践上都非常有用的解决问题的方法。

8. 引用

[1] Point Location, Wikipedia, http://en.wikipedia.org/wiki/Point_location

[2] Building of Trapezoidal Map from a set of non-intersecting lines, Jukka Kaartinen, http://cgm.cs.mcgill.ca/~athens/cs507/Projects/2002/JukkaKaartinen/

[3] Graphviz, Wikipedia, http://zh.wikipedia.org/zh/Graphviz