梯形图算法的实现

选题背景

平面排列问题是计算机很重要的组成部分，并且有着广泛的应用背景，如 GPS 等。其基本思想是利用平面上已知的一定数目的曲线排列，将平面划分为一些相互连接的单元，并满足一点的目的。这些单元可以包括点、曲线和面。利用这些单元就可以实现平面映射从而实现相关的应用。

点定位问题是平面排排列问题中比较重要而且比较基础的内容，它可以描述为：给定一定数目的曲线，通过一定的处理构建一个数据结构，这样对于给定的任何一点可以有效地找到和这个点对应的单元。

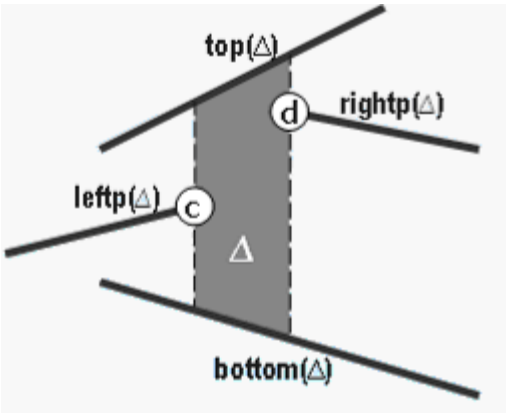
实现这个目的有一种很简单的算法就是遍历所有的边和定点，然后找到和要查询的点对应的单元。这种算法不需要任何其他的空间存储，简单方便，但是这种算法的复杂度为 O(n)，对于大规模问题来说，查询速度非常慢，而对于普通的查询来说，往往有 O(logn)的性能，算法还有很多提升的空间。

一种更有效的做法就是将搜索区域按照空间分成垂直的条带状，并对于每个带内部的进行查找，这样在 X 和 Y 方向上同时进行二分查找，可以在 O(logn)的时间内实现点的查询，只是这种算法需要的空间复杂度为 $O(n^2)$。对于这种方法，Snarnack 和 Tarjan 使用优先查找树的结构将算法的空间复杂度降低了许多。

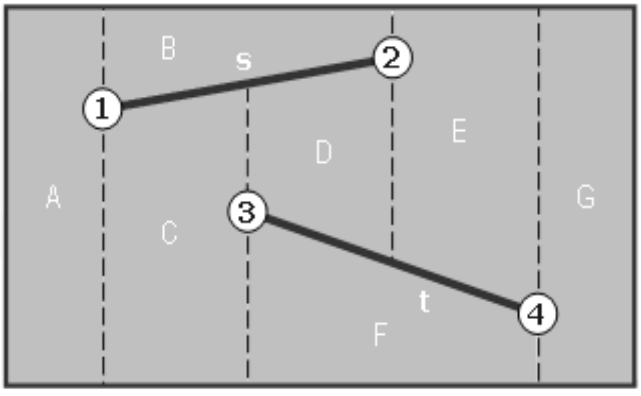
本文介绍的算法室友 Mulmuley 和 Seidel 基于随机插入构建(randomized incremental construction)思想而提出的，算法将搜索结构分解成一些竖直的梯形，并构建一个有向循环图作为查找结构，其空间复杂度为 O(n)，查询的时间复杂度的期望值为 O(logn)，在这里我们称之为梯形图算法。

算法描述

本次实验中的算法针对没有重合的线段排列中的点定位问题进行。依据线段的分布，将空间分成一些点、线段和梯形的集合，并构建与之对应的查找结构。在本算法中每个梯形的

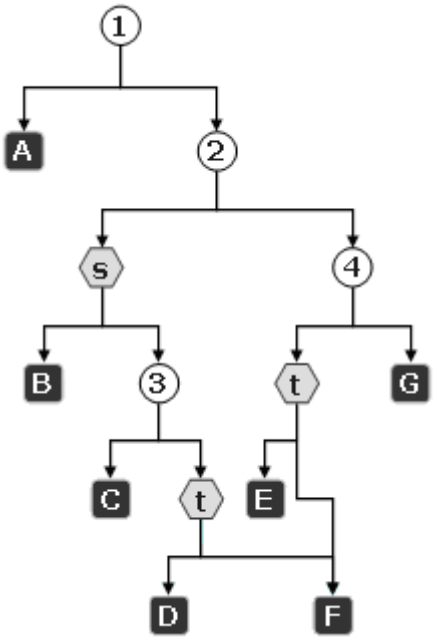


图片 1：梯形结构



图片 2：平面划分

结构如图 1 所示：有两条线段（或者是搜索区域的边界）构成了梯形的上下边，两个定点所在的竖直线段构成了梯形另外两边。整体的表面划分和搜索结构如图 2 和图 3 所示。具体的算法描述可以参见作者论文 [R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. Comput. Geom. Theory Appl., 1(1):51–64, 1991.]。



图片 3：搜索结构

算法实现

本次试验中我们在 VC6.0 框架下采用 MFC 对于梯形图算法进行实现和仿真。其中，对于平面图的划分采用梯形链接表的方式进行存储，但并辅助以一个有向循环图进行查找，算法主要包括 6 中数据结构：CSegmen，CTrapNode，CTrap，CSearchNode，SearchArcNode 和 CSearchGraph。其中 CSegment 作为辅助结构，用于存储线段，CTraNode 结构用于存储梯形的属性和指向相互邻接的左右梯形的指针；CTrap 中主要是一个存储 CTraNode 类型元素的动态链表，可以实现记录梯形图；CSearchNode、CSearchArcNode 和 CSearchGraph 用于实现有向图。其中 SearchNode 中有 4 项数值属性：一个标示结点类型的 Flag 和 3 个分可别指向 CPoint、CSegment 和 CTraNode 这 3 种数据的指针，但同时只有和 Flag 相对应的那一项非空。

算法采用逐步插入更新的方法实现。每一条线段插入后都会生成一个完整的平面划分图和查找结构。每次插入新的线段时，平面划分和相应的数据结构都要进行更新。依据新插入线段和需要更新的梯形之间的关系具体包括一下三种情况：线段在梯形内、线段从一边穿过梯形、线段贯穿梯形，分别采用不同的函数进行处理，具体操作参加邓老师课件[06.pl.e.TrapMap，12-14, 2009]。在每一步的实现过程中，要使用已经生成的查找表寻找和点对应的梯形，这样每一步的执行都要利用上一步生产的查找结构。

本算法实现过程没有利用任何现成代码，所有代码完全由自己编写完成。

技术难点和解决

在本次算法实现过程中，主要的困难存在于几个方面，对应的问题描述和以及解决方法如下。

1. 维护梯形图中拓扑关系

由于我们没有使用 DCEL 结构，而是跟据《计算几何——算法与应用》133、134 页中提示的，通过记录每个梯形左右最多四个邻居梯形指针来维护梯形图间的关系。所

以 CTrapNode 的属性有左右端点，上下线段 4 项，还有分别指向左上、左下、右上、右下 4 个邻居梯形的指针。但在实现过程中，维护起来不是很方便，很容易出错。而且因为在查找结构中的梯形也是利用 CTapNode，而在查找结构图中，其实梯形间的左右关系并没有用，所以浪费了很多空间，并且因为过多使用指针，所以很容易出错。最后在实现过程中效果很不好。

2. 退化情况的处理

这次我们的实验时间比较紧，经验不足，做的很不好，很多基本功能都没实现，所以没有处理退化的情况。

3. 查找结构的实现

查找结构属于有向无环图，并且相应的弧段有严格的左右方向，所以我们利用一种改造后的十字链表来实现查找结构，CSearchNode 的属性有：flag 表示结点的类型，可以为点、线段、梯形以及空类型四种，于此对应的有指向点、线段和梯形的三项指针，每个结点同时只有与 flag 相对应的那个指针非空;除了这些数据外,还有 3 个指针 firstIn，leftOut,rightOut 分别指向相应的弧段。因为弧段的同尾链域在这里没什么作用，而弧段本身的属于左弧段、右弧段要用到，所以我们在设计 CSearchArcNode,时是弧段类时，就去掉了同尾链域，而在 inf 域中记录弧段的类型。

由于没有经验，在后期调试时，才发现有很严重的错误——在加入弧段时，要首先在存储 CSearchNode 的数组中找到弧段的 tail、head 结点在数组中的位置，但因为查找图中的存在多个结点指向同一线段的情况，不能准确找到线段的存储位置。导致生成的弧段不正确，从而使查找图出错。这个问题刚开始没发现，在点查询时一直得不出正确结果，调试好久才发现。这个问题困扰我们很久，想了很久一直找不到有效的解决办法，最后时间不够，这个问题一直没解决，导致几乎大部分情况都不能运行成功。

4. 动态数组的使用

这次试验中，很多地方要用到数组，由于编程经验不足，在使用 MFC 的动态数组 CArray 和 CObArray 时遇到很多问题，虽然从中学习到不少经验，但很多问题处理的比较混乱，到最后导致试验完成的很不好。

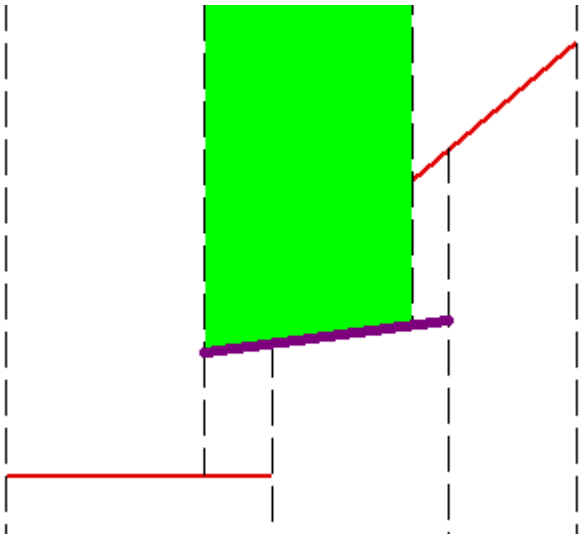
5. 随机线段生成

刚开始觉的该功能不是实验的核心内容，于是就准备放到最后实现，但因为后来一直被不能生成正确查找结构困扰，时间不够，所以该功能没有实现，只是有 2 不成熟的想法：（1）可以先生成不相交的矩形，然后在每个矩形内随机生成一条线段。

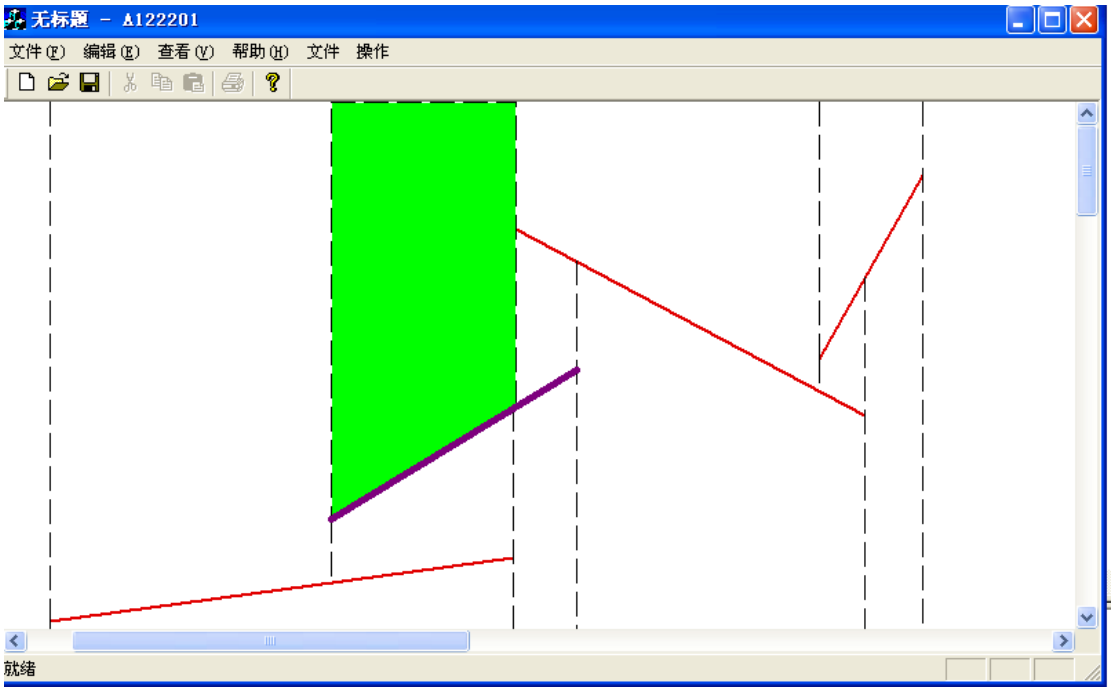
（2）随机生成点，然后对每个点找出与其余点最近距离，取它的一半做圆，可以生成平面不相交圆，每个圆内任取一条直径作为线段，则可生成随机不相交线段集。时间复杂度比较高。

实验测试

由于我们的实验做的很不好，只有很少几种简单的类型才可以实现正确结果



界面操作说明



线段为红色线条，梯形图为灰色虚线条。

1. 选择菜单栏的“文件->读取数据”选项，可从 txt 文件导入线段数据，数据的格式如下

   segmentNUM

   segment1Begin.x segment1Begin.y segment1End.x segment1End.y

   segment2Begin.x segment2Begin.y segment2End.x segment2End.y

   segment3Begin.x segment3Begin.y segment3End.x segment3End.y

   “文件->写入文件”可以实现上述格式的文件保存；

   “文件->清空数据”，可以实现清除系统数据。

   “文件->生成随机线段”，该功能没有实现。

2. “操作->手动输入线段”，实现屏幕输入线段；

   “操作->梯形剖分”实现按照输入的线段进行梯形剖分

   “操作->点查询”可以实现点查询，显出对应的梯形显绿色，线段显紫色。

   “操作->输出查询结果”，输出点查询的结果，即查询点上方第一条线段的首尾坐标。