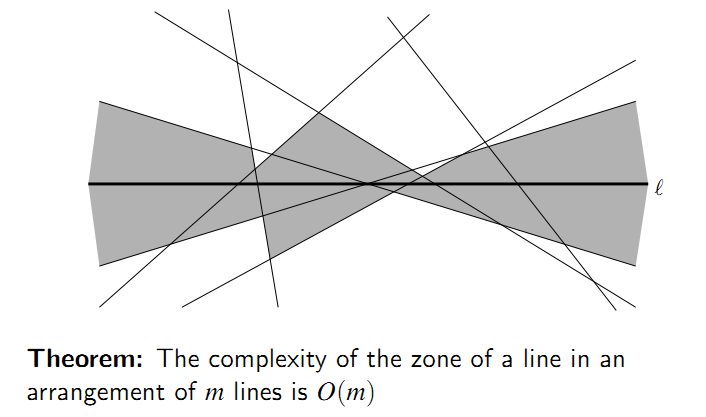
DCEL与对偶图

Doubly Connect Edge List (DCEL)是平面图的一种存储结构，对偶图Arrangement 描述了平面内点和线的对应关系。实验中将建立平面内的点、线与对偶图中的线、点之间的联系，并通过DCEL的数据存储形式来实现快速查询其他点对于一个点的极角序的算法，时间复杂度为O(n)（n为点的数量）。详细描述如下：

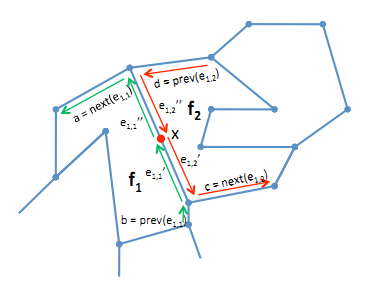
1. 对偶图

在平面内，可以将一个点P1(x1, y1)转化为对偶图中的直线x1\*X+Y-y1 = 0，将另一个点P2(x2, y2)转化为对偶图中的直线x2\*X+Y-y2 = 0，联立两个方程，解得交点坐标P(x0, y0)中x0 = (y2-y1)/(x2-x1)，即为平面内两个点的斜率（如果斜率不是无穷的话）。于是，平面内点与点之间的斜率关系就转化为对偶图中线与线交点横坐标的关系。当我们根据当前点转化成的对偶图中的直线，从“左”向“右”依次遍历整个对偶图时，根据与当前直线的交点的横坐标，即可得到一组点关于一个点的有序的极角序。而整个从“左”向“右”遍历的过程，则基于DCEL这一平面图存储结构，根据Zone Theorem，一条直线在对偶图中穿过的直线数量为O(m)，（m为直线的数量），保证了构造和查询的效率。

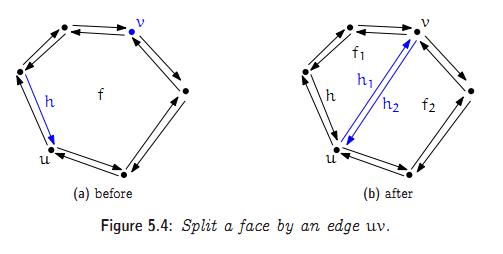


1. DCEL

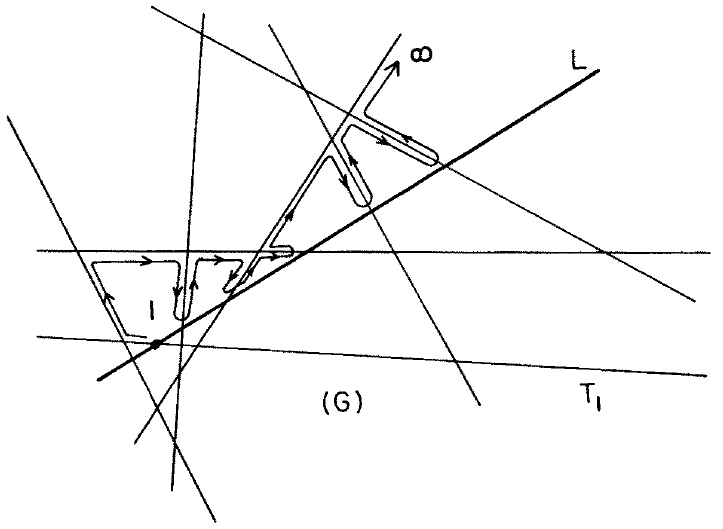
DCEL是存储平面图信息的一种数据结构， 实验中对DCEL的操作中，有两个重要的基本操作，一个是在边上增加一个顶点，即边的分割：



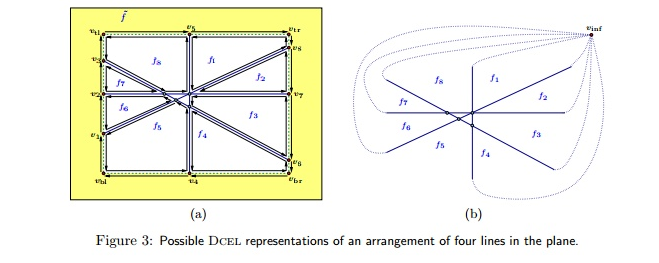
另一个是链接两个顶点，即面的分割：



两个重要的对外接口操作是在平面图中插入一条直线，以及按照从“左”向“右”查询一条直线穿过的所有的直线。查询操作与（1）中相关联，如下图所示：



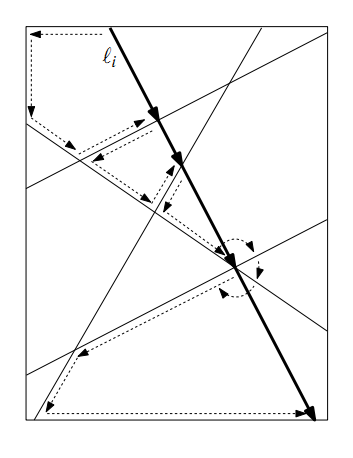
由于实验中DCEL存储的是对偶图中的直线，而直线是无边界的，故而需要特殊处理一下DCEL的存储结构，如图：



逻辑上的DCEL的形式应当为(b)图中的形式，而在程序运算过程中，为了更简单快捷地计算，会在所有直线的外围假想一个非常巨大的包围盒，包含所有直线的交点。

1. 实验流程

在预处理阶段，根据输入的地图信息，得到所有点，将所有点转化为对偶图中的直线，通过DCEL建立起平面图结构。插入和查询操作类似，过程都如下方所示：



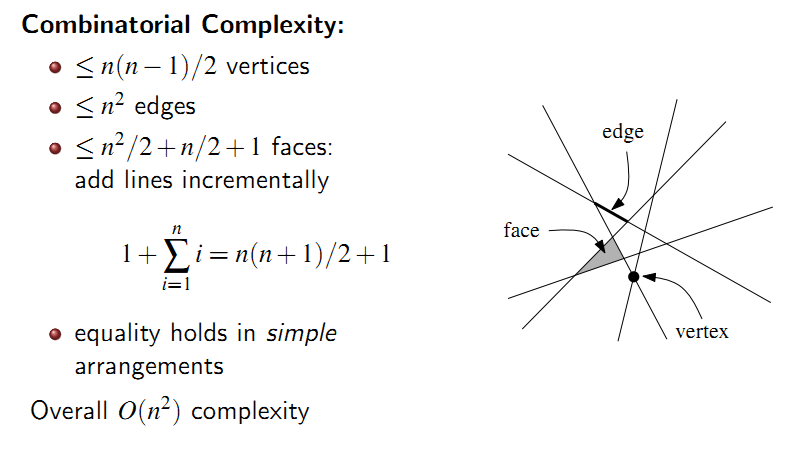
实时运行阶段，根据当前需要查询点转化为对偶图中的直线，到DCEL结构中按照横坐标序列从“左”到“右”查询所有直线，得到所有点关于当前点的斜率的关系。再根据获得的斜率关系以及点与当前要查询点之间的坐标关系来划分点的极角序为[0, PI)还是[PI, 2\*PI)。

从而，将一个斜率有序的数组转化为两个极角有序的数组，分裂后，再按照极角序，重新排列起来。

注：有些情况需要特殊处理，如在对偶图中两条直线平行的情况，对应于在原二维平面内，两个点所在直线平行于Y轴的情况。此种情况下，对于这些点的斜率的判断不会在DCEL求解过程中，而需要在外部单独计算，好在垂直情况下，极角序只有0度和PI度两种，根据计算的结果直接剥离开即可。还有就是X坐标相同时，按照Y坐标从小到大的方向去查询。

1. 实验分析

首先分析一下预处理阶段需要的时间和空间复杂度，如下图：



DCEL结构中，点、线、面都是O(n2)的空间复杂度，时间复杂度也是O(n2)。1000个点的测试输入，在DCEL存储结构中会有近50万个点，200万条边，还有50万个面。当点数达到5000时，DCEL存储结构中会有近1000万个点，4000万条边，还有1000万个面。而在DCEL中，每一个点，边，面都是一个存储结构，还会存有其他的信息，对内存空间要求很高。

预处理阶段，相对比较耗时间，考虑到平台是c++，大量的new操作会极大地降低预处理时间，我们采用了先批量new后再根据标号取用申请空间的方式来适当提高运行时速度。

运行时，考虑到此种算法空间复杂度需要O(n2)，有大量的信息常驻内存，时间复杂度为O(n)，但是常数操作比较多。而如果采用简单暴力的排序算法，在计算极角序的时候直接调用c++的std::sort的话，兴许会有不一样的效果，因为sort的时间复杂度，虽然近似为O(nlogn)，但是排序的结构非常简单，常数也十分地小，而空间复杂度，则相比于DCEL的存储，也大大的降低，并且，很多信息并不需要常驻内存。于是，在程序中，我们同时实现了使用std::sort进行的极角排序和采用对偶、DCEL来实现的极角排序。

实验通过手绘小数据，图像处理提取图片边缘转化为输入点的方式，对运行时程序进行sort和DCEL方法的运行时效率比较。结果发现，在很小规模的数据下（30点以内 ）DCEL会稍微比sort快一点，而到了几百几千的数据，DCEL与sort的速度不相上下，运行时间很小。下方是我们生成的测试数据表格：

测试数据表格

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Tiny\_map | Face\_map | Logo\_map | Thu\_map | Maze\_map | Text\_map |
| **规模类型** | **很小** | **小** | **中等** | **大** | **大** | **很大** |
| 顶点数量 | 13 | 92 | 1359 | 3519 | 4728 | 5796 |