**外扩法**

**一．输入图层**

Streets.shp：原始道路信息。

river\_erase.shp：用于在后处理进行擦除的水系图层。

test\_pt.shp：测试的点集。

**二．预处理**

（1）通过道路polyline生成polygon（streets\_polygon.shp）。

（2）由道路polygon生成coverage，得到拓扑关系（coverage文件夹）。

（3）由coverage提取具备拓扑关系的polygon（polygon.shp）和polyline（arc.shp）。

（4）空间连接，映射得到具备LANES属性的polyline（输入路径：Arc\_Join.shp）。

此部分挺快的。

**三．区域的合并**

主要有以下几个步骤：

1. 得到边和多边形的拓扑关系。这里有两个拓扑关系，一个是线的左右两端的多边形，这是直接有了的，读入进程序就可以了；第二个是多边形含有哪几个边，这个拓扑关系可以通过上一个拓扑关系推断出。这里由于方便上面的操作，建立的拓扑关系里面还包含了这条边的所属的道路等级信息。

2．通过输入的点图层和多边形图层进行空间查询操作，得到点所在的多边形作为搜索的起点。

3．由于原始多边形的属性中id均为0，需要进行一个更新操作。使从不同起点搜索的多边形id相同。

4. 迭代搜素。一条边的搜索终止条件为搜索到满足条件限制的边或者搜索到了整个区域的边界。这一步通过线的左右多边形拓扑表进行。

5. 输出搜索结果（extracted\_area.shp），并进行id的更新。

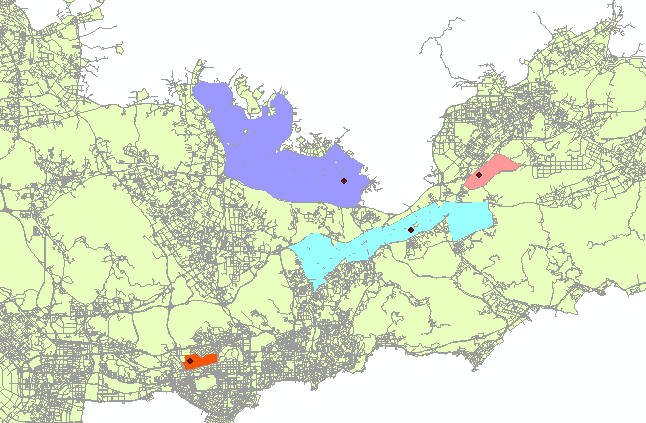
**四．后处理**

（1）图层擦除：根据输入的水系图层，对提取到的区域进行删除。

经过后处理后中的图层擦除后，得到最终结果图层extracted\_area\_erase.shp。

**五．结果与分析**

（1）输入点图层含有4个点（随机点的），不同颜色代表不同搜索区域。搜索结果大约包含4000多个小多边形（总的约37000）结果感觉不是很好，程序逻辑应该是没有问题，感觉可能是由于数据的原因造成。



总共耗时8分钟左右。绝大部分时间（至少90%）耗在了输出结果这一步，因为涉及到不断对于图层进行写操作，所以很慢。

（2）蓝色部分为水系擦除的部分。

