Voronoi 图的增量式构造与归并构造原理

Dezeming Family

2021 年 11 月 23 日

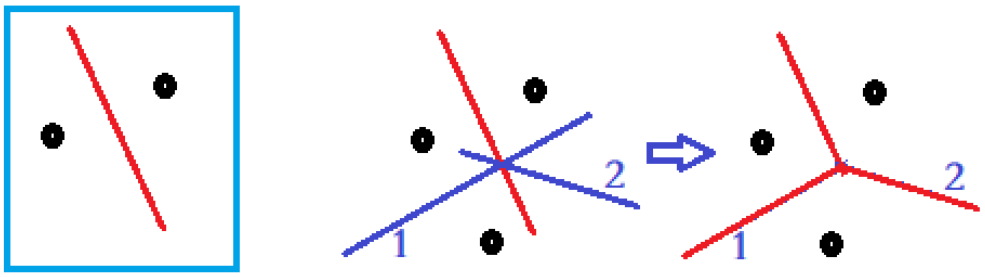
DezemingFamily 系列书和小册子因为是电子书，所以可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列书，可以从我们的网站 [https://dezeming.top/] 找到最新版。对书的内容建议和出现的错误欢迎在网站留言。

目录

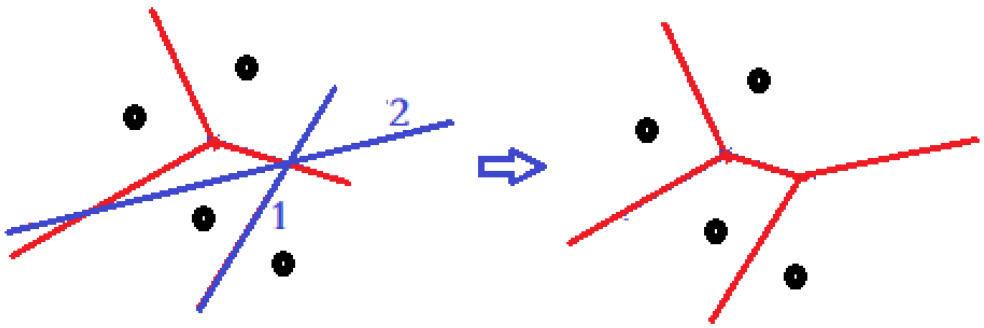
一 增量式构造方法

我们使用 DCEL 数据结构来进行访问和操作，因此可以从一个 cell 遍历上面的所有边，并且可以沿着边找到所有点以及可以访问到相邻的 cell。

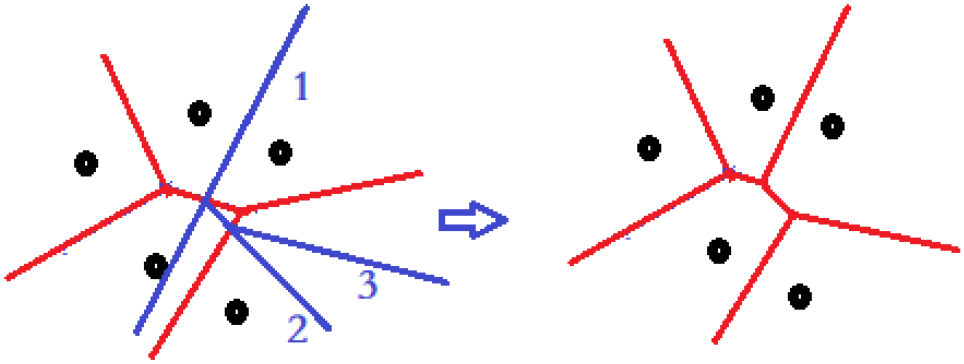
增量式构建比较简单。当有两个 site 时，就是垂直平分线。当加入第三个 site 时，找到它所属的 cell，然后做平分线；该平分线与 cell 的边相交于一个点，所以过渡到相邻的 cell，并再次做平分线。



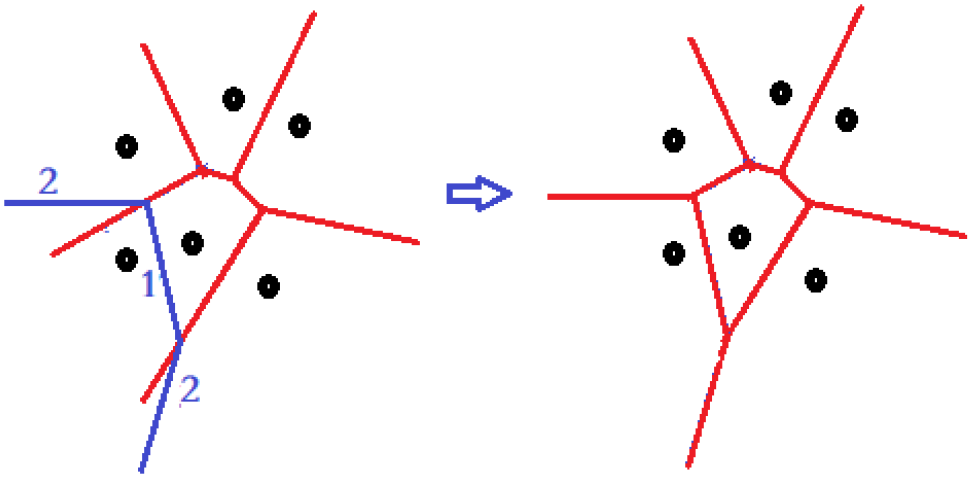
当再加入一个新的点，找到所属 cell，然后做平分线，如下图的 1 线，与当前 cell 交于一点，从而过渡到相邻 cell；然后与相邻 cell 的 site 再做平分线，这次与该 cell 并没有再交于其他点，所以划分终止。（注意只需要求当前过渡到的 cell 的交点，与其他 cell 是否存在交点无关。）



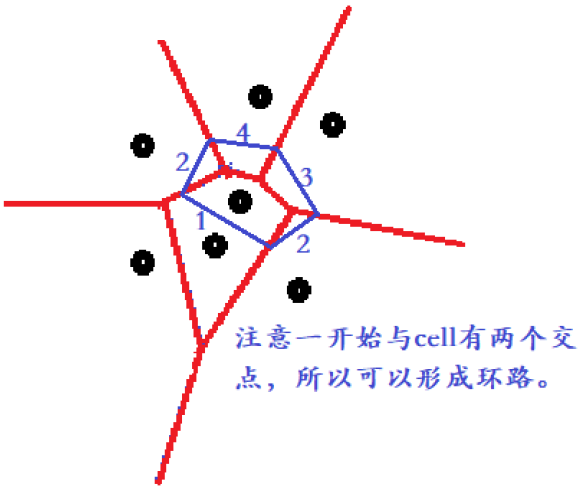
下面的构建过程也是如此，注意构建顺序：



下面的构建过程也是如此，新 site 与当前 cell 交于两个点，分别跟踪两个点到相邻 cell 做平分线：



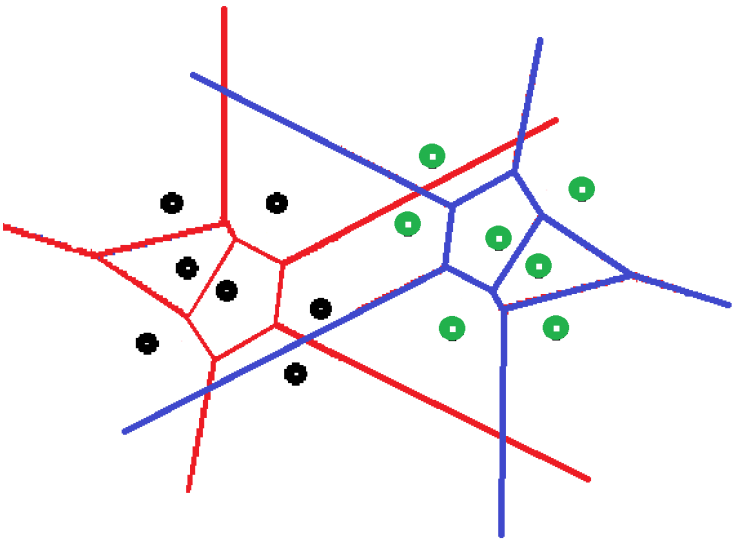
当新的 site 出现在下面的位置时，有趣的事情就发生了。由于与当前 cell 相交于两个点，因此我们可以看到它会形成一个环路。因为过程比较简单，我就不再画出新的划分了。



该算法的复杂度为 $O(n^2 \log n)$，因为一共需要增量插入 n 个 site，对于每个 site 而言，因为它有可能需要遍历周边的 cell，这个过程也算作线性的，即 $O(n)$，同时，对于周边的 cell，它需要与每个 cell 的边进行求交，考虑到凸多边形而且是有次序的（DCEL 结构），所以这个过程复杂度为 $O(\log n)$。所以总的时间复杂度是 $O(n^2 \log n)$。

二　归并法

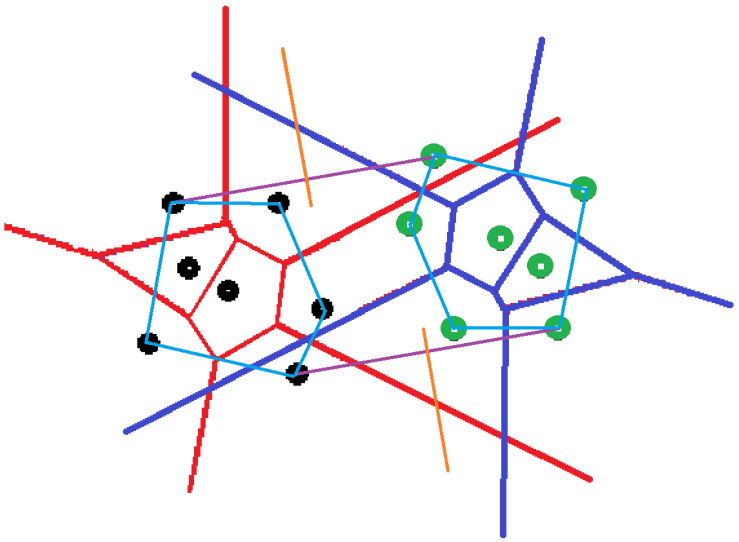
归并法是我认为一种很不直观的算法，在缺少图示过程中很难理解里面的细节。假如我们已经构建好了左右两部分，现在考虑怎么合并：



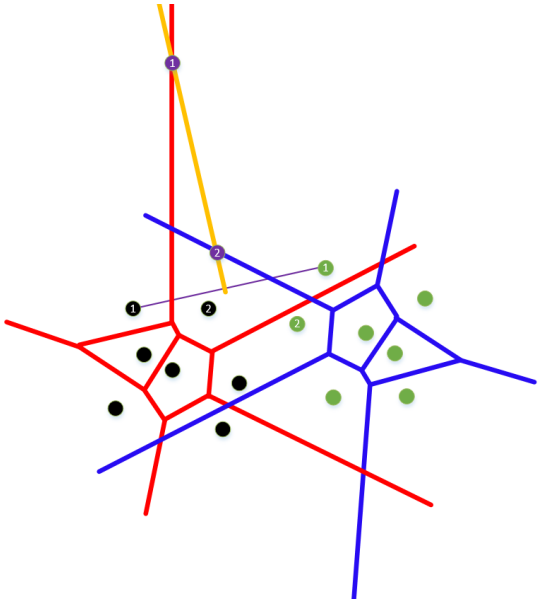
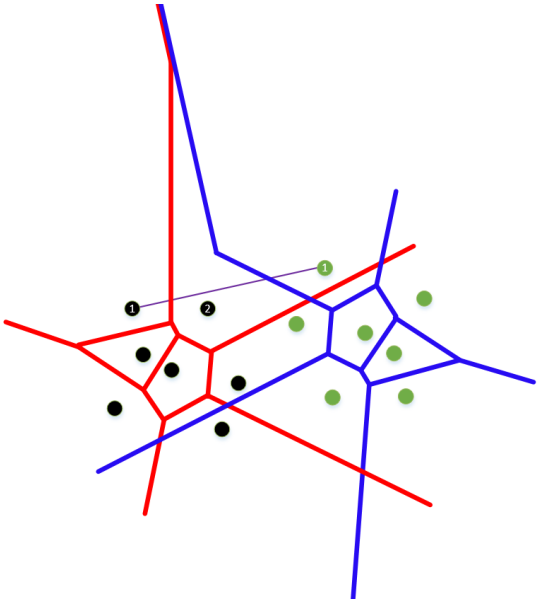
我们的目标是在合并的时候把这些冲突的 cell 给进行调节好。我们要构造一个折中线，该折中线上的任何点到左侧最近的 site 等于到右侧最近的 site。

该折中线距离左右两边的距离相等，且不会为 0，因此是单调从上往下（或者从下往上）的，不会折返，这个折中线我们称为 contour。

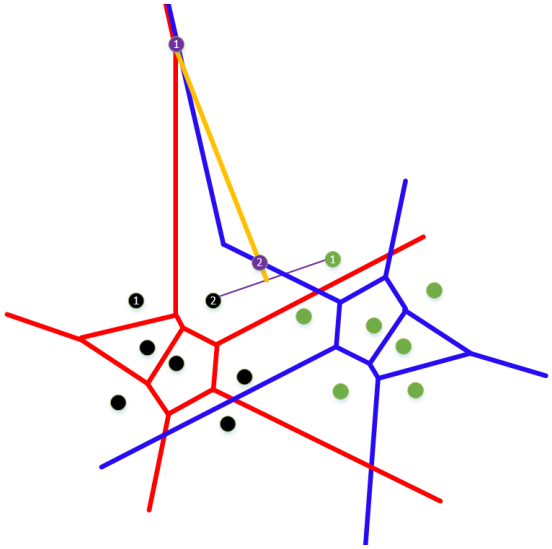
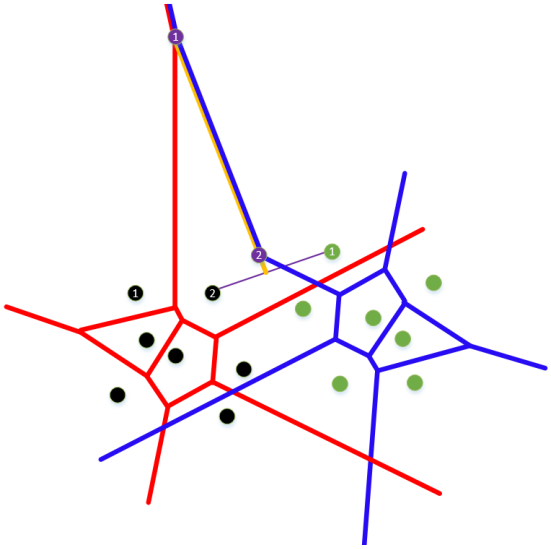
我们从上往下看，contour 的起点一定是两边的凸集的公切线的平分线（线性时间就能找到），如下图的橙色线：



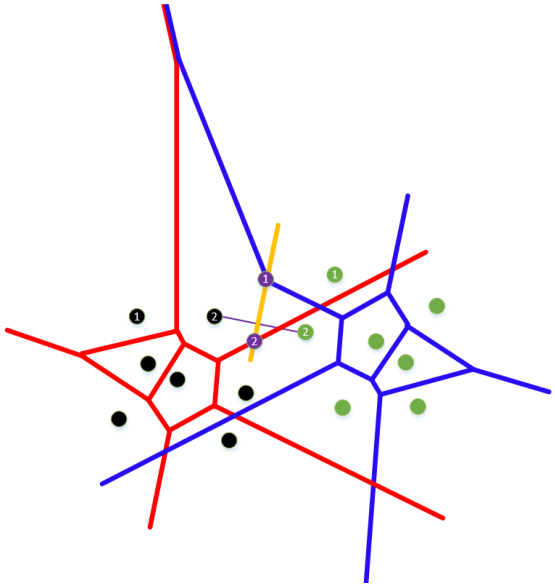
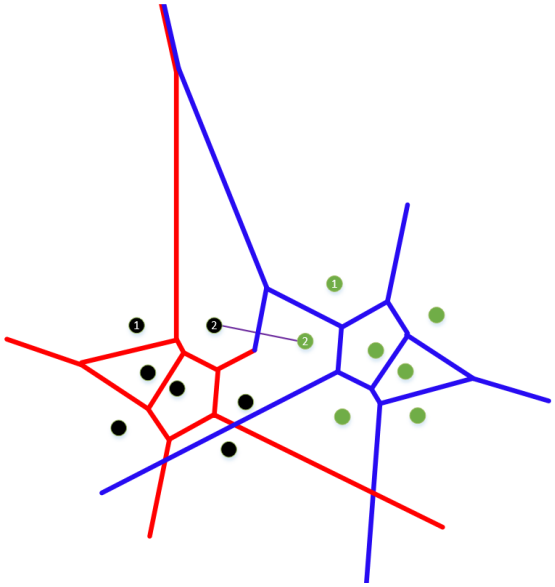
我们从这个橙色线开始，对它左右两边的 cell 进行裁切（一开始图省事没有用 visio 画图，结果没想到越画越麻烦）。这个 contour 会与两个 site 所在的 cell 求交，并分别裁切两边的 cell。因为橙线先与黑 1 所在的 cell 的边相交，所以左边翻墙到下一个 cell（即黑 2 所在的 cell）。

当前左边需要关注的 site 是黑 2，右边是绿 1。黑 2 与绿 1 做平分线，与黑 2 和绿 1 所在的 cell 边求交，并进行裁剪。注意下图的紫 1 其实是黑 1、黑 2 和绿 1 三个点共圆的圆心，即距离这三个点的距离相等。我们再次分别切割黑 2 和绿 2 所在的 site。我们不用管紫 1 这个交点，而是把紫 2 作为第一个交点，因为紫 2 是与绿 1 所在 cell 的边的交点，所以右边翻墙到下一个 cell（即绿 2 所在的 cell）。

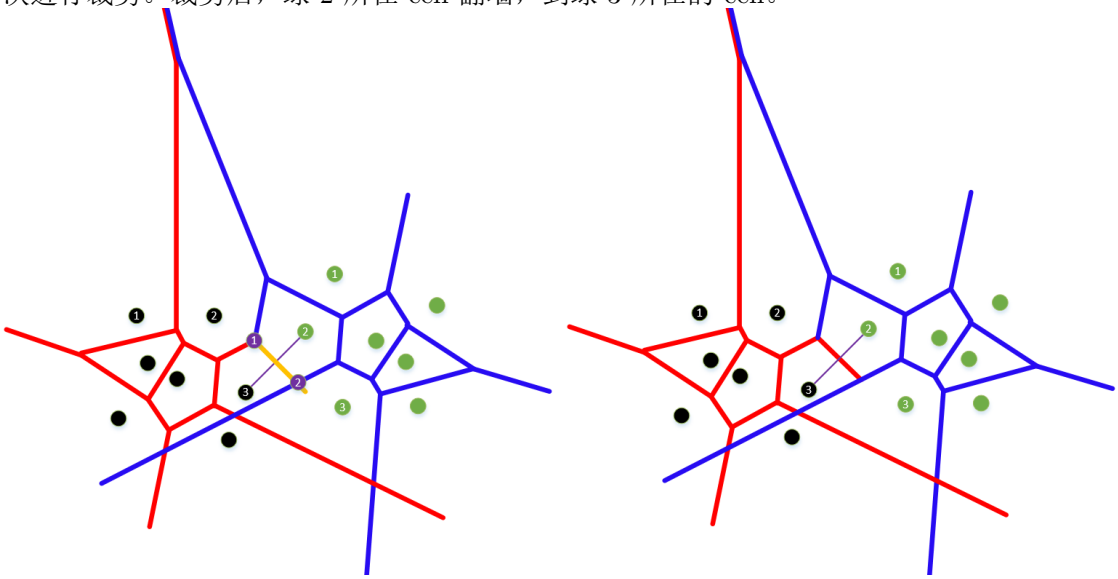



当前右边需要关注的 site 是绿 2。黑 2 与绿 2 重复上述过程，首先与红色边界交于紫 2，因此再次进行裁剪。裁剪后，黑 2 所在 cell 翻墙，到黑 3 所在的 cell。

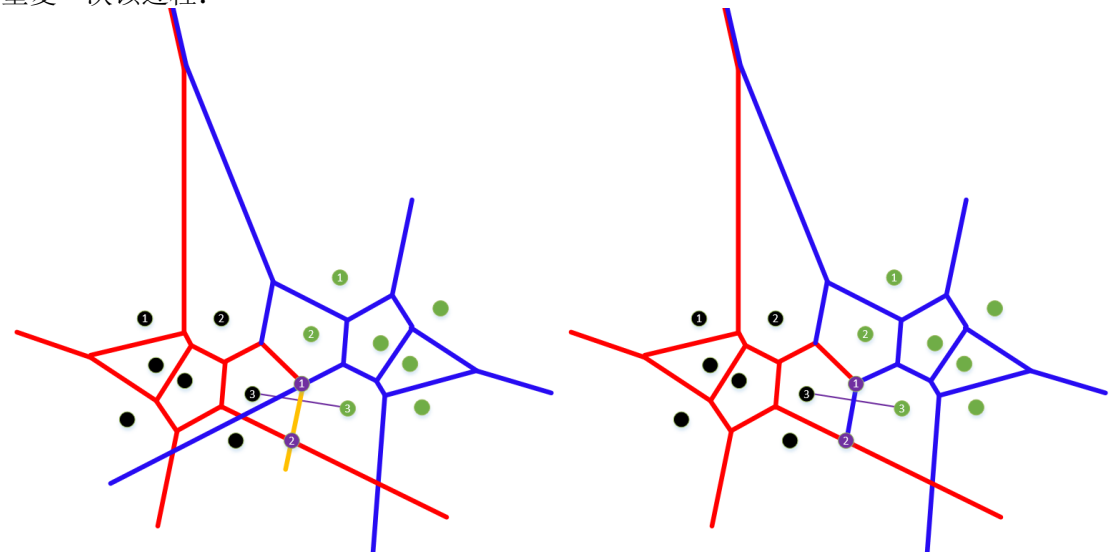



当前左边需要关注的 site 是黑 3，右边是绿 2。黑 3 与绿 2 重复上述过程，首先与红色边界交于紫 2，

因此再次进行裁剪。裁剪后，绿 2 所在 cell 翻墙，到绿 3 所在的 cell。



再重复一次该过程：

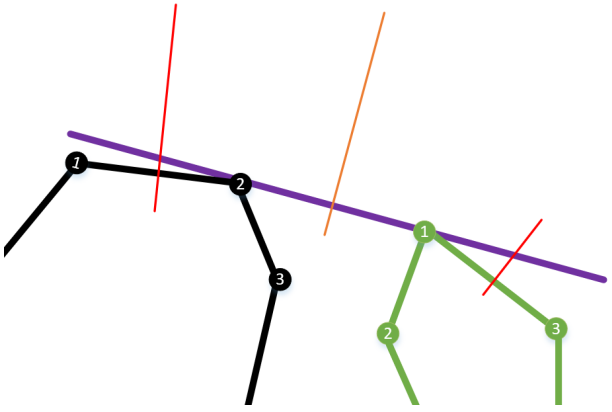


这样我们就对这整个归并方法有了清晰的认识。

三　为什么是凸集公切线

前面说过，contour 的起点一定是两边的凸集的公切线的平分线，这是为什么呢？

可以画图分析一下，不在凸集上的 site 或者在其他地方的 site 所在的 cell 不会是首先需要被裁剪的 cell，因为橙线并不会与这些 cell 相交：

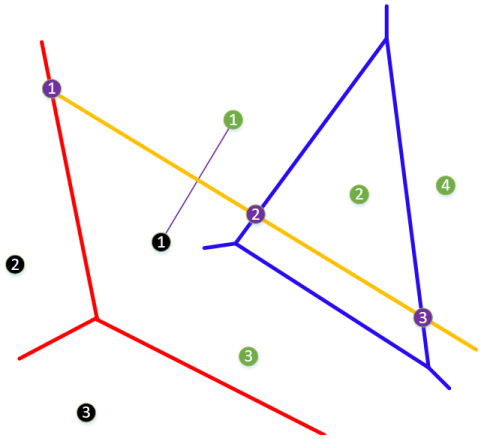


上图中，红线分别是分隔黑 1 和黑 2 所在 cell 的边界，以及分隔绿 1 和绿 3 所在 cell 的边界。橙线并不会与这些红线相交，意味着它不会改变黑 1 所在 cell 的范围，也不会改变绿 3 所在 cell 的范围。

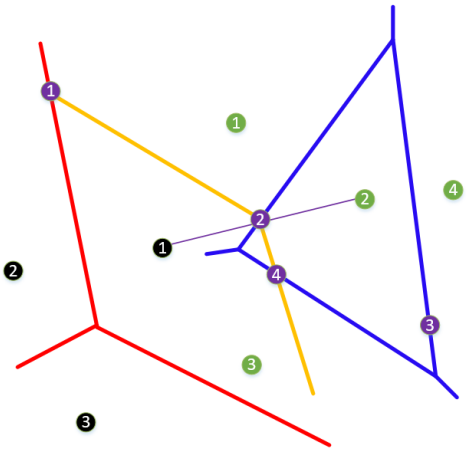
四　复杂度

关于复杂度的问题，思考一下，如果左边的 cell 在求交好几次都不会改动（没有翻墙事件发生），那么如果对面需要每次都计算与左边构建出的 cell 的所有边来求交，就看似会很浪费时间。但其实已经构建好的 contour 便是 cell 的边，而不会再被用来计算求交了。

下图中，橙色代表平分线，绿色表示构建好的 contour（新生成的 cell 边界）

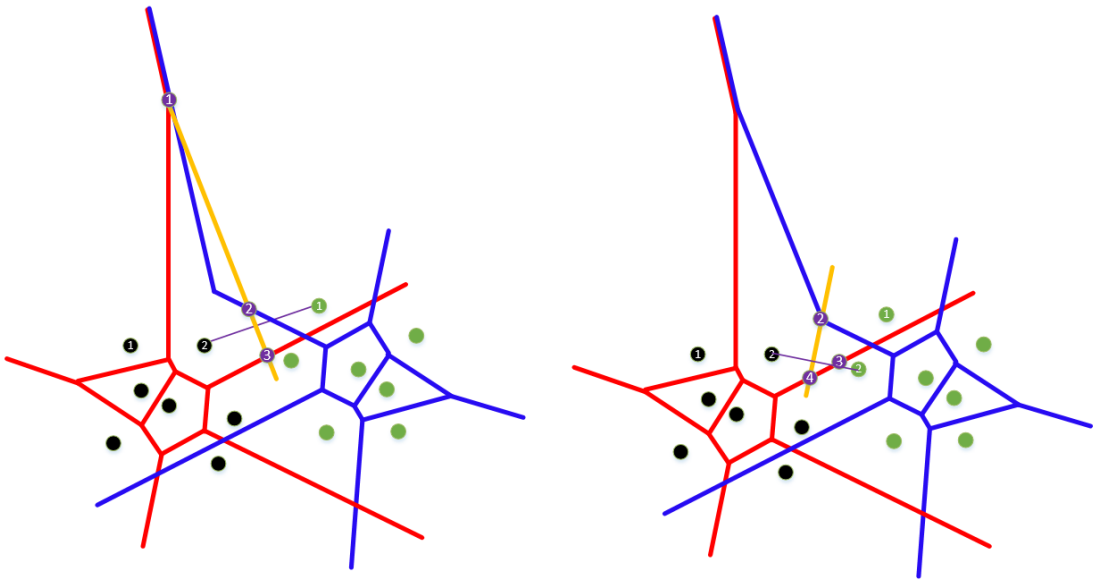


如上图，左边当前 site 为黑 1，右边为绿 1。平分线与绿 2 所在 cell 交于紫 2 和紫 3，因此右边的当前 site 变为绿 2。



借助了多边形的凸性，黑 1 和绿 2 的平分线只会交于紫 3 所在位置的后面的位置（注意还是有可能与紫 3 所在的边相交），这里是交于紫 4。

我们回到一开始的例子：



如上图左，当橙色线与黑 2 所在 cell 交于紫 3 的时候，我们就可以记录紫 3 所在边。到上图右时，橙线可以直接从紫 3 所在线计算交点，然后往下遍历，而不用再从一开始挨个遍历了。

参考文献

[1] 计算几何 • 邓俊辉 [清华大学]