算法 1: 增量Delaunay三角化生成算法

```
输入:
一个点集S;
输出:
一个Delaunay三角化T;
随机选取三个点生成第一个三角形，从S中移除这三个点;
while S非空 do
    取出一个点，记为p在当前三角化中定位该点;
    if p 在当前三角化的闭包中 then
        通过1-3和2-4边翻转将新点插入当前三角化中;
    else
        将新点与对应合适的闭包上的边生成一个新的三角形;
    end
    通过Lawson边翻转将新的三角化变成Delaunay三角化;
end
```

算法 2: 动态网格生成算法

输入:

一个点集S

一组描述物体边界运动的向量$\{v_s^t\}$，t代表当前步数，s代表对应点的序号

总步数$Maxstep$

输出:

一系列网格$\{M^t\}$

生成初始状态的网格;

$step = 0$;

输出该初始网格M^{step};

while $step < Maxstep$ **do**

  将再优化过程中添加的自由边界点移除，恢复强制边;

  移除边界运动过程中的冲突点和冲突边;

  通过移动强制点来完成对物体边界的移动：$S_i^{step+1} = S_i^{step} + v_{S_i}^{step}$;

  通过局部Lawson边翻转和Delaunay再优化提升局部网格质量;

  输出当前时间层的网格：M^{step};

  $step = step + 1$;

end

算法 3: 动态高维嵌入网格生成算法

输入:

一个点集S

一组描述物体边界运动的向量$\{v_s^t\}$ t代表当前步数，s代表对应点的序号

总步数$Maxstep$

动态嵌入函数$F(x,y,t), \quad t=1,2,\cdots,Maxstep$

输出:

一系列网格$\{M^t\}$

生成初始状态的网格;

$step=0$;

输出该初始网格M^{step};

while $step < Maxstep$ **do**

  将高维嵌入过程中添加的自由边界点移除，恢复强制边;

  移除边界运动过程中的冲突点和冲突边;

  通过移动强制点来完成对物体边界的移动：$S_i^{step+1}=S_i^{step}+v_{S_i}^{step}$;

  基于当前的动态嵌入函数$F(x,y,step)$进行高维嵌入;

  输出当前时间层的网格：M^{step};

  $step=step+1$

end