# 1. Gelly Streaming

## 1.1 解决问题

实现了4个准确算法：Triangle Count/Connected Components/Bipartitions/Centralized Weight

以Triangle Count和Connected Components为例，这两个算法都是**不需要进行迭代计算**的。都是以边为单位进行**增量处理**的。

具体的实现过程再述。

## 1.2 存在问题

不能实现的算法：SSSP/PageRank

这两个算法都是需要进行**迭代计算**的。

## 1.3 可否扩展

Graph Streaming上的迭代计算？

# 2. Gelly BSP

BSP模型的实现主要在如下三个包中：

* org.apache.flink.graph.pregel: VertexCentricIteration模型，只有一个Compute计算逻辑。
* org.apache.flink.graph.spregel: ScatterGatherIteration模型，有Gather和Scatter两个计算逻辑。
* org.apache.flink.graph.gsa: GatherSumApplyIteration模型，有Gather/Sum/Apply三个计算逻辑。

这三个迭代模型，本质上都是基于顶点的编程模型，底层依赖的是DataSet上的**DeltaIteration**.

# 3. Graph Streaming

图计算的本质和难点：**迭代计算**、**全局计算**。

迭代计算意味着不是一次计算就可获得结果的（TC、CC算法都是一次计算即可获得结果，而SSSP/PR算法需要多次反复迭代计算才能获取最终结果），目前我们可以做的一次计算，无法做迭代计算，是因为还没有找到对应的迭代模型。

全局计算意味着图的计算是牵一发而动全身的。（TC、CC算法都是局部计算即可获取结果的，Gelly Streaming的做法是将这条边所影响的两个顶点的数据放在一个节点上，那么TC、CC算法所需要的数据仅仅就是这两个顶点的数据，而且不会影响到其他顶点的数据，因此影响范围不会传播，而SSSP/PR算法是需要在整个连通子图上传播影响。）