Spark GraphX

1.两项重要的技术：Spark和图

* 图数据的类型包括网络结构数据，树状结构数据，关系型数据，Kitchen sink，和稀疏矩阵
* 图算法包括PageRank，推荐系统，最短路径，社区发现等

2. 简单介绍

* val graph = GraphLoader.edgeListFile(sc, “cit-Hepth.txt”)
  + GraphLoader是GraphX类库里的对象，有一个函数edgeListFile，有两个必填参数，第一个是SparkContext的sc，第二个是边列表文件的路径。
* graph.inDegrees.reduce((a,b) => if (a.\_2 > b.\_2) a else b)
  + 返回第二个值最大的一个点

3.基本知识（图术语解释）

* 基础
  + 有向图和无向图：所有边都有方向就是有向图，如果忽略边的方向就是无向图
  + 有环图和无环图：有环图是包含循环的，一系列定点连接成一个环，不恰当的算法实现会卡住，在环上永远循环下去。
  + 有标签的图和无标签的图：有变迁的图是对顶点或边有唯一标示，还有与之关联的数据（标签）。对顶点做了标签的图称为顶点标签图，对边做了标签的图称为边标签图。
  + 平行边和环：GraphX是伪图，会存在平行边和自环，需要通过groupEdges()和subgraph()来排除这两种情况
  + 二分图：整个图的顶点被分成两个不同的集合，所有源顶点是一个集合（之间没有联系），所有目标顶点是一个集合（之间也没有联系），在两个集合内都不存在相连的边。适合两个不同类型实体之间的关系建模。

4.GraphX基础

* 几个小点
  + 当一个scala的类或对象中定义了函数apply()时，在调用apply()时可以省略apply，即Graph.apply()简写为Graph()。
  + parallelize()和makeRDD()功能相同
* 顶点对象和边对象
  + graphX中，图的基础类为Graph，包含两个RDD：一个为边RDD，另一个为顶点RDD。优势：既可以将底层数据看作一个完整的图，使用图概念和图处理原理；也可以把它们看作独立的边RDD和顶点RDD，使用数据并行处理源语，如mapped、joined、transformed
* mapping操作
  + 简单的图转换
    - mapTriplets()
    - myGraph.mapTriplets(t => (t.attr, t.attr== “is-friends-with” && t.srcAttr.toLowerCase.contains(“a”))).triplets.collect
  + Map/Reduce
    - myGraph.aggregateMessages[Int](\_.sendToSrc(1),\_+\_).collect
    - def aggregateMessages[Msg](

sendMsg: EdgeContext[VD,ED,Msg]=>Unit,

mergMsg:(Msg,Msg)=>Msg)

:VertexRDD[Msg]

* + - sendMsg函数以EdgeContext作为输入参数，没有返回值（Scala的Unit相当于其他语言的void）
    - mergeMsg函数定义了如何将顶点收到的所有消息转换成需要的结果
    - sendToSrc：在边上将包含整数1的消息发送到源顶点。