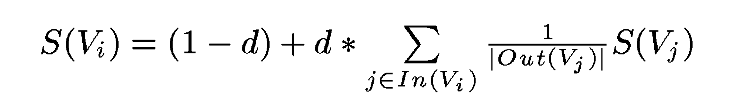
Pagerank graphx

# Pagerank

## 1.1、pagerank有向图计算公式

PageRank最开始用来计算网页的重要性。

公式如下：



S(Vi)是网页i的中重要性（PR值）。d是阻尼系数，一般设置为0.85。In(Vi)是存在指向网页i的链接的网页集合。Out(Vj)是网页j中的链接存在的链接指向的网页的集合。|Out(Vj)|是集合中元素的个数。

PageRank需要使用上面的公式多次迭代才能得到结果。初始时，可以设置每个网页的重要性为1。上面公式等号左边计算的结果是迭代后网页i的PR值，等号右边用到的PR值全是迭代前的。

详细例子参见（http://www.cnblogs.com/clover-siyecao/p/5726480.html）

## 1.2、pagerank无向图计算公式

IMG_256

其中weight(pj)是边（pi,pj）的权重，degree（pj）是顶点pj的度。

（如果无法确定边的权重，可以将图直接变为双向的。）

# 2、graphx实现pagerank算法

Graphx中可以使用pregel接口实现图计算中的迭代算法。在graphx中使用pregel接口实现pagerank算法如下：（参见：http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.graphx.Pregel$）

val pagerankGraph: Graph[Double, Double] = graph

// Associate the degree with each vertex

.outerJoinVertices(graph.outDegrees) {

(vid, vdata, deg) => deg.getOrElse(0)

}

// Set the weight on the edges based on the degree

.mapTriplets(e => 1.0 / e.srcAttr)

// Set the vertex attributes to the initial pagerank values

.mapVertices((id, attr) => 1.0)

def vertexProgram(id: VertexId, attr: Double, msgSum: Double): Double =

resetProb + (1.0 - resetProb) \* msgSum

def sendMessage(id: VertexId, edge: EdgeTriplet[Double, Double]): Iterator[(VertexId, Double)] =

Iterator((edge.dstId, edge.srcAttr \* edge.attr))

def messageCombiner(a: Double, b: Double): Double = a + b

val initialMessage = 0.0

// Execute Pregel for a fixed number of iterations.

Pregel(pagerankGraph, initialMessage, numIter)(

vertexProgram, sendMessage, messageCombiner)

可以看出这个算法是一个有向图算法，使用无向图计算时可以将图中的边变为双向的，在使用pagerank算法。

# pagerank计算

在此使用的是graphx中自带的pagerank算法，自带的pagerank算法并不是用pregel接口实现的，pregel接口的实现详细可以参见文档《[spark\_pregel](spark_pregel.docx)》。

计算数据的图可视化结果，如图1所示。

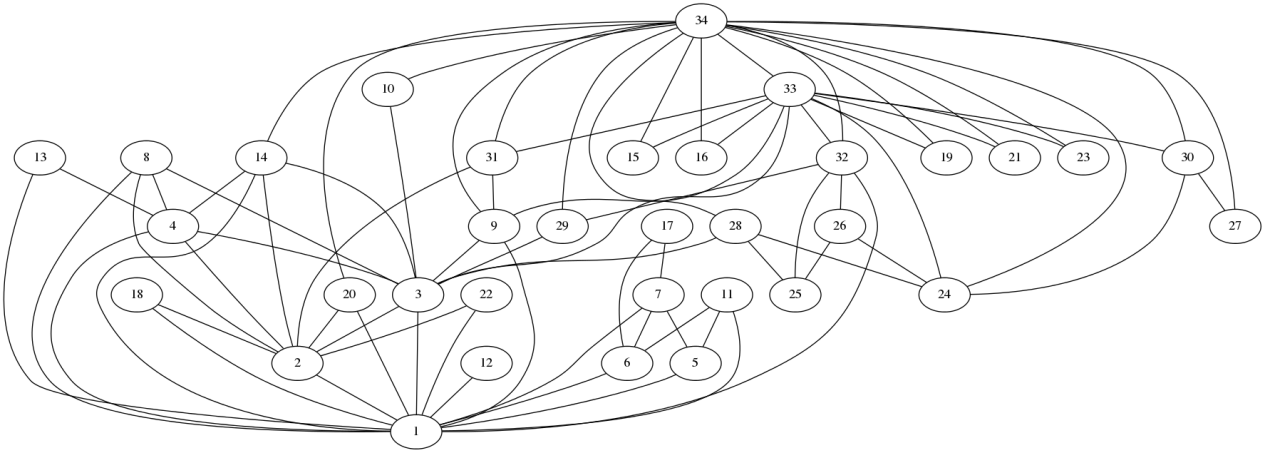


图 1 计算图的数据可视化

详细代码查看附页，图2为打印结果。

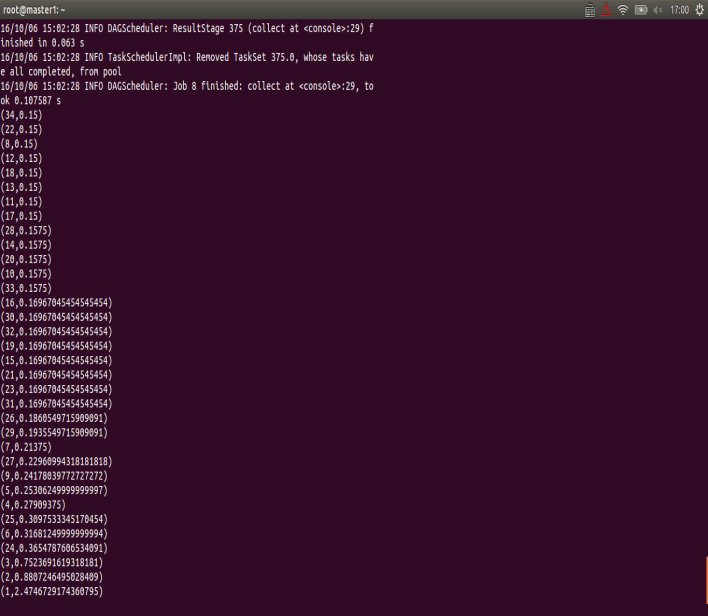


图 2 pagerank有向算法结果

使用无向图计算，即将图中的边变为双向的，其计算结果如图3所示。详细代码见附页。

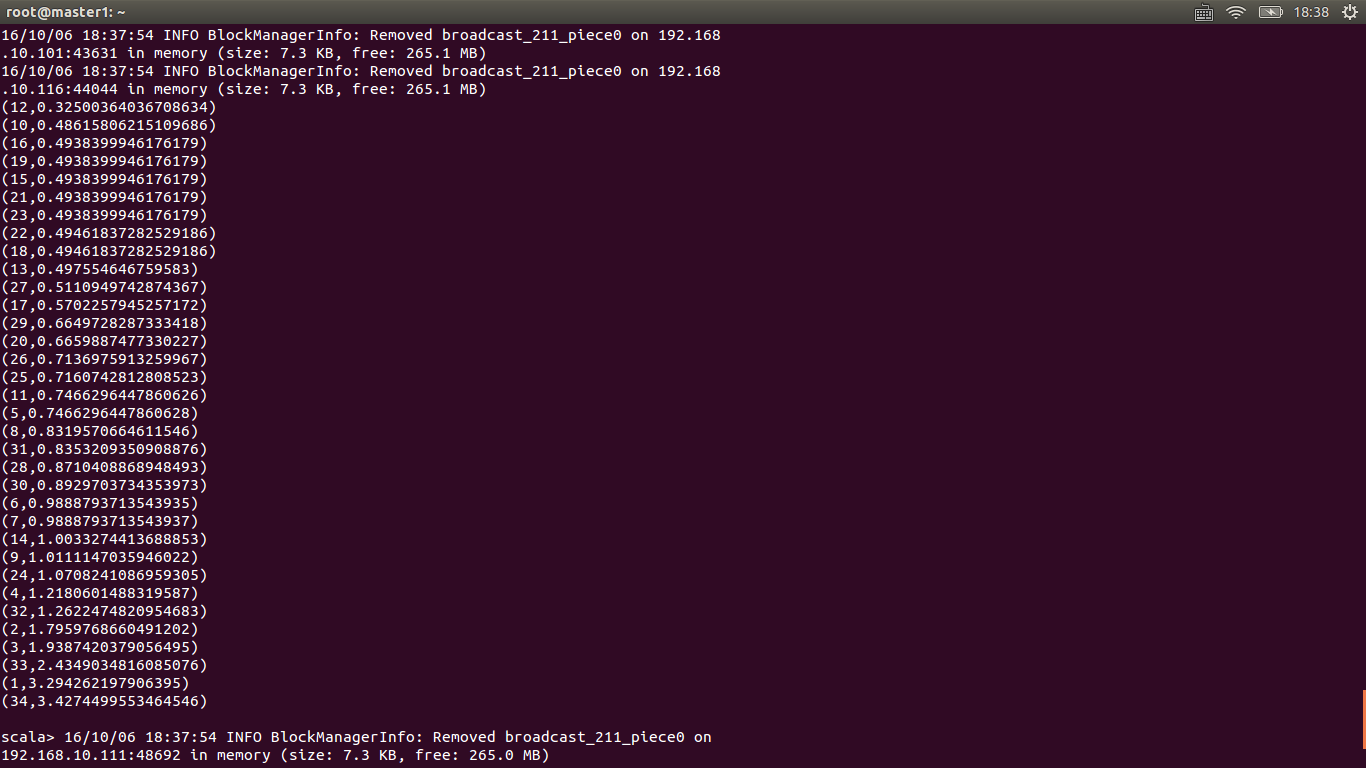


图 3 双向pagerank算法结果

# 问题分析

在有向图中pagerank问题：

问题：可以看见节点34的连接数较多，那么其重要性也应该是较大的，但是可以看见其重要性是最低的。

原因：pagerank算法是有向的，在源文件中34节点仅有出度，几乎没有入度，因此其重要性较低。

解决方法：1、将图变为双向的。2、自己重新实现pagerank算法。

# 附页：（代码）

有向图pagerank算法：

Import org.apache.spark.graphx.GraphLoader

val graph = GraphLoader.edgeListFile(sc, "hdfs://master1.jie.com:8020/user/jie/testdata/edges2.csv")

val ranks = graph.pageRank(0.001).vertices

ranks.collect.sortBy(v => v.\_2).foreach(println)

无向图pagerank算法：

Import org.apache.spark.graphx.\_

val graph = GraphLoader.edgeListFile(sc, "hdfs://master1.jie.com:8020/user/jie/testdata/edges2.csv")

val newedges = graph.edges.map(e=>Edge(e.dstId,e.srcId,e.attr))

Val newGraph = Graph(graph.vertices,graph.edges++newedges)

val ranks = newGraph.pageRank(0.001).vertices

ranks.collect.sortBy(v => v.\_2).foreach(println)

**参考资料：**

<http://www.cnblogs.com/clover-siyecao/p/5726480.html>

<http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.graphx.Pregel$>

<http://spark.apache.org/docs/latest/graphx-programming-guide.html#graph-algorithms>

http://jia1546.is-programmer.com/posts/32327.html

**参考代码：**