PageRank算法详解

今天我将给大家介绍PageRank算法,Spark GraphX官方文档介绍的图计算算法主要有三个,分别是PageRank算法,ConnectedComponents(连通体)算法,TriangleCount(三角形计数)算法,这些算法包含在org.apache.spark.graphx.lib包中，可以被直接访问。

PageRank，即网页排名，又称网页级别、Google左侧排名或佩奇排名,最先是由Google提出的,为Google的精确搜索做出了很大贡献,它是一种根据网页之间相互的超链接计算的技术,可以计算网页排名,而在Spark Graphx里面.它能度量一个图中每个顶点的重要程度.应用的场景很广,比如说社交系统的可以据此来计算网友粉丝排名,计算网页排名等等.pagerank的思路是让链接"投票",然后得出"投票"结果,比如说现在有A,B,C,D四个网页,B,C,D三个网页都指向A网页,那么A网页的PageRank值将是其它三个网页的总和,也就是

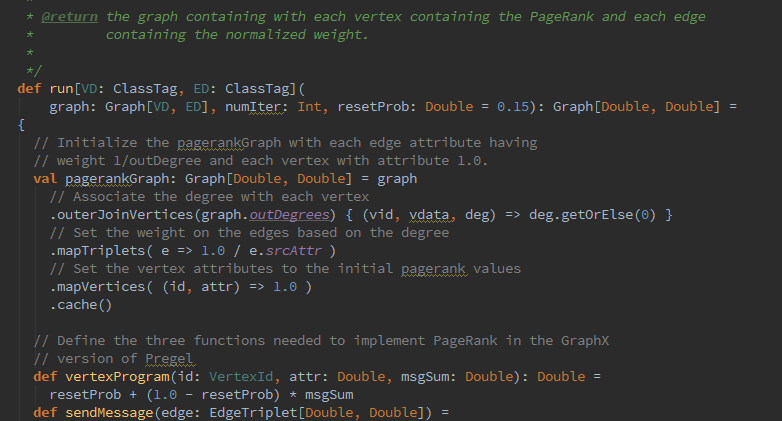
PR(A)=PR(B)+PR(C)+PR(D)

但是在实际情况中这样的"投票"方式是不准确的,因为每个网页的重要程度是不一样的,因此我们为每个网页加上了权值,现在我们在原来的假设上增加以下改变,我们将B网页指向C,D网页,将C网页也指向D网页,那么现在

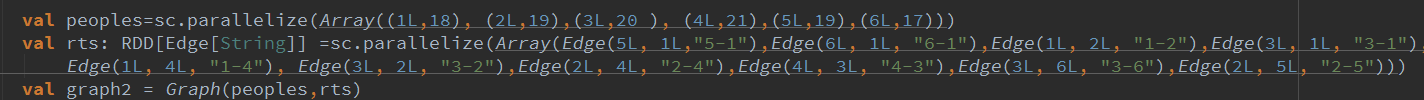
PR(A)=PR(B)/3+PR(C)/2+PR(D)/1

可以看出链接出去的网页将平分原网页的PR值,这只是最简单的情况,对复杂的情况来说我们可以将其转化为矩阵计算,假设有n个网页,每个网页的初始值为N0=[1/N,1/N,1/N...1/N],则N(r)=A\*N(r-1)而A是每个网页的链接矩阵,由数学知识可以知道PR是可以收敛的,将会无限趋近于N,这个N就是真实的排名情况,大致的原理就是这样.

Spark PageRank的源码如下



现在我写个Demo测试一下PageRank,首先构造一个图

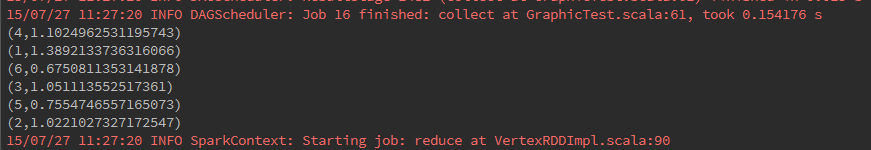


我们使用PageRank算法,可以调用GraphOps下面的pageRank,staticPageRank,personalizedPageRank,staticPersonalizedPageRank四个方法,它们内部封装了PageRank算法,我们直接使用就是

首先介绍第一种动态的PageRank,PageRank分动态和静态(staticPageRank),区别在于动态的PageRank是按照2次迭代的结果差异性来停止迭代,如果2次迭代的差异小于给定的参数趋近值则停止迭代,趋近值越小数据越精确；而静态的staticPageRank则是按照固定的次数迭代,方法参数中要带这个固定值,固定迭代次数越多,数据越精确.

graph2.pageRank(0.001,0.5).vertices.map(t=>(t.\_1,t.\_2)).collect().foreach(println)

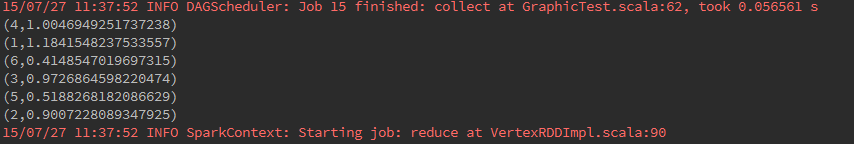
此处用的是动态的方式,pageRank带的第一个参数是一个Double类型的趋近值,小于这个趋近值迭代结束,第二个参数是可选的,表示的是图的各顶点的初始PR值,如果不写则默认为0.15.在进行pagerank算法时只与图结构的链接关系有关,与边的值和顶点的值都没有关系,程序结果如下:



静态的实现方式为:

graph2.staticPageRank(10).vertices.map(t=>(t.\_1,t.\_2)).collect().foreach(println)

staticPageRank的参数也有2个,第一个是Int类型的表示迭代次数,第二个和pageRank一样也是Double类型的表示初始PR值,默认为0.15,程序结果如下:

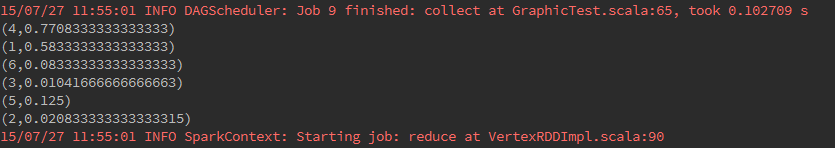


剩下的personalizedPageRank,和staticPersonalizedPageRank,与前面一对不同的是它是个性化的pagerank,与普通pagerank不同的是,它能指定从哪个顶点开始迭代,而且可以解决"PR黑洞"问题,回到我们开始的假设问题,有A,B,C,D四个网页,B,C,D三个网页都指向A网页,在这里,A网页只有指向它的没有指出去的,而B,C,D网页只有指出去的没有指进来的,专业点说A的出度为0,这样每一次迭代都会吞噬掉B,C,D的PR值,最终结果就是B,C,D的PR值为0,而personalizedPageRank能将这种出度为0的网页的PR值按照一定的比例分配给别的网页,从而避免"黑洞"的出现

personalizedPageRank也分动静态之分,首先看动态的:

graph2.personalizedPageRank(4,0.01,0.5).vertices.map(t=>(t.\_1,t.\_2)).collect().foreach(println)

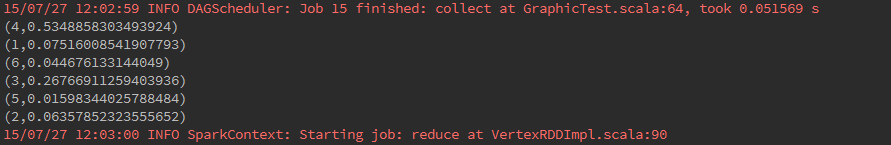
它有三个参数,第一个参数表示的是开始迭代的源点VertexId,第二个参数表示的是Double类型的趋近值,小于这个趋近值迭代结束,第三个参数是可选的,为Double类型的表示初始PR值,默认也为0.15,程序结果如下:



动态个性化pagerank实现方式为:

graph2.staticPersonalizedPageRank(4,10,0.5).vertices.map(t=>(t.\_1,t.\_2)).collect().foreach(println)

它也有三个参数,第一个和前面一样,表示的是开始迭代的源点VertexId,第二个参数表示的迭代的固定次数,第三个表示的是初始PR值,同样的是可选的,不写默认为0.15,程序结果如下:



好了,PageRank就介绍完了,你学会了吗?