- 人脉
- 算法

算法需求

输入的关系数据：

```
a,b,weight
b,c,weight
c,d,weight
b,a,weight
…
注：关系是有向的。a,b,weight 表示 a可以联系到b，联系的强度是weight。
```

目标输出：

```
a,c,b
b,d,c
x,y,(m1,m2,…)
…
注：包含所有二度人脉；x,y,(m1,m2) 表示y是x的二度人脉，中间人可以是多个(m1,m2,…)，按权重降序排列。二度人脉的权重是：(r1 * weight(x, m1) + r2 * weight(m1, y)) / 2，如： r1 = 1, r2 = 1/2。
```

算法1：用逆向关系做中间数据

mapreduce实现：启动两个mapreduce job。
job1 map:
将每一条关系数据，输出为正向和逆向两条：

```
a,b,weight
b,c,weight
…
->
a,(b,F,weight)
b,(a,B,weight)
b,(c,F,weight)
c,(b,B,weight)
…
注： F为 forward，B为 backward。中间数据大小翻倍。
```

job1 reduce:
聚合每个个体的正向和逆向关系列表：

```
a,(b,F,weight)
b,(a,B,weight)
b,(c,F,weight)
c,(b,B,weight)
…
->
a,([ (b,F,weight) ], [] )
b,( [(c,F,weight)], [ a,B,weight ] )
c,([ (b,B,weight) ], [])
…
注：每个个体一行，key为个体，value为两个数组：包括正向和逆向关系。
```

job2 map:

```
a,([ (b,F,weight) ], [] )
b,( [(c,F,weight)], [ a,B,weight ] )
c,([ (b,B,weight) ], [])
…
->
a,b,[ (c, weight) ]
…
注：主要是将两个数组的元素两两配对，weight加权平均，并输出。
```

job2 reduce:

```
a,b,[ (c, weight) ]
a,b,[ (z, weight) ]
…
->
a,b, [ (c, weight), (z, weight) ]
注：排序在reduce节点内做。
```

特点：数据量膨胀一倍，适合离线计算。根据需要，可进一步放到NoSql中快速查询，或者对目标个体做索引，方便人脉搜索。

在线计算

需求是，实时计算某个体的二度人脉。
使用分布式的图计算算法，如Spark GraphX，首先加载点和边的数据。
从起点个体出发，向它的一度关系个体发送消息并聚合，得到一度人脉个体列表。在从所有这

些一度列表出发，发送消息到所有二度个体并聚合。实际上是一种广度优先的算法。
参考： Graph.aggregateMessages (https://spark.apache.org/docs/1.3.1/api/scala/index.html#org.apache.spark.graphx.Graph)

参考

海量数据的二度人脉挖掘算法 (http://my.oschina.net/BreathL/blog/75112)
GraphX Programming Guide (https://spark.apache.org/docs/1.3.1/graphx-programming-guide.html)