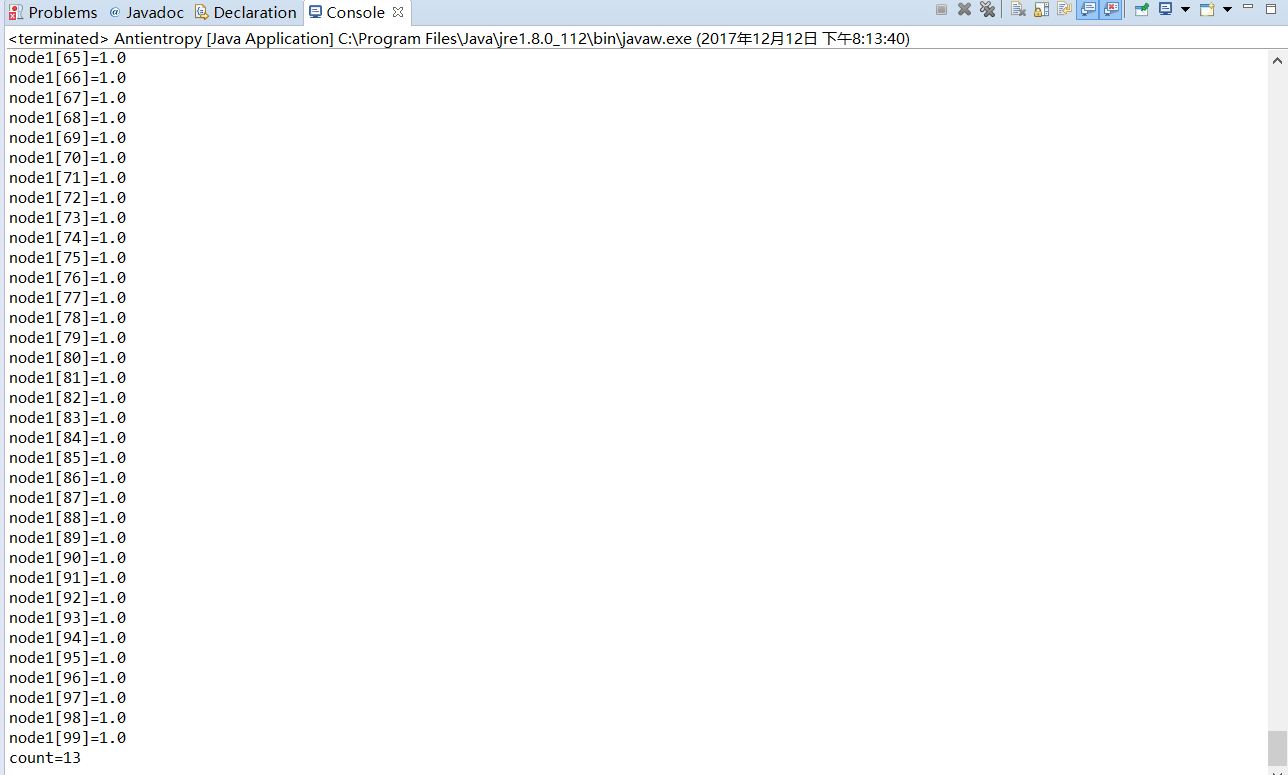
**Anti-entropy and Gossiping**

1. Anti-entropy（反熵）

① based on push

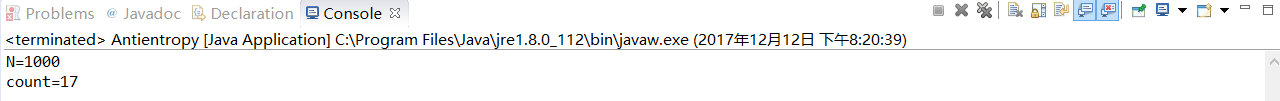
基于push的方法表示P只是把它自己的更新信息发出给Q；

代码为Antientropy(push).java，其中使用两个对象数组，一个用来保存全部的结点信息(node)，其中数据为1…N，另一个用来保存已感染的结点(node1)，可以通过N的大小调整结点个数，我们假定初始已感染的结点为node[0]，将其加入到node1[0]中，通过产生随机数来模拟已感染的结点随机选取结点的过程，当node1中的每个结点作为发起者与随机选择的另一个结点进行了一次更新信息交换时，这一轮通信结束，轮数count++；这样一直循环，直到所有结点全部通信过或等于node[0]为止，通信结束；运行结果如下图：

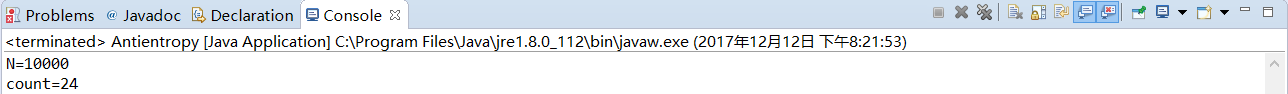


最后一轮通信结束后所有结点的信息均已更新，由于是随机选择结点进行通信，因此导致通信轮数不同，如：11 12 14（N=100）等等

总结：以上结果为N=100，初始已感染结点为1个时的结果，当N=1000，初始已感染结点为1个时，count=16 17 18 19 etc.



N=10000，初始已感染结点为1个时，count=23 24 25 etc.

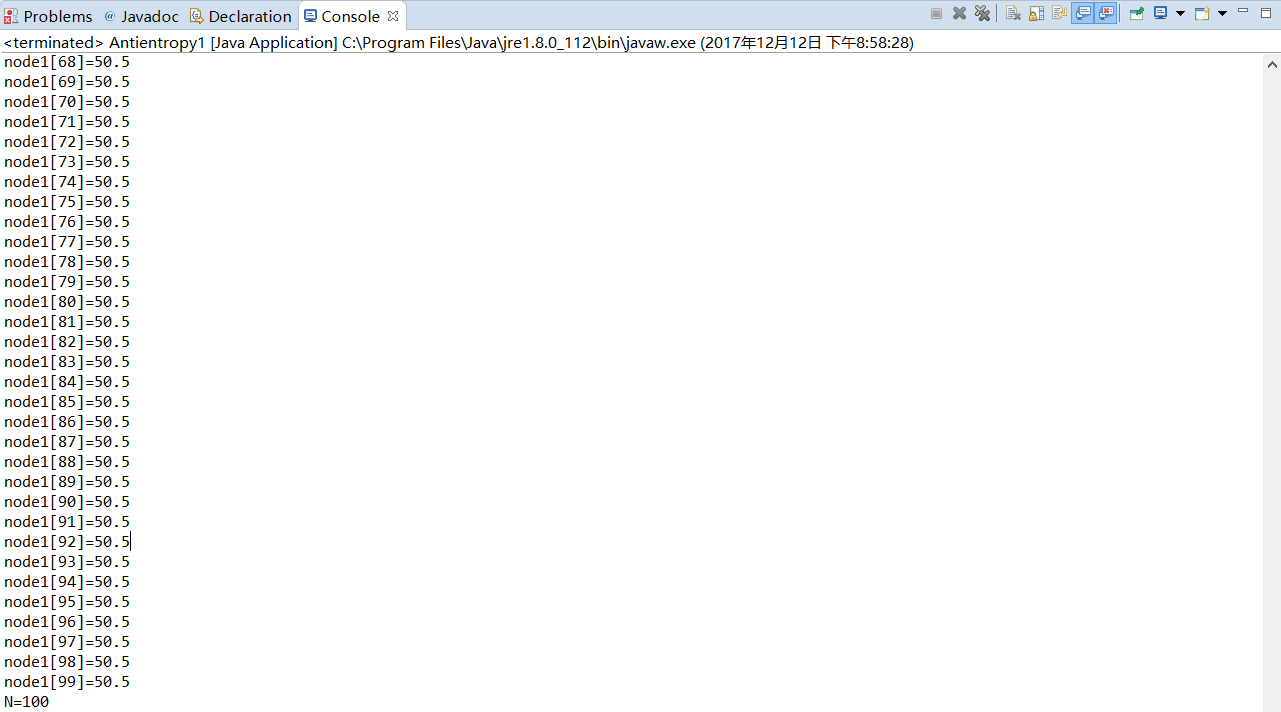


② based on push-pull

基于push-pull的方法表示P和Q相互发送更新信息给对方；

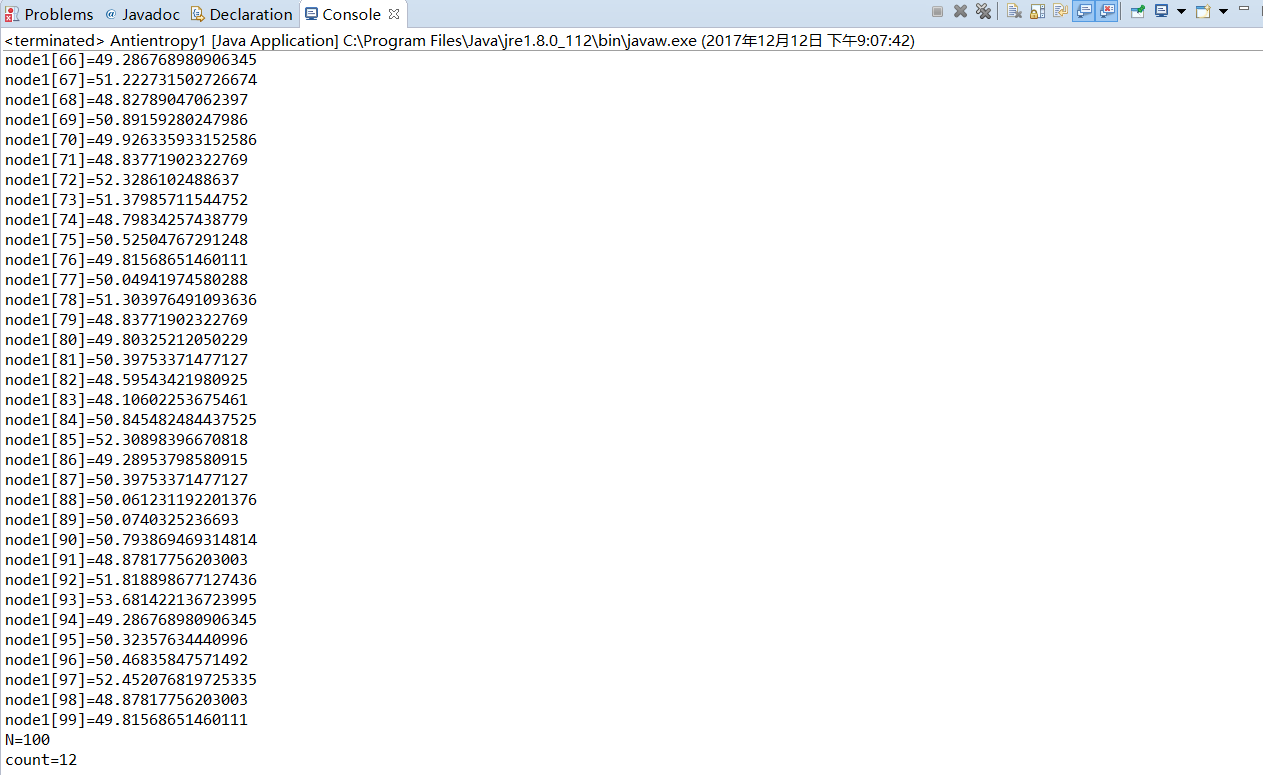
代码为Antientropy1(push-pull).java，基本代码结构与Antientropy(push).java相同，不同的是当两个结点P 与Q会相互发送更新信息给对方，本方法中通过将P于Q的值更新为（P+Q）/2来模拟此过程；此种方法参考书中125页应用部分，用于收集或汇集信息，当结点i与结点j通信时，它们相互如下更新其值：Xi,Xj🡨(Xi+Xj)/2，经过交换后，i和j具有相同的值，即所有初始值的平均值，传播速度是收敛的。下面运行程序来验证一下：

(N=100 node[i].getDate()=i(1…N))

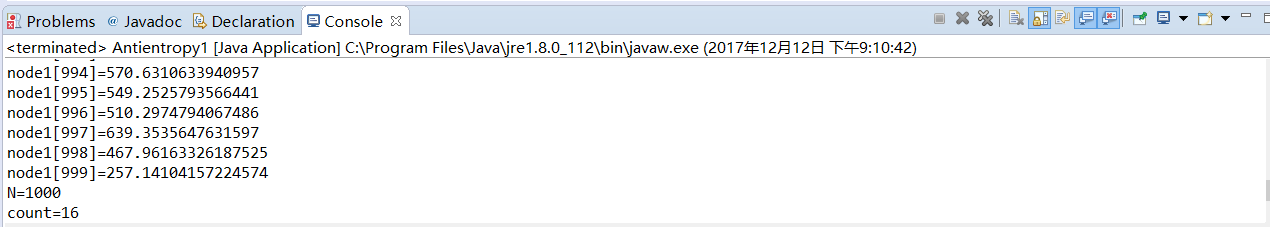


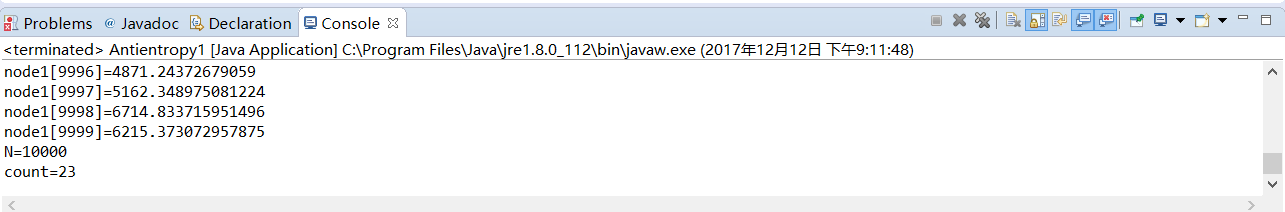
总结：当N=100，初始已感染结点为1个时，最终所有结点的值收敛于（1+…+N）/N，当最终要求数组中所有值均严格等于average才跳出循环时，由于average为double类型较精确，导致求得的轮数较大，为70轮左右；

因此更改为所有结点均为已感染状态时跳出循环时，运行结果如下：当N=100，初始已感染结点为1个时，所有结点均收到更新信息时，轮数为11 12 etc.



当N取1000，10000时分别测试：（若所有结点值都相同时跳出循环，最终结果仍收敛于average）

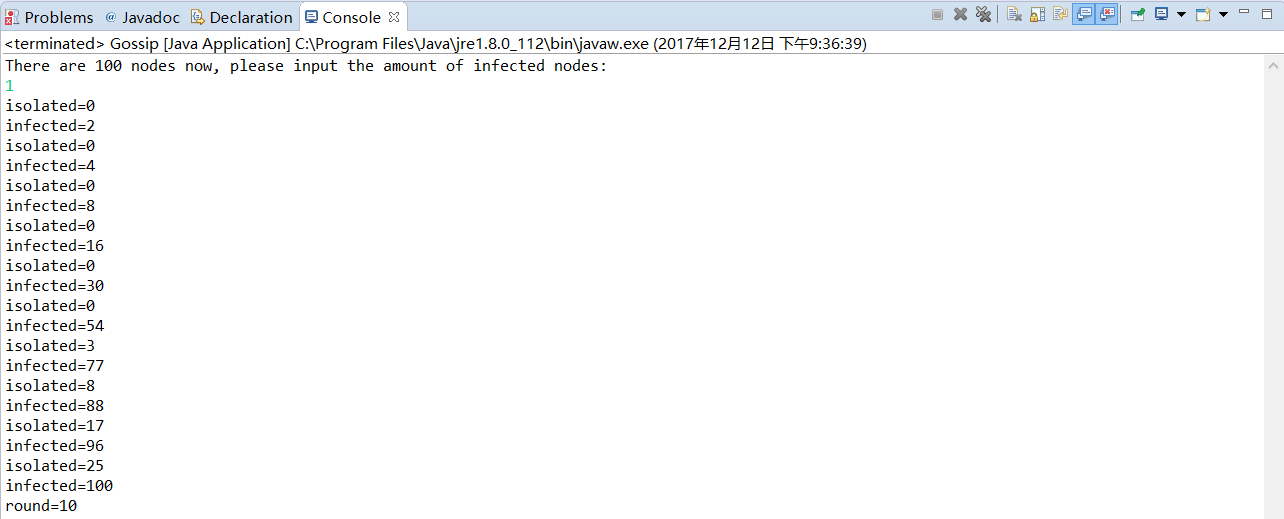




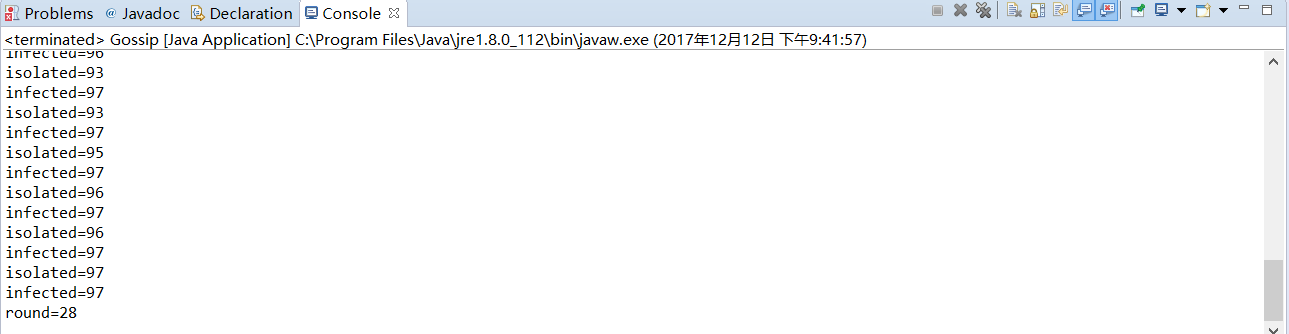
1. Gossping

Rumor spearing流言传播，简称gossiping，它的工作原理为：如果结点P正好刚更新数据项x,那么它将与任意结点Q通信，并把更新信息发送给Q，但是有可能Q被另一个结已更新了。在这种情况下，P可能不再传播该更新信息的概率为1/k，也就是说，它变为了已隔离的，代码为Gossip.java。结点保存两种状态，infected and isolated，他们均为boolean类型，当infected is true && isolated is true时，结点为已隔离状态，当infected is true && isolated is false，结点为已感染状态，当infected is false && isolated is false时，结点为易感染状态；使用了两个对象数组，一个用来在循环中判断，另一个用来在循环中修改，当一轮通讯结束时将两者进行同步；接下来开始循环，当所有结点全部通信过或仅有某一小部分结点未通信，其余结点已隔离时跳出循环，在循环中，已感染状态的结点分别生成随机数，即随机与其他结点通信，若该随机结点为非易感染状态，则产生[0，1)随机数判断其是否在[0，1/k)范围内来模拟其可能不再传播该更新信息的概率（1/k），若在[0，1/k)范围内，则该节点状态变为已隔离状态，否则继续。

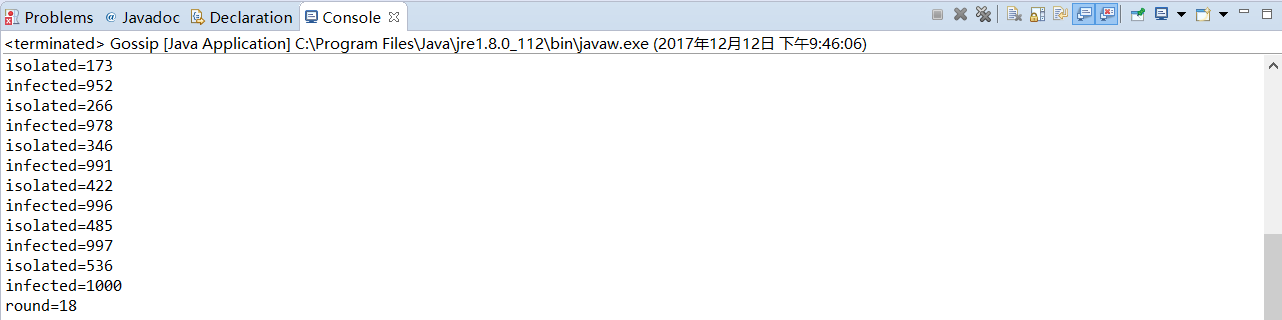
运行结果：N=100，k=10（概率为1/10），初始已感染结点为1个时运行结果如下：（isolated infected分别记录当前所有结点中已隔离状态，非易感染状态的结点的个数） round=10 11 12 13 etc. （为方便测试本程序可输入初始已感染结点个数）



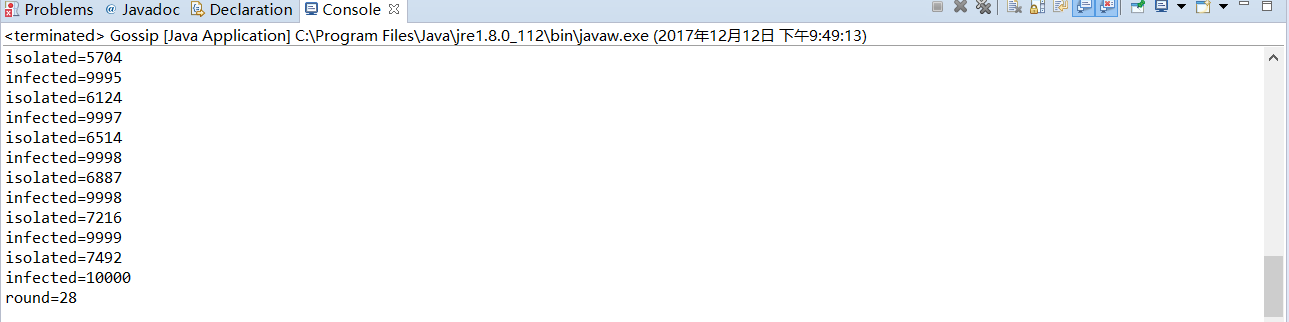
若仍N=100, 初始已感染结点为1个，我们调整k=4(1/4)时，结果为有3 or etc.个结点仍未已感染状态，符合书上127页结果：若k=4，ln(s)=-4.97，s<0.007，表明只有不到0.7%的结点仍未易感染的。



N=1000，k=10，初始已感染结点为1个：



N=10000，k=10，初始已感染结点为1个：

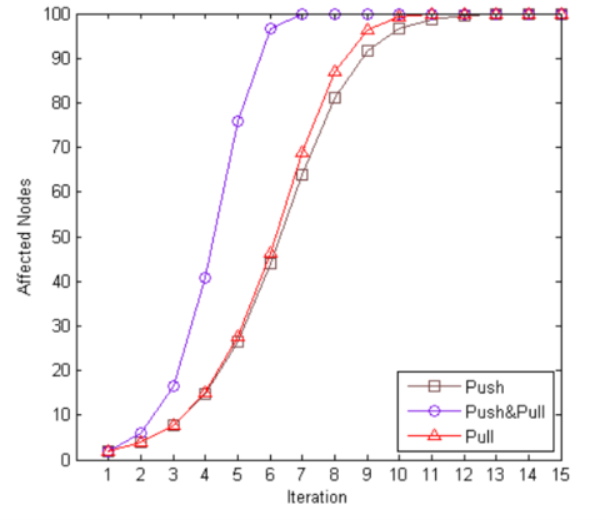


1. 总结

如果要快速传播更新信息，那么只使用基于push的方法是一个不好的选择；在基于push方法中，如果易感染的结点很多，那么其工作就更好；如果只有一个结点是易感染的，使用anti-entropy模型的任意一种方法，更新信息都会通过所有结点迅速传播，但push-pull最好。

该方法的一个特殊变体为gossiping，事实证明gossiping是快速传播新闻的很好方法，但是他不能保证所有结点都真的已更新了，可以证明，当有大量结点时，保持未更新的（易感染的）结点的系数满足一定的等式，详见书124页。

下图为一个在100个结点的集群中传播一个更新的模拟结果。在每次迭代中，每个节点只与一个随机选取的对等节点发生联系：



参考资料：

1. 分布式系统原理与范型（第二版）
2. Google and Baidu

东北大学软件学院

软件1507

20154994

周佳男