反熵与 Gossip 算法的实现和分析 实验报告

20165164 软英 1601 杨沛怡

（一） 数据设置

语言：Java
节点个数 n=节点最大值 max：1000，10000，10000
Gossip 算法中的 k：1.1，1.2，4
收敛条件中的 epsilon ：固定为 0.00000001

（二） 实验过程和结果

（1）伪代码

反熵：

1. 随机初始化 n 个节点
2. 对所有节点：
   选择除自己外的任意节点进行通信
3. 判断是否结束
   如果 最大节点和最小节点之差小于 epsilon
   结束
   否则
   返回 2

Gossip：

1. 随机初始化 n 个节点
2. 对所有未被隔离的节点 A：
   选择除自己外的任意节点 B
   如果 AB 之差小于 epsilon，认为相等
   A 的通信概率/=k
   否则
   AB 以 A 的概率进行通信
3. 判断是否结束
   如果 本轮通信中没有节点进行通信
   结束
   否则
   返回 2

（2）实验结果

（反熵次数，Gossip 次数）	n=max=100	n=max=1000	n=max=10000
k=1.1	（40，48）	（45，56）	（48，60）
k=1.2	（41，44）	（45，52）	（49，59）

k=4	（40，44）	（46，47）	（48，52）

（3）截图

以 k=1.1，n=max=1000 为例

```
1. using antiEntropy:
expected average: 507.908
Done. Counter=45
actural value of first node: 507.9079999965103

2. using gossiping:
expected average: 503.393
Done. Counter=56
actural value of first node: 503.3930000005698
```

（三） 结果分析

1. 节点个数的影响

为了简化参数，设置节点个数等于最大节点值。对比可知当节点越多，通信轮数越多。

2. 通信轮数的对比

做实验之前本以为 Gossip 加入了隔离状态，会结束得更快，但实验结果显示 Gossip 要比反熵慢，经过分析，我认为是因为 Gossip 的一轮通信中，实际进行通信的节点少于反熵，因此需要更多轮通信才能结束。

3. Gossip 中 k 的取值

经过对比，发现 k 较大时通信轮数较少，当 k 足够大后，影响变得很小， 说明：k 变大后，部分节点变成隔离态，不再通信，有利于达到结束条件。另外可能和通信概率减少的算法相关，本实验中选的是 1/k，不能保证所有节点都通信。