# 分布式算法

BY 唐志鹏 SA23011068

## 1 思考题

不是，证明：

考虑系统$S$，有一个变量$k$，初始值为0。只有一个动作：如果$k=1$，那么$k:=2$。设，$P$为断言$k<2$。$P$在$S$中总是真，但$P$不是不变量。因为，若$P$成立的状态（不可达）$k=1$被转换为$P$不成立的状态$k=2$，$P$并没有保持不变。因此，$P$不是不变量。

## 2 ch33 P9 Ex

先证明：一节点从$p_r$可达，则它曾经设置过自己的parent变量

图$G$是由parent和children确定的静态图，任意一个节点收到$M$才会加入到图中。根据算法，可达节点收到$M$时，会执行line5，由于是容许执行的，line7也会被执行，即：该可达节点设置了自己的parent变量。

再证明：一节点曾经设置过自己的parent变量，则它从$p_r$可达

根据算法，line7已经执行，因为是容许执行的，因此line5必然也被执行过，即：该节点收到过$M$，而$M$是$p_r$发出的，所以，它从$p_r$可达。

## 3 ch33 P30 Ex2.1

**同步模型：** 在convergecast的每个容许执行中，树中每个距离$p_r$为$d-t$的处理器至多在第$t$轮接收到所有孩子的消息。对$t$使用归纳法:

归纳基础：$t=1$，处理器距离$p_r$为$d-1$，该处理器要么为非叶节点，其孩子均为叶子节点，因此该节点在第1轮接收到所有孩子的消息。该处理器也可能为叶子节点，自然也满足结论。

归纳假设：假设树上每个距离$p_r$为$d-t+1$的处理器至多在第$t-1$轮接收到所有孩子的消息。

归纳步骤：设$p_i$到$p_r$的距离为$d-t$，那么其所有孩子到$p_r$的距离为$d-t+1$，由归纳假设，至多在$t-1$轮已经收到了其所有孩子的消息，由算法描述，$p_i$至多在第$t$轮接收到所有孩子的消息。

另外，最深的叶子节点的消息要传播到$p_r$至少在第$d$轮，因此，第$d$轮所有消息汇集到$p_r$，时间复杂度为$d$。

**异步模型：** 在convergecast的每个容许执行中，树中每个距离$p_r$为$d-t$的处理器至多在第$t$轮接收到所有孩子的消息。对$t$使用归纳法:

归纳基础：$t=1$，处理器距离$p_r$为$d-1$，该处理器要么为非叶节点，其孩子均为叶子节点，由异步模型的时间复杂性知，该节点至多在第1轮接收到所有孩子的消息。该处理器也可能为叶子节点，自然也满足结论。

归纳假设：假设树上每个距离$p_r$为$d-t+1$的处理器至多在第$t-1$轮接收到所有孩子的消息。

归纳步骤：设$p_i$到$p_r$的距离为$d-t$，那么其所有孩子到$p_r$的距离为$d-t+1$，由归纳假设，至多在$t-1$轮已经收到了其所有孩子的消息，由算法描述，$p_i$至多在第$t$轮接收到所有孩子的消息。

因此，$p_r$至多在第$d$轮收到所有孩子的消息，时间复杂度为$d$。

# 4 ch33 P30 Ex2.3

**连通性：** 算法构造的图$G$是连通的。否则，假设$G$中存在邻居节点$p_j$和$p_i$，其中，$p_j$从$p_r$可达，而$p_i$从$p_r$不可达。因为，$G$中的节点从$p_r$可达当且仅当它曾设置过自己的parent变量，所以，$p_j$必然设置过parent变量，$p_i$的parent变量为nil，且$p_i$属于$p_j$的unexplored集合。算法的line11和line14决定了$p_j$会向$p_i$发送消息M，使得$p_j$成为$p_i$的parent，于是，$p_i$从$p_r$可达，矛盾！因此，图$G$是连通的。

**无环：** 算法构造的图$G$是无环的。否则，假设$G$中存在一个环$p_1 p_2 \ldots p_i p_1$，则，$p_i$是从$p_1$可达的，且$p_1$的parent是$p_i$。令$p_1$是该环中最早接收到M的节点，那么，$p_1$会最早设置parent变量。根据算法，$p_i$在接收到消息M后会向$p_1$发送消息M，由于$p_1$的parent已经设置，根据算法line16，$p_1$会向$p_i$发送<reject>信息，不会将$p_i$设置为parent。矛盾！因此，图$G$是无环的。

**DFS:** 算法构造的图$G$是DFS树。只需证明：在有子节点和兄弟节点未访问时，子节点总是先加入到树中。设有节点$p_1, p_2, p_3$，$p_2, p_3$是$p_1$的直接相邻节点，且$p_1$在line12-14中先选择向$p_2$发送消息M，则，$p_1$当且仅当$p_2$向其返回一个<parent>后，才有可能向$p_3$发送消息M。而$p_2$仅在其向所有相邻节点发送过M后才会向$p_1$返回<parent>。所以，$p_2$的子节点永远先于$p_3$加入到树中。因此，图$G$是DFS树。

# 5 ch33 P30 Ex2.4

**同步模型：** 在每一轮中，只有一个消息（M or <parent> or <reject>）在传输，根据算法line6, 14, 16, 20, 25发送消息的语句可以发现：消息只发往一个处理器节点；除根节点外，所有处理器节点都是收到消息后才被激活。所以，不存在多个处理器在同一轮发送消息的情况。因此，时间复杂度与消息复杂度一致。

**异步模型：** 在一个时刻内至多有一个消息在传输，因此，时间复杂度也与消息复杂度一致。

**消息复杂度：** 对任意一边，可能传输的消息最多有四个：两条消息M，<parent>和<reject>，所以消息复杂度为$O(m)$

综上所述，时间复杂度为$O(m)$

## 6 ch33 P30 Ex2.5

**算法思想：** 维护一个集合Explored，记录已经收到消息M的处理器名称，并且该集合随消息M和<parent>一起发送。该方法避免了向已收到过消息M的处理器发送消息M，这样DFS树外的边不再耗时，时间复杂度也降为$O(n)$。

```
Code for processor P_i, (0 ≤ i ≤ n−1)
var:
    parent: init nil;
    children: init ∅;
    unexplored: init all the neighbors of P_i;
    Explored: init ∅

upon receiving no msg:
    if (i=r) and (parent=nil) then {
        parent:=i;
        ∀P_j∈unexplored;
        unexplored:= unexplored \ {P_j};
        Explored:=explored ∪ {P_i, P_j};
        send M and Explored to P_j
    }

upon reciving M and Explored form neighbor P_j:
    if parent=nil then {
        parent:=j;
        unexplored:= unexplored \ Explored ;
        if unexplored≠∅ then{
            ∀P_k ∈ unexplored;
            unexplored:= unexplored \ {P_k};
            Explored:=Explored∪{P_k};
            send M and Explored to P_k;
        }
        else {
            send <parent> and Explored to parent;
        }
    }

upon receiving <parent> and Explored from P_j:
    unexplored:= unexplored \ Explored;
    if unexplored=∅ then {
        if parent≠r then {
            send <parent> and Explored to parent;
            terminate;
        }
    }
    else {
        ∀P_k ∈ unexplored;
```

unexplored:= unexplored \ $\{P_k\}$;
Explored:=Explored$\cup\{P_k\}$;
send M and Explored to $P_k$;
}

# 7 ch34 P9 Ex3.1

假设R是大小为$n>1$的环（非均匀），A是其上的一个匿名算法，它选中某处理器为leader。因为环是同步的且只有一种初始配置，故在R上A只有唯一的合法执行。

下面用归纳法证明：在环R上算法A的容许执行里，对于每一轮$k$，所有处理器的状态在第$k$轮结束时是相同的。

归纳基础：$k=0$（第一轮之前），因为处理器在开始时都处在相同的初始状态，故结论是显然的。

归纳假设：结论对$k-1$轮成立。

归纳步骤：因为在$k-1$轮里各处理器处在相同状态，他们都发送相同的消息$m_r$到右边，同样的消息$m_l$到左边，所以在第$k$轮里，每处理器均接收右边的$m_l$，左边的$m_r$。因此，所有处理器在第$k$轮里接收同样的消息，又因为它们均执行同样的程序，故第$k$轮它们均处于同样的状态。

于是，若在某轮结束时，一个处理器宣布自己是leader(进入选中状态)，则其它处理器亦同样如此，这和A是一个leader选举算法的假定矛盾！对于同步环上的leader选举，匿名的、非一致性的算法。进一步地，对于同步环上的leader选举，不存在匿名的、一致性的算法。

# 8 ch34 P9 Ex3.2

由上述证明可知，同步环系统中不存在匿名的领导者选举算法。由于同步系统是异步系统的特殊情况，因此，异步环系统中也不存在匿名的领导者选举算法。

# 9 ch35 P39 Ex3.9

任取两个整数$0\le P,Q\le n-1$，则，

$$\begin{aligned} P &= \sum_{i=1}^{m} a_i\cdot 2^{i-1} \\ Q &= \sum_{i=1}^{m} b_i\cdot 2^{i-1} \end{aligned}$$

其中，$m=\lg n$，$0\le a_i,b_i\le 1$，则有，

$$\mathrm{rev}\,(P)=\sum_{i=1}^{m} a_i\cdot 2^{m-i}$$

设$P, Q$在同一片段上，$P_1, Q_1$也在同一片段上，且这两个片段相邻，并满足模运算的加法：

$$\begin{aligned} P_1 &= P + l \\ Q_1 &= Q + l \end{aligned}$$

其中，$l = 2^k$表示片段的长度。由$P, Q$在同一片段上知，

$$|P - Q| < l = 2^k$$

所以，存在$r\,(1 \leq r \leq k)$使得$a_r \neq b_r$，否则，$|P - Q| \geq l$。设$s = \min\,\{r\}$，则，

$$\operatorname{sign}(\operatorname{rev}(P) - \operatorname{rev}(Q)) = \operatorname{sign}(a_s - b_s)$$

而

$$\begin{aligned} P_1 &= P + l = \sum_{i=1}^{m} a_i \cdot 2^{i-1} + 2^k \\ Q_1 &= Q + l = \sum_{i=1}^{m} b_i \cdot 2^{i-1} + 2^k \end{aligned}$$

显然，$P, P_1$的前$k$位相同，$Q, Q_1$的前$k$位相同。由$1 \leq s \leq k$得，

$$\operatorname{sign}(\operatorname{rev}(P_1) - \operatorname{rev}(Q_1)) = \operatorname{sign}(a_s - b_s)$$

因此， 这两个相邻片段是序等价的。 根据等价关系的传递性， 所有片段都是序等价的。