分布式算法作业

联系：lightac@mail.ustc.edu.cn 作者：Zezhong

Problem 1

安全性（$safety$）板书：
思考：是否对于每一执行的每一断言都成立的断言/性质就是不变式呢？
答：

对于每一执行的每一个断言都成立的性质不一定是不变式。因为在这个过程之中可能会有状态无法达到，这样尽管每一次执行的每一个断言都成立，但是它仍然不是不变式。

不变式的定义：

断言P是S的一个不变式，如果(1)对所有的$\gamma \in I$，$P(\gamma)$成立，且(2)$P \Rightarrow P$
例子，假设一个整数变量x。假设x初始值为0。状态转移方式为：

```
    if x == 1 then x = 2
```

设断言：

$$x < 2 \tag{1}$$

显然，断言如同公式(1)所示都成立。但是不变式要求中要求每次转移过程之中保持不变，当$x = 1$式断言成立，当经过转移后$x = 2$时，断言并不成立。因此，不满足不变式定义中的(2)条件。

Problem 2

分布式算法$PPT\ \ ch33$：$P9\ \ Ex$
证明：图G里一结点从p_r可达当且仅当它曾设置过自己的$parent$变量。
答：

```
//算法2.2
//构造生成树（code for pi 0≤i≤n-1）
//初值：parent=nil；集合children和other均为ϕ
upon receiving no message:
```

```
5       if i=r and parent=nil then { //根尚未发送M
6           send M to all neighbors;
7           parent:=i;} //根的双亲置为自己
8   upon receiving M from neighbor pj
9       if parent=nil then {//pi此前未收到过M，M是pi收到的第1个msg
10          parent:=j;
11          send <parent> to pj; //pj是pi的双亲
12          send M to all neighbors except pj;
13      }else //pj不可能是pi的双亲，pi收到的M不是第1个msg
14          send<reject> to pj;
15  upon receiving <parent> from neighbor pj:
16      children:=children∪{ j }; //pj是pi的孩子，将j加入孩子集
17      if children∪other 包含了除parent外的所有邻居 then terminate;
18  upon receiving <reject> from neighbor pj:
19      other:=other∪{ j }; //将j加入other，通过非树边发送的msg。
20      if children∪other包含了除pi的双亲外的所有邻居 then terminate
```

1. **证明：必要性**

图G里一结点从p_r可达则它曾设置过自己的$parent$变量。由于图G是消息传播过程之中不断被生成的，因此任意一个结点在收到$message$之后才会被加入到图中，这样这个结点才会可达。当这个结点收到了$message$，由于是容许执行，程序会执行8至14行，判断它的$parent$是否为空。如果为空，则为设置自己的$parent$变量。如果不为空，则代表已经设置过了。

2. **证明：充分性**

图G里一结点设置过自己的$parent$变量则该结点从p_r可达。首先，$message$是从p_r发出的。如果结点设置过自己的$parent$变量则代表程序执行了代码的第10行，那么代码的第8行程序也执行过了。因此结点收到了$message$，综合$message$是从p_r发出的，可以推断出该结点是从p_r可达的。

Problem 3

分布式算法$PPT\ \ ch33$：$P30\ \ Ex2.1$

分析在同步和异步模型下，$convergecast$**算法的时间复杂性。**

答：

$Convergecast$**算法：**

每个叶子结点p_i发送x_i至双亲。//启动者

对每个非叶结点p_j，设p_j有k个孩子$p_{i1},...,p_{ik}$，p_j等待k个孩子的msg v_{i1}，v_{i2}，⋯，v_{ik}，当p_j收到所有孩子的msg之后将v_j=$max(x_j,v_{i1},...,v_{ik})$发送到$p_j$的双亲。

1 同步模型：

引理：在汇集算法的每个容许执行里，树中每个高为t子树根结点在第t轮里收到所有$children$的$message$。

利用数学归纳法：$t=1$时，每个叶子结点在第1轮里收到来自于自身的$message$。假设树上每个高为$t-1$的子树根结点在第$t-1$轮收到了$message$。假设p_i到叶子结点的距离为t，设p_j是p_i的$children$。因为p_j到叶子结点的距离为$t-1$。则p_j在$t-1$轮收到$message$。通过算法过程可知，在第t轮p_i将收到来自于p_j的$message$。

汇集是从所有结点收集信息至根。如果这个过程是同步的那么整个汇集过程的轮数为树的高度d。因此对于同步模型，存在汇集的时间复杂度为$O(d)$的算法。其中d为树的高度。

2 异步模型：

引理：在汇集算法的每个容许执行里，树中每个高为t子树根结点至多在第t轮里收到所有$children$的$message$。

利用数学归纳法：$t=1$时，每个叶子结点至多在第1轮里收到来自于自身的$message$。假设树上每个高为$t-1$的子树根结点至多在第$t-1$轮收到了$message$。假设p_i到叶子结点的距离为t，设p_j是p_i的$children$。因为p_j到叶子结点的距离为$t-1$。则p_j至多在$t-1$轮收到$message$。通过算法过程可知，至多在第t轮p_i将收到来自于p_j的$message$。

汇集是从所有结点收集信息至根。如果这个过程是异步的那么整个汇集过程的轮数至多为树的高度d。因此对于异步模型，存在汇集的时间复杂度为$O(d)$的算法。其中d为树的高度。

Problem 4

分布式算法$PPT\ \ ch33$：$P30\ \ Ex2.3$

证明$Alg2.3$构造一棵以P_r为根的DFS树。

答：

```
1  //算法2.3 构造DFS生成树，以Pr为根
2  Code for processor Pi, 0≤i ≤ n-1
3  var parent: init nil;
4  children: init ϕ;
5  unexplored: init all the neighbors of Pi
6  //未访问过的邻居集
7  upon receiving no msg:
8      if (i=r) and (parent=nil) then { //当Pi为根且未发送M时
9          parent := i; //将parent置为自身的标号
10         任意Pj ∈ unexplored;
11         将Pj从unexplored中删去; //若Pr是孤立结点,4-6应稍作修改
12         send M to Pj;
```

```
13     }//endif
14 upon receiving M from neighbor Pj:
15     if parent=nil then { //Pi此前未收到M
16         parent := j; //Pj是Pi的父亲
17         从unexplored中删Pj
18         if unexplored ≠ϕ then {
19             任意Pk ∈ unexplored;
20             将Pk从unexplored中删去;
21             send M to Pk;
22         } else send <parent> to parent; //当Pi的邻居均已访问过，返回到父亲
23     }else send <reject> to Pj; //当Pi已访问过时
24 upon receiving <parent> or <reject> from neighbor Pj:
25     if received <parent> then add j to children;
26     //Pj是Pi的孩子
27     if unexplored = ϕ then { //Pi的邻居均已访问
28         if parent ≠ i then send <parent> to parent;
29         //Pi非根，返回至双亲
30         terminate; //以Pi为根的DFS子树已构造好!
31     }else { //选择Pi的未访问过的邻居访问之
32         Pk ∈ unexplored;
33         将Pk从unexplored中删去;
34         send M to Pk;
35     }
```

1 连通性：

反证法。若图G不连通。则在图G中存在相邻结点P_j和P_i。P_j从P_r可达但P_i从P_r不可达。不可达意味着P_i的$parent$没有被设置为空且P_i不是P_j的$children$。但是，P_j可达意味着P_j的$parent$不为空且P_i属于$unexplored$集合之中。算法的14至21行决定了P_j会向P_i发送$message$并使得，P_i的$parent$设置为P_j且P_i成为P_j的$children$。因此这个过程之中产生矛盾，故图G是连通的。

2 无环性：

反证法。假设图G中存在一个环$P_1, P_2, P_3, ..., P_t, P_1$。不妨设$P_1$为环中最先收到$message$的点。当$P_t$收到消息时，它会将消息传递给$P_1$，此时$P_1$的$parent$已经不为空，所以根据代码第23行。$P_1$将向$P_t$发送一个$reject$消息且不会讲$P_t$设置为$parent$，同时由于$P_t$没有收到$P_1$发送的$parent$消息，因此不会将$P_1$设置成为自己的$children$。由于环的存在$P_1$是从$P_t$可达的。那么，$P_1$应该是$P_t$的$children$。产生矛盾。因此图$G$是无环的。

3 DFS树：

只需证明在有子结点和兄弟结点没有访问时，子结点总是优先加入树中即可。设存在结点P_x，P_y与P_t直接相连。假设P_x为访问的子节点（程序在第18到21行先访问的），P_y为兄弟结点。那么P_t只会在P_x向P_t发送一个

$parent$只会才会向P_y发送$message$（第24行）。又由程序可知P_x向所有的相邻结点都发送过$message$之后才会向P_t发送$parent$（第27至30行）。因此，P_x的子结点永远优先于P_y先收到消息，先加入树G。综上，G为DFS树。

Problem 5

分布式算法$PPT\ \ ch33$：$P30\ \ Ex2.4$

证明$Alg2.3$的时间复杂性为$O(m)$。

答：

消息复杂度：

对于任意一条边来说，可能传递的消息最多有4个，即两个M其中2个相应M的消息（$parent$，$reject$），因此消息复杂度为$O(m)$。其中m为图的边数。

1 同步模型：

在每一轮中，根据算法可知，有且只有一条消息（M，$parent$，$reject$）在传递，从代码的第12、21、23、28、34行发送消息的语句中分析可以发现消息只发送一个结点，除去根节点外，所有的处理器都是在收到消息之后才能被激活，所以，不存在多个处理器在同一轮发消息的情况，因此时间复杂度与消息复杂度一致均为$O(m)$。其中m为图的边数。

2 异步模型：

对于异步算法分析算法可知，一个时刻内至多一条消息（M，$parent$，$reject$）在传递，因此时间复杂度与消息复杂度是一致的。均为$O(m)$。其中m为图的边数。

Problem 6

分布式算法$PPT\ \ ch33$：$P30\ \ Ex2.5$

修改$Alg2.3$获得一新算法，使构造DFS树的时间复杂性为$O(n)$，并证明。

答：

1 方法一：

可以考虑在每个处理器中维护一本地变量（变量之中存储访问信息），同时添加一消息类型，在处理器P_i转发$message$时，发送消息$notice$通知其余的未访问过的邻居，这样其邻居在转发$message$时便不会向P_i转发，这样时间复杂度为$O(n)$。其中n是图中的点数。

2 方法二：
在消息$message$和$parent$信息中维护一发送数组，记录已经转发过$message$的处理器名称，这样可以使得每个处理器被发送一次。
两种方式都是避免向已转发过$message$的处理器发送消息$message$，这样DFS树外的边不再耗时，时间复杂度也降为$O(n)$。

Problem 7

分布式算法$PPT\ \ ch34$：$P9\ \ Ex3.1$
证明同步环系统中不存在匿名的、一致性的领导者选举算法。
答：

$Lemma3.1$：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。

证明过程：在匿名系统中，每个处理器在系统中具有相同的状态机。设R是大小为$n>1$的环（非均匀），由 $Lemma3.1$ 可知，设算法A 是使环R上某个处理器为 $leader$ 的算法。因为环是同步的，且只有一种初始配置。在每轮里，各处理器均发出同样的$message$，所以在各轮里各个处理器接收到相同的$message$，则状态改变也相同。这也表明R上A唯一合法执行。所以所有处理器要么同为$leader$，要么同时不为$leader$。故同步环系统中匿名的、一致性的领导者选举算法的算法是不存在的。

Problem 8

分布式算法$PPT\ \ ch34$：$P9\ \ Ex3.2$
证明异步环系统中不存在匿名的领导者选举算法。
答：

证明过程：每个处理器的初始状态和状态机相同，除了接收消息的时间可能不同外，接收到的消息序列也相同。所以最终处理器的状态也是一致的。由于处理器处理一条消息至多需要 1 单位时间，若某时刻某个处理器宣布自己是 $leader$（接收到了m条消息），则在有限时间内（m个时间单位内），其它处理器也会宣布自己是 $leader$。故异步环系统中匿名的领导者选举算法是不存在的。

Problem 9

分布式算法$PPT\ \ ch35$：$P39\ \ Ex3.9$

证明：若将环R_n^{Rev}划分为长度为j(j是2的方幂)的连续片断，则所有这些片断是次序等价的。

答：

序等价定义：两个环$x_0, x_1, ..., x_{n-1}$和$y_0, y_1, ..., y_{n-1}$是(次)序等价的，若对每个i和j，$x_i<x_j$，当且仅当$y_i<y_j$。

证明过程：若j为片段的长度，不妨设$j = 2^k$。假设m_1和m_2在同一个片段上，n_1和n_2在同一个片段上，且这个两个片段相邻。显然m_1与m_2有公式(2)和(3)的表示方法。

$$m_1 = \sum_{i=0}^{logj-1} a_i \times 2^i \tag{2}$$

$$m_2 = \sum_{i=0}^{logj-1} b_i \times 2^i \tag{3}$$

m_1和m_2在同一个片段上，可得：

$$|m_1 - m_2| < j = 2^k \tag{4}$$

由公式(4)分析可知一定存在$t \in [1, k]$使得$a_t \neq b_t$。反证法，若任意$t \in [0, k]$使得$a_t \neq b_t$，则由于$m_1 \neq m_2$，因此$|m_1 - m_2| \geq j$。与m_1和m_2在同一个片段上发生矛盾。取$t_0 = min\{t\}$，最小下标经过反转之后得到最大下标，因此：

$$sign(rev(m_1) - rev(m_2)) = sign(a_{t_0} - b_{t_0}) \tag{5}$$

其中$rev(m)$是m的二进制表示的逆序列。同时，

$$n_1 = m_1 + j = \sum_{i=0}^{logj-1} a_i \times 2^i + 2^k \tag{6}$$

$$n_2 = m_2 + j = \sum_{i=0}^{logj-1} b_i \times 2^i + 2^k \tag{7}$$

由公式(6)和(7)可知，m_1和n_1以及m_2和n_2这两对前k位相同。由于$t_0 \in [0, k]$，因此有：

$$sign(rev(n_1) - rev(n_2)) = sign(a_{t_0} - b_{t_0}) \tag{8}$$

综上，这两个片段是序等价，根据等价关系具有传递性，可知所有的片段都是次序等价的。