Ex 2.1 分析在同步和异步模型下，convergecast 算法的时间复杂性
解：（参考 PPT 中关于 broadcast 的证明，下面仅给出同步模型下的分析）
同步模型下：在 convergecast 算法的每个容许执行里，树中每个高度为 t 的处理器在第 t 轮里接收到所有孩子节点的消息。使用归纳法证明：

归纳前提：当 d = 1 时，Pr 的孩子就是叶子节点，所以在第 1 轮时，Pr 接收到所有孩子节点的消息。

归纳假设：树中每个高度为 t-1 的处理器在第 t-1 轮里接收到所有孩子节点的消息。

归纳步骤：设 Pi 高度为 t，则 Pi 的所有孩子节点高度为 t-1。由归纳假设，在第 t-1 轮，Pi 的所有孩子节点都能收到其孩子节点的消息。根据算法描述，在第 t 轮，Pi 将收到所有孩子节点的消息。

如果记 d 为树的高度，根据上述讨论，同步 convergecast 算法的时间复杂度为 d。

Ex 2.2 证明在引理 2.6 中，一个处理器在图 G 中是从 P_r 可达的，当且仅当它的 parent 变量曾被赋过值。

解：

充分性：一个处理器 Pi 在图 G 中是从 Pr 可达的，则它能收到 Pr 的 msg，根据算法描述，当 Pi 第一次收到 msg 时，将会对 parent 变量赋值。

必要性：若结点的parent变量被赋值过，则可知算法的第7行被执行，由于是容许执行，故算法的第5行也被执行，即收到过msg，而msg是从Pr发出的，所以该结点是可达的。

Ex 2.3 证明 Alg2.3 构造了一棵以 pr 为根的 DFS 树
解：
先证 Alg2.3 构造了一棵以 pr 为根的有向生成树；（参考 PPT，依次证明连通性和无环性）

再证该生成树为 DFS 树
只需证明在有子结点与兄弟结点未访问时，子结点总是先加入树中。设有节点P1，P2和P3。P2和P3是P1的直接相邻节点。P1在第12~14行中先选择向P2发送M，则P1当且仅当P2向其返回一个<parent>（第17行，第22行）时才有可能向P3发送M。而P2仅在其向所有的相邻节点发送过M后才会向P1返回<parent>。所以P2的子节点是永远先于P3加入树中的，即G是DFS树。

Ex2.4 证明 Alg2.3 的时间复杂性为 O(m)。
解：
同步模型：每一轮中，根据算法，有且只有一个消息(M or Parent or Reject)在传输，从算法的第 6 、14、16、20、25 行发送消息的语句中可以发现：消息只发往一个处理器结点，除根结点外，所有的处理器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，所以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：对任一边，可能传输的消息最多有 4 个，即 2 个 M ，2 个相应 M 的消息（Parent or Reject），所以消息复杂度为 O(m)

Ex 2.5 修改 Alg2.3 获得一新算法，使构造 DFS 树的时间复杂性为 O(n)。

解：**（算法思想）**（1）在每个处理器中维护一个本地变量，同时添加一个消息类型，在处理器 Pi 转发 M 时，发送消息 N 通知其余的未访问过的邻居，这样其邻居在转发 M 时便不会向 Pi 转发。（2）在消息 M 和<parent>中维护一个发送数组，记录已经转发过 M 的处理器名称。两种方式都是避免向已转发过 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 Pi (0≤i≤n-1)
var parent: init nil;        //变量
    children: init Φ
    unexplored: int Pi 的所有邻居
    explored: init Φ //加入已发送 M 的邻居集合

upon receiving no msg:       //消息处理
  if (i=r) and (parent=nil) then {
    parent:=i;
    对任意的 Pj∈unexplored{
        unexplored:= unexplored \ {Pj} //删去 Pj
        explored:=explored∪{Pj} //已发送 M 的邻居并上 Pj
        send m to Pj
    }
upon reciving m form neighbor Pj:
    if parent=nil then {
        parent:=j;
        unexplored:= unexplored except Pj
        if unexplored≠Φ then{
            对任意的 Pk∈unexplored and Pk 不属于 explored:
            unexplored:= unexplored \ {Pk}
            explored:=explored∪{Pk}
            send m to Pk
        }
        else

            send <parent> to parent;
}
upon receiving <parent> from pj
    if unexplored≠Φ then {
        if parent ≠ r then
            send <parent> to parent
            terminate;
    }
    else {
        对任意的 Pk∈unexplored and Pk 不属于 explored:
        unexplored:= unexplored \ {Pk}
        explored:=explored∪{Pk}
        send m to Pk
}
```

Ex 3.1 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.

解：在同步系统中，一个算法以轮的形式进行。每轮中所有待发送 msg 被传递，随后每个处理器进行一步计算。根据 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。也就是说，在匿名环中，处理器间始终保持对称，他们的初始状态一致，在每一轮中处理器接收相同的 msg，执行相同的操作。所以到终止状态时，所有处理器的状态仍然是一样的。如果选中一个处理器为 leader，那么其他处理器也被选中。这是不合理的。因此在同步环系统中不存在匿名的、一致性的 leader 选举算法。

Ex3.2 证明异步环系统中不存在匿名的领导者选举算法

解：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，则在有限时间内，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。

3.3 若将环 Rrev 划分为长度为 j(j 是 2 的方幂)的连续片段，则所有这些片段是次序等价的。

证： 对一个整数 $P(0 \le P \le n-1)$,可以表示为：$P=\sum_{i=0}^{m} a_i * 2^i$，其中 $m = \lg n$

则有 $rev(P)= \sum_{i=0}^{m} a_i * 2^{m-i}$。

设P、Q在同一个片段上，P′、Q′在同一片段上，且设这两个片段时相邻的，由模运算的加法可得：

$$P'=P+2^k$$
$$Q'=Q+2^k,$$

2^k表示片段的长度。又有

$$P=\sum_{i=0}^{m} a_i * 2^i$$
$$Q=\sum_{j=0}^{m} b_j * 2^j$$

且P、Q在同一个片段上，有

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

故存在 $a_r \ne b_r$，$0 \le r < k$。不然，$|P-Q| > 2^k$，与P、Q在同一个片段矛盾。

现设 $s = \min\{r\}$，那么代入 rev(P)与 rev(Q)，可得

$$\text{sign}\left(rev(P) - rev(Q)\right) = \text{sign}\ (a_r - b_r),$$

又有

$$P'=\sum_{i=0}^{m} a_i * 2^i+2^k$$
$$Q'=\sum_{j=0}^{m} b_j * 2^j+2^k,$$

可得，P与P′前k位（第0位至第k−1位）相同，Q与Q′前k位（第0位至第k−1位）相同。且P′、Q′亦在同一个片段上，有

$$|P'-Q'| < 2^k$$

存在 $a_t \ne b_t$，$0 \le t < k$。那么，

$$\text{sign}\left(rev(P') - rev(Q')\right) = \text{sign}\ (a_t - b_t)$$

由此可知，这两个片段是序等价的，片段选取的随机性及等价的传递关系，即可得到所有的片段为序等价的。