**Algorithm 1** Block creation

**function** $BlockCreation()$
    $TODO$

**Algorithm 2** Block proposal

**function** $BlockProposal(Block\ B,\ Accounts\ A)$
    **for** $a \in A$ **do**
        $< sorthash, \pi, j > \leftarrow Sortition(a.sk, seed, t, role, a.w[], W)$
        **if** $j > 0$ **then**
            $priority \leftarrow Min_{n \in [1,j]} Hash(sorthash||n)$
            $GOSSIP\_MESSAGE(priority, SIGNED_{a.sk}(< sorthash, \pi >))$
    $msgs \leftarrow incomingMsgs[round, step = 0].iterator()$
    **while** $elapsedtime < proposaltime$ **do**
        $m \leftarrow msgs.next()$
        $VerifySortition(m.sorthash, m.pi, m.pk)$
        **if** $m.priority < minSeenPriority$ **then**
            $minSeenPriority = m.priority$
    **while** $elapsedtime < fullblocktime$ **do**
        $m \leftarrow msgs.next()$
        $< priority, sorthash, \pi > \leftarrow m$

**Algorithm 3** Soft Vote

**function** $SoftVote$
    $CommitteeVote(ctx, round, REDUCTION\_ONE, tstep, hblock)$
    $hblock \leftarrow CountVotes(ctx, round, REDUCTION\_ONE, Tstep, tstep, lblock + lstep)$
    $empty\_hash \leftarrow H(Empty(round, H(ctx.last\_block)))$
    **if** $hblock = TIMEOUT$ **then**
    $CommitteeVote(ctx, round, REDUCTION\_TWO, tstep, empty\_hash)$ **else**
    $CommitteeVote(ctx, round, REDUCTION\_TWO, tstep, hblock1)$
    $hblock \leftarrow CountVotes(ctx, round, REDUCTION\_TWO, Tstep, tstep, lblock + lstep)$
    **if** $hblock = TIMEOUT$ **then return** $empty\_hash$ **else return** $hblock$

**Algorithm 4** CertifyVote

```
1: function CertifyVote(hblock)
2:     while step < 256 do
3:
4:         < hblock, bConfirmed > ← BLOCK_STEP(hblock, step)
5:         if bConfirmed then return hblock
6:         step++
7:
8:         < hblock, bConfirmed > ← EMPTY_STEP(hblock, step)
9:         if bConfirmed then return hblock
10:        step++
11:
12:        CommonCoinFlipVote(hblock, step)
13:        step++
14:
15:        HangForever()
16:        =0
```

**Algorithm 5** BlockConfirmation

```
1: function BlockConfirmation(hblock)
2:     r ← CountVotes(ctx, round, FINAL, Tfinal, tfinal, lstep)
3:     if r = hblock then
4:     return < FINAL, BlockOfHash(hblock) > else
5:     return < TENTATIVE, BlockOfHash(hblock) >
   =0
```