Algorithm 1: Handling Inconsistent Blockchain Views (G_0)

```
   Input: Blockchain graph $G_0$
   Output: Main View $v_0$
1  $V_0 \leftarrow$ FindViews($G_0$) // All different views
2  foreach $v \in V_0$ do
3      foreach $v' \in$ Subview($v$) do
4          $W(v') \leftarrow 0$
           // Initialize mining power of each subview
5      end
6  end
7  foreach $B \in G$ do
8      foreach $v \in V_0$ do
9          foreach $v' \in$ Subview($v$) do
10             if $B \in v$ then
11                 $W(v') \leftarrow W(v') +$ Miningpower($B$)
12             end
13         end
14     end
       // Compute mining power invested in the view
15 end
16 foreach $v \in V_0$ do
17     if Subview($v$) $= \varnothing$ then
18         return $v$
19     else
20         Update $v \leftarrow \underset{v' \in \text{subview}(v)}{\arg\max} W(v')$
21         Go to line 17
22     end
23 end
24 return $v_0$
```

Algorithm 2: FindViews(G)

```
Input: Graph $G$
Output: View Set $V_s$
$V_s \leftarrow \varnothing$
$leaf\_group \leftarrow$ GroupLeaf($G$)
foreach $\mathcal{S} \in leaf\_group$ do
    $u, v \leftarrow \varnothing$
    $v, v.subview \leftarrow$ AddPre($\mathcal{S}, G, v, U$)
    $V_s \leftarrow V_s \cup \{v\}$
end
Function Addpre($\mathcal{S}, G, v, U$):
    $v \leftarrow v \cup \mathcal{S}$
    $U \leftarrow U \cup \{v\}$
    if $\mathcal{S}$ is genesis then
        return $v, U$
    else
        Addpre($\mathcal{S}.predecessors, G, v, U$)
    end
```