Algorithm 1 SCC 更新算法-SCC 合并

Input: SCC 图 G_S, 更新节点 S_i

Output: 更新后的 SCC 图 G_{Snew}, 被合并的节点集合 S, 合并后新节点的编号 S_{new}

```
1:  function SCCMerge($G_S$, $S_i$)
2:      $S \leftarrow \emptyset$
3:      $G_{Snew} \leftarrow G_S$
4:      $S_{old} \leftarrow -1, S_{new} \leftarrow -1$
5:      $cycle$ = findCycle($G_{Snew}, S_i$)
6:      while $|cycle| != 0$ do
7:          $G_{Snew}, S_{new}$ = Merge($G_{Snew}$, $cycle$)
8:          if $|S| == 0$ then
9:              $S = S \cup cycle$
10:         else
11:             $S = S \cup (cycle - S_{old})$
12:         end if
13:         $S_{old} = S_{new}$
14:         $cycle$ = findCycle($S_{new}, S_{new}$)
15:     end while
16:     return $G_{Snew}, S, S_new$
17: end function
```

Algorithm 2 SCC 更新算法-寻找环路

Input: SCC 图 G_S, 更新节点 S_i

Output: 返回一个包含环的节点集合 C

```
 1: function findCycle($G_S, S_i$)
 2:     $C \leftarrow \emptyset, visited \leftarrow \emptyset$
 3:     for each node $v$ in $G_S$ do
 4:         $visited \cup \{v : FALSE\}$
 5:     end for
 6:     stack $S$
 7:     $S$.push($S_i$)
 8:     while $!S.empty()$ do
 9:         $N \leftarrow S$.pop()
10:         for each outcome edge $e$ of node $N$ do
11:             if $e.dst == S_i$ then
12:                 $C \leftarrow S$
13:                 break
14:             end if
15:             if $!visited[e.dst]$ then
16:                 $visited[e.dst] = TRUE$
17:                 $S$.push($e.dst$)
18:             end if
19:         end for
20:     end while
21:     return $C$
22: end function
```

Algorithm 3 SCC 更新算法-合并环路

```
Input: SCC 图 $G_S$, 在环中的节点集合 $cycle$
Output: 返回更新过后的 SCC 图 $G_{Snew}$, 新节点的编号 $S_{new}$
 1: function MERGE($G_S, cycle$)
 2:     SCCNode $N_{new}$
 3:     for each node $N_c$ in $cycle$ do
 4:         $N_{new}.nodeSet \leftarrow N_{new}.nodeSet \cup N_c.nodeSet$
 5:     end for
 6:                 ▷ 需要注意节点 ID 的重用, SCC 节点的主键是原始图节点集合
 7:     for each node $S$ in all SCC Graphs $[G_{S0}, G_{S1}, ..., G_{SN}]$ do
 8:         if $S.nodeSet == N_{new}.nodeSet$ then
 9:             $N_{new}.nodeID \leftarrow S.nodeID$
10:             $reused = TRUE$
11:         end if
12:     end for
13:     if !$reused$ then                    ▷ 节点 ID 未被重用, 获取一个新的 ID
14:         $N_{new}.nodeID$ = GETNEWSCCID
15:     end if
16:     for each node $N_s$ in $G_S$ do
17:         if $N_s$ in $cycle$ then                ▷ 需要将所有的出边合并到新节点中
18:             for each outcome edge $e$ of $N_s$ do
19:                 $N_{new}$.insertOutcomeEdge($e.dst$)
20:             end for$G_{Snew} \leftarrow G_{Snew} - N_s$
21:         else                         ▷ 需要把出边在 $cycle$ 中的节点改为 $N_{new}$
22:             for each outcome edge $e$ of $N_s$ do
23:                 if $e.dst$ in $cycle$ then
24:                     $N_s$.deleteOutcomeEdge($e.dst$)
25:                 end if
26:             end for
27:             if exist edge $e$ where $e.dst$ in $cycle$ then
28:                 $N_s$.insertOutcomeEdge($N_{new}.nodeID$)
29:             end if
30:         end if
31:     end for
32: end function
```