Marden Clustering Algorithm

```
 1: function APPLYALG(network G_orig)
 2:     allClusters ← []
 3:     G ← G_orig
 4:     change ← true
 5:     while change do
 6:         L ← list of G's nodes in order of highest to lowest node degree
 7:         C ← []
 8:         while L not empty do
 9:             mark all nodes of G as unchecked
10:             seed ← first element of L
11:             seedAdjList ← list of nodes adjacent to seed
12:             cluster ← [seed]
13:             cluster ← FINDCLUSTERS(seed, seed, seedAdjList, cluster)
14:             remove all nodes in cluster from L
15:             add cluster to C
16:         end while
17:         G_new ← condensed network of G where the nodes are the clusters in C
18:         if G == G_new then
19:             change ← false
20:             append [C] to allClusters
21:         else
22:             G ← G_new
23:         end if
24:     end while
25:     return allClusters
26: end function

27: function FINDCLUSTERS(n, seed, seedAdjList, cluster)
28:     for each node nAdj adjacent to n do
29:         if nAdj is unchecked then
30:             mark nAdj as checked
31:             score ← FINDSCORE(nAdj, seed, seedAdjList)
32:             if score ≥ 0 then
33:                 add nAdj to cluster
34:                 FINDCLUSTERS(nAdj, seedAdjList, cluster)
35:             end if
36:         end if
37:     end for
38:     return cluster
39: end function

40: function FINDSCORE(n, seed, seedAdjList)
41:     score ← 0
42:     seedAdj ← false
43:     for each node nAdj adjacent to n do
44:         if nAdj in seedAdjList then
45:             score ← score+weight of edge connecting nAdj and n
46:         end if
47:         if nAdj == seed then
48:             seedAdj ← true
49:         end if
50:     end for
51:     if seedAdj and n in seedAdjList then
52:         score ← score+weight of edge connecting n and seed
53:     end if
54:     return score
55: end function
```