
Algorithm 1 Randomized sampling algorithm for estimating the impact of surveillance regulation.

```
1: procedure REACH(degree, hops, hubs)
2:   if hops = 0 then
3:     return 1 ▷ With no hops, include just the seed.
4:   end if
5:   if hops = 1 then
6:     return degree + 1 ▷ With one hop, include the neighbors and seed.
7:   end if
8:   if hops = 2 then
9:     connectedHubs ← {}
10:    connectedPopulation ← 0
11:    for hub in hubs do
12:      if count(connectedHubs) = degree then
13:        break
14:      end if
15:      if random() < connectivity(hub) then
16:        connectedHubs = connectedHubs ∪ hub
17:        connectedPopulation + = connectivity(hub)
18:      end if
19:    end for
20:    reach ← connectedPopulation · subscriberBase ▷ Estimate two-hop connectivity through hubs.
21:    reach + = (degree - count(connectedHubs)) · (degree - 1) ▷ Estimate two-hop connectivity through individual subscribers.
22:    reach + = degree ▷ Estimate one-hop connectivity.
23:    reach + = 1 ▷ Include the seed.
24:    return min(reach, subscriberBase)
25:  end if
26:  if hops = 3 then
27:    connectedHubs ← {}
28:    connectedPopulation ← 0
29:    for hub in hubs do
30:      if count(connectedHubs) = degree then
31:        break
32:      end if
33:      if random() < connectivity(hub) then
34:        connectedHubs = connectedHubs ∪ hub
35:        connectedPopulation + = connectivity(hub)
36:      end if
37:    end for
38:    reach ← connectedPopulation · subscriberBase · degree ▷ Estimate three-hop connectivity through first-hop hubs.
39:    firstHopHubs ← connectedHubs
40:    remainingHubs ← hubs - firstHopHubs
41:    connectedPopulation ← 0
42:    for i in [0, degree - count(firstHopHubs)] do
43:      secondHopHubs ← {}
44:      for hub in remainingHubs do
45:        if random() < connectivity(hub) then
46:          secondHopHubs = secondHopHubs ∪ hub
47:          if hub not in connectedHubs then
48:            connectedPopulation + = connectivity(hub)
49:            connectedHubs = connectedHubs ∪ hub
50:          end if
51:        end if
52:      reach + = 1 ▷ Add the individual one-hop node.
53:      reach + = (degree - 1 - count(secondHopHubs)) · degree ▷ Add the second- and third-hop nodes that are individual subscribers.
54:      reach + = count(secondHopHubs) ▷ Add the second-hop nodes that are hubs.
55:    end for
56:    reach + = connectedPopulation · subscriberBase · degree ▷ Add the third-hop nodes through second-hop hubs.
57:  end for
58:  return min(reach, subscriberBase)
59: end if
60: end procedure
```
