```
 1 /* Main procedure */
 2 bc[i] ← 0, for i = 0..n − 1;
 3 Set the adjacency arrays a₁[i], a₂[i], for i = 0..2m − 1;
 4 d[i], σ[i], δ[i] ← 0, for i = 0..n − 1;
 5 p[i][j] ← 0, for i = 0..n − 1, j = 0..n − 1;
 6 Set up APSP kernel execution configuration: grid₁, threads₁;
 7 Set up back propagation kernel execution configuration: grid₂, threads₂;
 8 for i ∈ 0..n − 1 do
 9 │   /* APSP */
10 │   continue ← true;
11 │   dist ← 0;
12 │   while continue do
13 │   │   apsp_kernel<<<grid₁,threads₁>>>(a₁, a₂, d, σ, p, done, dist);
14 │   │   dist++;
15 │   end
16 │   /* Back propagation */
17 │   done ← false;
18 │   while dist>1 do
19 │   │   back_prop_kernel<<<grid₁,threads₁>>>(a₁, a₂, d, σ, δ, p, dist);
20 │   │   Sync thread blocks;
21 │   │   back_sum_kernel<<<grid₂,threads₂>>>(i, dist, d, δ, bc);
22 │   │   dist--;
23 │   end
24 end
25 return bc
26 /* APSP kernel */
27 procedure apsp_kernel(a₁, a₂, d, σ, p, done, dist)
28 foreach thread i do
29 │   u ← a₁[i], w ← a₂[i]; /* set the node ids for edge i */
30 │   if d[u]==dist then
31 │   │   if d[w]==-1 then
32 │   │   │   continue ← true; d[w] ← dist + 1;
33 │   │   end
34 │   │   if d[u]==dist+1 then
35 │   │   │   p[w][u] ← 1;
36 │   │   │   atomicAdd(σ[w],σ[u]);
37 │   │   end
38 │   end
39 end
40 /* Back propagation kernel */
41 procedure back_prop_kernel(a₁, a₂, d, σ, δ, p, dist)
42 foreach thread i do
43 │   u ← a₁[i]; w ← a₂[i]; /* set the node ids for edge i */
44 │   if d[u]==dist-1 then
45 │   │   if p[u][w]==1 then
46 │   │   │   atomicAdd(δ[w], σ[w]/σ[u] * (1 + δ[u]));
47 │   │   end
48 │   end
49 end
50 procedure back_sum_kernel(s, dist, d, δ, bc, n)
51 foreach thread i do
52 │   if i≠s && d[i]== dist-1 then
53 │   │   bc[i] ← bc[i] + δ[i];
54 │   end
55 end
```