Stage 2 - BFS traversal starting at u_{low}

```
while Q not empty do
    dequeue v ← Q;
    for all neighbor w of v do
        if d[w] = (d[v] + 1) then
            if t[w] = Not-Touched then
                enqueue w → Q_BFS;
                enqueue w → Q[d[w]];
                t[w] ← Down;
                d[w] ← d[v] + 1;
                dP[w] ← dP[v];
            else
                dP[w] ← dP[w] + dP[v];
            σ̂[w] ← σ̂[w] + dP[v];
```

Stage 3 - modified dependency accumulation

```
δ̂[v] ← 0, v ∈ ∀V; level ← V;
while level>0 do
    while Q[level] not empty do
        dequeue w ← Q[level];
        for all v ∈ P[w] do
            if t[v] =Not-Touched then
                enqueue v → Q[level − 1];
                t[v] ← Up;
                δ̂[v] ← δ[v];
            δ̂[v] ← δ̂[v] + (σ̂[v]/σ̂[w])(1 + δ̂[w]);
            if t[v] = Up ∧(v ≠ u_high ∨ w ≠ u_low) then
                δ̂[v] ← δ̂[v] − (σ[v]/σ[w])(1 + δ[w]);
            if w ≠ r then
                C_B[w] ← C_B[w] + δ̂[w] − δ[w];
    level ← level − 1;
σ[v] ← σ̂[v], v ∈ ∀V;
for v ∈ V do
    if t[v] ≠ Not-Touched then
        δ[v] ← δ̂[v], v ∈ ∀V
```