Algorithm 1 The pseudocode for Evaluate

```
num = 0, push(root, S)
DFN[root] = num; LOW[root] = num; ST_SIZE[root] = 1
VISISTED[root] = true; num = num + 1
repeat
    Vertex v = peek(S)
    y = next unvisited neighbor of v
    if y exists then
        mark y as visited, push(y, S)
        DFN[y] = num; LOW[y] = DFN[y]; PARENT[y] = v;
        ST_SIZE[y] = 1; IMPACT[y] = 0
        num = num + 1
    else
        pop(S)
        for all neighbors w of v do
            if DFN[w] < DFN[v] and PARENT[v] ≠ w then
                LOW[v] = min(LOW[v], DFN[w])
            else if PARENT[w] == v then
                LOW[v] = min(LOW[v], LOW[w])
                if !COUNTED[w] and (PARENT[v] ≠ w or v is root) then
                    COUNTED[w] = true
                    ST_SIZE[v] = ST_SIZE[v] + ST_SIZE[w]
                end if
                if LOW][w] ≥ DFN[v] and v ≠ root then
                    mark v as an articulation point
                    CUT_SIZE[v] = CUT_SIZE[v] + ST_SIZE[w]
                    IMPACT[v] = IMPACT[v] + f(ST_SIZE[w])
                end if
            end if
            if v is root and has more than one child then
                mark v as an articulation point
            end if
        end for
    end if
until Stack is empty
for each visited and counted vertex v do
    if v is an articulation point then
        IMPACT[v] = IMPACT[v] + f(num - CUT_SIZE[v])
    else
        IMPACT[v] = IMPACT[v] + f(num - 1)
    end if
    Maintain the vertex v* with the minimum IMPACT value
end for
return v*
```