Algorithm 2 `BOM_generator(n, root, depth, max_parents, min_demand, max_demand, seed)`

Let G be a directed graph
Let $leaf\_nodes$ be a list
Let $demand$ be a dictionary

$G \leftarrow$ `DiGraph()`
for $i \leftarrow 0$ **to** $root - 1$ **do**
    G.`add_node`(i)
end for
for $i \leftarrow root$ **to** $n - 1$ **do**
    $Possible\_parents \leftarrow G$.`nodes`$()$
    $num\_parents \leftarrow \min(|Possible\_parents|, max\_parents)$
    $Parents \leftarrow$ `Uniform_sample`$(Possible\_parents, num\_parents)$
    G.`add_node`(i)
    for each $parent \in Parents$ **do**
        G.`add_edge`$(parent, i,$ `weight` $=$ `Uniform_integer`$(1, 10))$
    end for
end for
for each $node \in G$.`nodes`$()$ **do** ▷ Detecting non-connected nodes and adding edges
    if G.`in_degree`$(node) == 0$ **and** $node \geq root$ **then**
        $parent \leftarrow$ `Uniform_node_choice`(G)
        if $parent \neq node$ **then**
            G.`add_edge`$(parent, node,$ `weight` $=$ `Uniform_integer`$(1, 10))$
        end if
    end if
end for
for each $node \in G$ **do**
    if G.`out_degree`$(node) == 0$ **then**
        $leaf\_nodes$.`add`$(node)$
    end if
end for
if `is_weakly_connected`(G) **then**
    $components \leftarrow G$.`weakly_connected_components`(G)
    if $|components| > 1$ **then**
        for $i \leftarrow 0$ **to** $|components| - 2$ **do**
            $src \leftarrow components[i][-1]$
            $dest \leftarrow components[i + 1][0]$
            G.`add_edge`$(src, dest,$ `weight` $=$ `Uniform_integer`$(1, 10))$
        end for
    end if
end if
for each $i \in G$.`nodes`$()$ **do**
    if $i \in leaf\_nodes$ **then**
        $demand[i] \leftarrow$ `Uniform`(min_demand, max_demand)
    else
        $demand[i] \leftarrow 0$
    end if
end for

return BOM