Input: Two-dimensional flow direction ($\mathbb{F}$), drainage area ($\mathbb{A}$), upstream accumulation length ($\mathbb{L}$), critical drainage area (A_{cr}), and critical length (l_{cr})

Output: *Lines* is a list of centerlines, each represented by a list of grid cell indicies

`/* Initialization of the delineation */`

Let *Open* be an empty queue of lists of grid cell indicies;
foreach outlet (i, j) in the study area **do**
    if $\mathbb{A}(i, j) > 0$ **then**
        Open.push(single-element list $[(i, j),]$);

`/* Delineation of river centerlines */`

while *Open* is not empty **do**
    $L = Open$.pop();
    if L has no upstreams **then**
        Let the expected length $l \leftarrow \mathbb{L}(i, j)$, where (i, j) is the last grid cell of L;
    else
        Let the expected length $l \leftarrow \mathbb{L}(i, j) - \sum \mathbb{L}(i', j')$, where (i, j) is the last grid cell of L, (i', j') is the last grid cell of the upstream of L;
    loop
        Let N be a set of grid cells. Each of N: (1) is the neighbor of L, (2) flow into L, and (3) have not been included in any line in *Lines* or in *Open*;
        if Length of $L \geq l$ **or** N is empty **then**
            Lines.append(L);
            break;
        Find the grid cell (i_2, j_2) in N that has the largest $\mathbb{A}$;
        if $\mathbb{A}(i_2, j_2) \leq A_{\text{cr}}$ **then**
            Lines.append(L);
            break;
        Find (i', j') that is the grid cell that (i_2, j_2) flows to, (i_1, j_1) is the grid cell in L that also flows to (i', j');
        if (i', j') is the last grid cell of L **then**
            L.append((i_2, j_2));
        else
            `/* Bifurcation */`
            if Length of $L < l_{\text{cr}}$ **then**
                Lines.append(L);
                break;
            Split L into two sub-list: L_1 is the list before (i', j') (inclusive) and L_2 is the one after (i', j') (exclusive);
            Open.push(L_1);
            Open.push(L_2);
            Open.push(one-element list $[(i_1, j_1),]$);
            break;