1 Hadamard

Algorithm 1 CSR Hadamard Computation

Require: A($m \times n$), B($m \times n$)
Ensure: C($m \times n$)

```
c_pos ← 0
for row = 0 to m − 1 do
    C_IA[row] ← c_pos
    col_A_pivot ← A_IA[row]
    col_B_pivot ← B_IA[row]
    col_A_limit ← A_IA[row + 1]
    col_B_limit ← B_IA[row + 1]
    A_line_size ← col_A_limit − col_A_pivot
    B_line_size ← col_B_limit − col_B_pivot

    if A_line_size > B_line_size then
        for col_A_pivot to col_A_limit − 1 do
            for col_B_pivot to col_B_limit − 1, such that (A_JA[col_A_pivot] < B_JA[col_B_pivot]) do
                ++col_B_pivot
            end for
            if A_JA[col_A_pivot] == B_JA[col_B_pivot] then
                C_csr_values[c_pos] ← A_csr_values[c_pos] × B_csr_values[c_pos]
                C_JA[c_pos] ← col_A_pivot
                ++c_pos
            end if
            ++col_A_pivot
        end for
    else
        for col_B_pivot to col_B_limit − 1 do
            for col_A_pivot to col_A_limit − 1, such that (A_JA[col_A_pivot] < B_JA[col_B_pivot]) do
                ++col_A_pivot
            end for
            if A_JA[col_A_pivot] == B_JA[col_B_pivot] then
                C_csr_values[c_pos] ← A_csr_values[c_pos] × B_csr_values[c_pos]
                C_JA[c_pos] ← col_B_pivot
                ++c_pos
            end if
            ++col_B_pivot
        end for
    end if
end for

C_IA[at_row] = c_pos
```

2 Kathri-Rao

Algorithm 2 CSR Kathri-Rao Computation

Require: A($m \times n$), B($o \times n$)
Ensure: C($(m * o) \times n$)

$A1 \leftarrow A_{CSCformat}$
$B1 \leftarrow B_{CSCformat}$

Require: A1($n \times m$), B1($n \times o$)

$c1\_pos \leftarrow 0$
for $at\_column = 0$ **to** $number\_columns - 1$ **do**
    $C1\_IA[at\_column] \leftarrow c1\_pos$
    $line\_A\_pivot \leftarrow A1\_IA[at\_column]$
    $line\_B\_pivot \leftarrow B1\_IA[at\_column]$
    $line\_A\_limit \leftarrow A1\_IA[at\_column + 1]$
    $line\_B\_limit = B1\_IA[at\_column + 1]$

    for $line\_A\_pivot = 0$ **to** $line\_A\_limit - 1$ **do**
        $line\_B\_pivot \leftarrow B1\_IA[at\_column]$
        for $line\_B\_pivot = 0$ **to** $line\_B\_limit - 1$ **do**
            $C\_csc\_values[c1\_pos] \leftarrow A\_csc\_values[c1\_pos] \times B\_csc\_values[c1\_pos]$
            $C1\_JA[c1\_pos] \leftarrow line\_A\_pivot \times line\_B\_pivot$
            $++c1\_pos$
        end for
    end for
end for

$C1\_IA[at\_column] = c1\_pos$

$C \leftarrow C1^T$

3 Kronecker

Algorithm 3 CSR Kronecker Computation

Require: A($m \times n$), B($o \times p$)
Ensure: C($(m * o) \times (n * p)$)

$A1 \leftarrow A^T$
$B1 \leftarrow B^T$

Require: A1($n \times m$), B1($n \times o$)

$c1\_pos \leftarrow 0$
for $at\_column = 0$ **to** $number\_columns - 1$ **do**
    $C1\_IA[at\_column] \leftarrow c1\_pos$
    $line\_A\_pivot \leftarrow A1\_IA[at\_column]$
    $line\_B\_pivot \leftarrow B1\_IA[at\_column]$
    $line\_A\_limit \leftarrow A1\_IA[at\_column + 1]$
    $line\_B\_limit = B1\_IA[at\_column + 1]$

    for $line\_A\_pivot = 0$ **to** $line\_A\_limit - 1$ **do**
        $line\_B\_pivot \leftarrow B1\_IA[at\_column]$
        for $line\_B\_pivot = 0$ **to** $line\_B\_limit - 1$ **do**
            $C\_csc\_values[c1\_pos] \leftarrow A\_csc\_values[c1\_pos] \times B\_csc\_values[c1\_pos]$
            $C1\_JA[c1\_pos] \leftarrow line\_A\_pivot \times line\_B\_pivot$
            $++c1\_pos$
        end for
    end for
end for

$C1\_IA[at\_column] = c1\_pos$

$C \leftarrow C1^T$