1 Hadamard

Algorithm 1 CSR Hadamard Computation

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

```
1:  c_pos ← 0
2:  for row = 0 to m − 1 do
3:      C_IA[row] ← c_pos
4:      col_A_pivot ← A_IA[row]
5:      col_B_pivot ← B_IA[row]
6:      col_A_limit ← A_IA[row + 1]
7:      col_B_limit ← B_IA[row + 1]
8:      A_line_size ← col_A_limit − col_A_pivot
9:      B_line_size ← col_B_limit − col_B_pivot

10:     if A_line_size > B_line_size then
11:         for col_A_pivot to col_A_limit − 1 do
12:             for col_B_pivot to col_B_limit − 1, such that (A_JA[col_A_pivot] < B_JA[col_B_pivot]) do
13:                 ++ col_B_pivot
14:             end for
15:             if A_JA[col_A_pivot] == B_JA[col_B_pivot] then
16:                 C_csr_values[c_pos] ← A_csr_values[c_pos] × B_csr_values[c_pos]
17:                 C_JA[c_pos] ← col_A_pivot
18:                 ++ c_pos
19:             end if
20:             ++ col_A_pivot
21:         end for
22:     else
23:         for col_B_pivot to col_B_limit − 1 do
24:             for col_A_pivot to col_A_limit − 1, such that (A_JA[col_A_pivot] < B_JA[col_B_pivot]) do
25:                 ++ col_A_pivot
26:             end for
27:             if A_JA[col_A_pivot] == B_JA[col_B_pivot] then
28:                 C_csr_values[c_pos] ← A_csr_values[c_pos] × B_csr_values[c_pos]
29:                 C_JA[c_pos] ← col_B_pivot
30:                 ++ c_pos
31:             end if
32:             ++ col_B_pivot
33:         end for
34:     end if
35: end for

36: 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 $A\_at\_column = 0$ **to** $A\_number\_columns - 1$ **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 $B\_at\_column = 0$ **to** $B\_number\_columns - 1$ **do**

        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
end for

$C1\_IA[A\_at\_column + B\_at\_column] = c1\_pos$

$C \leftarrow C1^T$