

1 Hadamard

Algorithm 1 CSR Hadamard Computation

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

Ensure: $C(m \times n)$

```
1:  $c\_pos \leftarrow 0$ 
2: for  $row = 0$  to  $m - 1$  do
3:    $C\_IA[row] \leftarrow c\_pos$ 
4:    $col\_A\_pivot \leftarrow A\_IA[row]$ 
5:    $col\_B\_pivot \leftarrow B\_IA[row]$ 
6:    $col\_A\_limit \leftarrow A\_IA[row + 1]$ 
7:    $col\_B\_limit \leftarrow B\_IA[row + 1]$ 
8:    $A\_line\_size \leftarrow col\_A\_limit - col\_A\_pivot$ 
9:    $B\_line\_size \leftarrow col\_B\_limit - col\_B\_pivot$ 

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:         $C\_csr\_values[c\_pos] \leftarrow A\_csr\_values[c\_pos] \times B\_csr\_values[c\_pos]$ 
17:         $C\_JA[c\_pos] \leftarrow col\_A\_pivot$ 
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:         $C\_csr\_values[c\_pos] \leftarrow A\_csr\_values[c\_pos] \times B\_csr\_values[c\_pos]$ 
29:         $C\_JA[c\_pos] \leftarrow col\_B\_pivot$ 
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: $C((m * o) \times n)$

1: $A1 \leftarrow A_{CSCformat}$

2: $B1 \leftarrow B_{CSCformat}$

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

3: $c1_pos \leftarrow 0$

4: **for** $at_column = 0$ **to** $number_columns - 1$ **do**

5: $C1_IA[at_column] \leftarrow c1_pos$

6: $line_A_pivot \leftarrow A1_IA[at_column]$

7: $line_B_pivot \leftarrow B1_IA[at_column]$

8: $line_A_limit \leftarrow A1_IA[at_column + 1]$

9: $line_B_limit \leftarrow B1_IA[at_column + 1]$

10: **for** $line_A_pivot = 0$ **to** $line_A_limit - 1$ **do**

11: $line_B_pivot \leftarrow B1_IA[at_column]$

12: **for** $line_B_pivot = 0$ **to** $line_B_limit - 1$ **do**

13: $C_csc_values[c1_pos] \leftarrow A_csc_values[c1_pos] \times B_csc_values[c1_pos]$

14: $C1_JA[c1_pos] \leftarrow line_A_pivot \times line_B_pivot$

15: $++ c1_pos$

16: **end for**

17: **end for**

18: **end for**

19: $C1_IA[at_column] = c1_pos$

20: $C \leftarrow C1^T$

3 Kronecker

Algorithm 3 CSR Kronecker Computation

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

Ensure: $C((m * o) \times (n * p))$

1: $A1 \leftarrow A^T$
2: $B1 \leftarrow B^T$

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

3: $c1_pos \leftarrow 0$
4: **for** $at_column = 0$ **to** $number_columns - 1$ **do**
5: $C1_IA[at_column] \leftarrow c1_pos$
6: $line_A_pivot \leftarrow A1_IA[at_column]$
7: $line_B_pivot \leftarrow B1_IA[at_column]$
8: $line_A_limit \leftarrow A1_IA[at_column + 1]$
9: $line_B_limit \leftarrow B1_IA[at_column + 1]$

10: **for** $line_A_pivot = 0$ **to** $line_A_limit - 1$ **do**
11: $line_B_pivot \leftarrow B1_IA[at_column]$
12: **for** $line_B_pivot = 0$ **to** $line_B_limit - 1$ **do**
13: $C_csc_values[c1_pos] \leftarrow A_csc_values[c1_pos] \times B_csc_values[c1_pos]$
14: $C1_JA[c1_pos] \leftarrow line_A_pivot \times line_B_pivot$
15: $++ c1_pos$
16: **end for**
17: **end for**
18: **end for**

19: $C1_IA[at_column] = c1_pos$

20: $C \leftarrow C1^T$
