Algorithm 1 $C_i = A_i \times B$

Input: A_i, B
Output: C_i (result of $A_i \times B$)

$nnzPerColScan\_left \leftarrow A.nnzPerColScan$ (this is computed in matrix setup)
$mat\_send \leftarrow local\ B$
for $k = myrank : myrank + nprocs$ **do**
    $mat\_recv \leftarrow Irecv(remote\ B)\ from\ right\ neighbor$
    $Isend(mat\_send)\ to\ left\ neighbor$
    $nnzPerColScan\_right \leftarrow compute\ it\ for\ mat\_send$
    $C_i \leftarrow$ RECURS_MATMULT(
        $A_i, mat\_send,$
        $nnzPerColScan\_left[0], nnzPerColScan\_left[1],$
        $nnzPerColScan\_right[0], nnzPerColScan\_right[1])$
    $wait\ for\ Isend\ and\ Irecv\ to\ finish$
    $swap(mat\_send, mat\_recv)$
end for
sort C_i and remove duplicates.

$nnzPerColScan\_left[0]$ and $nnzPerColScan\_left[1]$ are being used to know the starting and ending index for nonzeros of each column of A. The same for $nnzPerColScan\_right$ about B.

In the recursive function, if A is being split vertically, $nnzPerColScan\_left$ will also be split to half, In this case, B should be split horizontally, so we go through half of the nonzeros of B and create $nnzPerColScan\_middle$ to know when each column ends for the top block and starts for the bottom block. Then call the recursive function as following:

Algorithm 2 Calling RECURS_MATMULTfor when A is being split vertically

$C_i \leftarrow$ RECURS_MATMULT(
    $A_{i,1}, B_{i,1},$
    $nnzPerColScan\_leftStart[0], nnzPerColScan\_leftEnd[0],$
    $nnzPerColScan\_rightStart, nnzPerColScan\_middle)$
$C_i \leftarrow$ RECURS_MATMULT(
    $A_{i,2}, B_{i,2},$
    $nnzPerColScan\_leftStart[A\_col\_size\_half], nnzPerColScan\_leftEnd[A\_col\_size\_half],$
    $nnzPerColScan\_middle, nnzPerColScan\_rightEnd)$