QuickSort(start, fin, comp)
   if( start >= fin)
      return

   (equal_start, equal_fin) ← Partition(start, fin, comp)

   QuickSort(start, equal_start − 1, comp) //recursively sort all values less than pivot
   QuickSort(equal_start, equal_fin, comp + 1)//recursively sort values = pivot by next component
   QuickSort(equal_fin+1, fin, comp)//recursively sort all values greater than pivot

//end

Partition(start, fin, comp)

   Pivot ← substring starting at (start + comp*comp_size) of length = comp_size
   Partition array with indices (pointers) into three categories
   1. Whose components are less than pivot
   2. Whose components are equal to pivot (starting at index equal_start, ending with equal_fin)
   3. Whose components are greater than pivot.

   Return values equal_start and equal_fin

//end

See example below
Step 1: \( S = ATATA TTAG \$

collect indices of all suffixes into an array:

\[
\text{Ptr} = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
\]

Step 2: Sort suffixes by sorting only indices of corresponding suffixes: component size = 2, the first component of each index points to the first component of size 2 character.

\[
\text{Ptr} = [1, 2, 3, 0, 5, 6, 7, 8, 9, 10]
\]

draw partition on pivot value

pivot = AT (component corresponding to index at Fhr)

after partition

\[
\text{Ptr} = [1, 2, 3, 0, 5, 6, 7, 8, 9, 10]
\]

Sorting is done in-place, \( \text{Ptr} \) after sorting [10 8 1 3 5 9 7 2 4 6]