

# INTRODUCTION TO ALGORITHMS AND DATA STRUCTURES: PSEUDOCODE SKETCHES OF RADIX-SORT VARIANTS

ANDRZEJ WĄSOWSKI

IT UNIVERSITY OF COPENHAGEN

FEBRUARY 25, 2005

This note supplements: Arne Andersson. Stefan Nilsson. *Implementing radix-sort*. Journal of Experimental Algorithmics. Vol. 3, 1998. **Warning:** The following pseudocodes are abstractions of what the paper presents. They do not include all the tricks (some of them are language specific). Most importantly calls to a comparison-based sort algorithm on small sublists are not shown. You are welcome to study original C implementations accompanying the paper. Also many assumptions are not spelled out, read the paper if in doubt. This material is fresh, and as such may contain errors. Please report to me if you find any.

We are sorting  $n$  strings over alphabet  $\Sigma = \{1..m\}$ , according to the lexicographic ordering. Refer to CLRS, chpt. 8.3 for a pseudo code of the LSD RADIX-SORT.

```
MSD-RECURSIVE-RADIX-SORT( $A, pos$ )
  if  $length[A] < 2$  or  $pos > max-length[A]$ 
    then return  $A$ 
  new array  $B[1..m]$ 
   $\triangleright B$  is initialized with empty lists
  BUCKETING( $A, B, pos$ )
   $res \leftarrow \langle \rangle$ 
  for  $i \leftarrow 1$  to  $m$ 
    do Append MSD-RECURSIVE-RADIXSORT( $B(i), pos + 1$ )
      after the current content of  $res$ 
  return  $res$ 
```

```
BUCKETING( $A, B, pos$ )
  for  $i \leftarrow 1$  to  $length[A]$ 
    do Append  $A(i)$  after the current content of  $B(A(i)_{pos})$ 
   $\triangleright B$  is returned by reference
```

Function (or rather an attribute) *max-length* returns the length of the longest string in  $A$ . It should be implemented to do it in constant time. If  $x$  is a string, then  $x_i$  denotes the  $i$ 'th character in this string.

### MSD-RADIX-SORT-STACK( $A$ )

▷ Assume an existence of a global stack served by POP and PUSH  
▷ Array  $A$  has an additional attribute  $pos[A]$ .  
 $res \leftarrow \langle \rangle$   
 $pos[A] \leftarrow 0$   
PUSH( $A$ )  
**while** stack is not empty  
    **do**  $A \leftarrow$  POP()  
        **if**  $length[A] = 1$  or  $pos[A] > max-length[A]$   
            **then** Prepend  $A$  before the current content of  $res$ .  
            **else** BUCKETING-STACK( $A$ )  
**return**  $res$

### BUCKETING-STACK( $A$ )

▷ initially  $B$  contains empty lists, a static array may be used  
array  $B[1..m]$   
 $p \leftarrow pos[A]+1$   
 $prevch \leftarrow A(1)_p$   
 $beg \leftarrow 1$   
**for**  $i \leftarrow 1$  **to**  $length[A]$   
    **do**  $ch \leftarrow A(i)_p$   
        **if**  $prevch \neq ch$   
            **then** Append  $A(beg..i-1)$  to the end of  $B(prevch)$   
                 $beg \leftarrow i$   
                 $prevch \leftarrow ch$   
Append  $A(beg..length[A])$  to the end of  $B(prevch)$   
  
**for**  $i \leftarrow 1$  **to**  $m$   
    **do if**  $B(i) \neq \langle \rangle$   
        **then**  $pos[B(i)] \leftarrow p$   
            PUSH( $B(i)$ )  
             $B(i) \leftarrow \langle \rangle$   
▷ values are returned via global stack  
**return**

**Exercise.** Implement FORWARD-RADIX-SORT in pseudocode.