**Radix sort**

Radix sort is a stable sorting algorithm used mainly for sorting strings of the same length.

## Description

The fundamental principle of radix sort stems from the definition of the stable sort – sorting algorithm is stable, if it maintains the order of keys, which are equal.

Radix sort iteratively orders all the strings by their n-th character – in the first iteration, the strings are ordered by their last character. In the second run, the strings are ordered in respect to their penultimate character. And because the sort is stable, the strings, which have the same penultimate character, are still sorted in accordance to their last characters. After n-th run the strings are sorted in respect to all character positions.

## Asymptotic complexity

The [asymptotic complexity](http://en.algoritmy.net/article/44682/Asymptotic-complexity) of radix sort is O(m \cdot C(n)), where mis the length of sorted strings and C(n)is the complexity of the inner implementation of the a stable sort (eg. [counting sort](http://en.algoritmy.net/article/40549/Counting-sort)).

## Code

pseudo code

1.function radixSort(String s)

2.for i in (s.length - 1)  -> 0 do

3.stableSort(s[i])

java

/\*\*

\* Radix sort (ascending)

\* Inner stable sort: counting sort

\* @param array array to be sorted

\* @param dimension fixed length of sorted strings

\* @return sorted array

\*/

public static String[] radixSort(String[] array, int dimension){

for(int i = dimension - 1; i >= 0 ; i--){

array = countingSortForRadix(array, i); //order strings by characters at their i-th position

}

return array;

}

/\*\*

\* Counting sort for radix sort

\* @param array array to be sorted

\* @param position position (key) used for the sorting

\* @return sorted array

\*/

public static String[] countingSortForRadix(String[] array, int position){

String[] aux = new String[array.length];

char min = array[0].charAt(position);

char max = array[0].charAt(position);

for(int i = 1; i < array.length; i++){

if(array[i].charAt(position) < min) min = array[i].charAt(position);

else if(array[i].charAt(position) > max) max = array[i].charAt(position);

}

int[] counts = new int[max - min + 1];

for(int i = 0; i < array.length; i++){

counts[array[i].charAt(position) - min]++;

}

counts[0]--;

for(int i = 1; i < counts.length; i++){

counts[i] = counts[i] + counts[i-1];

}

for(int i = array.length - 1; i >=0; i--){

aux[counts[array[i].charAt(position) - min]--] = array[i];

}

return aux;

}

example

