

# ITS202: Algorithms and Data Structures

## String Sorts

Ms. Sonam Wangmo

Gyalpozhing College of Information Technology  
Royal University of Bhutan

October 20, 2020

We consider two fundamentally different approaches to string sorting.

- ① LSD referred to as least-significant-digit (LSD) string sorts.
- ② MSD referred to as most-significant-digit (MSD) string sorts.

# LSD radix sort

## Concepts

- Consider characters from right to left.
- Sort using  $d$ th character as the key (Using key-indexed counting)
- LSD sorts fixed length strings in ascending order.
- Sorting such string can be done using key-indexed counting.
- If the strings are each of length  $W$ , we sort the strings  $W$  times with key-indexed counting, using each of the positions as the key, proceeding from right to left.

# LSD radix sort

```
1      public class LSD
2      {
3          public static void sort(String[] a, int
4              W)
5          {
6              //Sort a[] on leading W characters.
7              int N = a.length;
8              int R = 256;
9              String[] aux = new String[N];
10             for (int d = W-1; d >= 0; d--)
11             {
12                 // Sort by key-indexed counting on
13                 // dth char.
14                 int[] count = new int[R+1];
15                 // Compute frequency counts.
16                 for (int i = 0; i < N; i++)
```

# LSD radix sort

```
1         count[a[i].charAt(d) + 1]++;
2         for (int r = 0; r < R; r++)
3             // Transform counts to indices.
4             count[r+1] += count[r];
5         for (int i = 0; i < N; i++)
6             // Distribute.
7             aux[count[a[i].charAt(d)]++] =
                a[i];
8         for (int i = 0; i < N; i++)
9             // Copy back.
10            a[i] = aux[i];
11    }
12 }
13 }
```

# LSD radix sort

To sort an array  $a[]$  of strings that each have exactly  $W$  characters, we do  $W$  key-indexed counting sorts: one for each character position, proceeding from right to left.

# LSD radix sort

input (W=7)	d=6	d=5	d=4	d=3	d=2	d=1	d=0	output
4PGC938	2IYE230	3CIO720	2IYE230	2RLA629	1ICK750	3ATW723	1ICK750	1ICK750
2IYE230	3CIO720	3CIO720	4JZY524	2RLA629	1ICK750	3CIO720	1ICK750	1ICK750
3CIO720	1ICK750	3ATW723	2RLA629	4PGC938	4PGC938	3CIO720	10HV845	10HV845
1ICK750	1ICK750	4JZY524	2RLA629	2IYE230	10HV845	1ICK750	10HV845	10HV845
10HV845	3CIO720	2RLA629	3CIO720	1ICK750	10HV845	1ICK750	10HV845	10HV845
4JZY524	3ATW723	2RLA629	3CIO720	1ICK750	10HV845	2IYE230	2IYE230	2IYE230
1ICK750	4JZY524	2IYE230	3ATW723	3CIO720	3CIO720	4JZY524	2RLA629	2RLA629
3CIO720	10HV845	4PGC938	1ICK750	3CIO720	3CIO720	10HV845	2RLA629	2RLA629
10HV845	10HV845	10HV845	1ICK750	10HV845	2RLA629	10HV845	3ATW723	3ATW723
10HV845	10HV845	10HV845	10HV845	10HV845	2RLA629	10HV845	3CIO720	3CIO720
2RLA629	4PGC938	10HV845	10HV845	10HV845	3ATW723	4PGC938	3CIO720	3CIO720
2RLA629	2RLA629	1ICK750	10HV845	3ATW723	2IYE230	2RLA629	4JZY524	4JZY524
3ATW723	2RLA629	1ICK750	4PGC938	4JZY524	4JZY524	2RLA629	4PGC938	4PGC938

Figure 1: LSD

# LSD radix sort

Q. What if strings are not all of same length?



# LSD radix sort

Q. What if strings are not all of same length?

## Concepts

- Consider characters from left to right order.
- Sort using  $d$ th character as the key (Using key-indexed counting)
- A general-purpose string sort, where strings are not necessarily all the same length
- Sorting such string can be done using key-indexed counting.
- If the strings are each of length  $W$ , we sort the strings  $W$  times with key-indexed counting, using each of the positions as the key, proceeding from right to left.

# MSD radix sort

input

she	are	are	are	are	are	are	are	are
sells	by	by	by	by	by	by	by	by
seashells	she	sells	seashells	sea	sea	sea	sea	sea
by	sells	seashells	sea	seashells	seashells	seashells	seashells	seashells
the	seashells	sea	seashells	seashells	seashells	seashells	seashells	seashells
sea	sea	sells	sells	sells	sells	sells	sells	sells
shore	shore	seashells	sells	sells	sells	sells	sells	sells
the	shells	she	she	she	she	she	she	she
shells	she	shore	shore	shore	shore	shore	shells	shells
she	sells	shells	shells	shells	shells	shells	shore	shore
sells	surely	she	she	she	she	she	she	she
are	seashells	surely	surely	surely	surely	surely	surely	surely
surely	the	the	the	the	the	the	the	the
seashells	the	the	the	the	the	the	the	the

Annotations: "To" points to the 's' in 'sells' of the first row. "hi" points to the 'h' in 'the' of the second row.

are	are	are	are	are	are	are	output
by	by	by	by	by	by	by	by
sea	sea	sea	sea	sea	sea	sea	sea
seashells	seashells	seashells	seashells	seashells	seashells	seashells	seashells
seashells	seashells	seashells	seashells	seashells	seashells	seashells	seashells
sells	sells	sells	sells	sells	sells	sells	sells
sells	sells	sells	sells	sells	sells	sells	sells
she	she	she	she	she	she	she	she
shells	shells	shells	shells	shells	shells	shells	shells
she	she	she	she	she	shells	shells	shells
shore	shore	shore	shore	shore	shore	shore	shore
surely	surely	surely	surely	surely	surely	surely	surely
the	the	the	the	the	the	the	the
the	the	the	the	the	the	the	the

Annotations: "need to examine every character in equal keys" points to the 's' in 'seashells' of the first row. "end of string goes before any char value" points to the 'e' in 'the' of the second row.

Figure 2