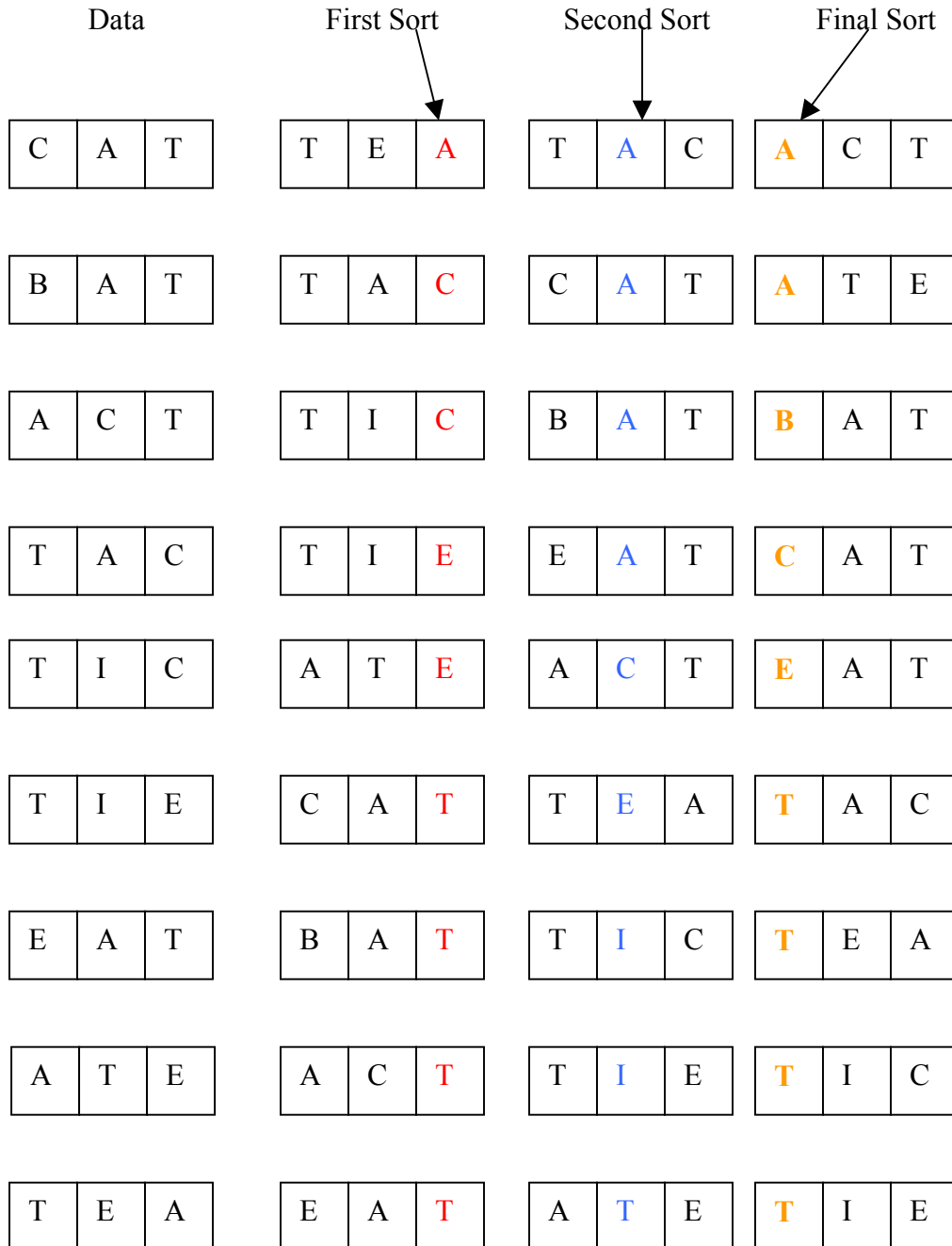


# Radix Sort

*Input:*  $n$  items, each with  $m$  “digits”. Each digit is from the set  $\{1, 2, \dots, k\}$ .

*Output:* Lexicographically sorted order.

Example: Main order: sort on **the least significant digit** first.



**Main sorting order:** sort on **the least significant digit** first.

**Why not the first significant digit first?** Avoid making buckets inside another bucket. It will be complicate.

**Stable Sort:** When the elements have same value, they will keep the same order in the output as their input. For example, in the second sort of the above example, TAC and CAT have same A in the middle and we put them in the order of last sort that was C is before T at the last digit.

**Algorithm:**

```
void RadixSort ()
{
    for(i=m; i>=1; i--)
        sort the input list using a stable sort
        based on the ith digit.
}
```

**Radix Sort is based on Counting Sort.** For each digit sorting, the counting sort needs  $O(n+k)$  time. There are  $m$  digits. So, the **total time** for Radix Sort is  $O(m(n+k))$

$O(m(n+k))$ :  $m$  = number of digits (columns)

$n$  = number of items to be sorted

$k$  = alphabet size. (How many different kinds of alphabets.)