input (w = 2)	d=1	d=0	
no	pa	ai	ai
is	pe	al	al
th	of	co	co
ti	th	fo	fo
fo	th	go	go
al	no	is	is
go	is	no	no
pe	th	of	of
to	ti	pa	pa
co	ai	pe	pe
to	al	th	th
th	fo	th	th
ai	go	th	th
of	to	ti	ti
th	co	to	to
pa	to	to	to

Figure 1: Trace of LSD string sort for Exercise 2.

Sergio E. Garcia Tapia
Algorithms by Sedgewick and Wayne (4th edition) [SW11]
January 21, 2025

5.1: String Sorts

Exercise 1. Develop a sort implementation that counts the number of different key values, the uses a symbol table to apply key-indexed counting to sort the array. (This method is *not* for sure when the number of different keys is large).

Solution. It was unclear to me whether the sort implementation should allow for any type of key. I decided to limit the implementation to arrays of integers.

See `com.segarciat.algs4.ch5.sec1.ex01`.

Exercise 2. Give a trace for LSD string sort for the keys:

```
no is th ti fo al go pe to co to th ai of th pa
```

Solution. See Figure 1.

Exercise 3. Give a trace for MSD string sort for the keys

```
no is th ti fo al go pe to co to th ai of th pa
```

Solution. See Figure 2. The `CUTOFF` subarray length is `0` (no cutoff). Hence, we assume that insertion sort is used when the subarray is length 1, which would cause the method to immediately return anyway (because a 1-element array is sorted). Note that I omitted the recursive calls on singleton subarrays, such as the one containing `co`. I also omitted calls when the ends of the strings are reached because MST skips recursive calls on such strings.

Exercise 4. Give a trace for 3-way string quicksort for the keys

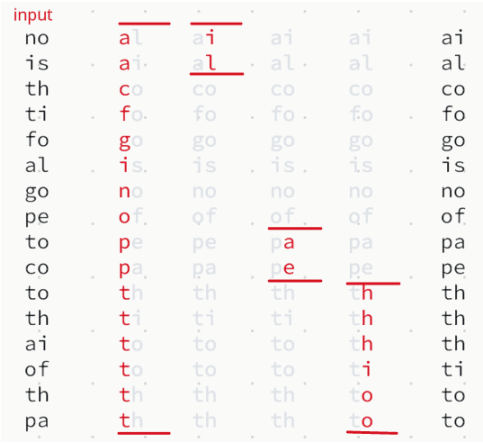

Figure 2: Trace of MSD string sort for Exercise 3.

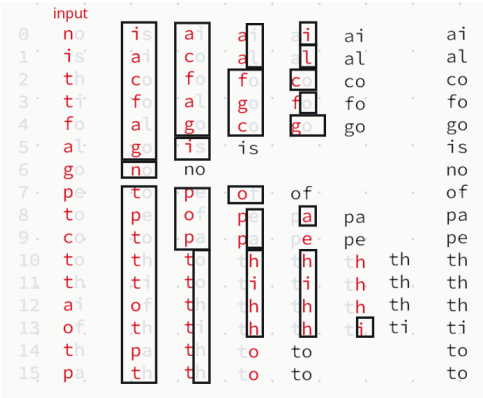

Figure 3: Trace of 3-way string quicksort for Exercise 4.

```
no is th ti fo al go pe to co to th ai of th pa
```

Solution. See Figure 3.

Exercise 5. Give a trace for MSD string sort for the keys

```
now is the time for all good people to come to the aid of
```

Solution. See Figure 4.

Exercise 7. Develop an implementation of key-indexed counting that makes use of an array of `Queue` objects.

Solution. See `com.segarciat.algs4.ch5.sec1.ex07`.

Exercise 8. Give the number of characters examined by MSD string sort and 3-way string quicksort for a file with n keys `a`, `aa`, `aaa`, `aaaa`, `aaaa`, ...

Solution. 3-way string quicksort would require first n, then n again, the $n-1$, and so on, which comes out to about $\Theta(n^2)$. That is, all characters are examined.

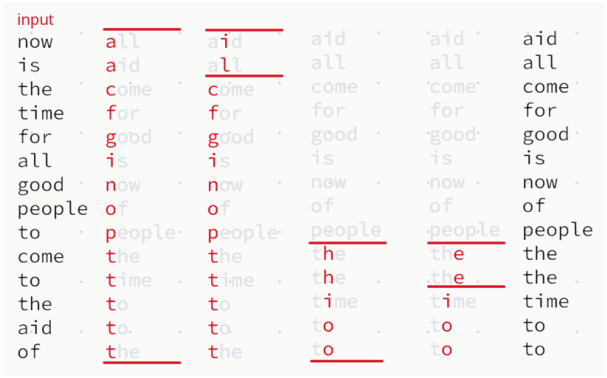

Figure 4: Trace of MSD string sort for Exercise 5.

Meanwhile, MSD would also examine all the characters. Unlike 3-way string quicksort, MSD would also incur the cost of initializing the `count` arrays, which it would do nR times.

Exercise 9. Develop an implementation of LSD string sort that works for variable-length strings.

Solution. See `com.segarciat.algs4.ch5.sec1.ex09`.

Exercise 12. *Alphabet.* Develop an implementation of the `Alphabet` API that is given on page 698 and use it to develop LSD and MSD sorts for general alphabets.

Solution. See `com.segarciat.algs4.ch5.sec1.ex12`.

References

[SW11] Robert Sedgewick and Kevin Wayne. *Algorithms*. 4th ed. Addison-Wesley, 2011. ISBN: 9780321573513.