

Sorting Algorithms — Stability

Let A be an array, and let $<$ be a strict weak ordering on the elements of A .

A sorting algorithm is *stable* if

$$i < j \text{ and } A[i] \sim A[j] \text{ implies } \pi(i) < \pi(j).$$

where π is the sorting permutation (sorting moves $A[i]$ to position $\pi(i)$.)

Informally, stability means that equivalent elements retain their relative positions, after sorting. (If the elements have class/structure type and are ordered according to their value on one field — the *key* field — then *equivalent* elements are elements with equal keys.)

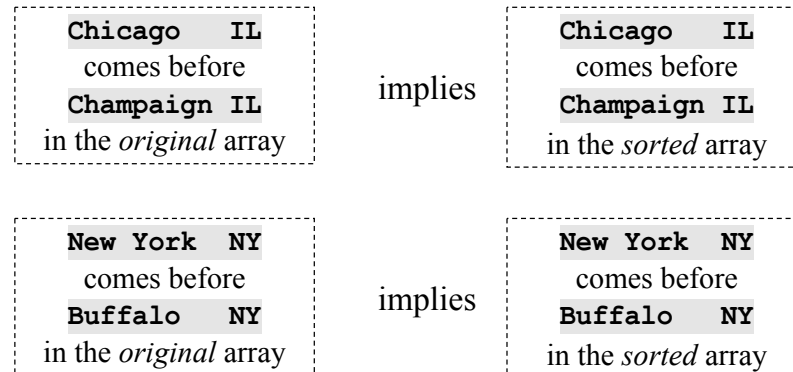
Suppose we apply a sorting algorithm with this array as input, using the state (NY, IL, etc) as the key.

New York	NY
Chicago	IL
Detroit	MI
Buffalo	NY
Milwaukee	WI
Champaign	IL

There are four correct outputs.

Chicago IL	Champaign IL	Chicago IL	Champaign IL
Champaign IL	Chicago IL	Champaign IL	Chicago IL
Detroit MI	Detroit MI	Detroit MI	Detroit MI
New York NY	New York NY	Buffalo NY	Buffalo NY
Buffalo NY	Buffalo NY	New York NY	New York NY
Milwaukee WI	Milwaukee WI	Milwaukee WI	Milwaukee WI

But for a *stable* sorting algorithm, *only the first output is correct.*



In practice, we sometimes need a stable sorting algorithm.

We might need sort the array A above alphabetically by state, with all cities in a single state appearing in alphabetical order.

Our sorting software might allow sorting on only one field at a time.

- We may
- 1) First sort the array A alphabetically by city.
 - 2) Then sort the array A alphabetically by state, using a stable sorting algorithm.

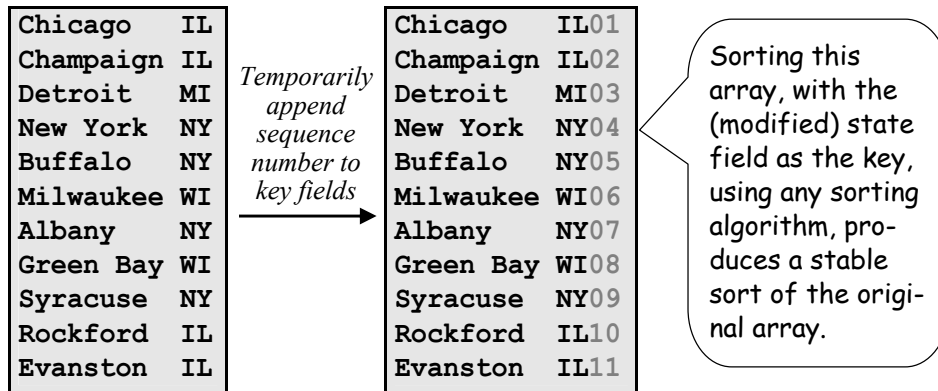
Chicago IL Champaign IL Detroit MI New York NY Buffalo NY Milwaukee WI Albany NY Green Bay WI Syracuse NY Rockford IL Evanston IL	Sort alpha- betically by city.	Albany NY Buffalo NY Champaign IL Chicago IL Detroit MI Evanston IL Green Bay WI Milwaukee WI New York NY Rockford IL Syracuse NY	Stable sort alpha- betically by state.	Champaign IL Chicago IL Evanston IL Rockford IL Detroit MI Albany NY Buffalo NY New York NY Syracuse NY Green Bay WI Milwaukee WI
---	---	---	--	---

Unfortunately, many of the (otherwise) best sorting algorithms are not stable.

For example, *quicksort* and *heapsort* are not stable. (*Mergesort* property implemented is stable.)

Any sorting algorithm may be made stable, at a price: The price is $\Theta(n)$ extra space, and moderately increased running time (less than doubled, most likely).

We have to (temporarily) append a sequence number to the key of each element of the array. The sequence number serves as a tie-breaker.



The C++ standard library provides a choice between a stable sorting template function `stable_sort()`, and a (presumably faster) non-stable sorting template function `sort()`.

The Java library (class `java.util.Arrays`) provides static methods for sorting objects, that are guaranteed to be stable.