Stability in sorting algorithms

Stability is mainly important when we have key value pairs with duplicate keys possible (like people names as keys and their details as values). And we wish to sort these objects by keys.

What is it?

A sorting algorithm is said to be stable if two objects with equal keys appear in the same order in sorted output as they appear in the input array to be sorted.

Formally stability may be defined as,

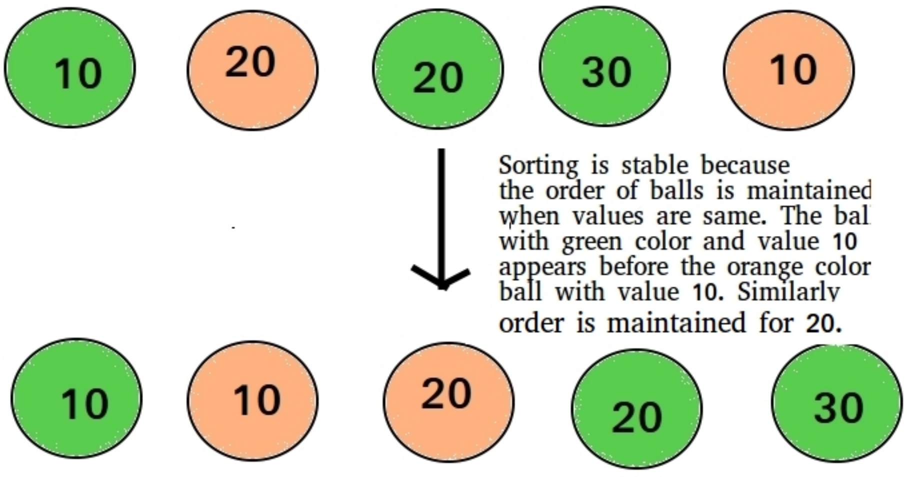
Let A be an array, and let < be a strict weak ordering on the elements of A.

A sorting algorithm is stable if-

$i < j and A[i] \equiv A[j] implies \pi(i) < \pi(j)$

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.


10
20
20
30
10
Sorting is stable because the order of balls is maintained when values are same. The bal with green color and value 10 appears before the orange color ball with value 10. Similarly order is maintained for 20.
10
10
20
20
30

Do we care for simple arrays like array of integers?

When equal elements are indistinguishable, such as with integers, or more generally, any data where the entire element is the key, stability is not an issue. Stability is also not an issue if all keys are different.

An example where it is useful

Consider the following dataset of Student Names and their respective class sections.

$(Dave,A)(Alice,B)(Ken,A)(Eric,B)(Carol,A)$

If we sort this data according to name only, then it is highly unlikely that the resulting dataset will be grouped according to sections as well.

$(Alice,B)(Carol,A)(Dave,A)(Eric,B)(Ken,A)$

So we might have to sort again to obtain list of students section wise too. But in doing so, if the sorting algorithm is not stable, we might get a result like this-

$(Carol,A)(Dave,A)(Ken,A)(Eric,B)(Alice,B)$

The dataset is now sorted according to sections, but not according to names.

In the name-sorted dataset, the tuple $(Alice,B)$ was before $(Eric,B)$, but since the sorting algorithm is not stable, the relative order is lost. If on the other hand we used a stable sorting algorithm, the result would be-

$(Carol,A)(Dave,A)(Ken,A)(Alice,B)(Eric,B)$

Here the relative order between different tuples is maintained. It may be the case that the relative order is maintained in an Unstable Sort but that is highly unlikely.

Which sorting algorithms are stable?

Some Sorting Algorithms are stable by nature, such as Bubble Sort , Insertion Sort ,Merge Sort, Count Sort etc.

Comparison based stable sorts such as Merge Sort and Insertion Sort, maintain stability by ensuring that-

Element $A[j]$ comes before $A[i]$ if and only if $A[j]<A[i]$, here i, j are indices and $i<j$.

Since [Text]i Other non-comparison based sorts such as Counting Sort maintain stability by ensuring that the Sorted Array is filled in a reverse order so that elements with equivalent keys have the same relative position. Some sorts such as Radix Sort depend on another sort, with the only requirement that the other sort should be stable.

Which sorting algorithms are unstable?

Quick Sort ,Heap Sort etc., can be made stable by also taking the position of the elements into consideration. This change may be done in a way which does not compromise a lot on the performance and takes some extra space, possibly $\Theta(n)$.

Can we make any sorting algorithm stable?

Any given sorting algo which is not stable can be modified to be stable. There can be sorting algo specific ways to make it stable, but in general, any comparison based sorting algorithm which is not stable by nature can be modified to be stable by changing the key comparison operation so that the comparison of two keys considers position as a factor for objects with equal keys.