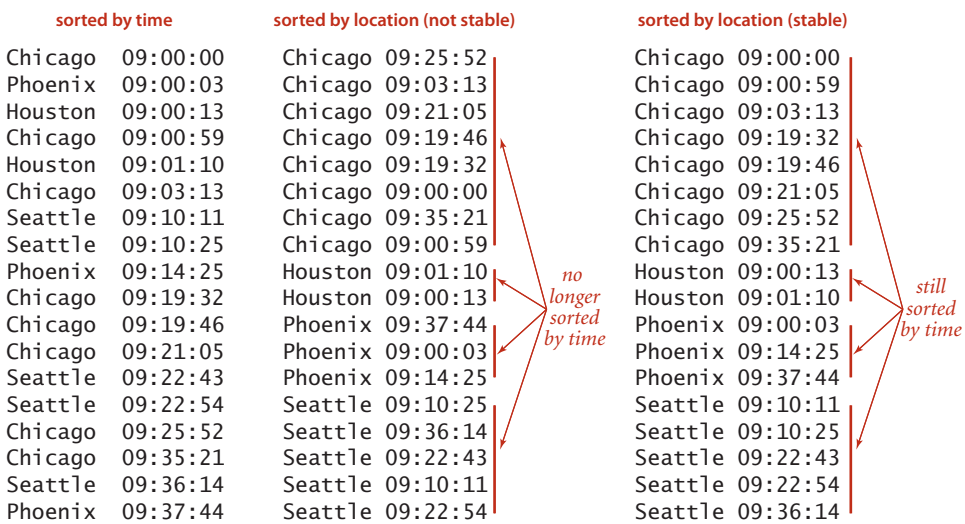
Stability. A sorting method is *stable* if it preserves the relative order of equal keys in the array. This property is frequently important. For example, consider an internet commerce application where we have to process a large number of events that have locations and timestamps. To begin, suppose that we store events in an array as they arrive, so they are in order of the timestamp in the array. Now suppose that the application requires that the transactions be separated out by location for further processing. One easy way to do so is to sort the array by location. If the sort is unstable, the transactions for each city may *not* necessarily be in order by timestamp after the sort. Often, programmers who are unfamiliar with stability are surprised, when they first encounter the situation, by the way an unstable algorithm seems to scramble the data. Some of the sorting methods that we have considered in this chapter are stable (insertion sort and mergesort); many are not (selection sort, shellsort, quicksort, and heapsort). There are ways to trick any sort into stable behavior (see EXERCISE 2.5.18), but using a stable algorithm is generally preferable when stability is an essential requirement. It is easy to take stability for granted; actually, no practical method in common use achieves stability without using significant extra time or space (researchers have developed algorithms that do so, but applications programmers have judged them too complicated to be useful).



Stability when sorting on a second key