Radix Drawbacks:

1. Memory: If using Counting sort inside, and if you have a large number of possible values for each digit like in hex (16) then the memory size required is large. And this is the case for every digit.
2. Flexibility: All numbers can be handled with quicksort. Radix might have trouble with negative numbers

If the numbers are bit patterns of length k . log(n) for constant k then radix and best comparison sort have same time complexity (radix almost always loses on space complexity). If the length of numbers of greater than order O(lg n) with random bit patterns then radix is slower.

The shaded area in the dotted image in Order Statistics with Worst Case Linear Time Statistic Determination is denoting the elements that are certainly greater than the median of medians and not ALL the elements that are greater.

In counting sort we calculate the cumulative array because the elements might not be only integers. In using the method of storing counts and then creating a new array the elements of the new array are not extracted from the old array but are simply entered. If the elements of the array are objects with associated keys then that will not be possible so a new array with “rearranged” elements is instead created.

In bucket sort the main thing is that we expect the number of elements in each bucket to be the same. So insertion sort takes constant time to sort the elements.

A stable sort is one that preserves order of equal elements while sorting. Eg: Insertion sort and merge sort. Not examples: Heap sort and Quicksort.