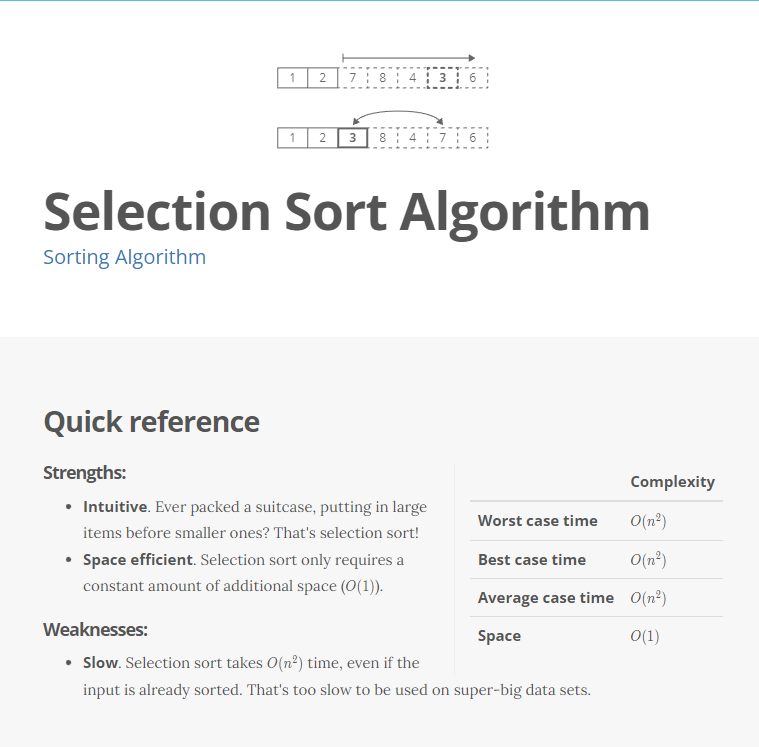
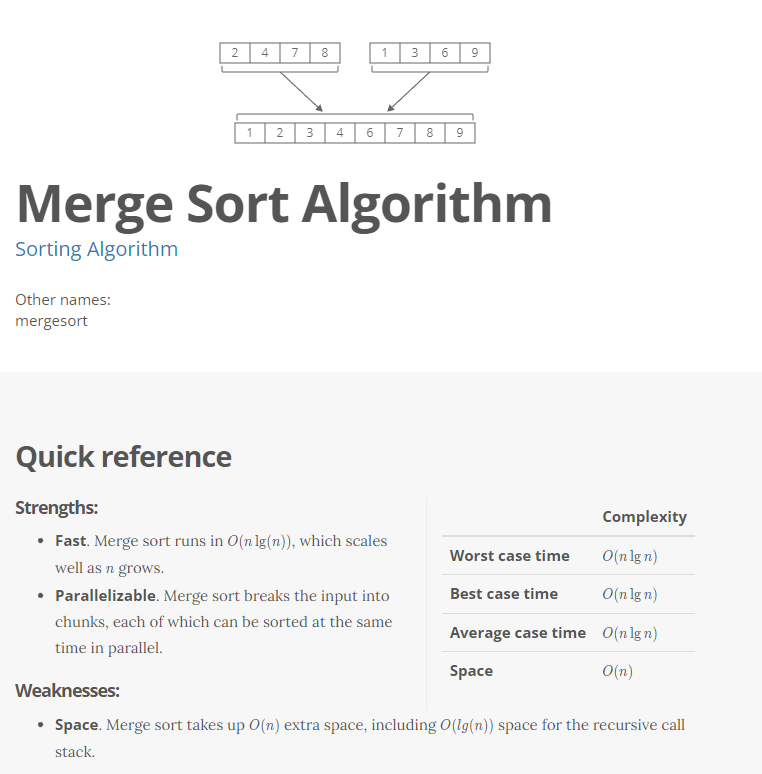
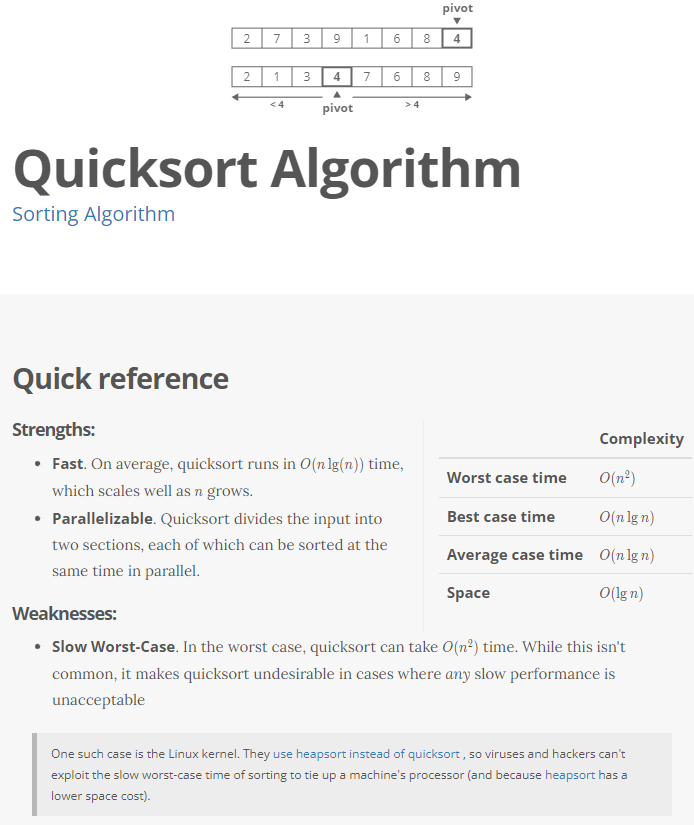


Sorting Algorithms

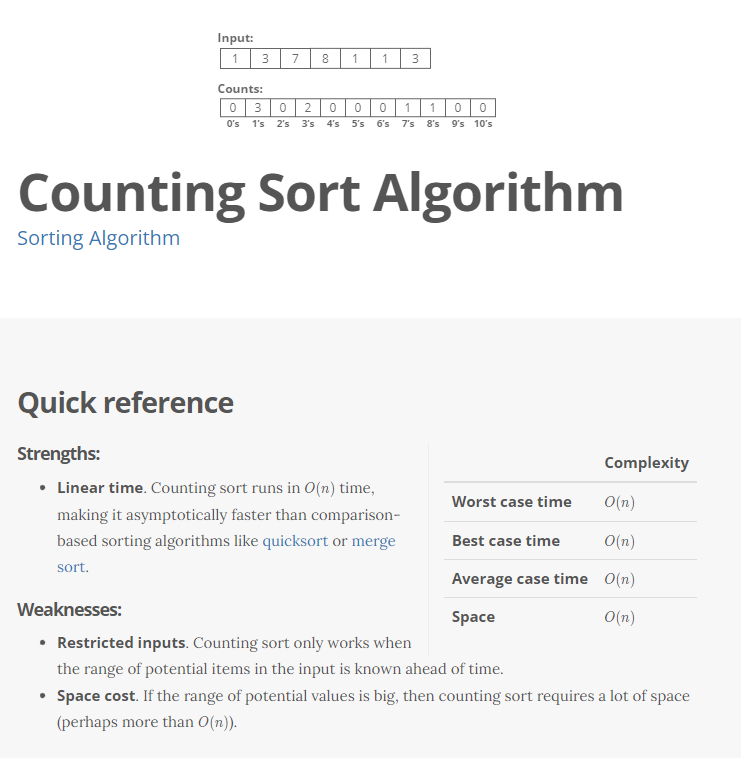


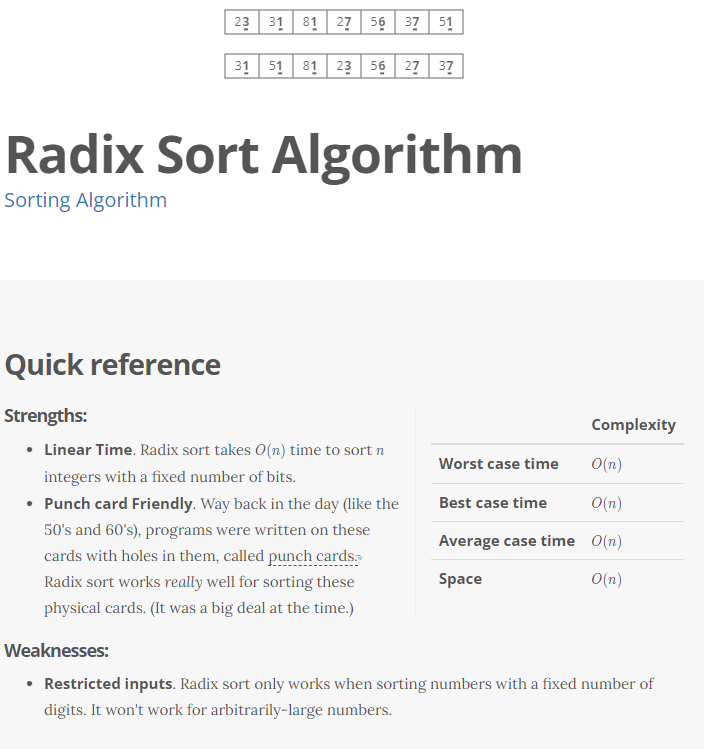












**Which Sorting Algorithm Should I Use?**

It depends. Each algorithm comes with its own set of pros and cons.

* **Quicksort** is a good default choice. It tends to be fast in practice, and with some small tweaks its dreaded O(n^2)*O*(*n*2) worst-case time complexity becomes very unlikely. A tried and true favorite.
* **Heapsort** is a good choice if you can't tolerate a worst-case time complexity of O(n^2)*O*(*n*2) or need low space costs. The Linux kernel [uses heapsort instead of quicksort](https://github.com/torvalds/linux/blob/master/lib/sort.c#L194)for both of those reasons.
* **Merge sort** is a good choice if you want [a stable sorting algorithm](https://www.interviewcake.com/concept/stable-sort). Also, merge sort can easily be extended to handle data sets that can't fit in RAM, where the bottleneck cost is reading and writing the input on disk, not comparing and swapping individual items.
* **Radix sort** *looks* fast, with its O(n)*O*(*n*) worst-case time complexity. But, if you're using it to sort binary numbers, then there's a hidden constant factor that's usually 32 or 64 (depending on how many bits your numbers are). That's often *way* bigger than O(\lg(n))*O*(lg(*n*)), meaning radix sort tends to be slow in practice.
* **Counting sort** is a good choice in scenarios where there are small number of distinct values to be sorted. This is pretty rare in practice, and counting sort doesn't get much use.

**Each sorting algorithm has tradeoffs. You can't have it all.**

So you have to know *what's important* in the problem you're working on. How large is your input? How many distinct values are in your input? How much space overhead is acceptable? Can you afford O(n^2)*O*(*n*2) worst-case runtime?

Once you know what's important, you can pick the sorting algorithm that does it best. Being able to compare different algorithms and weigh their pros and cons is the mark of a strong computer programmer and a definite plus when interviewing.