DAILY DSA | DAY-6 | SORTING ALGORITHMS |
-GOPALKRISHNA A

In the past few days, we have seen various data structures starting from arrays to tuples & sets. Now we will start with one of the exciting topics **"Sorting algorithms"**

Sorting algorithms:

- Sorting doesn't always come for free!
- The efficiency of most sorting algorithms is based on the number of comparisons it has to perform between input elements

Advantages that benefit from having sorted input:

- **Find a particular value:** Determining the existence or index of a given value is **(Ologn)** operation in a sorted list
- **Finding minimum or maximum values:** Rather than searching over the entire input, finding minimum and maximum elements with sorted input becomes an **O(n)**
- **Finding duplicate or missing values:** Since similar values are grouped together in sorted lists, it is easier to determine the frequency of each value

Popular & Efficient Data Sorting Algorithms:

Algorithm	Average runtime	Worst runtime	When to use
Insertion sort	N**2	N**2	Better for short lists, stable
Quick sort	N log N	N**2	Often fast, in-place, not stable
Merge sort	N log N	N log N	Stable, but not in place
Heap sort	N log N	N log N	In-place, better than Quicksort
Radix sort	N*K	N*K	The non-comparison sort that runs in linear time is stable, not in place

Do you know?

- Most of the common out-of-the-box sorting methods used in the standard library are **"QUICK SORT"**

When to use a different sorting algorithm rather than the standard library?

- Do you need to sort the entire list or just maintain minimum/maximum k elements? Consider using **heap sort** to reduce time & space complexity
- Is your data already partially or nearly sorted? Consider using **Insertion sort** which is the most efficient

- Do you need to sort in place or create a copy of the original output? **Insertion sort** and **quicksort** can be performed.
- Do you need a stable sort, where sorted items are retained in the same order? **Merge sort and insertion sort** are stable while quick sort is not