Sorting Algorithms

Sorting algorithms are a set of instructions that take an array or list as an input and arrange the items into a particular order.

Sorts are most commonly in numerical or a form of alphabetical (called lexicographical) order, and can be in ascending (A-Z, 0-9) or descending (Z-A, 9-0) order.

Why Sorting Algorithms are Important

Since sorting can often reduce the complexity of a problem, it is an important algorithm in Computer Science. These algorithms have direct applications in searching algorithms, database algorithms, divide and conquer methods, data structure algorithms, and many more.

Some Common Sorting Algorithms

Some of the most common sorting algorithms are:

- Selection Sort
- Bubble Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort
- Counting Sort
- Radix Sort
- Bucket Sort

Classification of Sorting Algorithm

Sorting algorithms can be categorized based on the following parameters:

1. Based on Number of Swaps or Inversion This is the number of times the algorithm swaps elements to sort the input. `Selection Sort` requires the minimum number of swaps.
2. Based on Number of Comparisons This is the number of times the algorithm compares elements to sort the input. Using Big-O notation (https://guide.freecodecamp.org/computer-science/notation/big-o-notation/), the sorting algorithm examples listed above require at least `O(nlogn)` comparisons in the best case and `O(n^2)` comparisons in the worst case for most of the outputs.
3. Based on Recursion or Non-Recursion Some sorting algorithms, such as `Quick Sort`, use recursive techniques to sort the input. Other sorting algorithms, such as `Selection Sort` or `Insertion Sort`, use non-recursive techniques. Finally, some sorting algorithm, such as `Merge`

`Sort`, make use of both recursive as well as non-recursive techniques to sort the input.

4. Based on Stability Sorting algorithms are said to be `stable` if the algorithm maintains the relative order of elements with equal keys. In other words, two equivalent elements remain in the same order in the sorted output as they were in the input.
5. `Insertion sort`, `Merge Sort`, and `Bubble Sort` are stable
6. `Heap Sort` and `Quick Sort` are not stable
7. Based on Extra Space Requirement Sorting algorithms are said to be `in place` if they require a constant `O(1)` extra space for sorting.
8. `Insertion sort` and `Quick-sort` are `in place` sort as we move the elements about the pivot and do not actually use a separate array which is NOT the case in merge sort where the size of the input must be allocated beforehand to store the output during the sort.
9. `Merge Sort` is an example of `out place` sort as it require extra memory space for it's operations.

Best possible time complexity for any comparison based sorting

Any comparison based sorting algorithm must make at least nLog2n comparisons to sort the input array, and Heapsort and merge sort are asymptotically optimal comparison sorts.This can be easily proved by drawing the desicion tree diagram.