Searching

Linear search : it is a simple algorithm that finds an element in a list by sequentially checking each element until a match is found or the end of the list is reached.

Here's how it work with an example

Imagine you have a list of numbers: [2, 5, 9, 1, 7, 3] and you want to find the number 7.

linear search process:

1. Start at the begining of the list (index 0) and compare first element (2) with the target value 7.
2. Since 2 is not equal to 7, move to the next element (index 1) and compare 5 with 7.
3. Continue this process, comparing each element until you find a match or reach the end of the list
4. in this case, you will find that 7 is at index 6.

If you reach the end of the list without finding a match, the search is unsuccessful.

Binary search:- Binary search is an efficient algorithm for finding an item from a sorted list of items. It works by repeatedly dividing the search interval in half. If the value of the search key is less than the value in the middle of the interval, the search continues in the lower half, otherwise, it continues in the upper half. This process repeats until the value is found or the search interval is empty. Here is how binary search works step-by-step.

1. Find the middle element of the list
2. If the target element is equal to the middle element, return the index of the middle element.
3. If the target element is smaller than the middle element, repeat the search on the left half of the list.
4. If the target element is greater than the middle element, repeat the search on the right half of the list.
5. Repeat the process until the element is found or the list is reduced to zero.

linear search

1. it works unsorted or sorted data
2. Best for small data sets
3. Simple and easy to implement
4. less efficient for large data sets

Binary search

1. it works only on sorted data.
2. Best for large sorted data sets
3. Requires sorting and more complex logic.
4. Highly efficient for large datasets.

Selection sort

it sorts an array by repeatedly selecting the smallest element from the unsorted portion and swapping it with the first unsorted element. This process continues until the entire array is sorted.

1. First we find the smallest element and swap it with the first element. This way we get the smallest element at its correct position.
2. Then we find the smallest among remaining elements and swap it with the second element.
3. we keep doing this until we get all elements moved to correct position.

Sort the following numbers using selection sort algorithm.

89, 45, 68, 90, 29, 34, 17

89	17	17	17	17	17	17
45	45	29	29	29	29	29
68	68	68	34	34	34	34
90	90	90	90	45	45	45
29	29	45	45	90	68	68
34	34	34	68	68	90	89
17	89	89	89	89	89	90

Bubble Sort

In this sorting, compare the adjacent elements of the list and swap if they are in the wrong order. we sort the array using multiple passes. After the first pass, the maximum element goes to end. Same way, after second pass, the second largest element goes to second last position and so on. In every pass, we process only those elements that have already not moved to correct position.

Sort following numbers using bubble sort

40, 30, 50, 60, 10, 20

40	30	30	30	30	30	30	30	30	30
30	40	40	40	40	40	40	40	40	40
50	50	50	50	50	50	50	50	10	10
60	60	60	60	10	10	10	10	50	20
10	10	10	10	60	20	20	20	20	50
20	20	20	20	20	60	60	60	60	60

30	30	30	10	10
40	10	10	30	20
10	40	20	20	30
20	20	40	40	40
50	50	50	50	50
60	60	60	60	60

Insertion sort

it works by iteratively inserting each element of an unsorted list into its correct position in a sorted portion of the list.

steps of insertion sort

1. consider the first element as sorted. Then pick the second element and compare it with the first. If it is smaller, insert it before the first element. If it is larger, leave it where it is.
2. Now, consider, the third element. Compare it with the sorted portion (the first two elements). Insert it in the correct position in the sorted portion of the array.

3. Continue this process for each element, picking one element at a time and inserting it into the sorted portion of the array.

Sort the following numbers using insertion sort technique.

70, 40, 60, 50, 20, 30, 10

70	40	40	40	40	40	40	40	40
40	70	60	60	50	50	50	50	20
60	60	70	50	60	60	20	20	50
50	50	50	70	70	20	60	60	60
20	20	20	20	20	70	70	70	70
30	30	30	30	30	30	30	30	30
10	10	10	10	10	10	10	10	10

20	20	20	20	20	20	20	20	10
40	40	40	40	30	30	30	10	20
50	50	50	30	40	40	40	30	30
60	60	30	50	50	50	40	40	40
70	30	60	60	60	60	50	50	50
30	70	70	70	70	10	60	60	60
10	10	10	10	10	70	70	70	70

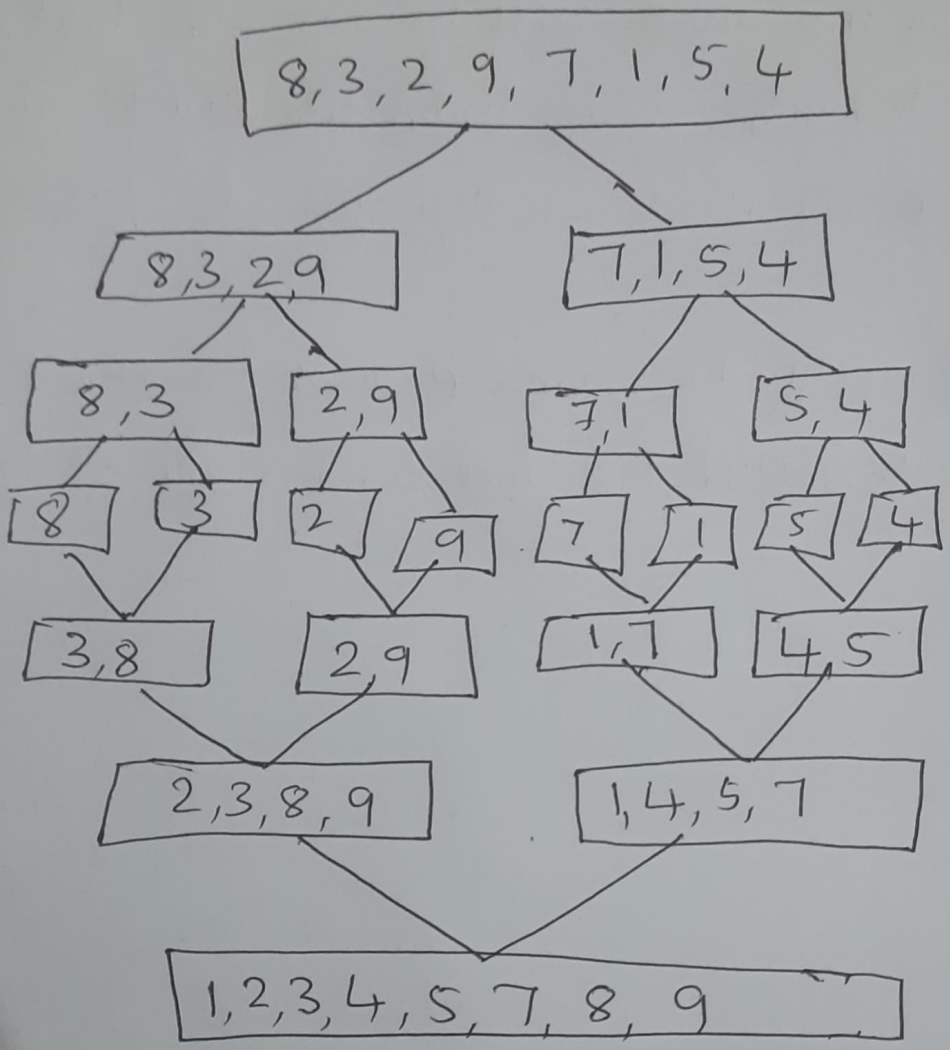
Merge sort

The merge sort works as follows:

1. Divide the sequence of elements into two equal parts
2. Recursively sort the elements on the left part.
3. Recursively sort the elements on the right part.
4. merge the sorted left and right parts into a single sorted array.

Sort the following numbers using merge sort.

8, 3, 2, 9, 7, 1, 5, 4



quick sort

In this sorting, picks an element as a pivot and partions the given array around the picked pivot by placing the pivot in its correct position in the sorted array. There are mainly three steps in the algorithm.

1. Select an element from the array as the Pivot.
2. Partition the array. After partioning, all elements smaller than the pivot will be on its left, and all elements greater than the pivot will be on its right. The pivot is then in its correct position.
3. Recursively apply the same process to the two partioned sub-arrays. i.e left and right of the pivot.