**1. What is Sorting?**

Sorting is the process of arranging elements in a specific order, typically ascending or descending. It is commonly used in data processing to make data retrieval and analysis easier and more efficient.

* **Internal Sorting:** When all data to be sorted fits into memory (RAM), it is known as internal sorting. Examples include bubble sort, insertion sort, and quick sort.
* **External Sorting:** When data is too large to fit into memory, it is divided and processed in smaller chunks. External sorting algorithms, like merge sort for disk-based sorting, are designed to handle large data by using external storage like disks.

When sorting:

* **Numbers** are arranged in ascending (default) or descending order.
* **Strings** are arranged in dictionary (alphabetical) order by default.

**2. Various Sorting Algorithms**

**1. Bubble Sort**

* **Definition:** Bubble Sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent pairs, and swaps them if they are in the wrong order.
* **Graph (Complexity):** O(n²) for time in the worst and average cases, O(1) for space.

**2. Modified Bubble Sort**

* **Definition:** Modified Bubble Sort optimizes the basic bubble sort by adding a flag. If no swaps are made during a pass, the list is already sorted, and the algorithm stops early.

**3. Selection Sort**

* **Definition:** Selection Sort repeatedly finds the minimum element from the unsorted part and places it at the beginning of the sorted part.
* **Graph (Complexity):** O(n²) for time in all cases, O(1) for space.

**4. Insertion Sort**

* **Definition:** Insertion Sort builds the final sorted array one item at a time. It takes each element and places it in its correct position among already sorted elements.
* **Graph (Complexity):** O(n²) time complexity in the worst case, O(n) in the best case (when nearly sorted), O(1) for space.

**5. Quick Sort**

* **Definition:** Quick Sort is a divide-and-conquer algorithm. It selects a 'pivot' element and partitions the array around the pivot, placing elements smaller than the pivot on one side and larger elements on the other, then recursively sorts the partitions.
* **Graph (Complexity):** O(n log n) on average, O(n²) in the worst case, O(log n) for space due to recursive calls.

**6. Merge Sort**

* **Definition:** Merge Sort is a divide-and-conquer algorithm that divides the list into two halves, sorts each half recursively, and then merges the two sorted halves into a single sorted list.
* **Graph (Complexity):** O(n log n) for time in all cases, O(n) for space due to auxiliary array use.

**7. Heap Sort**

* **Definition:** Heap Sort builds a max heap from the data, then removes the maximum element (root of the heap) and places it at the end of the array. The process is repeated until all elements are sorted.
* **Graph (Complexity):** O(n log n) for time in all cases, O(1) for space.

**3. Bubble Sort Program**

Here’s a simple Python implementation of Bubble Sort:

python

Copy code

def bubble\_sort(arr):

n = len(arr)

for i in range(n):

# Last i elements are already sorted

for j in range(0, n-i-1):

# Swap if the element found is greater than the next element

if arr[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

return arr

# Example usage

array = [64, 34, 25, 12, 22, 11, 90]

sorted\_array = bubble\_sort(array)

print("Sorted array:", sorted\_array)

**What is Modified Bubble Sort?**

In Modified Bubble Sort, we introduce a flag variable to check if any swaps were made during an iteration. If no swaps were made, the list is already sorted, and we can stop early, reducing unnecessary comparisons. This optimization makes it faster when the list is nearly sorted.

Modified Bubble Sort Program:

python

Copy code

def modified\_bubble\_sort(arr):

n = len(arr)

for i in range(n):

swapped = False

for j in range(0, n-i-1):

if arr[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

swapped = True

# If no two elements were swapped in the inner loop, break

if not swapped:

break

return arr

# Example usage

array = [64, 34, 25, 12, 22, 11, 90]

sorted\_array = modified\_bubble\_sort(array)

print("Sorted array with modified bubble sort:", sorted\_array)