Comparison of some commonly used sorting algorithms based on their key characteristics:

## 1. Time Complexity:

- Bubble Sort: $O(n^2)$
- Selection Sort: $O(n^2)$
- Insertion Sort: $O(n^2)$
- Merge Sort: O(n log n)
- Quick Sort: O(n log n) average case, $O(n^2)$ worst case
- Heap Sort: O(n log n)
- Radix Sort: O(d * (n + k)), where ‘d’ is the maximum number of digits, ‘n’ is the number of elements, and ‘k’ is the range of digits.

## 2. Space Complexity:

- Bubble Sort: O(1)
- Selection Sort: O(1)
- Insertion Sort: O(1)
- Merge Sort: O(n)
- Quick Sort: O(log n) for the recursive call stack (in-place sorting can achieve O(1) auxiliary space)
- Heap Sort: O(1)
- Radix Sort: O(n + k), where ‘n’ is the number of elements and ‘k’ is the range of digits.

## 3. Stability:

- Bubble Sort: Stable

- Selection Sort: Not stable
- Insertion Sort: Stable
- Merge Sort: Stable
- Quick Sort: Not stable
- Heap Sort: Not stable
- Radix Sort: Stable

## 4. Best Case Scenario:

- Bubble Sort: O(n) when the list is already sorted
- Selection Sort: $O(n^2)$
- Insertion Sort: O(n) when the list is already sorted
- Merge Sort: O(n log n)
- Quick Sort: O(n log n)
- Heap Sort: O(n log n)
- Radix Sort: O(d * (n + k)), where ‘d’ is the maximum number of digits, ‘n’ is the number of elements, and ‘k’ is the range of digits.

## 5. Worst Case Scenario:

- Bubble Sort: $O(n^2)$
- Selection Sort: $O(n^2)$
- Insertion Sort: $O(n^2)$
- Merge Sort: O(n log n)
- Quick Sort: $O(n^2)$
- Heap Sort: O(n log n)

- Radix Sort: O(d * (n + k)), where ‘d’ is the maximum number of digits, ‘n’ is the number of elements, and ‘k’ is the range of digits.