1. Bubble Sort:
- Komplexität: Im besten Fall (bereits sortierte Liste) hat Bubble Sort eine lineare Zeitkomplexität von O(n), im durchschnittlichen und schlechtesten Fall hat es jedoch eine quadratische Zeitkomplexität von O(n^2).
- Speicherbedarf: Bubble Sort sortiert die Elemente in-place, d.h. es benötigt keinen zusätzlichen Speicher.
- Stabilität: Bubble Sort ist stabil, da es gleiche Elemente nicht vertauscht.
- Art des Algorithmus: Bubble Sort ist ein vergleichsbasiertes Sortierverfahren, bei dem benachbarte Elemente paarweise verglichen und bei Bedarf vertauscht werden.
2. Selection Sort:
- Komplexität: Selection Sort hat unabhängig von der Eingabe eine quadratische Zeitkomplexität von O(n^2), sowohl im besten als auch im durchschnittlichen und schlechtesten Fall.
- Speicherbedarf: Selection Sort sortiert die Elemente in-place, ohne zusätzlichen Speicherbedarf.
- Stabilität: Selection Sort ist instabil, da die Reihenfolge gleicher Elemente verändert werden kann.
- Art des Algorithmus: Selection Sort ist ein vergleichsbasiertes Sortierverfahren, bei dem das kleinste Element in jedem Durchlauf ausgewählt und an die richtige Position verschoben wird.
3. Insertion Sort:
- Komplexität: Im besten Fall (bereits sortierte Liste) hat Insertion Sort eine lineare Zeitkomplexität von O(n), im durchschnittlichen und schlechtesten Fall hat es jedoch eine quadratische Zeitkomplexität von O(n^2).
- Speicherbedarf: Insertion Sort sortiert die Elemente in-place, ohne zusätzlichen Speicherbedarf.
- Stabilität: Insertion Sort ist stabil, da es gleiche Elemente nicht vertauscht.
- Art des Algorithmus: Insertion Sort ist ein vergleichsbasiertes Sortierverfahren, bei dem jedes Element an die richtige Position in einem bereits sortierten Teil der Liste eingefügt wird.
4. Quick Sort:
- Komplexität: Im durchschnittlichen Fall hat Quick Sort eine durchschnittliche Zeitkomplexität von O(n log n), im schlechtesten Fall kann es jedoch eine quadratische Zeitkomplexität von O(n^2) haben. Im besten Fall liegt die Zeitkomplexität ebenfalls bei O(n log n), wenn eine gute Pivot-Strategie verwendet wird.
- Speicherbedarf: Quick Sort benötigt zusätzlichen Speicher für die Rekursionsaufrufe im Stack.
- Stabilität: Quick Sort ist in der Regel instabil, da die Reihenfolge gleicher Elemente beim Partitionieren verändert werden kann. Es gibt jedoch Variationen des Algorithmus, die Stabilität erreichen können.
- Art des Algorithmus: Quick Sort ist ein vergleichsbasiertes Sortierverfahren, das auf dem Prinzip der Teile-und-Herrsche (Divide-and-Conquer) basiert. Es wählt

ein Element als Pivot, partitioniert die Liste um das Pivot und wendet den Algorithmus rekursiv auf die beiden Teillisten an.

5. Merge Sort:

- Komplexität: Merge Sort hat eine stabile Zeitkomplexität von O(n log n) in allen Fällen (best, average, worst).
- Speicherbedarf: Merge Sort benötigt zusätzlichen Speicher in Form einer temporären Liste, um die Teillisten zu fusionieren.
- Stabilität: Merge Sort ist stabil, da die Reihenfolge gleicher Elemente beim Zusammenführen der Teillisten erhalten bleibt.
- Art des Algorithmus: Merge Sort ist ein vergleichsbasiertes Sortierverfahren, das auf dem Prinzip der Teile-und-Herrsche basiert. Es teilt die Liste in immer kleinere Teillisten auf, sortiert sie einzeln und fusioniert sie dann, um die sortierte Gesamtliste zu erhalten.

Andere Sortierverfahren wie Bucketsort und Radixsort, die direkt die Sortierschlüssel verwenden, sind hier nicht näher beschrieben, da sie spezifische Anforderungen an die Art der Eingabe haben und nicht allgemein für beliebige Listen verwendet werden können.