Name: Adwait Purao
Uid: 2021300101   Batch: B2

Insertion sort divides the input array into sorted and unsorted parts, picks an element from the unsorted part and inserts it into the correct position in the sorted part. It has a time complexity of O(n^2) in the worst case but can perform better on partially sorted arrays. It is simple to understand, efficient for small data sets or nearly sorted data, in-place, and stable. However, it is inefficient for large or reverse-ordered data sets and has a lot of swaps, making it slow on modern computers.
Selection sort repeatedly finds the minimum element in the unsorted part and swaps it with the leftmost element of the unsorted part. It has a time complexity of O(n^2) in all cases and requires fewer swaps but more comparisons than insertion sort. It is simple to understand, efficient for small or nearly sorted data sets, and in-place. However, it is inefficient for large data sets, has many comparisons, and is unstable. The choice between selection sort and insertion sort depends on the characteristics of the input data and the problem requirements.