BUBBLE SORT

How does Bubble Sort Work?

Bubble Sort commences by initiating a traversal from the initial element of the array. For each element, an iterative loop navigates through the subsequent elements once. During this process, comparisons are executed. In the subsequent loop, if a smaller element compared to the one in the first loop is identified, they interchange positions. This sequence persists, progressively arranging the list in ascending order, with the smallest element gradually moving towards the beginning. In the second loop, the condition "j < array.length - i – 1" with the "1" signifies that checking the final index is unnecessary since the largest element consistently finds its place there after each iteration. The variable "i" accounts for the fact that scrutiny of the segments that are already correctly positioned towards the end of the indices is superfluous.

Time Complexity

Because of its utilization of two nested loops, the Bubble Sort algorithm exhibits a time complexity of O(N^2).

Space Complexity

Bubble Sort boasts a space complexity of O(1) due to its in-place sorting nature, eliminating the need for additional memory that scales with the input size.

Discussion

Bubble sort has low space complexity but suffers from high time complexity, making it suitable mainly for small datasets. Its inefficiency becomes pronounced as dataset size increases. For larger datasets, more optimized algorithms are preferred.

Here is the visualization result of the bubble sort implementation, where adjacent elements are swapped to gradually arrange the array: