**Analysis of time complexities**

**Explanation of Time Complexity:**

1. **Worst-Case Scenario:**

* In the worst-case scenario, the array is in reverse sorted order, and each pass of the algorithm requires comparing and swapping almost every pair of elements.
* Each pass consists of two phases: an odd phase and an even phase. Each phase iterates through approximately n/2 pairs of elements.
* In each pass, there are two nested loops iterating over the odd-indexed and even-indexed elements, respectively. Each loop performs comparisons and swaps, leading to a total of approximately n/2 comparisons and swaps per pass.
* Since there can be up to n passes until the array is fully sorted, the total number of comparisons and swaps is roughly (n/2) \* n = O(n^2).

1. **Best-Case Scenario:**

* In the best case scenario, the array is already sorted and comparisons as in the worst case, resulting in the same time complexity of O(n^2). In the best-case scenario, the array is already sorted, and no swaps are required during any pass of the algorithm.
* However, the algorithm still performs the same number of passes and comparisons as in the worst case, resulting in the same time complexity of O(n^2).

1. **Average-Case Scenario:**

* The average-case time complexity of Odd-Even sort is also O(n^2).
* Although the actual number of comparisons and swaps may vary depending on the initial order of elements, the algorithm's behavior tends towards O(n^2) on average.

**Conclusion:**

The Odd-Even sort algorithm exhibits quadratic time complexity O(n^2), making it less efficient compared to many other sorting algorithms like quicksort or merge sort, which have average-case time complexities of O(n log n). However, Odd-Even sort's simplicity and ease of implementation make it suitable for educational purposes or for sorting small datasets where efficiency is not a primary concern.