The Algorithm chosen: Radix Sort Algorithm.

Radix Sort is a non-comparative integer sorting algorithm that processes digits of the numbers one at a time, from the least significant digit (LSD) to the most significant digit (MSD), or vice versa. It leverages a stable subroutine, typically Counting Sort, to sort based on each digit.

Optimization 1:

- **Original Approach:** Radix Sort often uses a base-10 system, which results in multiple passes over the data, equal to the number of digits in the largest number.
- **Refinement:** Use a higher base (e.g., base-16 or base-256) to reduce the number of passes. This reduces the number of iterations over the dataset, as fewer digits are required to represent each number in higher bases.
- **Theoretical Impact:** By reducing the number of passes, the overall time complexity can be reduced from O(d*(n+k)) to O(d/(log b (10)) * (n+b)) where b is the new base.
- **Empirical Comparison:** A higher base version of Radix Sort typically shows a noticeable decrease in execution time, especially for larger datasets with numbers containing many digits.

Result of applying that optimization:

```
(base) lokeshbudda@Lokeshs-MacBook-Air Q2 % ./optimizedRadixsort
Choose input method:
1. Generate Large Random Numbers (1000000 elements)
2. Read from Input File
1

Base-10 Radix Sort Results:
Comparisons: 0
Swaps: 7000000
Basic operations: 21000063
Execution time: 159 ms
Memory usage: 4000000 bytes

Base-16 Radix Sort Results:
Comparisons: 0
Swaps: 5000000
Basic operations: 15000075
Execution time: 113 ms
Memory usage: 4000000 bytes
```

The test case with 1 million elements, ranging from 1 to 1,000,000. It demonstrates a significant performance difference between the Base-10 and Base-16 versions. Base-16 faster due to fewer passes needed over the array.