Least Significant Digit (LSD)

1. **Uniform Length Requirement:**
    - **Strings:** All strings must have the same number of characters.
    - **Integers:** Each integer is considered as a fixed-length sequence of bytes.
2. **Processing Digits from Right to Left:**
    - The algorithm starts with the least significant digit (the rightmost position) and sorts the array based solely on the values at that position.
    - It then proceeds to the next digit to the left, repeating the sorting process. This is done for each digit until the most significant digit is sorted.
3. **Using Key-Indexed Counting (Counting Sort):**
    - For each digit position, the algorithm uses key-indexed counting, a technique that counts the frequency of each possible digit (or character) value.
    - These counts are then accumulated to determine the starting index (or bucket) for each digit in a temporary storage array.
    - Items are distributed into this temporary array according to their digit at the current position.
    - After the distribution, the sorted order based on that digit is copied back to the original array.
4. **Stability:**
    - The key to LSD Radix Sort is that each pass uses a stable sort, meaning that items with the same digit value maintain their relative order from previous passes.
    - This stability ensures that the sorting done on less significant digits is preserved as the algorithm processes more significant digits.
5. **Completion:**
    - After processing all digit positions from rightmost to leftmost, the array is fully sorted in lexicographical order (for strings) or numerical order (for integers).

Characteristics and Advantages

- **Efficiency:**
  The algorithm is efficient for fixed-length data since it processes each digit in a linear pass. Its performance depends primarily on the number of items and the number of digits in each item.

- **Memory Usage:**
  It requires additional memory proportional to the number of items plus the range of possible digit values (for example, 256 for extended ASCII).

- **Language Agnostic:**
  The process described here focuses on the underlying concepts—counting frequencies, accumulating counts, distributing items, and repeating the process for each digit—making it applicable in any programming language.