Radix Sort

Introduction

Radix Sort is a non-comparative sorting algorithm that works by distributing elements into different buckets based on their digits or characters. It is particularly effective when sorting integers or strings with fixed-length representations. This sorting technique relies on the idea of sorting elements digit by digit, from the least significant digit (LSD) to the most significant digit (MSD), or vice versa.

Logic

1. **Least Significant Digit (LSD) Radix Sort**:
    - Begin by sorting the elements based on their least significant digit (e.g., rightmost digit for integers or characters for strings).
    - Create a set of buckets, one for each possible digit value (0-9 for base 10 numbers).
    - Distribute elements into these buckets based on their LSD.
    - Gather elements back into a single array, maintaining the order of distribution.
    - Repeat the process for the next significant digit (tens, hundreds, etc.) until the most significant digit is processed.
2. **Most Significant Digit (MSD) Radix Sort**:
    - Start by sorting elements based on their most significant digit.
    - Divide the elements into buckets based on the value of the MSD.
    - Recursively apply the MSD Radix Sort to each bucket until all digits are processed.
    - Concatenate the sorted buckets to obtain the final sorted array.

C++ implementation of Radix sort: Radix Sort.
Python implementation of Radix sort: Radix Sort.
Java implementation of Radix sort: Radix Sort.
Javascipt implementation of Radix sort: Radix Sort.

Pseudo Code

```
1. Find the maximum digit in the data to determine the number of passes required for sorting
2. Do the following for each pass:
   - Create buckets (lists) to hold the digits (0 to 9 for the decimal system).
   - Scan the input array, count the occurrences of each digit, and store them in the corres
   - Replace the input array with values in buckets in ascending order.
```

Complexity

- **Time Complexity**:

  - Radix Sort has a time complexity of O(n*k) where n is the number of elements to be sorted, and k is the number of digits (or characters) in the maximum value.
  - Counting Sort, which is used as a subroutine, has a time complexity of O(n + k) for each digit pass.
- **Space Complexity**:
  - The space complexity of Radix Sort is O(n + k), where n is the number of elements and k is the base of the numbering system (e.g., 10 for decimal).

Advantages

- Radix Sort is a stable sort, preserving the relative order of equal elements.
- It is particularly efficient for sorting large datasets with a limited range of values.
- It can be applied to integers, strings, or other data types.
- Radix Sort does not rely on comparison operations, making it faster than many comparison-based sorting algorithms in certain cases.

Considerations

- Radix Sort is most efficient when the range of values in the dataset is not significantly larger than the number of elements.
- It may not be suitable for sorting data with variable-length representations.
- The choice of LSD or MSD Radix Sort depends on the data and the desired sorting order (ascending or descending).

Disadvantages

- Radix Sort may require additional memory for storing buckets, making it less memory-efficient for very large datasets.
- For large datasets with a wide range of values, it can become inefficient due to a high number of passes.

Limitations

- Radix Sort is not a general-purpose sorting algorithm and may not be the best choice for all scenarios.
- It is primarily suitable for sorting non-negative integers or fixed-length strings.
- The choice of base (e.g., base 10 for decimal numbers) affects the algorithm's performance.

Edge Cases

- Radix Sort works well for sorting integers with a fixed number of digits or characters. It may not perform efficiently for datasets with variable-length representations.
- When sorting strings, it is essential to handle strings of different lengths correctly to avoid unexpected results.

External Resources

- Wikipedia - Radix Sort
- GeeksforGeeks - Radix Sort
- Sorting Algorithm Animations - Radix Sort

These external resources provide in-depth explanations, visualizations, and additional examples of Radix Sort.