Counting Sort

Introduction

Counting Sort is an efficient sorting algorithm used to sort elements in linear time when the range of input values is small. It works by counting the occurrences of each element and using that information to place elements in their correct sorted order.

Logic

1. Determine the range of input values (min and max).
2. Create a frequency array to store the count of each unique element.
3. Count the occurrences of each element in the input array and update the frequency array.
4. Accumulate the frequencies to determine the final positions of elements.
5. Place each element in the correct position in the output array.

C++ implementation of Count sort: CountSort Python implementation of Count sort: CountSort Java implementation of Count sort: CountSort Javascript implementation of Count sort: CountSort

Advantages

- Counting Sort is efficient for small ranges of input values.
- It is a stable sorting algorithm, preserving the relative order of equal elements.
- It works well when the range of input values is known and not significantly larger than the number of elements.

Disadvantages & Limitations

- Counting Sort is not suitable for sorting large ranges of input values, as it requires creating a large frequency array.
- It's not a comparison-based sorting algorithm, so it's not applicable to general sorting scenarios.
- Memory consumption can be a concern when dealing with a wide range of values.

Time and Space Complexity

- **Time Complexity:**
  - O(n + k), where n is the number of elements and k is the range of input values.
- **Space Complexity:**
  - O(k), where k is the range of input values.

#sorting [[SORTING]]