Countıng algorıthm of e-educatıon based on modern technologıes

Kenan Gasimzadeh

Abstract
Count sort is a sorting algorithm that sorts the elements of an array in logical order, counting the number of times that each unique element exists in the array, and can be used mostly in cases where the range of values in the input array is relatively small compared to the size of the array. We will talk about its disadvantages, comparison with other algorithms, and areas of application.

Key words: Countıng Algorıthm, digital signal, correct sorted position.

Counting sort is a simple yet fast sorting algorithm used in computer science. The counting algorithm is mainly used to sort numbers in a small range and to solve many counting problems. Sort by count is a sorting algorithm commonly used when the range of values in an array is relatively small. It is a comparison-free algorithm that works by counting the number of occurrences of each distinct element in the input array and then using this information to place each element in the correct sorted position.

Overall, counting sort is a useful and efficient algorithm for sorting arrays of integers with a small range of values, and is a valuable tool in many areas of computer science and engineering.counting variety has many applications in various fields of computer science and engineering. Here are some examples:

1. Digital Signal Processing
2. Sim Compatibility
3. Image Processing
4. Database Management
5. Computational Biology

1) A count sort is used in digital signal processing programs to sort and filter signals by frequency. It is especially useful when the frequency range is relatively small.

2) The count array is used in string matching algorithms to efficiently count the occurrence of each character in a string. This can help improve the efficiency of string matching algorithms such as the boyer-moore algorithm.

3) Used in image processing programs to sort and filter image data based on color or intensity. It can be especially useful when processing images with a small color range or intensity values.

4) Computational sorting can be used to efficiently sort and search data based on specific fields in database management systems. it can be especially useful when working with large datasets that have a small range of values in a particular field.

5) Counting sort is used in computational biology programs to efficiently sort and search dna sequences based on nucleotide base pairs. it can be especially useful when looking for patterns in dna sequences.

Time Complexity Of Sorting By Counting:

Total Complexity = O(Max)+O(Size)+O(Max)+O(Size) = O(Max+Size)

1) Worst Case Complexity: O(N+K)
2) Best Case Complexity: O(N+K)
3) Average Job Complexity: O(N+K)

In all the above cases, the complexity is the same because the algorithm runs n+k times regardless of how the elements are placed in the array.

Problems with sorting by counting:

1) limited to small input ranges
2) not stable for incomplete data
3) requires integer keys

1- Count sort only works for input data with small range of values. If the input data consists of elements much larger than the input size, the count sort can be inefficient in terms of memory usage.

[1000000,500,1000000,200,300]
2- If we sort this list using counting sort, it is not guaranteed to preserve the order of elements with the same number. this can be a problem if you need to keep the original order of elements with the same number.
[“Apple”,“Banana”,“Pear”,“Kiwi”,“Orange”]
3-Count sort cannot be applied directly to this list because it requires integer keys. one solution would be to convert floating-point numbers to integers by multiplying them by a factor and then rounding them to the nearest integer. however, this can be computationally expensive and lead to accuracy problems.
[3.5, 2. 0, 1.5,4 .0, 3.0]
Advantages of Sorting By Count:
Fast: count sort can run very fast depending on the range of items to be sorted.
constant memory usage: counting sort uses memory equal to the number of elements to be sorted and uses less memory than some other sorting algorithms due to constant memory usage.
Simple: count sort is relatively simple and easy to understand compared to other sorting algorithms.
Disadvantages of sorting by counting:
Memory problem: sorting a count requires as much memory as the number of items to sort. Therefore, there may be a memory shortage for very large datasets.
Consistency: an enumeration sort is constant, meaning that elements with the same value are sorted in the order of input.
Mixed elements: a count sort assumes that the elements to be sorted are distinct and unique. that is, trying to sort the same element twice may result in wrong results.2, 2, 4, 41, 5, 5.
differences in the count-by-sort algorithm:
Stability - count sort is a stable sorting algorithm, that is, it preserves the relative order of equal elements. this feature makes it useful in various applications such as database management and record keeping.
Space complexity – count sorting requires additional space proportional to the range of input values to store the count array, which can make it less space efficient than other sorting algorithms in some cases.
In-Place Sort - Count Sort İs Not İn-Place Sort Algorithm Because İt Requires Additional Memory To Store An Array Of Numbers.
Counting sort algorithm:
1. Set the range of values in the input array
2. Create a count array.
3. Count the repetition of each element.
4. Change the calculation array.
5. Extract the sorted array.
6. Return the sorted array.

```
// Finding the Largest Number In An Array
   For (Int I = 1; I < Size; I++) {If (Array[I] > Max)
         Max = Array [I];
   }
```

```
// Reset the İndices Of All The Numbers İn The Array
For (Int I = 0; I <= Max; ++I) {
      Count [I] = 0;
   }
```

```
// Finding The Number Of Repeated Elements İn An Array
   For (Int I = 0; I < Size; I++) {
      Count[Array[I]]++;
   }
-----------------------------------------------------------
// Stores The Total Count Of Each Array
   For (Int I = 1; I <= Max; I++)
                                                {
                                                     Count[I] += Count[I - 1];
                                                }
---------------------------------------------------------
/* Look Up The İndex Of Each Element Of The Actual Array İn The Count Array And Place The Elements
İn The Output Array*/
For (Int I = Size - 1; I >= 0; I--)
   {
      Output[Count[Array[I]] - 1] = Array[I];
      Count[Array[I]]--;
   }
-----------------------------------------------------------------
// Sorting An Array By Index
   For (Int I = 0; I < Size; I++) {
      Array[I] = Output[I];
   }
}
-------------------------------------------------------------
// Printing A Sorted Array
Void Display(Int Array[], Int Size) {
   For (Int I = 0; I < Size; I++)
      Cout << Array[I] << " ";
   Cout << Endl;
}
----------------------------------------------------------
```

Conclusion

Sort by count is a sorting algorithm that operates by counting the number of occurrences of each distinct element in the input array and then using arithmetic to calculate the position of each element in the sorted output array. It has (n+k) time complexity, where n is the number of elements in the input array and k is the range of the input data. In general, counting sort is a useful algorithm for efficiently sorting small ranges of integers. However, this may not be the best choice for sorting large datasets or incomplete data.

References

[1] Clifford Stein Rivest Leiserson Cormen Introduction To Algorithms Third Edition Page 194
[2] Https://Www.Programiz.Com/Dsa/Counting-Sort
[3] Geeksforgeeks Https://Www.Geeksforgeeks.Org/Counting-Sort