Entropy optimal sorting algorithm

Nachiket Deo

August 2019

1 Introduction

Sorting is primary operations in computer science and has wide applications across the field.Their are variety of algorithms that have been developed that employ different design strategies.Once such algorithm is quicksort,in-place algorithm with divide and conquer strategy,widely known as fastest algorithm in terms of space and time complexity.In this paper we are going to explore a type of algorithm that uses three-way partitioning. In usual algorithm we sort the data that is less than pivot to it's left and the data that is greater than pivot to it's right. However, the class of algorithm we are exploring takes into account the keys that are equal to pivot as well.

2 Partitioning details

This partitioning techniques divides the array into three parts.The part named 'I' is Figure 1 contains the elements that are less than pivot. The part 'II' contains the elements that are equal to pivot and the part with label '?' has elements that are yet to be discovered. The last part with label 'III' contains the elements that are greater than pivot. This technique is particularly helpful where the number of distinct elements in array are less. The performance of the sorting algorithm is enhanced by the fact that after partitioning we recursively call the elements lesser than pivot and elements greater than pivot, leaving the elements that are equal to pivot from further processing as they are already in the correct position. This reduces the number of comparisons and swaps required in further iterations.

I	II	?	III

Figure 1: Three Way Partition

3 Previous Works

The Three-Way partition algorithm was designed by the Edsger Dijkstra as a solution for Dutch National Flag problem.It is elegant method by swapping the value that is less than pivot and collecting elements that are equal together and then recursively sorting the elements that are less than pivot and elements that are greater than pivot.The disadvantage of the algorithm is that when the data doesn't contain few distinct elements the number of swaps increase dramatically and overall performance is degraded,which is much considerably lower than regular Quick-sort. The number of comparisons are also increased thus, making it bad choice for such type of data. The another type of partitioning technique that was developed was known as Bentley - Mcilroy. It improves to sort performance by reduction of the number of swaps in the case where data doesn't contain few distinct elements. This algorithm involves moving the elements that are equal to pivot to the end of the array. If the elements is equal to the first element of the array then we move it to beginning of the array.If the element is to the end of the array then we move to the end. We also keep track of the number of elements that we move to beginning or the end of the array using variables.At the end of the partitioning process we get back the elements that we have transferred to the both ends of the array at their appropriate positions. As you can see this process involves lot of swaps as we move the repeated elements twice. This provides us a useful technique to reduce the swaps when the number of elements are distinct.

4 Algorithm

Algorithm 1 Three Way Partition Algorithm

```
procedure PARTITION($a[], low, high$)
    $pivot \leftarrow low$
    $i \leftarrow low + 1$
    $j \leftarrow low + 1$
    $PivotCounter \leftarrow pivot$
    while $i \leq high + 1$ do
        if $a[i] \leq a[pivot]$ then
            $swap(a[i], a[j])$
            $i \leftarrow i + 1$
            $j \leftarrow j + 1$
        else if $a[i] > pivot$ then
            $i \leftarrow i + 1$
        else
            $swap(a[i], a[j])$
            $swap(a[j], a[PivotCounter + 1])$
            $i \leftarrow i + 1$
            $j \leftarrow j + 1$
            $PivotCounter \leftarrow PivotCounter + 1$
        end if
    end while
    $k \leftarrow j - 1$
    $num \leftarrow 0$
    $i \leftarrow l$
    while $i \leq PivotCounter$ do
        $swap(a[k], a[i])$
        $k \leftarrow k - 1$
        $num \leftarrow num + 1$
    end while
    $Pivot \leftarrow (j - 1) - (num + 1)$
    $PivotCounter \leftarrow (j - 1)$
end procedure
```