**بسم الله الرحمن الرحيم**



**Computer Science Department**

**COMP2321**

**Data Structure In C**

**Project#4:Sorting Algorthims**

# 

# 

*Student Name:*

***Majd Khasib***

*ID #:*

***1162331***

*Instructors Name:*

***Iyad Jaber***

* Abstract:

In this report we are going to talk about some of the most used sorting algorithms ,their properties with examples and how they are compared.

Contents

[1). Counting Sort: 4](#_Toc532796683)

[2). Bucket Sort: 6](#_Toc532796684)

[3.)Cocktail Sort: 8](#_Toc532796685)

[4).Gnome Sort: 9](#_Toc532796686)

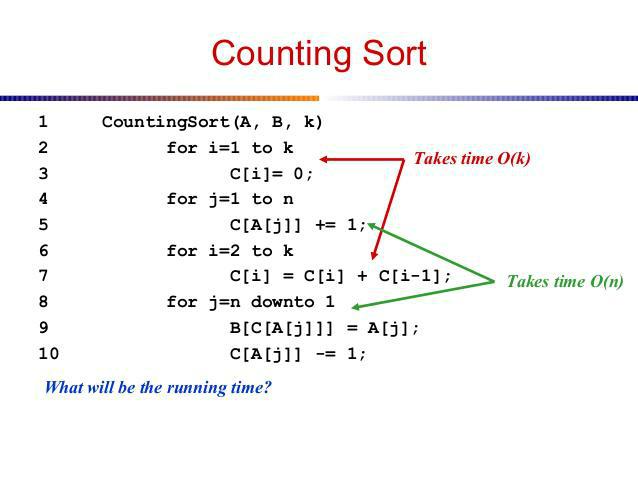
[5).Comb Sort: 11](#_Toc532796687)

[ Summary: 14](#_Toc532796688)

[ References: 15](#_Toc532796689)

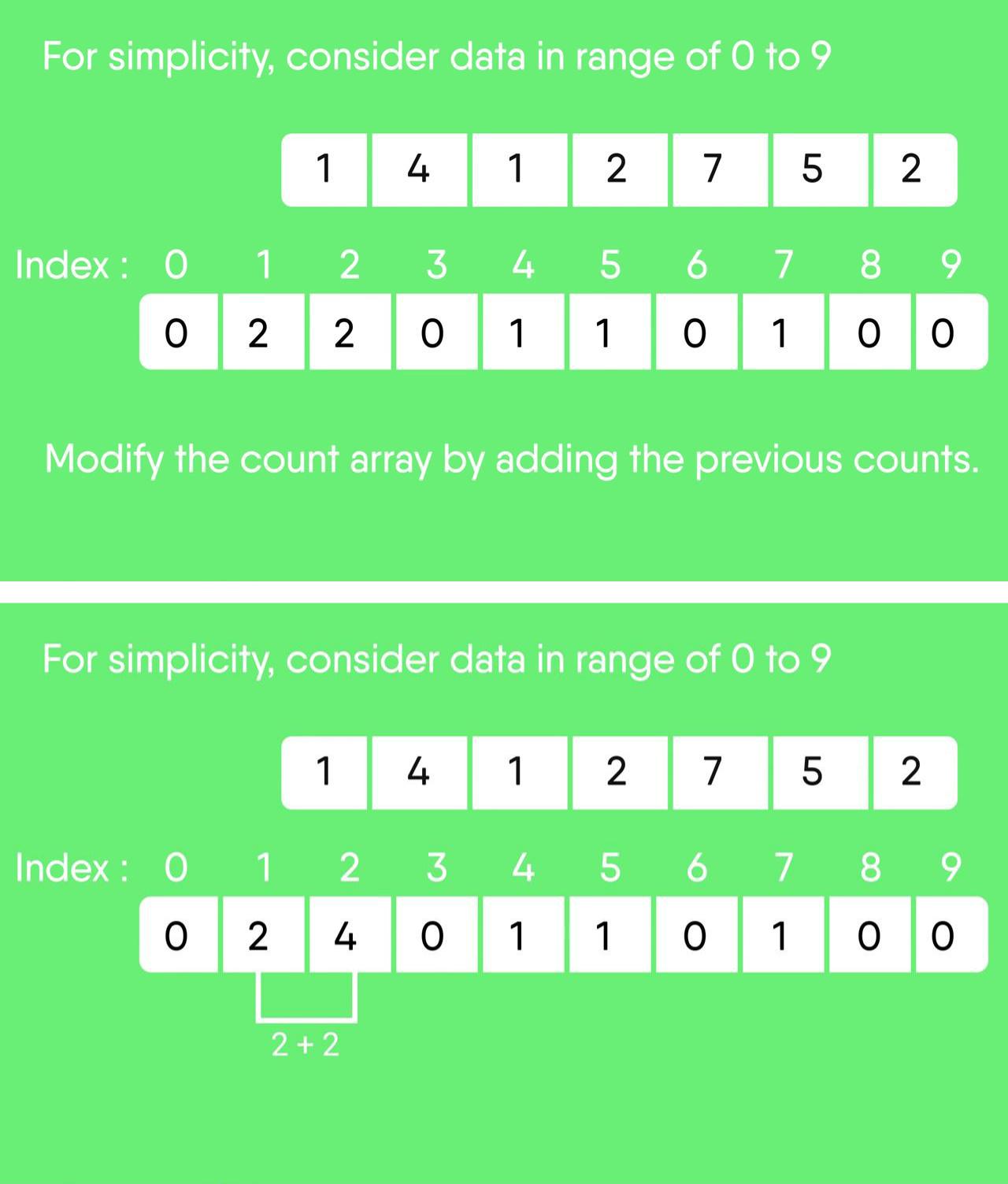
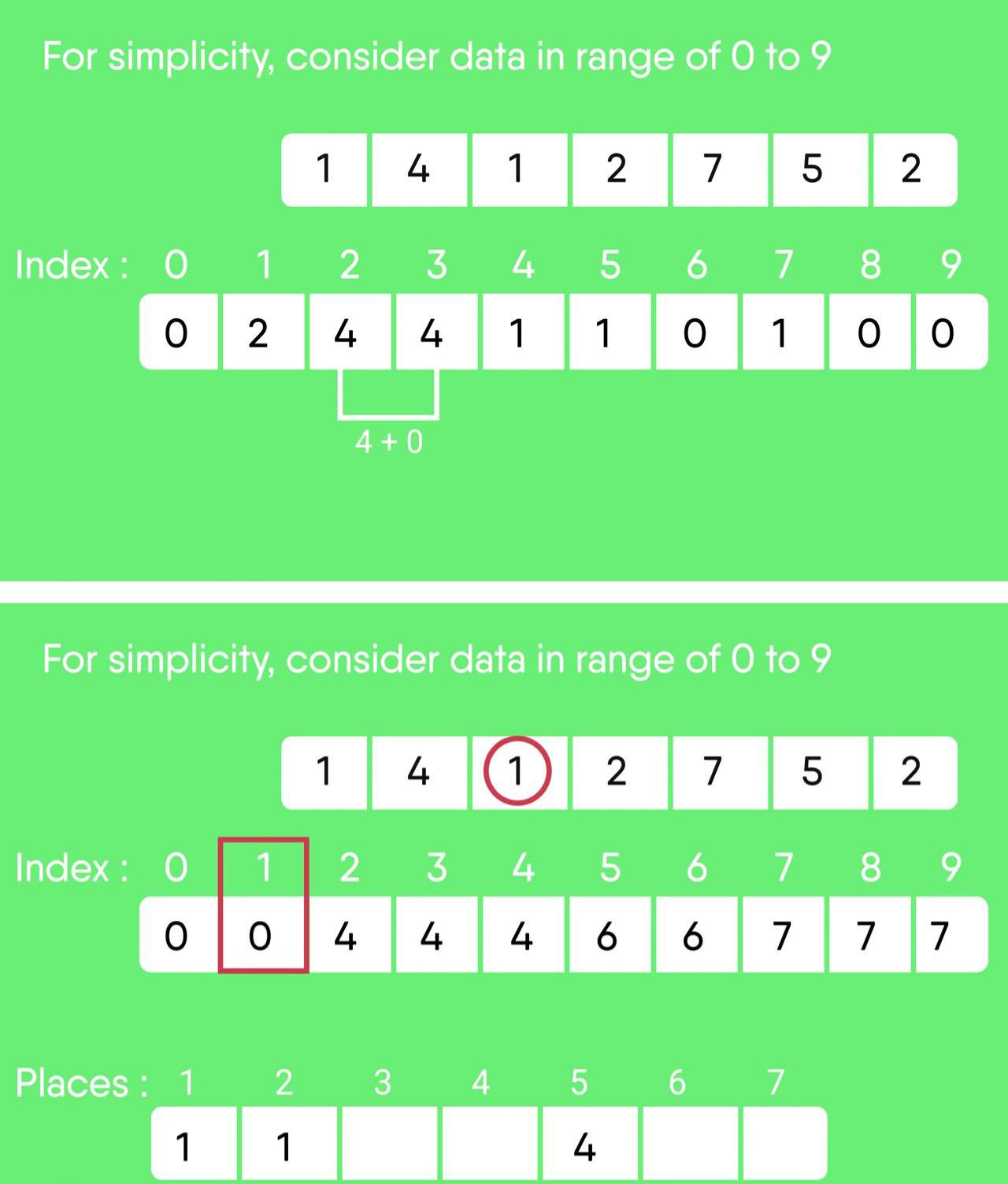
# 1). Counting Sort:

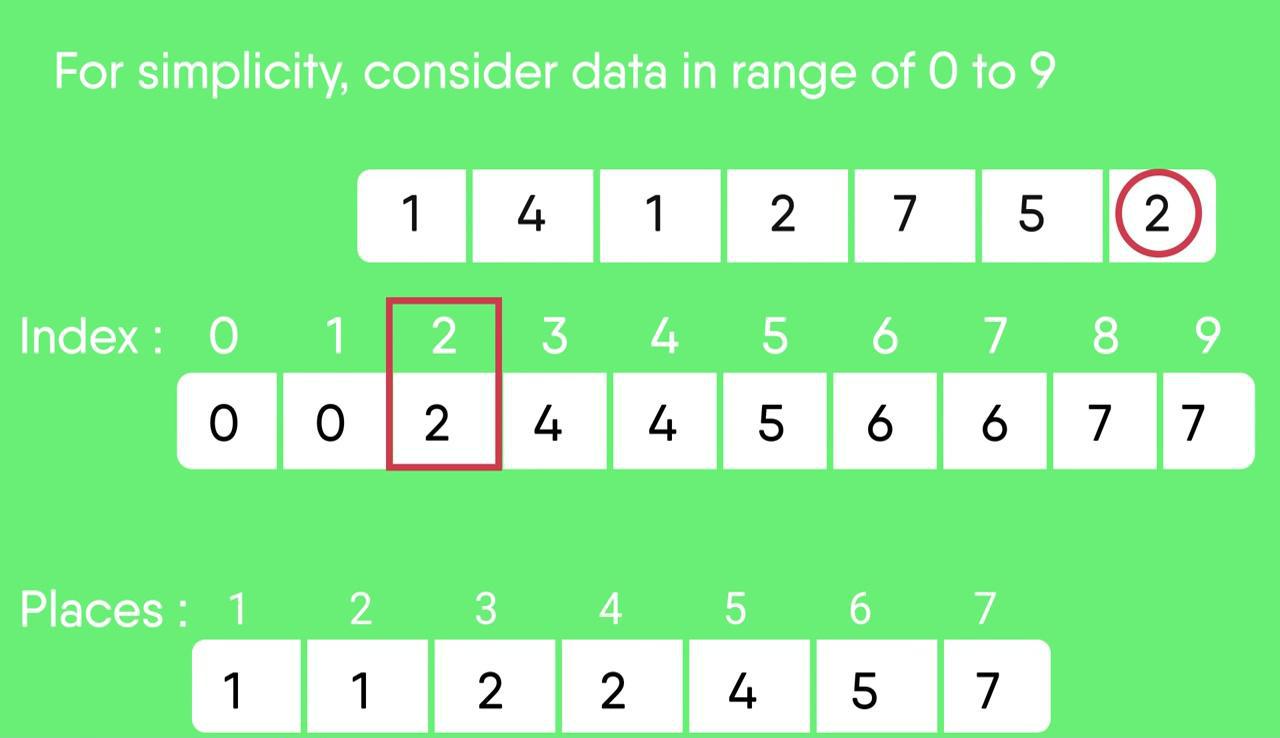
Counting Sort is easy to be implemented. It needs a range of the data to be sorted, uses the number of occurrences of each element as an index to get the sorted array.

The following pseudo code shows how the algorithm works:

To talk about the algorithm properties ,we can see in the blow table.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **The type of case** |  | Time complexity |
|  | **Worst (Reverse)**  **Average(Random)**  **Best(sorted** |  | O(n) |
|  | **Space** |  | Space=max(A[i]) |
|  | **Stability** |  | **Stable**(if line 8 was changed from start to end it become **unstable**) |

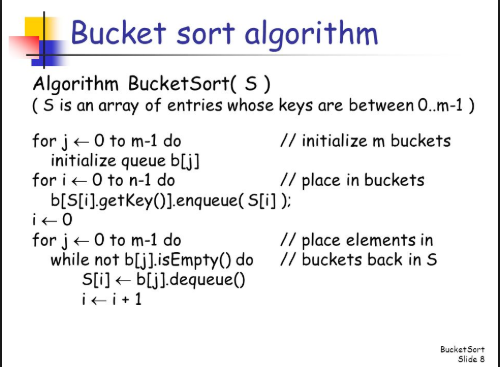
The following is an example of the sort technique :



# 2). Bucket Sort:

Bucket sort is mainly useful when input is uniformly distributed over a range. To use the bucket sort we need to build a bucket to store the elements of the array using a given way. After the elements is being stored in buckets ,each bucket is sorted using a given sorting method(let’s say insertion). Finally, The buckets will have the values sorted and to be moved into the original array.

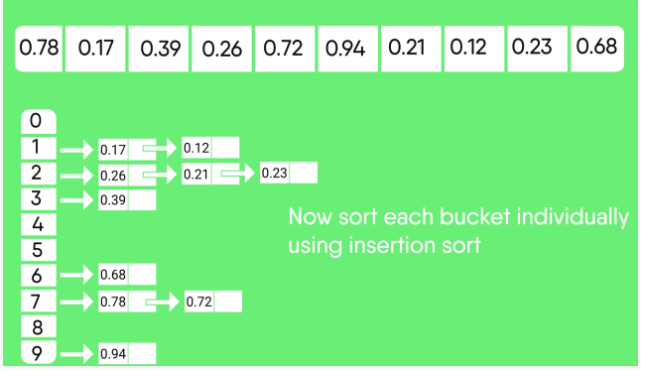
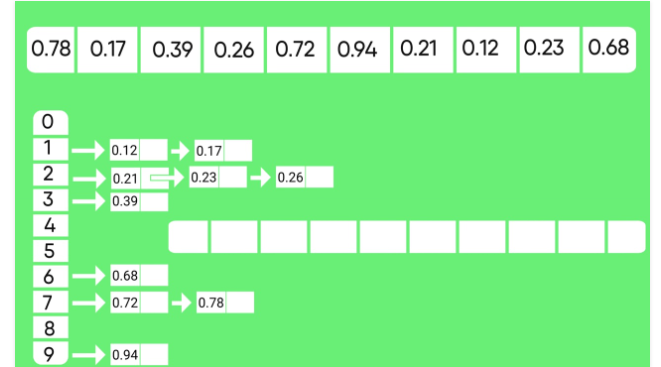
The following pseudo code shows the algorithm :



To talk about the algorithm properties ,see the following table:

|  |  |
| --- | --- |
| The type of case | Time complexity |
| Best  Average  worst | O(N+R),where N is number of elements ,R Number of buckets |
| space | Space=N+R |
| Stability | Stable |

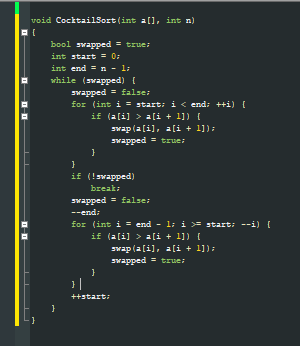
This is an example of Bucket sort:



# 3.)Cocktail Sort:

Cocktail Sort is a sorting algorithm that traverses the array from both directions at each iteration using a variation of Bubble sort, in a given iteration the largest number will hold at last in forward direction, while the smallest number will be at first of the start index of the given iteration.

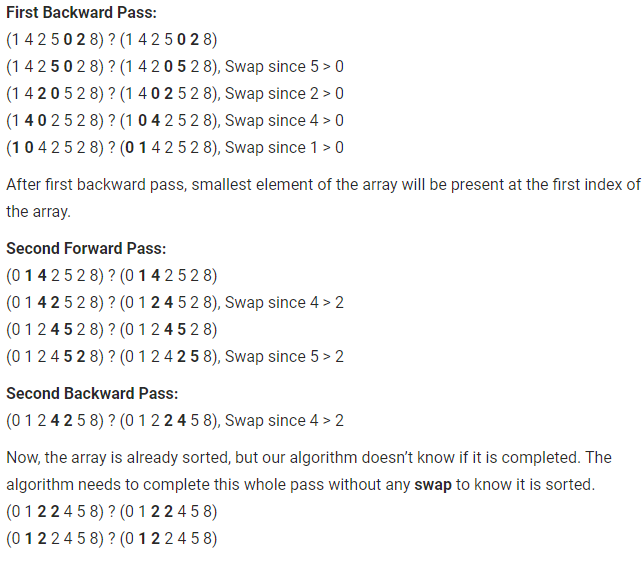
The following C code shows how the algorithm works:



The below table shows the algorithm properties :

|  |  |
| --- | --- |
| The type of case | Time complexity |
| Best(sorted)  Worst(Reversed) | O(n)  O(n\*n) |
| space | Space=C,c is a constant to hold swap varaible |
| Stability | Stable |

This example illustrates how the algorithm works:



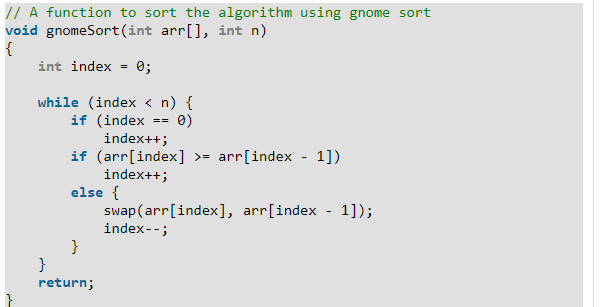
# 4).Gnome Sort:

Gnome Sort also called Stupid sort is based on the concept of a Garden Gnome sorting his flower pots. A garden gnome sorts the flower pots by the following method-

He looks at the flower pot next to him and the previous one; if they are in the right order he steps one pot forward, otherwise he swaps them and steps one pot backwards.

If there is no previous pot (he is at the starting of the pot line), he steps forwards; if there is no pot next to him (he is at the end of the pot line), he is done.

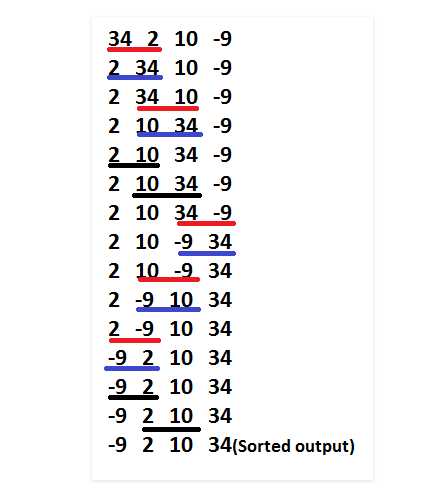
The following C code shows how the algorithm works:



The following table shows the properties of the algorithm:

|  |  |
| --- | --- |
| The type of case | Time complexity |
| Best(sorted)  Worst(Reversed) | O(n)  O(n\*n) |
| space | Space=C,c is a constant to hold swap varaible |
| Stability | Stable |

The following example illustrates the algorithm:



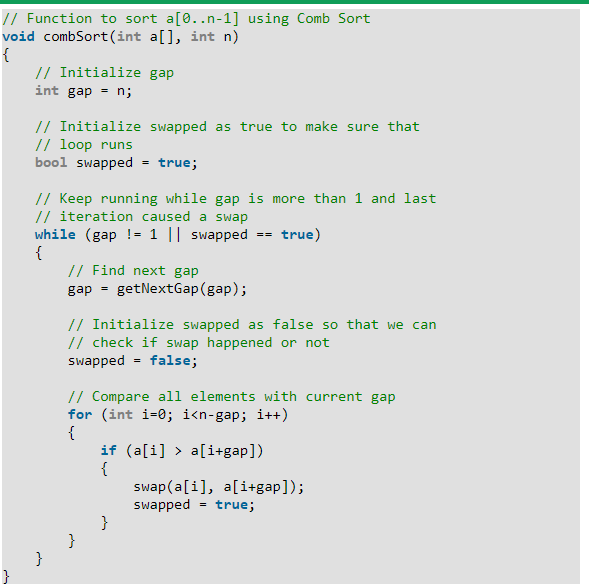
# 5).Comb Sort:

Comb Sort is mainly an improvement over Bubble Sort. Bubble sort always compares adjacent values. So all [inversions](https://www.geeksforgeeks.org/counting-inversions/) are removed one by one. Comb Sort improves on Bubble Sort by using gap of size more than 1. The gap starts with a large value and shrinks by a factor of 1.3 in every iteration until it reaches the value 1.

Thus Comb Sort removes more than one [inversion counts](https://www.geeksforgeeks.org/counting-inversions/) with one swap and performs better than Bubble Sort.

The shrink factor has been empirically found to be 1.3 (by testing Comb sort on over 200,000 random lists) [Source: [Wiki](https://en.wikipedia.org/wiki/Comb_sort)]

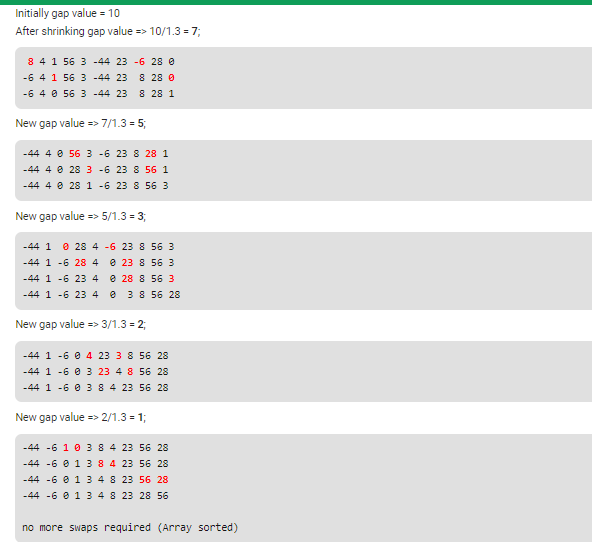
The pseudo code for the Comb sort is int the following figure:



The properties for the comp sort is:

|  |  |
| --- | --- |
| The type of case | Time complexity |
| Best(sorted)  Worst(Reversed) | O(n)  O(n\*n) |
| space | Space=C,c is a constant to hold swap varaible |
| Stability | Not Stable |

The following example shows the algorithm:



# Summary:

The following table shows time complexity, space and stability of the used algorithms.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Best** | **Average** | **Worst** | **Space** | **Stability** |
| **Counting** | n | n | n | 1 | Yes |
| **Bucket** | N+R | N+R | N+R | N+R | Yes |
| **Cocktail** | n | n\*n | n\*n | 1 | Yes |
| **Gnome** | n | n\*n | n\*n | 1 | Yes |
| **Comb** | n | n\*n /2^p | n\*n | 1 | No |

From the above table we can see the counting sort is the best sorting algorithm between them. However, it can be used for float or negative numbers.

# References:

[1]- http://en.wikipedia.org

[2]- <https://geeksforgeeks.org>

[3]- http://stackoverflow.com