Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if t hey are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

Bubble Sort Algorithm:
1- traverse from left and compare adjacent elements and the higher one is placed at right side.
2- In this way, the largest element is moved to the rightmost end at first.
3- This process is then continued to find the second largest and place it and so on until the data is sorted.
1

Implementaion for the bubble sort in Python:

```
# Optimized Python program for implementation of Bubble Sort


def bubbleSort(arr):
    n = len(arr)

    # Traverse through all array elements
    for i in range(n):
        swapped = False

        # Last i elements are already in place
        for j in range(0, n-i-1):

            # Traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True
        if (swapped == False):
            break


# Driver code to test above
if __name__ == "__main__":
    arr = [64, 34, 25, 12, 22, 11, 90]

    bubbleSort(arr)

    print("Sorted array:")
    for i in range(len(arr)):
        print("%d" % arr[i], end=" ")
```

Advantages:
1- Bubble sort is easy to understand and implement.
2- It does not require any additional memory space.
3- It is a stable sorting algorithm, meaning that elements with the same key value maintain their relative or der in the sorted output.

Disadvantages:
1- Bubble sort has a time complexity of O(N2) which makes it very slow for large data sets.
2- Bubble sort is a comparison-based sorting algorithm, which means that it requires a comparison operat or to determine the relative order of elements in the input data set. It can limit the efficiency of the algorith m in certain cases.