Assignment - 08

Pseudocode and Flowchart for Sorting Algorithm

Write pseudocode and create a flowchart for a bubble sort algorithm. Provide a brief explanation of how the algorithm works and a simple array of integers to demonstrate a dry run of your algorithm.

Bubble Sort Algorithm: -

Bubble sort is the simplest sorting algorithm and is useful for small amounts of data, Bubble sort implementation is based on swapping the adjacent elements repeatedly if they are not sorted. Bubble sort's time complexity in both of the cases (average and worst-case) is quite high. For large amounts of data, the use of Bubble sort is not recommended.

The basic logic behind this algorithm is that the computer selects the first element and performs swapping by the adjacent element if required based on the kind of sorting i.e. ascending and descending till it reaches the last element this is known as a pass. The computer performs multiple such passes till all the elements are sorted or no more swapping is possible.

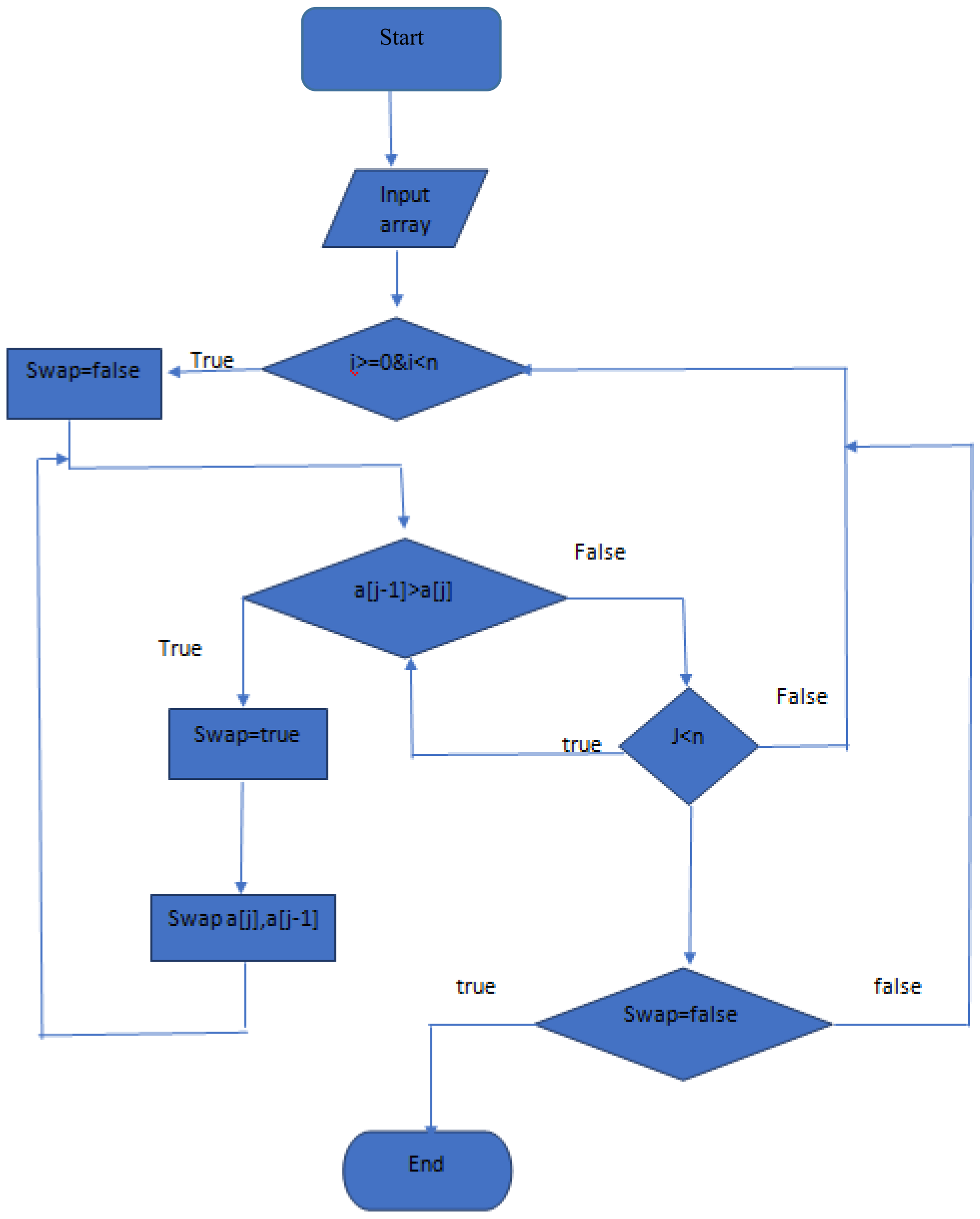
Bubble Sort Algorithm Pseudo Code:

Consider the below-given pseudo code for implementing a bubble sort:

```
Bubble Sort(a[],n)
For i=0 to n-1
Swap=false
For j=i+1 to n
   if a[j-1] >a[j]
      Swap(a[j-1],a[j])
      Swap=true
   Break if not swapped
```

Bubble Sort Algorithm Flow chart:

To help you understand better you can look at the flowchart for the bubble sort given below:



Dry Run Demonstration: -

Consider a simple array of integers: [5, 3, 8, 4, 2]

First Pass:

Compare 5 and 3, swap since 5 > 3: [3, 5, 8, 4, 2]

Compare 5 and 8, no swap since 5 < 8: [3, 5, 8, 4, 2]

Compare 8 and 4, swap since 8 > 4: [3, 5, 4, 8, 2]

Compare 8 and 2, swap since 8 > 2: [3, 5, 4, 2, 8]

Second Pass:

Compare 3 and 5, no swap since 3 < 5: [3, 5, 4, 2, 8]

Compare 5 and 4, swap since 5 > 4: [3, 4, 5, 2, 8]

Compare 5 and 2, swap since 5 > 2: [3, 4, 2, 5, 8]

Compare 5 and 8, no swap since 5 < 8: [3, 4, 2, 5, 8]

Third Pass:

Compare 3 and 4, no swap since 3 < 4: [3, 4, 2, 5, 8]

Compare 4 and 2, swap since 4 > 2: [3, 2, 4, 5, 8]

Compare 4 and 5, no swap since 4 < 5: [3, 2, 4, 5, 8]

Fourth Pass:

Compare 3 and 2, swap since 3 > 2: [2, 3, 4, 5, 8]

At this point, the list is sorted, but the algorithm does not know it yet and will go through the entire list one more time without any swap, confirming that the list is sorted.