**Assignment 1: 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**.

Algorithm for bubble sort:

1. Start

2. Declare an array 'a' of size n

3. Assign values to the array 'a'

4. Set i = 1

5. If i < n, go to step 6; otherwise, go to step 13

6. Set j = 0

7. If j < n - i, go to step 8; otherwise, go to step 5

8. If a[j] > a[j+1], then do

Swap a[j] and a[j+1]

9. Increment j by 1.

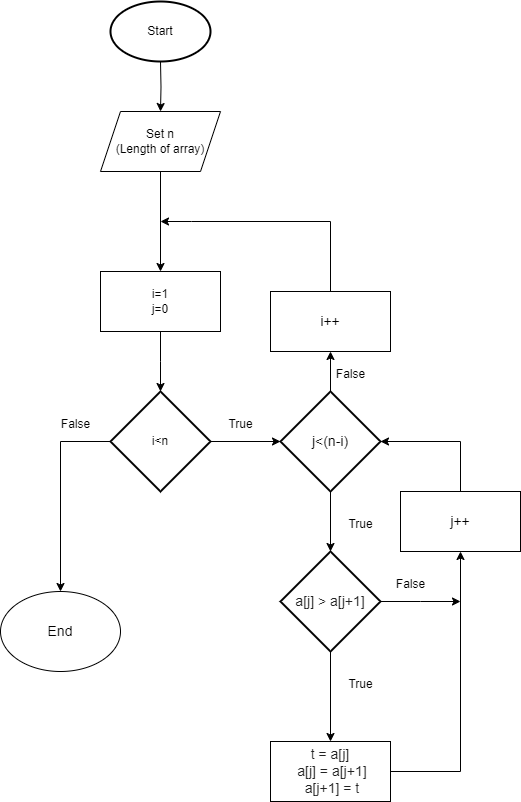
10. Go to step 7

11. Increment i by 1

12. Go to step 5

13. End

Flowchart:



Psedocode:

1. Begin

2. Declare an array 'a' of size n

3. Assign values to the array 'a'

4. Set i = 1.

5. While i > n

Set j = 0.

While j >(n - i)

If a[j] > a[j+1]

Swap a[j] and a[j+1].

Increment j by 1.

Increment i by 1.

6. Display the elements of the array 'a' after sorting

**Brief explanation of how the algorithm:**

Bubble sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.

Here's a dry run of the algorithm using the array [7, 5, 4, 2, 6]:

1. Start with `i = 1`. Since `i < n`, proceed to step 4.

2. Set `j = 0`.

3. Since `j < n - i`, proceed to step 8.

4. Check if `a[j] > a[j+1]` for the current values of `j`. If `a[j] > a[j+1]` (7 > 5), swap them.

5. Increment `j` by 1.

6. Repeat steps 3-5 until `j` reaches `n - i`.

7. Increment `i` by 1 and go back to step 3.

8. Repeat steps 2-7 until `i` reaches `n`.

At each iteration of the outer loop (`i`), the largest unsorted element bubbles up to its correct position at the end of the array. After each pass, the largest element in the unsorted portion of the array is moved to its correct position, so the sorted portion of the array grows larger until the entire array is sorted.

Using the array [7, 5, 4, 2, 6], the steps would be as follows:

1. Start with [7, 5, 4, 2, 6]. After the first pass, the largest element '7' is moved to the end.

2. After the second pass, '6' is moved to the second last position.

3. After the third pass, '5' is moved to the third last position.

4. After the fourth pass, '4' is moved to the fourth last position.

5. After the fifth pass, '2' remains at the start as it is the smallest element.

The sorted array is then [2, 4, 5, 6, 7].