**Assignment 1**: Pseudo code and Flowchart for Sorting Algorithm – Write Pseudo code 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.

**Pseudo code:**

Start

Set i=1, j=0

Read Array a

for i in range from 1 to n-1:

for j in range from 0 to n-i:

if a[j]>a[j+1] then

swap a[j] and a[j+1]

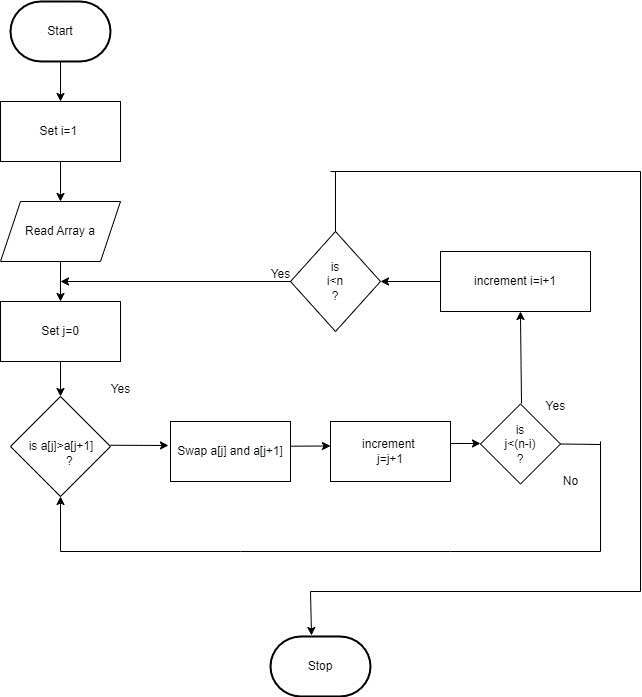
end if

end for

end for

end

**Flowchart:**



**Dry Run:** Consider an array: [15, 3, 18, 4, 2]

First iteration:

Comparing 5 and 3, swap: [3, 15, 18, 4, 2]

Comparing 5 and 8, no swap

Comparing 8 and 4, swap: [3, 15, 4, 18, 2]

Comparing 8 and 2, swap: [3, 15, 4, 2, 18]

Second iteration:

Comparing 3 and 5, no swap

Comparing 5 and 4, swap: [3, 4, 15, 2, 18]

Comparing 5 and 2, swap: [3, 4, 2, 5, 18]

Third iteration:

Comparing 3 and 4, no swap

Comparing 4 and 2, swap: [3, 2, 4, 15, 18]

Fourth iteration:

Comparing 3 and 2, swap: [2, 3, 4, 15, 18]

Final sorted array: [2, 3, 4, 15, 18]

**Assignment 2**: Recursive Function and Efficiency Analysis - Write a recursive function pseudo code and calculate the nth Fibonacci number and use Big O notation to analyze its efficiency. Compare this with an iterative approach and discuss the pros and cons in terms of space and time complexity.

**Pseudo code**:

start

read n

function Fibonacci(n)

If n<=1 then

return n

else

return Fibonacci(n-1)+Fibonacci(n-2)

end if

end function

Efficiency analysis:

1. recursive approach:

Time complexity: O(2^n)

Space complexity: O(n)

1. iterative approach:

Time complexity: O(n)

Space complexity: O(1)