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.

Void bubblesort( int arr[], int n)

{

Int I, j, temp, flag=0;

For (i=0; i<n; i++)

{

For (j=0 ; j<n-i-I; j++)

{

If(arr[j]>arr[j+1])

{

Swap=arr[j]

temp=arr[j]

arr[j]=arr[j+1]

arr[j+1]=temp

flag=1;

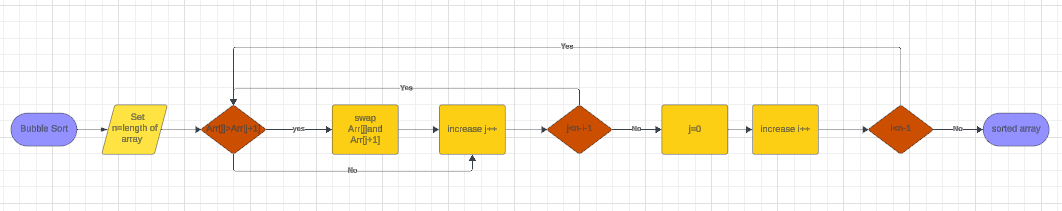
}

End loop

End loop

End fuction

Flowchart:



Explanation:

Bubble Sort works repeatedly compares each pair of adjacent element and swapping them if they are in wrong order. First we compare first element with the second element if first element is greater than second element then we swap the both the element and this process goes so on for each element in the array.

Dry Run:

Array= [10, 9, 11, 8]

Flag=0, i=0,j=0

If 10>9 then swap the number

Array= [9,10,11,8]

J=1

If 10>11 not true then there will not be swapping

J=2

If 11>8 true then swap the number

Arry=[9,10,8,11]

Flag=1 , i=1,j=0

9>10 false no swapping

J=1

10>8 true swap the number

Array=[9,8,10,11]

J=2

10>11 false no swapping

Flag=2 , i=2 , j=0

9>8 true swap the number

Array=[8,9,10,11]

Array is sorted

Assignment 2: Recursive Function and Efficiency Analysis - Write a recursive function pseudocode 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.

function fibonacci(n)

if n <= 1

return n

else

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

Start

input num

result = call fibonacci(num)

display result

end

Efficiency using Big O notation:

The time complexity of this recursive function is O(2^n). This is because each call to the fibonacci function branches into two more calls, resulting in an exponential growth in the number of function calls as n increases. The space complexity is also O(n) due to the recursive calls consuming stack It's easy to understand and implement.

Pros & cons:

It's easy to understand and implement.

It has exponential time complexity so it makes it inefficient for large values of n. It also consumes a lot of space due to recursive calls.