Day 2 Assignment

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.

// Pseudocode for bubble Sort:

Step 1: start

Step2: Taking the first Element of the array and compare with next element until the max element reaches the last position of the array. After the first pass max element reaches last position of the array.

Step 3: Step 2 will keep on executing until it reaches the iteration reaches the length of the array on every iteration the max element reaches it position

Step 4: At end of execution of step 3 and step 4 will get sorted Array

Step 5: End of the sorting.

//Algorithm for Bubble Sort:

Step1: start

Step 2: get input from the user that is Array[]

Step 3: Declaration of variables i,j

Step4: Using for loop setting i=0 and max value is i<=array.length and increamenting by 1

Step5: Declaring the inner for loop setting j=0 and max value will be n-i-1

Step6: Having if condition and checking the array[j] is > array[j+1] if it is greater then swap the position of the both element.

Step 7:We will get the sorted array and using printf statement we get the output.

// Dry run of algorithm

Initial array:[5,2,9,3,7]

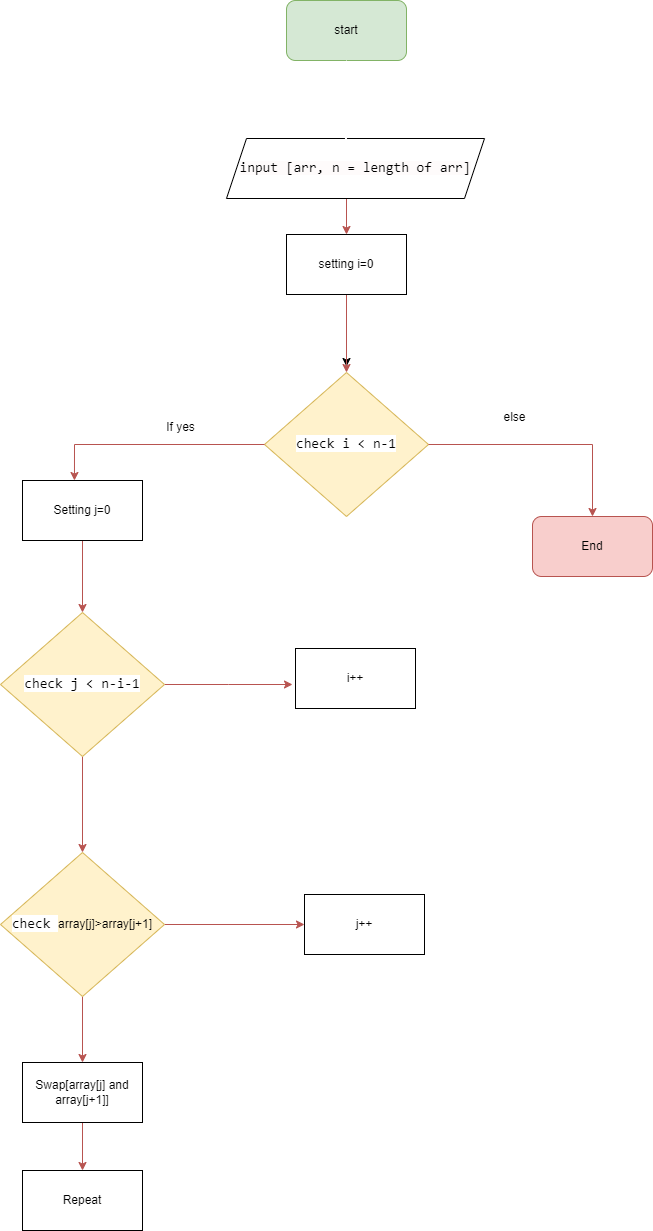
First iteration:[2,5,3,7,9]

Second iteration:[2,3,5,7,9]

Third iteration:[2,3,5,7,9]

The array is now sorted.

// Flowchart Bubble Sort



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.

//pseudocode for Fibonacci number

Step1: start

Step2: get number from user

Step3:function fibonacci(number) if number is zero then return zero

Step4:if number =1 then return 1

Step5:else return Fibonacci(number-1) +Fibonacci (number-2)

Step6: End.

// algorithm for Fibonacci number

Step1: start

Step2: function fibonacci(number) Get input from user number

Step3: if n==0 , return 0

Step4:if n == 1, return 1

Step5: set a=0,b=1

Step6: else return fibonacci(number-1) + fibonacci(number-2)

Step7: end

Efficiency Analysis using Big O Notation:

The time complexity of the recursive Fibonacci algorithm O(2^n).

Each recursive call, the function makes two additional recursive calls (n - 1 and n - 2),

As n increases the number of function calls grows exponentially making the algorithm

inefficient for large values of n.

Comparison with Iterative Approach:

In iterative approach to calculating Fibonacci numbers has a time complexity of O(n), making it much more efficient than the recursive approach.

Recursive Approach:

Pros: Concise easy to understand, and elegant.

Cons: Inefficient for large values of n due to exponential time complexity and potential stack overflow issues.

Iterative Approach:

Pros: Efficient time complexity of O(n), suitable for large values of n.

Cons: May be more complex and less intuitive than the recursive approach.

Flowchart:

