DAA

Assignment - 2

Shashank Bagda
92100133020

1) What is an Algorithm?

→ An algorithm is a step-by-step procedure or a set of rules for solving a specific problem or accomplishing a particular task. It is well-defined sequence of instructions that takes some input, performs a series of operations and produces the desired output. It can be implemented in various programming languages and used to solve range of problems.

2) Explain the need for an algorithm?

→ a) Efficiency - They are designed to optimize the use of computational resources such as time & memory.

b) Reproducibility - They can be followed by anyone to achieve the same result for a given a input.

c) Scalability - As problems grow in complexity and size, algorithms provide a way to handle larger inputs and still produce results in a reasonable amount of time.

d) Accuracy - Algorithms can be rigorously analyzed and tested for correctness ensuring that they produce the correct output for all valid inputs.

3) 1 : Initialize sum = 0
2 : For i in range 0 to 9, else go to 5
3 : sum = sum + age[i]
4 : End loop
5 : average = sum / 10
6 : return average
No of primitive operation = 11
Here overall Asymptotic Complexity is O(1)

4) 1 : total amount = 0
2 : currency = 0
3 : total amount = total amount + (2.0 * 50)
4 : total amount = total amount + (1.5 * 35)
5 : total amount = total amount + (2.5 * 10)
6 : total amount = total amount + (1.0 * 15)
7 : amount returned = currency - total amount
8 : total items = 2.0 + 1.5 + 2.5 + 1.0
9 : return amount returned & total items
No of primitive operation = 14
Here overall Asymptotic complexity is O(1)

5) 1 : factorial = 1
2 : i from 1 to N, else 5
3 : factorial = factorial * i
4 : End
5 : return factorial

No of primitive operation = 3N + 2
overall Asymptotic Complexity = O(N)

6) 1: sum = 0
2: i in range 1 to 100, else 5
3: sum = sum + i
4: End
5: return sum

No of primitive operation = 202
Overall Asymptotic complexity = O(1)

7) 1: largest = number
2: For i in range 2 to N
3: current > largest, set largest = current
4: End
5: return largest

No of primitive operation = 4N + 2
Asymptotic complexity = O(N)

8) 1: N <= 1 return false
2: For i in range 2 to $\sqrt{N}$
3: N is div by i, return false
4: End
5: return true

primitive operation = $4\sqrt{N} + 3$
Asymptotic complexity = $O(\sqrt{N})$

9)

```
fib[0] = 1 , fib [1] = 1
For i in range 2 to 49
    fib[i] = fib[i-1] + fib[i-2]
End
return fib
```

primitive operation = 98
Asymptotic complexity = O(N)

10)

```
binary = " "
while n > 0 , else : 8
    remainder = n % 2
    binary = remainder + binary
    n = n / 2
end
return binary
```

primitive operation = $\log_2(n) + 5$
Asymptotic complexity = $O(\log N)$

11) The algorithm of the given question is as follows

1. The algorithm starts with the first two element of the Fibonacci sequence [1, 1]

2. It then iteratively calculates the next Fibonacci number by adding the last two numbers in the sequence & append it to the sequence

classmate

Date
Page

3. The algorithm performs 3 primitive operations in each iteration of the loop.
4. The loop runs from 'n-2' iterations to generate the first 50 Fibonacci numbers.
5. The overall asymptotic complexity of this algorithm is O(n) because the time complexity grows linearly with the input size.

12) 1. The algorithm uses a while loop to repeatedly divide the decimal number n by 2 and concatenate the remainder to the left side of the binary representation

2. The loop runs until n becomes 0

3. In each iteration, the algorithm performs 3 primitive operations

4. The overall asymptotic complexity of this algorithm is O(log n) because the number of iterations required to convert the decimal number to binary is logarithmic in the size of the input number 'n'.