Summative Assessment

Prepared by Zuberi Ayyub Msemo

African Leadership University

Facilitator: Derrick Odonkor

Course: Advanced Algorithms

Faculty: Computer Science

Date: Dec 12, 2021

Question 1 :

a) Pseudocode;

Approach 1: using for loop

```
DECLARE:  an input function to get user input(n) to calculate the sum      O(1)
ASSIGN: int variable to function declared above                              O(1)
LOOP: run a loop till the entered number using the range and for loop        O(n)
        CALCULATE: the sum of the curr no by using sum = sum + curr          O(1)
        UPDATE: the sum of curr in each loop                                 O(1)
  RETURN: the sum of the last curr number as the sum of n number             O(1)
```

Time Complexity :

From the algorithm above, the idea is to iterate through a given range of numbers till the assigned input is reached ***O(n)*** , While iterating the sum is calculated, updated at each stage,

Mathematically,

$$T(n) = O(1) + O(1) + O(n) + O(1) + O(1) + O(1)$$

$$T(n) = 5O(1) + O(n)$$

$$\mathbf{T(n) = O(n)}$$

Approach 2: using recursive function

```
DEFINE: function to find the sum of n numbers                  O(1)
IF : the input is zero                                         O(1)
        RETURN : the input                                     O(1)
ELSE :  call the find  num + sum function with num -1          O(1)
ASSIGN:  user input to num variable                            O(1)
RETURN : the find sum result                                   O(1)
```

Time Complexity:

In this scenario, each time they find sum function calls and find the sum, it runs recursively and decreases the num by 1 at each iteration, in doing so, it runs in O(1) time at each point.

$$\mathbf{T(n) = O(1)}$$

b) **Algorithm behaviour:**

As the number of inputs increases the algorithm's response time due to the fact that the larger the number, the larger number of iterations the algorithm will execute, in each iteration the sum is calculated and stored. Throughout this process, the algorithms will behave slowly to larger numbers compared to small numbers.

In perspective, When the algorithms were tested with 10, 10000, 1000000, 1000000000 as inputs, it was relatively easy and fast to calculate the sum of 10 , 10000, 100000 compared to 1000000000 as a considerable amount of iterations were required to accomplish the task.

c) Source Code:
https://github.com/Zubrah/AA-Summativr/blob/main/Question01/Approach01.py

Question 2: ALU Grading Policy

things to notice:

- The function developed takes inputs and strips them and compares them with the grading policy assigned and when satisfied with a certain category it prints the output in the line below the input.
- The first number prompted is not associated with the result but just to initialize the function to run and ask the user to input a certain grade to start.
- When the grade is inserted the result will be displayed as the expected grade as per grading policy.

Source Code:
https://github.com/Zubrah/AA-Summativr/blob/main/Question02/Assignment02.py

Question 3 :

a) **Pseudocode**

```
ENCRYPTION FUNCTION
DEFINE: key of the encryption
TAKE: inputs from the user and change to upper case
ASSIGN:  each letter in alphabet a number between 0 and 25 a=0,b=1,c=2......, z=25
CHANGE:  message into 2 x 1 letter vectors, Change each vector into 2 x 1 numeric vectors
MULTIPLY:  each numeric vector by encryption matrix
CONVERT:  product vectors to letters
RETURN: the result of the encrypted message
```

```
DECRYPTION FUNCTION
PASS:  encrypted message and key as arguments
CONVERT: encrypted message to upper case
CALCULATE:  the determinant of given encryption matrix A, det A
            Make sure that det A has a modular inverse for Mod 26
            the adjoint of A, adj A
FIND: the corresponding ciphertext letter in the alphabet
ASSIGN: the corresponding value to result in a list
RETURN: the result as a decrypted message
```

b) Factors for Simplicity and Clarity

From the algorithm implemented, the use of matrices to generate the key and assign values in columns and rows make it simple to generate a random key from a pair of 26 letters of alphabets with 2 or 3 jumps key each. Converting to the upper case ensures that the original message is stored and can be encrypted and decrypted with the same key, a small change in the key input will result in an unexpected message/result.

c) Running Time of the Algorithm

Comparing the two algorithms with respect to the two keys(2 and 3) we found that using the 3 as the key becomes relatively slow compared to the use of 2 since the inverse of the algorithm and changes the matrices/jumps with respect to the key provided. The higher the key the slower the algorithm it would be.

d) Source Code:
https://github.com/Zubrah/AA-Summativr/blob/main/Question03/Approach01.py

Question 4 :
Source Code:
https://github.com/Zubrah/AA-Summativr/blob/main/Question04/Assignment04.py

Question 5:
Source Code:
https://github.com/Zubrah/AA-Summativr/blob/main/Question05/Assignment05.py