COMP6481 – Winter 2024

Programming and Problem Solving

Assignment # 1

Part 1

**Question 1:**

**[A]:**

Algorithm as a pseudo code for polyalphabetic cipher Encryption:

//initialize the function for encryption and it take plaintext as a string and key as an int value.

function Polyalphabeticcipherencryption (plaintext, key)

//Initialize the string to store the ciphertext

Ciphertext = ‘‘

// start the for loop to iterate over the plaintext

for each character in plaintext:

numeric value = numericvalue(character) // convert the char into numeric value

// logic to convert plaintext to ciphertext

shift = (numeric value + key +( current position mod 3 )) mod 26

// Now convert all numeric value to letter again

Encryted\_text = shifttoletter(shift)

ciphertext = ciphertext + encrypted\_text

// returning cipher text

return ciphertext

**Input:**

Plaintext as a string

Key as an int value

**Output**:

Ciphertext as a string

[B]:

Algorithm as a pseudo code for polyalphabetic cipher decryption:

//initialize the function for decryption and it take ciphertext as a string and key as an int value.

function Polyalphabeticcipherdecryption (ciphertext, key)

//Initialize the string to store the plaintext

plaintext = ‘‘

// start the for loop to iterate over the ciphertext

for each character in ciphertext:

numeric value = numericvalue(character) // convert the char into numeric value

// logic to convert ciphertext to plaintext

shift = (numeric value - key - ( current position mod 3 )) mod 26

// Now convert all numeric value to letter again

decrypted text = shifttoletter(shift)

Plaintext = plaintext + decrypted text

// returning plain text

return plaintext

**Input:**

ciphertext as a string

Key as an int value

**Output**:

Plaintext as a string

**[C]:**

Time Complexity of encrypted and decrypted algorithm are O(n) because we iterate the loop for read cipher text and plain text char and the other operation like shift and convert to numeric value has constant time.

[D]: Space Complexity of encrypted and decrypted algorithm are O(n) because we use string builder or scanner class to store string as a cipher text or plain text.

Question 2:

Algorithm for greater and less element

//function to sum take input Array A and X value

function Sumbiggerandsmaller(A, x)

// initializing int value for both sum

sumbigger = 0;

sumsmaller = 0;

// Checking the num in array

for each num in A:

// if number is greater than x then number is added to the sumbigger

if num>x:

sumbigger = sumbigger + num

// if number is less than x then number is added to the sumSmaller

else

sumsmaller = sumsmaller + num

return sumbigger, sumsmaller

[A] Time Complexity of this algorithm is O(n) because we are going to iterate over the all array element.

[B] Space complexity is O(1)

Question 3:

Algorithm of sumbigger and sumsamller for sorted array:

//function to sum take input Array A and X value

function Sumbiggerandsmaller(A, x)

// initializing int value for both sum

sumbigger = 0.

sumsmaller = 0;

//Intializing two pointer low and high

Low = 0

High = 0

// Traverse the array and check the condition

While low < high:

If A[low] + A[High] > x

//add A[high] to sumbigger and drecrement high

Else if A[low] + A[High] < x

//add A[high] to sumsmaller and increment low

Else a[low] + a[high] = x

// a[low] and a[high] both are at center

//Return the sum of both

Return sumbigger, sumsmaller

[A] the time Complexity in this case is also the O(n).

[B] the Space Complexity in this case is O(1).

Question 4:

Algorithm of Reversetwoconsicutive

// function take array as an argument

Function Reverse(A):

// determine the size of array

N = A.length()

// calculate the mid point of array

Mid = n/2;

// Intializing the leftend and rightend with checking condition

if mid is even :

leftend = Mid

rightend = Mid

else:

leftend = Mid -1

rightend = Mid -1

// Reverse the two consecutive element

For I = 0 to leftend:

//Swap the element

Swap A[i] to A[i+1]

For I = 0 to rightend to n-2:

//Sum of the element

A[i+1] = A[i] + A[i+1]

// if n is odd

If n is Odd:

// mid = mid -1

//Return a modifier array

Return A;

A : Time Complexity : O(n)

B: Space Complexity : O(1)