Data Structures and Algorithms

Comp 200

Fall 2022



Department of Computer Science

Forman Christian College University

Lab 4

Time Complexity and Recursion

|  |  |
| --- | --- |
| **DESCRIPTION** | **MARKS ALLOCATED** |
| Attendance | 25% |
| Task Completion | 35% |
| Viva | 35% |
| Submission | 15% |

### Marks will be deducted in case if students have not completed the assigned task.

**Note that these marks are max in each category. We may assign less than the given percentage of marks in case students have not successfully completed all the requirements.**

**This lab is time constrained. Please note that you must finish and submit your work within given time**.

**Question 1**: [Weightage: 15%]

This given python program rearranges positive number at even indexes and negative numbers at odd indexes in the same array. Find the total time complexity of the given program.

def rearrange(arr, n): 1

i = -1 1

for j in range(n): n

if (arr[j] < 0): 2n

i += 1 n

Arr[i], arr[j] = arr[j], arr[i] 6n

pos, neg = i+1, 0 2

while (pos < n and neg < pos and arr[neg] < 0): 4(n+1)

arr[neg], arr[pos] = arr[pos], arr[neg] 6(n)

pos += 1 n

neg += 2 n

def printArray(arr, n): 1

for i in range(n): n

print(arr[i], end=" ") 2n

# Driver code

If \_\_name\_\_ == “\_\_main\_\_”:

arr = [-1, 2, -3, 4, 5, 6, -7, 8, 9]

n = len(arr)

printArray(arr, n)

rearrange(arr, n)

printArray(arr, n)

5 + 21n + 4(n+1)

**=O(n)**

**Question 2**: [Weightage: 15%]

This function finds sum of elements whose sum is closest to zero. Find the time complexity.

def minAbsSumPair(array, array\_size): **1**

#At least two element are required in an array

if array\_size < 2: 1

print("Invalid Input") 1

return 1

min\_left = 0 1

min\_right = 1 1

min\_sum = array[0] + array[1] 3

for left in range (0, array\_size - 1): n

for right in range (left + 1, array\_size): n^2

sum = array[left] + array[right] 3(n^2)

if abs(min\_sum) > abs(sum): n^2

min\_sum = sum n^2

min\_left = left n^2

min\_right = right n^2

print("The two elements whose sum is minimum are",

arr[min\_left], "and ", arr[min\_right]) n

# Driver code

arr = [1, 60, -10, 70, -80, 85]

minAbsSumPair (arr, len(arr));

9 + 2n + 8n^2

O(n^2)

**Question 3**: [Weightage: 15%]

Find the time and space complexity of following recursive function.

def factorial(n): 1

# Base case

if n==0: 1

return 1 1

# Recursive case

else:

return n\*factorial(n-1) 2n

#Driver code

print(factorial(5))

time:

3 + 2n

O(n)

Space:

O(n)

**Question 4**: [Weightage: 15%]

Find the time and space complexity of following code.

def fibonacci(n): 1

if n == 0: 1

return 0 1

elif n == 1: 1

return 1 1

else:

return fibonacci(n - 1) + fibonacci(n - 2) 2^n +5

# Driver code

print(fibonacci(7))

time:

5 + (2^n+5)

O(2^n)

Space:

O(n)

**Question 5**: [Weightage: 15%]

Find the time and space complexity of following code.

def list\_sum\_recursive(input\_list): 1

# Base case

if input\_list == []: 1

return 0 1

# Recursive case

else:

head = input\_list[0] 2

smaller\_list = input\_list[1:] 2

return head + list\_sum\_recursive(smaller\_list) n+7

#Driver Code

print(list\_sum\_recursive([1, 2, 3]))

Time:

7 + n+7

O(n)

Space:

O(n)

**Question 6**: [Weightage: 15%]

Tower of Hanoi is a mathematical puzzle where we have three rods (A, B, and C) and N disks. Initially, all the disks are stacked in decreasing value of diameter i.e., the smallest disk is placed on the top and they are on rod A. The objective of the puzzle is to move the entire stack to another rod (here considered C), obeying the following simple rules:

* Only one disk can be moved at a time.
* Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
* No disk may be placed on top of a smaller disk.

Below is solution of the puzzle in Python. You are required to compute time and space complexity of Tower of Hanoi function.

def tower\_of\_hanoi(n, start, end, aux): 1

# Base case

if(n == 1): 1

print("Move disk 1 from", start, "to", end) 1

return 1

tower\_of\_hanoi(n - 1, start, aux, end) 2^n

print("Move disk", n, "from", start, "to", end) 1

tower\_of\_hanoi(n - 1, aux, end, start) 2^n

#Driver function

n = int(input("Enter the number of disks: "))

tower\_of\_hanoi(n, 'A', 'B', 'C')

time:

2(2^n) + 5

O(2^n)

Space:

O(n)