United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Midterm Exam Total Marks: **30** Fall 2021

Course Code: CSE 2217 Course Title: Algorithms

Time: 1 hour 45 minutes

Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules.

There are **FOUR** questions. **Answer all of them**. Show full simulation/tabulations wherever necessary. Figures in the right-hand margin indicate full marks.

1. **(a)** Derive the best-case and the worst-case running-time equations for the following function `has_common` and express those in Big-Oh (O) notation. Also provide the best-case and the worst-case examples of the arrays A and B with n =4 and m=5 for the function `has_common`. **[4]**

```
function has_common(A, B):
  1.    m = A.length
  2.    n = B.length
  3.    for i=1 to m do
  4.      for j=1 to n do
  5.        if A[i] == B[j] then
  6.          return true
  7.      end
  8.    end
  9.    return false
```

(b) Derive the exact-cost equation for the running-time of the following function `bubble_sort` and prove that it is in $O(n^2)$. **[2+2]**

```
function bubble_sort(A):
  1.    n = A.length
  2.    for i=1 to n-1 do
  3.      for j=1 to n-i do
  4.        if A[j] > A[j+1] then
  5.          swap A[j] and A[j+1]
  6.      end
  7.    end
  8.    for i=1 to n do
  9.      print(A[i])
 10.    end
```

2. **(a)** Determine a good asymptotic upper bound for the following recurrence equation: $T(n) = 2T(\frac{n}{2}) + O(1)$, where $T(1) = O(1)$. **[2]**

(b) Consider a modified version of the Merge sort algorithm as follows: **[2]**

If the array size is less than or equal to 2, then it sorts the array at constant time. Otherwise, it divides the array of size n into 3 subarrays, each with a size of $n/3$. This division takes $O(n^2)$ time. Then the algorithm sorts the subarrays recursively, and then merges their solutions at time $O(n)$. Write a recurrence relation for the running-time $T(n)$ of this algorithm.

3. **(a)** Given an array A = {-2, 8, -4, 2, -6, 4}, find the maximum-sum continuous sub-array using divide-and-conquer approach. You must show the recursion tree and clearly mention left, right and crossing sum for each tree node. **[3]**

(b) What is the activity selection problem? Is it true that the activity selection problem has one unique optimal solution? **[2]**

(c) "Data encoded using Huffman coding is uniquely decodable"- is the statement true or false? Justify your answer. **[1]**

(d) Information to be transmitted over the internet contains the following characters with their associated frequencies as shown in the following table:

Character	a	e	l	n	o	s	t
Frequency	45	65	13	45	18	22	53

Use Huffman technique to answer the following questions:

i. Build the Huffman code tree for the message and find the codeword for each character. **[3]**

ii. If the data consists of only these characters, what is the total number of bits to be transmitted? What is the percentage saving if the data is sent with 8-bit ASCII values without compression? **[1]**

4. **(a)** What is Optimal Substructure property? How Dynamic Programming differs from Divide-and-Conquer problems in terms of subproblems? **[1]**

(b) Suppose, on a cold winter morning, you went to Starbucks to buy a hot chocolate coffee. After paying the bill, the cashier needed to give you a total change of 7 cents. There is a huge supply of 1 cent, 2 cents, and 5 cents coins available in the cashbox. You don't want to carry many coins, so you asked her to return the change using a minimum number of coins. Determine how many coins she should return in this scenario by applying the Dynamic Programming Process? **[3]**

(c) Determine the maximum profit for the 0-1 Knapsack problem given in the following table using Dynamic Programming**.** **[4]**

Knapsack Weight: 8 kg

Objects	(1)	(2)	(3)	(4)
Weight (kg)	5	4	6	3
Profit ($)	11	10	12	9

Also show that **Greedy Strategy** for the 0-1 Knapsack problem fails to achieve the maximum profit for the same objects.