CMP 418: Algorithm and Complexity Analysis (3 Credit Units)
Problem Sets

Part A: Solving Recurrence

Problem 1-1

For each group of functions, arrange the functions in the group in increasing order of growth. That is, arrange them as a sequence f1, f2... such that f1 = O(f2), f2 = O(f3), f3 = O(f4),.... For each group, add a short explanation to explain your ordering to earn full marks.

Question: Group 0

$f_1 = n^2$, $f_2 = n$, $f_3 = n \log n$, $f_4 = 2^n$, $f_5 = (\log n^2)^2$

Question: Group 1

$f_1 = \log((\log n)^3)$ $f_2 = (\log n)^{3(\log\ 3n)}$ $f_3 = 3^{\log n}$ $f_4 = n^{3^{\log n}}$

$f_5 = \log 3n^{n^3}$ $f_6 = (\log\log n)^3$

Question: Group 3

$f_1 = 4^{3n}$, $f_2 = 2^{n^4}$, $f_3 = 2^{3^{n+1}}$, $f_4 = 2^{3^n}$, $f_4 = 2^{5n}$

Problem 1-2

Prove the following assertions by using the definitions of the notations involved, or disprove them by giving a specific counter example.

(a). If t (n) ∈ O(g(n)), then g(n) ∈ Ω(t (n)).

(b). Θ(αg(n)) = Θ(g(n)), where α >0.

(c). Θ(g(n)) = O(g(n)) ∩ Ω(g(n)).

(d). For any two nonnegative functions t (n) and g(n) defined on the set of nonnegative integers, either t (n) ∈ O(g(n)), or t (n) ∈ Ω(g(n)), or both

Problem 2-1. Back-solving recurrences

(a) If $T(n) = \Theta(n^2 \log n)$ is a solution to $T(n) = aT(n/2) + \Theta(n^2)$, where a is a positive integer, find all possible values of a.

(b) If T(n) = Θ(n2) is a solution to T(n) = aT(n/3) + Θ(n), where a is a positive integer, find all possible values of a.

(c) If T(n) = Θ(n2) is a solution to T(n) = 4T(n/b) + Θ(n2), where b is a positive real number, find all possible values of b.

(d) If $T(n) = \Theta(n^{6.006})$ is a solution to $T(n) = 5T(n/b) + \Theta(n^5)$, where b is a positive real number, find all possible values of b.

(e) If T(n) = Θ(n2) is a solution to T(n) = 6T(n/6)+f(n), find one possible function f.

Problem 2-2. Recurrence Tree

1. Derive the complexity class of the following recursive function using recursion tree
   a. $T(n) = T(n/2) + T(n/4) + n^2$
   b. $T(n) = 4T(n/2) + n/\lg n$

Part B: Sorting Problem

Problem 2-1

Consider the algorithm for the sorting problem that sorts an array by using the frequency of each of its elements, to put the element in its appropriate position in the sorted array:

ALGORITHM *ComparisonCountingSort(A*[0..*n* − 1]*)*
//Sorts an array by comparison counting
//Input: Array *A*[0..*n* − 1] of orderable values
//Output: Array *S*[0..*n* − 1] of *A*'s elements sorted in non-decreasing order
(1) for *i* ←0 to *n* − 1 do
(2) *Count*[*i*]←0
(3) for *i* ←0 to *n* − 2 do
(4) for *j* ←*i* + 1 to *n* − 1 do
(5) if (*A*[*i*]<*A*[*j*])
(6) *Count*[*j*]←*Count*[*j*]+ 1
(7) else *Count*[*i*]←*Count*[*i*]+ 1
(8) for *i* ←0 to *n* − 1 do
(9) *S*[*Count*[*i*]]←*A*[*i*]
(10) return *S*

a. Use this algorithm to sort the array A[50, 30, 80, 10, 40, 60, 20] by specifying the content of the arrays below:
b. What is the basic operation?
c. What is it computing?
d. Is the algorithm Stable?
e. Is it in place?
f. Derive the complexity class of the algorithm?

Problem 2-2

Use quicksort to sort the array given below. Show all the steps involve on each iterations.

E	X	A	M	P	L	E	S

Part C: Searching Problem

Problem 3-1

How many comparison (both successful and unsuccessful) will be made by the brute force algorithm in searching for each of the following patterns in the binary tree of one thousand zeros?

a. 0 0 0 0 1
b. 0 1 0 1 0

Problem 3-1

What do you understand by brute force analysis of algorithm? State its strengths and weakness

a. Determine the minimum number of character comparisons made by the brute-force algorithm in searching for the pattern BRANDING in the text THERE_IS_MORE_TO_LIFE_THAN_INCREASING_ITS_SPEED

Assume that the length of the text which is 47 characters long is known before the search starts.

Problem 3-2

Given the following list, 1, 8, 6, 5, 3, 7, 4, 2

i. Construct a heap for the list by successive key insertions
ii. Sort the list by using Heapsort

Problem 3-3

```
ALGORITHM Problem(A[0..n-1])
for i <- 1 to n-1 do
        j <- i-1
        while j ≥ 0 and A[j] ≤ A[j+1] do
                SWAP(A[j], A[j+1])
                j <- j-1
```

Trace the behavior of the algorithm on the following input array by completing the passes below. If you need additional passes draw then. 68, 55, 44, 79, 19, 9. Note that the routine SWAP(a, b) interchanges the values a and b.

	0	1	2	3	4	5
Initial Array	68	55	44	79	19	9
Pass 1:						
Pass 2:						
Pass 3:						
Pass 4:						
Pass 5:						
Pass 6:						

a. What is the basic operations?
b. Determine the complexity class of this algorithm.
c. Is the algorithm in place?
d. is the algorithm stable? If yes justify your answer otherwise which modification should be performed (without affecting the correctness) in the algorithm to make it stable?

Part D: General

Problem 4-1

The knapsack problem is stated as follow: Given n items of known weights $w_1, w_2, \ldots, w_n$ and values $v_1, v_2, \ldots, v_n$ and a knapsack of capacity W, find the most valuable subset of the items that fit into the knapsack.

The following table shows a small instance of the Knapsack problem having five items with their respective weights and values.

Item No.	Item Weight	Item Value
1	2	$15
2	3	$20
3	1	$10
4	2	$12

Assuming a knapsack of capacity of 5 units of weight, answer the following questions.
a. Find the most valuable subset of the items that fit into the knapsack using exhaustive search.
b. Find the most valuable subset of the items that fit into the knapsack using Dynamic Programming.

Problem 4-2

The Longest Common Subsequence (LCS) Problem is to find the longest subsequence common to all sequences in a set of sequences (often just two). Note that subsequence is different from a substring. For example, the LCS(KADUNA, KANO)= KAN, and is of length 3.

a. State the recurrence relation that define the optimal substructure of an LCS
b. State an algorithm that compute the length of the LCS based on the recurrence defined in (i).
c. Given two string KADUNA, KAKNO show how the length of their longest common subsequence can be obtained using dynamic programming.

Problem 4-3

Karu LGA has decided to recognize BHU main Library and you have been hired. Unfortunately the books were not well kept and your first assignment is to arrange the in sorted order according to their catalogue number. It was then noticed that some books have the same catalogue number which is not supposed to be the case since catalogue number provides a unique identity for each book. The management then decided that for each duplicate, the catalogue number should be maintained for one of the book and the remaining books with same catalogue number should be returned to the catalogue section of the library to be reassigned a new number. You therefore decide to combine the sorting and removing duplicate in order to make the work faster.

a. Propose an algorithm that will perform this task

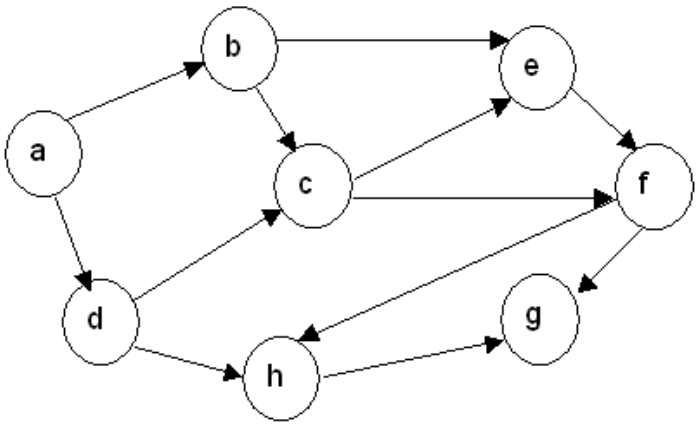
b. Is the proposed algorithm in place?

c. Evaluate the performance of the your algorithm

d. An alternative solution to the problem is by first sorting the books through an efficient sorting technique that has complexity of $\Theta(n\log n)$ the check only consecutive books: if two books or more have the same catalogue number, then they must be next to each other. Show that the complexity of this alternative solution is $\Theta(n\log n)$, where n is the total number of books in the library.

Problem 4-4

a. Use the following graph to answer the questions below.



Complete the following table by writing the order of arranging the vertices according to the traversal/algorithm indicated. You must arrange the vertices of the graph in sorted order (i.e., dictionary order) as much as possible.

b. Depth-first traversal (from vertex a)

c. Breadth-first traversal (from vertex a)

d. Topological ordering of vertices (from the appropriate vertex) using the source removal algorithm.