NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

SEMESTER 2 (2011/2012)

EXAMINATION FOR

CS3230: Design and Analysis of Algorithms

May 2012 **Time Allowed: 2 Hours**

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **EIGHT (8)** problems and comprises **SEVEN (7)** printed pages, including this page.
2. Answer **ALL** questions within the space in this booklet.
3. This is an **Open Book** examination.
4. All your asymptotic bounds must be in its simplest form. Namely, if the correct answer is $O(n^2)$ and you answered $O(n^2 + n)$, you will be penalized.

Write Your MATRICULATION NO:

□□□□□□□□□□

This portion is for examiner's use only

Question	Marks	Score
Problem 1,2,3	6+4+4=14	
Problem 4,5	4+6=10	
Problem 6	8	
Problem 7	8	
Problem 8	10	
Total	50	

--*Do not write above this line.*------------------------------------

Problem 1. (6 marks)
Solve the following recurrence relations and write out the **asymptotic tight bounds** for T(n). No need to write out the derivation process. In all cases below, T(x) denotes T(floor of x) if x is not an integer. In all cases below, the recurrence relation is for n > 0. For n = 0: T(0) = 1.

$T(n) = T(n-1) + 13$;   Your answer is T(n) = _____________

$T(n) = 13T(n/13) + n^{13}$;   Your answer is T(n) = _____________

$T(n) = T(\log_2(n)) + 13n$;   Your answer is T(n) = _____________

Problem 2. (4 marks)
a) Can you find two functions f(n) and g(n), such that i) $f(n) = O(n^5)$, ii) $g(n) = \Omega(n^3)$, and iii) $f(n)-g(n) = O(n^2)$? (Note that this means f(n), g(n), and f(n)-g(n) must be all non-negative.) Indicate "yes"/"no" first. If "yes", write your f(n) and g(n) below (your answer must satisfy all three properties to earn any marks). If "no", give a proof showing why this is not possible (no marks will be given without a proof).

b) Can you find two functions f(n) and g(n), such that i) $f(n) = O(n^3)$, ii) $g(n) = \Omega(n^5)$, and iii) $f(n)-g(n) = \Omega(n^2)$? (Note that this means f(n), g(n), and f(n)-g(n) must be all non-negative.) Indicate "yes"/"no" first. If "yes", write your f(n) and g(n) below (your answer must satisfy all three properties to earn any marks). If "no", give a proof showing why this is not possible (no marks will be given without a proof).

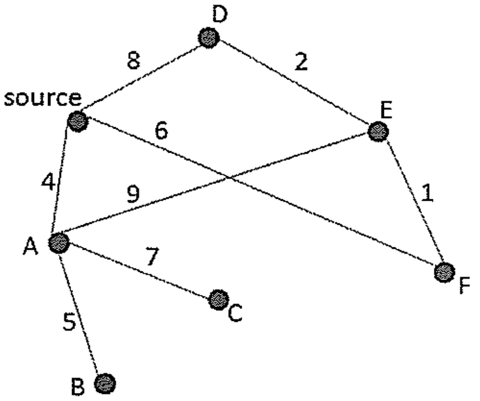
Problem 3. (4 marks)
When reasoning about NP-complete problems, we often use polynomial-time reductions. Explain clearly why it is important for the reduction to have at most polynomial time complexity.

-------------------------------------*Do not write above this line.*-------------------------------------

Problem 4. (4 marks)

Under what scenarios the worst-case time complexity of MergeSort is asymptotically lower than that of CountingSort? And under what scenarios the worst-case time complexity of CoutingSort is asymptotically lower than that of MergeSort?

Problem 5. (6 marks)



Imagine that you run Dijkstra's single-source shortest path algorithm on the above graph. For each iteration in this algorithm, a new node is added to the set S where S contains all those nodes whose shortest paths have finalized. The first node added to S is the source node, with 0 being the length of the shortest path from the source to itself. The second node added to S is node A, with 4 being the length of the shortest path from the source to A. Fill in below the 3rd, 4th, and 5th node added to S, together with the length of the shortest path from the source to that node.

2nd node added to S is __*A*__, and the length of the shortest path from source to that node is _4__

3rd node added to S is ____, and the length of the shortest path from source to that node is ____

4th node added to S is ____, and the length of the shortest path from source to that node is ____

5th node added to S is ____; and the length of the shortest path from source to that node is ____

--*Do not write above this line.*--------------------------------------

Problem 6. (8 marks)

Consider Huffman's algorithm for constructing optimal coding tree. That algorithm implicitly assumes that each code symbol is a binary bit, and it minimizes the total number of symbols used to represent the original data. There we only have 2 different code symbols (i.e., 0 and 1). Now imagine that we have 3 different code symbols: x, y, and z. For example, if the original data is ABCCD, one possible prefix code (which may or may not be optimal) would be A=x, B=y, C=zy, D=zx. Using this prefix code to encode the original data will give us xyzyzyzx, which uses total 8 symbols.

In such a new setting with 3 different code symbols, we still want to construct optimal prefix code so that the total number of symbols used to represent the original data is minimized. For this problem, we will assume that the total number of distinct characters in the original data is $1+2k$ for some integer k (i.e., you start with $1+2k$ frequency counts). Clearly explain how Huffman's algorithm can be modified to find the optimal coding tree in such a new setting. You also need to clearly explain how to modify the original proof for Huffman's algorithm so that it can now prove the optimality of your modified Huffman's algorithm.

Explain clearly how Huffman's algorithm should be modified:

Explain clearly how the original proof should be expanded/modified so that it can prove that your modified Huffman's algorithm produces optimal code tree:

---------------------------------------*Do not write above this line.*---------------------------------------

Problem 7. (8 marks)

Consider the coin change problem where we need to make change for a certain amount *A*, using coins with dominations d_1, d_2, ..., d_n. We require that the solution cannot use any denomination more than once. This means that it will not be possible to make change in some cases. For example, if we only have two denominations of 2 and 5, then it is impossible to make change for the amount of 4. For any given amount *A*, and any given denominations d_1, d_2, ..., d_n, our goal here is to determine whether it is possible to make change for the amount *A* using these denominations, under the restriction that each denomination can be used at most once. You are asked to design a dynamic programming algorithm to achieve this goal. Your algorithm should be deterministic and should have a worst-case time complexity of $O(nA)$. In your answer, you should first derive the recurrence relation used for dynamic programming, then provide concise pseudo-code, and finally prove the correctness and time complexity of your algorithm.

Write your recurrence relation below (remember to define the notations you introduce) and provide a concise explanation:

Write your pseudo-code below:

Write the proof on correctness and time complexity below:

-------------------------------------*Do not write above this line.*-------------------------------------

Problem 8. (10 marks)

Consider the following problem where the input is an arbitrary undirected graph with n vertices and m edges. Furthermore, the input has labeled $2k$ (distinct) vertices in the graph as s[1], s[2], ..., s[k], t[1], t[2], ...t[k]. (The input may choose these $2k$ vertices in an arbitrary way.) The remaining n-$2k$ vertices have no label. Here k can be any integer between 1 and $n/2$. The goal is to determine whether there exist k *vertex-disjoint* paths where the ith path goes from s[i] to t[i], and to output those k paths if they do exist. (For simplicity, you can assume that a path is not allowed to *visit* the same vertex more than once. A node is *visited* by a path if it is present on that path. The first and the last node on a path are also considered to have been visited by the path.) A set of k paths are *vertex-disjoint* if and only if no vertex in the graph is visited by more than one path in that set. Can you design a deterministic algorithm to solve this problem with $O(n \cdot m^2)$ worst-case time complexity?

If you can, indicate "YES" first, and then present algorithm. You should first give a high-level description of the ideas behind your algorithm, then provide concise pseudo-code, and finally prove the correctness and time complexity of your algorithm. If you feel that this cannot be done, indicate "NO" first, and then prove that the problem is NP-hard. Only indicating "YES"/"NO" will not earn you any marks. Partial answers may earn partial marks.

Indicate "YES"/"NO" first: ________________________________

Write your detailed answer below:

--*Do not write above this line.*--------------------------------------

(This page is intentionally left blank for you to write the answer for Problem 8.)

END OF EXAM PAPER