CS_332
DESIGNING ALGORITHMS
(Attempt 2 questions out of 3)

Question 1

(a) Define the O, Ω and Θ notations.

[4 marks]

(b) Prove that $(3n+1)^2 = O(n^2)$ using the definition of O from part (a).

[3 marks]

(c) Show that O is *transitive*; that is, show that if $f(n) = O\Big(g(n)\Big)$ and $g(n) = O\Big(h(n)\Big)$ then $f(n) = O\Big(h(n)\Big)$.

[3 marks]

(d) The runtime $T(n)$ of many divide-and-conquer algorithms may be described by a recurrence relation of the form

$$T(n) \;=\; aT(n/b) + \Theta(n^c),$$

for constants $a \geq 1$, $b > 1$ and $c \geq 0$, where n/b represents either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$.

State a Θ-expression for the function $T(n)$, distinguishing three cases according to the values of a, b and c.

[3 marks]

(e) Suppose you are given an ordered list A of n distinct numbers that has been *circularly shifted* k positions to the right.

For example, if the original ordered list is $\langle 5, 15, 27, 29, 35, 42 \rangle$ then the list $\langle 35, 42, 5, 15, 27, 29 \rangle$ has been circularly shifted k=2 positions, while the list $\langle 27, 29, 35, 42, 5, 15 \rangle$ has been circularly shifted by k=4 positions.

(i) Suppose k is known. Give a $O(1)$ algorithm to find the largest number in A.

[2 marks]

(ii) Suppose k is not known. Give a $O(\lg n)$ algorithm to find the largest number in A. Explain your algorithm, describe it in pseudocode, and justify its runtime.

For partial marks, you may give a $O(n)$ algorithm.

[10 marks]

Question 2

This question concerns the Making Change problem in which we wish to determine the least number of coins which make up an amount N when using coins of a given denomination $d_1, d_2, \ldots, d_n$. The greedy strategy, of repeatedly including in the solution the largest coin whose value doesn't exceed the remaining amount, fails in general, and we must rely on a dynamic programming solution.

(a) Suppose that the coins are worth 1, 2, 4, 8, 16 and 32; that is, each coin is worth ***exactly*** twice the next lower denomination coin. Show that the greedy strategy works in this case.

(Hint: would you ever use two coins of the same denomination?)

[4 marks]

(b) Suppose that each coin is worth ***at least*** twice the next lower denomination coin. Show that the greedy strategy doesn't necessary work in this case.

(Hint: consider coins of denomination 1, 5 and 11).

[4 marks]

(c) The English coinage before decimalization included half-crowns (30 pence), florins (24 pence), shillings (12 pence), sixpences (6 pence), threepences (3 pence), and pennies (1 pence). Show that with these coins, the greedy algorithm for making change does not necessarily produce an optimal solution.

[4 marks]

(d) Describe the dynamic programming algorithm for making change. In particular:

(i) Give the recursive equation for $c[i, j]$, the minimum number of coins needed to make up j pence using only the first i types of coins. Explain clearly every part of your equation.

[5 marks]

(ii) Give pseudocode for the dynamic program which computes the values $c[i, j]$.

[5 marks]

(iii) Analyse the running time and space requirement of this algorithm.

[3 marks]

Question 3

(a) Explain the phrase "information-theoretic lower bound" as it applies to comparison-based algorithms.

[5 mark]

(b) You are given two sorted lists of integers, each of length n, and you have to find the median of all of the $2n$ numbers. (The median is the n^{th}-smallest element.) For example, the median of the elements in $\langle 5, 9, 16, 21, 27\rangle$ and $\langle 3, 5, 8, 11, 14\rangle$ is 9.

(i) Argue that *any* comparison-based algorithm for this problem must perform at least $1 + \lceil \lg n \rceil$ comparisons.

(Hint: how many of the $2n$ values in the lists may be the median of the $2n$ elements?)

[5 marks]

(ii) Describe an algorithm which makes $1 + \lceil \lg n \rceil$ comparisons in the worst case.

(Hint: by comparing the medians of the two lists, you can either return an answer immediately, or recursively find the median of the numbers in two half-sized lists.)

[5 marks]

(c) You are given a black box with the following properties: It answers comparison questions put to it about the items in a list $\langle a_1, a_2, \ldots, a_n\rangle$. Regardless of the questions asked by the algorithm, after $\lceil \lg(n!) \rceil - 1$ questions, the black box produces two distinct orderings of the a_is which are consistent with all of its answers.

(i) Describe an algorithm which implements this black box function.

(Hint: think about how an adversary works.)

[5 marks]

(ii) An algorithm for purging (deleting) duplicates prints a list of the distinct items in its input list. Show that any comparison-based algorithm for purging duplicates makes $\Omega(n \lg n)$ comparisons in the worst case.

(Hint: Use an adversary strategy which uses the above black box.)

[5 marks]