# CS60003 Algorithm Design and Analysis, Autumn 2009–10

## Class Test 1

Maximum marks: 30 | September 08, 2009 | Total time: 1 hour

---

**Roll no:** ____________________ **Name:** ____________________________________________________________

[ *Write your answers in the question paper itself. Be brief and precise. Answer all questions.* ]

**1.** Suggest how you can force the quick sort algorithm to run in $O(n \log n)$ time in the worst case. **(5)**

**2.** An array $A = [a_1, a_2, \ldots, a_n]$ is called 2-sorted if the array $[a_1 + a_2, a_2 + a_3, a_3 + a_4, \ldots, a_{n-1} + a_n]$ is sorted.

**(a)** Give an example of an array of 10 integers, that is 2-sorted, but not sorted. **(5)**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

**(b)** Use a reduction argument to prove that any comparison-based algorithm for 2-sorting an array of $n$ elements must take $\Omega(n \log n)$ time in the worst case. **(5)**

**3.** A game is played between you and the computer. The game starts with a row of coins of values $c_1, c_2, \ldots, c_n$. All the values $c_i$ are known to you since the beginning of the game. The moves alternate between you and the computer. You make the first move. The player making a move is required to take a coin from one of the two ends. You are provided with an added option of skipping your move, but at most once in the entire game. (Your opponent does not have this option.) Your profit is the total value of all the coins you collect. The following steps lead to a polynomial-time dynamic-programming algorithm to compute your maximum guaranteed profit (that is, the maximum amount of money that you can definitely win).

**(a)** Suppose that at some point of time, the coins left are $c_i, c_{i+1}, \ldots, c_j$, and it is your turn to make a move. Let $P_1(i, j)$ denote your maximum guaranteed profit from this point (until the end of the game), given that you have already used the option of skipping a move. In this case, you have to pick either $c_i$ or $c_j$. On the other hand, suppose that you have not already exercised your option of skipping a move, that is, you may now pick either $c_i$ or $c_j$ or none of them. Let $P_2(i, j)$ be your maximum guaranteed profit from this point. Express these profits in terms of your profits for $(i, j-1)$, $(i, j-2)$, $(i+1, j)$, $(i+1, j-1)$ and $(i+2, j)$. Notice that your opponent does not cooperate with you in order to maximize your profit. In other words, although you make moves to maximize your profit, you do not have any control over the moves of your opponent. **(4)**

$P_1(i, j) =$

$P_2(i, j) =$

**(b)** Describe how you can initialize the profit values. **(4)**

For each $i = 1, 2, \ldots, n$, initialize:

$P_1(i, i) =$ $P_2(i, i) =$

Also for each $i = 1, 2, \ldots, n-1$, initialize:

$P_1(i, i+1) =$ $P_2(i, i+1) =$

**(c)** What is the final value you like to compute? **(2)**

**(d)** Describe an iterative algorithm to compute the final value from the initial values. **(3)**

**(e)** Analyze the running time of your algorithm. **(2)**

## ROUGH WORK