Algorithms

Duration: 120 Minutes
Maximum Marks: 50

1. Find $f(n)$ such that $T(n) = \Theta(f(n))$ (without using any master theorem)
$$T(n) \;=\; T(3n/5) + T(n/3) + n$$
(10 marks)

2. You are given a list $a_1 < a_2 < \ldots < a_N$. Assume that each $a_i \geq 1$. Your aim is to find the smallest number m such that m does not appear in the list $a_1, a_2, \ldots a_N$. Describe an efficient algorithm and its complexity. *(10 marks)*

3. In this task you are given a list of tasks $1, 2, \ldots N$. Each task comes with a length l_i, as well as a deadline d_i, for $1 \leq i \leq N$. The tasks can only be scheduled one at a time and once task i is scheduled it executes to completion (taking time l_i). If a schedule results in task i finishing at time t_i then its net lateness is $t_i - d_i$. The total lateness of a schedule is the sum of the net lateness of all the tasks. You have to determine the order in which the tasks have to be scheduled so that the total lateness is minimized and prove that your algorithm is correct. (Note that total lateness can be negative) *(10 marks)*

4. You are given a sequence of pairs of integers $(a_1, b_1), (a_2, b_2), \ldots (a_n, b_n)$. We would like a sequence $i_1 < i_2 < \ldots < i_k$ such that, for all $1 \leq j < k$, either $a_{i_j} \leq a_{i_{j+1}}$ or $b_{i_j} \leq b_{i_{j+1}}$. Our aim is compute the longest possible such sequence, however, only need to compute the length of the longest such sequence. Describe an algorithm to solve this problem and explain its complexity. *(10 marks)*

5. Let $G = (V, E)$ be a connected weighted graph. Suppose $v = v_1, v_2, \ldots, v_k = v$ is a cycle in this graph. Suppose $e = (v_i, v_{i+1})$ has weight strictly larger than the weight of all the other edges in this cycle. That is $wt(v_i, v_{i+1}) > wt(v_j, v_{j+1})$ for all $1 \leq j \leq k-1$, $i \neq j$. Prove that there is NO minimum cost spanning tree containing e. Does this suggest an algorithm to find MCSTs (at least in graphs where all edge weights are distinct)? *(10 marks)*

6. Let $a_1, a_2, \ldots, a_n$ be a sequence of integers. Recall the highest segment sum problem solved in the class. Here, we want to solve a variant of the problem that allows you to pick a segment and drop at most one element while computing the sum (why would you want to drop an element to get the highest sum?). Write an algorithm that computes the highest "special segment sum" where a special segment is either a segment of the numbers $a_i, a_{i+1}, \ldots, a_j$ with $i \leq j$ or such a sequence with one number from $a_i, a_{i+1}, ..., a_j$ omitted. *(10 marks)*