CSC373 **Term Test ; March 2** **Spring, 2020**

Name _____ Student No. _____

Indicate your tutorial time and room.

AIDS ALLOWED: One page (two sides) of handwritten notes
Answer ALL questions on test paper. Use backs of sheets for additional space. For every question, briefly justify your answer. You can use your knowledge from lectures, tutorials, the texts, or the assignments set as part of any answer. No cell phones or smart watches (or any device that can communicate) are allowed. It is an academic offense to have any such device visible.

REMINDER: You get 20% for any question or subquestion if you state "I do not know how to answer this question". You get 10% for any question which you just leave blank.

Important: Students taking the test from 4-6 must remain in the test room until 5:20. Students taking the test from 5-7 must arrive by 5:20 but can begin at 5:10.

1. (10 points)

   Suppose we are given an algorithm that can multiply two 4×4 matrices in ℓ non-scalar multiplications. What is the largest value of ℓ such that we can use the 4×4 matrix multiplication algorithm so as to design a divide and conquer algorithm that will be faster than Strassen's $\approx n^{2.81}$ time algorithm? Explain your answer.

   Solution sketch: $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ \ell T(n/4) + O(n^2), & \text{if } n > 1 \end{cases}$
   By Master's Theorem: $a = \ell, b = 4, d = 2$ so that $T(n) = O(n^{\log_4 7})$.
   We need $\log_4 \ell < \log_2 7 \implies 4^{\log_4 \ell} < 4^{\log_2 7} \implies \ell < (2^2)^{\log_2 7} = 2^{(\log_2 7)\cdot 2} = 7^2 = 49$.

Name _____ Student No. _____

2. (15 points) Suppose we have an array $A[1..n]$ of positive and negative integers. We want to design a divide and conquer algorithm to determine whether or not there is a sub-array $A[1 \leq i \leq j \leq n]$ such that $\sum_{i \leq k \leq j} A[k] \geq 2 \cdot \max_\ell A[\ell]$. If so, the algorithm should return an appropriate pair of indices i, j satisfying the stated condition; if not, then return "null" Your algorithm should run in time that is asymptotically less that n^2, the time for a naive brute force search algorithm.

   - (5 points)
     Explain your algorithm and the high level ideas underlying it.

     Solution sketch: Split array A to two roughly equal size arrays A_1, A_2 and recursively find maximum subarray in each, as well as the maximum subarray that crosses the two midpoints of A_1 and A_2. If any of the arrays satisfies the conditions return relevant indexes, otherwise return $NULL$.

   - (10 points) Analyze the complexity of your algorithm.

     Solution sketch:
     $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n/2) + O(n), & \text{if } n > 1 \end{cases}$
     Using Masters Theorem $T(n) \in O(n \log(n))$

Name _____ Student No. _____

3. (15 points) The following questions concern Huffman optimal prefix coding.

   - (5 points) Provide an optimal Huffman prefix tree for the following set of symbol occurrence probabilities: $Prob[a] = 1/4, Prob[b] = 1/4, Prob[c] = 1/8, Prob[d] = 1/8, Prob[e] = 1/8, Prob[f] = 1/8$

     Solution sketch:
     Sort (in non-decreasing order) symbols according to their probabilities and apply Huffman prefix-free algorithm as explained in the class.

   - (10 points) Suppose we have 4 symbols each of which occurs with probability strictly less than 1/3. Using a proof by contradiction, show that in an optimal prefix coding that no symbol can have a 1 bit code word (i.e., be at depth 1 in a Huffman optimal prefix tree.)

     Solution sketch:

     First we can show that the optimal tree will have depth 2. The smallest two symbols must have combined probability more than 1/3 (or else the sum of probabilities would be less than 1). If we have a symol at depth 1, then the remaining 3 symbols will require depth 2 so that the entire tree will have depth 3.

Name _____ Student No. _____

4. (15 points) Suppose the Canadian Armed Forces is conducting a training exercise for n soldiers in an elite unit. Each of soldiers must first climb up and down a 10 meter rope and then run a kilometer on a track with n lanes. There is only one rope so only one soldier can be on the rope at any time whereas it doesn't matter how many soldiers are on the track. Suppose we know the time $c_i > 0$ (resp. $r_i > 0$) for soldier i to climb the rope (resp, to complete a kilometer the run). The goal is to minimize the completion time when all soldiers have completed the exercise.

   - (5 points) Suppose we perform a greedy algorithm by sorting the order in which the soldiers will climb the rope so that $r_i + c_i \geq r_{i+1} + c_{i+1}$. Show that this algorithm will not always produce an optimal schedule.

     Solution sketch: If we use the suggested algorithm on a setup with $c_1 = 5, c_2 = 2, c_3 = 1, r_1 = 4, r_2 = 5$, and $r_3 = 5$, then the necessary time will be more than when we sort by non-increasing order of running times.

   - (10 points) Design a greedy scheduling algorithm so as to produce an optimal schedule for the order in which the soldiers will climb up and down the rope. Briefly sketch the outline of a proof showing the optimality of your algorithm.

     Solution sketch: Order non-increasingly by running time. Use exchange argument to prove optimality of this solution.

Name _____ Student No. _____

5. (15 points) Consider the following variant of the knapsack problem. We are given a knapsack size bound of B and a set $\{J_1, J_2, \ldots, J_n\}$ of n input items where $J_i = (s_i, v_i)$. Each item has a size $s_i \leq n^2$ and an arbitrary value v_i. You can assume all arithmetic operations and comparisons take time $O(1)$. The objective is to compute the value of an optimal solution OPT where OPT is a multi-set (i.e., elements c an appear more than once in a a multi-set) such that $\sum_{i:J_i \in OPT} n_i s_i \leq B$ and every item J_i occurs $n_i \in \{0, 1, 4\}$ times in OPT; that is, an item in OPT occurs once or four times (or is not used in OPT).

   Design an optimal dynamic programming algorithm for this knapsack variant whose time complexity is $O(p(n))$ for some polynomial $p(n)$ time bound. What is the time bound of your algorithm?

   Solution sketch: $M[i, b_i]$ = maximum value obtainable using $J_1, \ldots, J_i$ with size limit b.

$$M'[i,b] = \begin{cases} 0, & i = 0 \\ 0, & b < 0 \\ max\{M'[i-1,b], M'[i-1,b-s_i] + v_i, M'[i-1,b-4s_i] + 4v_i\}, & otherwise \end{cases}$$

   $T(n) \in O(nb)$ for $b = n \cdot n^2$ so $T(n) \in O(n^4)$.

Name _____ Student No. _____

6. (15 points) You are going on a long trip. You start on the road at mile post 0. Along the way there are n hotels, at mile posts $a_1 < a_2 < \ldots < a_n$, where each a_i is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which of the hotels you stop at. You must stop at the final hotel (at distance a_n), which is your destination.

   You would ideally like to travel 100 miles a day, but this may not be possible (depending on the spacing of the hotels). If you travel x miles during a day, the penalty for that day is $(100 - x)^2$. You want to plan your trip so as to minimize the total penalty; that is, the sum, over all travel days, of the daily penalties. Give an efficient dynamic programming algorithm that determines the optimal sequence of hotels at which to stop. What is the time complexity of your algorithm.

   Solution sketch: M[i] = the minimum penalty to arrive to hotel i.

$$M'[i] = \begin{cases} 0, & i = 0 \\ min\{M'[j], (100 - (a_i - a_j))^2 : j < i \wedge a_i - a_j \leq 100\}, & i > 0 \end{cases}$$

   M' has n entries and computing each entry is of $O(n)$. Thus, $T(n) \in O(n^2)$

Name _____ Student No. _____

7. (15 points)

There is a round robin tennis tournament amongst n players in which each player plays all the other players once. Thus there are $\binom{n}{2}$ games that need to be played. If a player wins a game she gets 1 point; a losing player gets 0. Suppose at the end of the tournament, player i obtains s_i points. We want to determine if the vector $(s_1, s_2, \ldots, s_n)$ can be the outcome of the tournament. For example when $n = 4$, (3,3,0,0) *cannot* be the outcome of the tournament while (3,2,1,0) can be an outcome. Note that $\sum_i s_i = \binom{n}{2}$.

For some polynomial time bound $p(n)$, design an algorithm that runs in time $O(p(n))$ that can determine if a given vector $(s_1, s_2, \ldots, s_n)$ can be an outcome. (You do not have to state the polynomial time bound but give some indication as to why your algorithm runs in polynomial time.)

Hint: Set up an appropriate flow network with integer capacities and use a maximum flow algorithm to check if the max flow is $\binom{n}{2}$.

Solution sketch: Construct a bipartite flow network $N = (V, E)$ with $V = V^M \cup V^P \cup \{s, t\}$ where V^M represents matches, V^P represents players, and s and t represent start and terminal nodes respectively. Notice that $|V^M| = \binom{n}{2}$ and $|V^P| = n$, the number of players. For players $i \neq j$ connect $V^M_{i,j}$ to V^P_i and to V^P_j each with capacity 1. Connect s to all nodes in V^M with capacity 1. Connect all nodes in V^P to t where edge (V^P_i, t) has capacity s_i. A given outcome is feasible if the max flow in N is $\binom{n}{2}$.