CS/ECE 374 A (Spring 2022)
Final Exam

Instructions:

- Date/Time: May 12 Thursday 8:00am–11:00am Central Time.
- Except for your two double-sided handwritten $8.5'' \times 11''$ cheat sheets, exams are **closed-everything**. In particular: No medically unnecessarily electronic devices are allowed in exams, including smart watches and headphones/earbuds.
- You will write your solutions on paper, scan your solutions using a scanning app, convert your scans to PDF, and upload the PDF to Gradescope. (You are **not** allowed to write your solutions on tablets, unlike in some past semesters.) Gradescope will only accept PDF submissions. Please make sure your solution to each numbered problem starts on a new page of your submitted PDF file. Please do not scan your cheat sheets or scratch paper.
- You have **two and a half hours (i.e., 150 minutes)** to write your solutions. We are providing 30 minutes at the end of each exam period for students to scan and upload their exams, and to deal with any unexpected technical difficulties that may arise. Gradescope will stop accepting submissions precisely at 11:00am. We will not accept submissions after the Gradescope deadline. **All work on the exam must stop 30 minutes before the Gradescope deadline, i.e., at 10:30am.**
- Please make sure that your scans are clear and easy to read. We very strongly recommend that you write with black ink on unlined white paper. Submissions consisting of raw photos or low-quality scans will be penalized.
- If you are ready to scan your solutions before 10:30am, send a private message to the host of your Zoom call ("Ready to scan") and wait for confirmation before leaving the Zoom call.
- If something goes seriously wrong during the exam, send email to Timothy (tmc@illinois.edu) as soon as possible explaining the situation. If you have already finished the exam but cannot submit to Gradescope for some reason, include a complete scan of your exam as a PDF file in your email. If you are in the middle of the exam, send email to Timothy, continue working until the time limit, and then send a second email with your completed exam as a PDF file. Please do not email raw photos.
- You are reminded about the course's, the department's, and the university's academic integrity policies.
- Avoid the deadly sins. Write complete solutions, not examples. Declare all your variables. Write concisely and precisely.
- Good luck!

---

1. [22 PTS] *True or False questions.*

   (a) [2 PTS] True or False: The language $\{x1x : x \in \{0,1\}^*\}$ is regular. Justification is not necessary.

   (b) [3 PTS] True or False: The language generated by the following context-free grammar is precisely $\{x1x : x \in \{0,1\}^*$ is a palindrome$\}$:

   $$\begin{aligned} S &\rightarrow A1A \\ A &\rightarrow 0A0 \mid 1A1 \mid \varepsilon \mid 0 \mid 1 \end{aligned}$$

   Briefly justify your answer.

   (c) [3 PTS] Suppose that Algorithm A has running time $\Theta(n^2)$, and Algorithm B has running time satisfying the recurrence $T_B(n) = 3T_B(n/2) + 374n$ with base case $T_B(1) = 2022$. True or False: Algorithm B is faster than Algorithm A for a sufficiently large n. Briefly justify your answer.

   (d) [3 PTS] True or False: If there is a polynomial-time reduction from Problem A to Problem B, and Problem B is NP-complete, then Problem A is NP-complete. Briefly justify your answer.

   (e) [3 PTS] Assume P $\neq$ NP. True or False: Given an undirected weighted graph G and a value ℓ, the problem of deciding whether G contains a tree that uses all the vertices and has total weight at most ℓ is NP-complete. Briefly justify your answer.

   (f) [3 PTS] True or False: For every language L, if L is undecidable, then the complement of L is decidable. Briefly justify your answer.

   (g) [2 PTS] True or False: The language

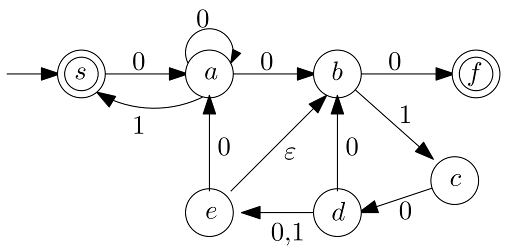
   $$\{\langle M\rangle : \text{the Turing machine } M \text{ accepts exactly two palindromes of each length}\}$$

   is decidable. (Here, $\langle M\rangle$ denotes an encoding of M.) Justification is not necessary.

   (h) [3 PTS] True or False: For every language L, if L is finite, then L is decidable. Briefly justify your answer.

2. [20 PTS] *Automata and graph algorithms.*

   (a) [5 PTS] (Easy) In the NFA M shown, give a string accepted by M that has strictly fewer 0's than 1's.



   (b) [10 PTS] More generally, suppose we are given an NFA $M = (\Sigma, Q, s, \delta, A)$, with alphabet $\Sigma = \{0, 1\}$, a set Q of n states, a start state $s \in Q$, a transition function $\delta : Q \times \{0, 1, \varepsilon\} \to 2^Q$, and accepting states $A \subseteq Q$. We want to decide whether there exists a string accepted by M that has strictly fewer 0's than 1's. Give an efficient algorithm to solve this problem, by constructing a graph and using the Bellman–Ford algorithm. Remember to define the vertices and edges and weights of your graph precisely. Analyze the running time as a function of n.

   Note: you may use the following known fact: in a weighted graph where negative edge weights are allowed, there is a version of the Bellman–Ford algorithm (with the same running time as the original) that computes the shortest-path distance $d[s, v]$ from a source vertex s to every vertex v, where the distance $d[s, v]$ is defined as $-\infty$ if there is a walk from s to v that contains a negative-weight cycle. Also, you may assume that Bellman–Ford can handle graphs that may have self-loops and parallel edges.

   (c) [5 PTS] Given an NFA M with n states, we want to decide whether there exists a string that is **not** accepted by M and has strictly fewer 0's than 1's. Give a (not necessarily efficient) algorithm to solve this problem. Analyze the running time as a function of n. Hint: Use part (b)[1] and known automata constructions.

3. [16 PTS] *Dynamic programming.* We have three types of toys: n_1 toys of value c_1, n_2 toys of value c_2, and n_3 toys of value c_3. We are also given a positive integer H. (You may assume that the given values c_1, c_2, c_3 are positive integers at most H.) We would like to put the toys in gift bags, maximizing the number of bags, such that each gift bag has value at least H.

   In order to develop a dynamic programming algorithm for this problem, we define the following subproblems: For $i \leq n_1$, $j \leq n_2$, $k \leq n_3$, and $h \leq H$, let $f(i, j, k, h)$ be the maximum number of bags that can be created from i toys of value c_1, j toys of value c_2, and k toys of value c_3, such that the first bag has value at least h and each of the other bags has value at least H. (In other words, think of the first bag as already pre-loaded with value $H - h$.) The answer we want is $f(n_1, n_2, n_3, H)$.

[1]In multi-part questions, even if you are unable to do part (b) correctly, you can still do part (c) by assuming the result from part (b).

(a) [10 PTS] Give a recursive formula for f. Remember to include appropriate base cases, and briefly justify your answer.

(b) [3 PTS] Specify a valid evaluation order for your formula. (Pseudocode is not necessary.)

(c) [3 PTS] Bound the running time needed to compute $f(n_1, n_2, n_3, H)$, as a function of n_1, n_2, n_3, H. Briefly justify your answer.

4. [21 PTS] *Greedy algorithms.* Consider the following problem: We are given a set S of n intervals $\{[a_1, b_1], \ldots, [a_n, b_n]\}$ and an additional interval $[A, B]$. (You may assume that all the a_i's and b_i's are distinct.) We want to choose a subset T^* of S with the smallest number of intervals such that the union of the intervals in T^* cover $[A, B]$.

Example: for $S = \{[1,8], [2,7], [6,20], [10,30], [18,36]\}$ and $[A,B] = [5,35]$, one optimal solution is $T^* = \{[2,7], [6,20], [18,36]\}$, which uses 3 intervals and has union $[2,36]$ which covers $[5,35]$. (This is not the only optimal solution; there may be other solutions using the same number of intervals.)

(a) [5 PTS] (Easy) Consider the following greedy strategy: select the interval which covers the longest length of the uncovered portion of $[A, B]$; then remove this interval, and repeat until all of $[A, B]$ is covered. Give a counterexample showing that this algorithm does not always give an optimal solution.

(b) [8 PTS] Consider an optimal solution T^*. Consider the interval $[a_i, b_i]$ from S with the largest b_i such that $a_i \leq A \leq b_i$. Prove that if T^* does not use the interval $[a_i, b_i]$, then we can modify T^* to get another optimal solution that uses $[a_i, b_i]$.

(c) [8 PTS] (Easy) Using part (b),[2] describe a correct greedy algorithm to solve the problem. Analyze the running time. (There is no need to prove correctness, since it should follow from (b).)

5. [21 PTS] *NP-completeness.* Consider the following problem AMERICAN-HAM-CYCLE:

Input: a directed graph G where each edge is colored red, white, or blue, and the number of vertices n is divisible by 3.

Output: yes iff there exists a cycle C such that C visits every vertex in G exactly once, and the colors of the edges along C alternate according to the pattern red-white-blue-red-white-blue- $\cdots$ (i.e., the i-th edge in C is red if $i \equiv 1 \bmod 3$, white if $i \equiv 2 \bmod 3$, and blue if $i \equiv 0 \bmod 3$).

(a) [5 PTS] (Easy) Prove that AMERICAN-HAM-CYCLE is in NP.

Note: all you need to do is to state what is the certificate, and what is the condition to verify, and the time needed for the verification procedure.

[2]See earlier footnote.

(b) [16 PTS] Prove that AMERICAN-HAM-CYCLE is NP-hard.

Note: you may use the fact that the standard HAMILTONIAN-CYCLE problem is NP-complete (where the input is a directed graph H, and the output is yes iff there exists a cycle that visits every vertex of H exactly once). Partial credit may be given for following the format and main steps of an NP-hardness proof (specifying the direction of reduction, indicating what is given and what you are constructing, and what it means for the reduction to be correct, etc.).