CS/ECE 374: Algorithms & Models of Computation, Spring 2019

Final **Monday, May 6, 13:30–16:30**

Name:	
NetID:	

⇐ Please PRINT

Which exam room to go to based on your discussion section.

Lincoln Theater		DCL 1320	MSEB 100
AYA 9am Yipu AYB 10am Xilin AYC 11am Xilin AYD noon Mitch AYE 1pm Ravi	AYJ 1pm Shant AYF 2pm Konstantinos AYG 3pm Robert BYE 3pm Jiaming	AYH 4pm Robert AYK 2pm Shant BYA 9am Zhongyi BYC 1pm Shu	BYB 10am Zhongyi BYF 4pm Jiaming BYD 2pm Shu

- ***Don't panic!***
- Please print your name, print your NetID, and circle your discussion section in the boxes above.
- There are seven questions – you should answer all of them.
- If you brought anything except your writing implements, your double-sided **handwritten** (in the original) 8½" × 11" cheat sheet, and your university ID, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
  - Submit your cheat sheet together with your exam. We will not return or scan the cheat sheets, so photocopy them before the exam if you want a copy.
  - If you are NOT using a cheat sheet, please indicate so in large friendly letters on this page.
- Please read all the questions before starting to answer them. Please ask for clarification if any question is unclear.
- **This exam lasts 175 minutes.** The clock started when you got the exam.
- If you run out of space for an answer, feel free to use the blank pages at the back of this booklet, but please tell us where to look.
- As usual, answering any (sub)problem with "I don't know" (and nothing else) is worth 25% partial credit. Correct, complete, but **slightly** sub-optimal solutions are *always* worth more than 25%. Solutions that are exponentially (or dramatically) slower than the expected solution would get no points at all. A blank answer is not the same as "I don't know".
- Total IDK points for the whole exam would not exceed 10.
- Give complete solutions, not examples. Declare all your variables. If you don't know the answer admit it and use IDK. Write short concise answers.
- **Style counts.** Please use the backs of the pages or the blank pages at the end for scratch work, so that your actual answers are clear.
- Please return ***all*** paper with your answer booklet: your question sheet, your cheat sheet, and all scratch paper.
- ***Good luck!***

NETID: NAME:

1 (10 PTS.) SHORT QUESTIONS AND HOPEFULLY SHORT ANSWERS.

No justification is required for your answers.

1.A. (5 PTS.) Give an asymptotically tight bound for the following recurrence.

$$T(n) = \sum_{i=1}^{10} T(n_i) + O(n) \quad \text{for } n > 200, \qquad \text{and} \qquad T(n) = 1 \quad \text{for } 1 \le n \le 200,$$

where $n_1 + n_2 + \cdots n_{10} = n$, and $n/20 \le n_i \le (9/10)n$ for all i.

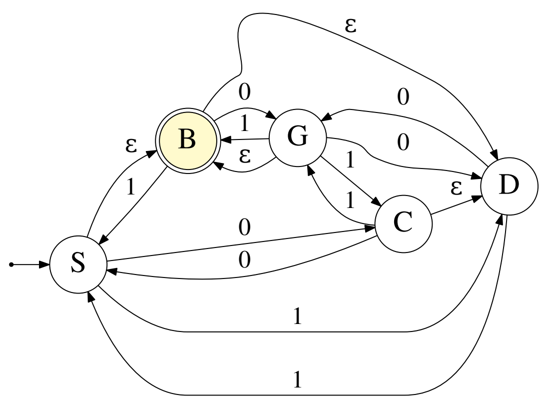
1.B. (5 PTS.) Describe (in detail) a DFA for the language below. Label the states and/or briefly explain their meaning.

$$\{ w \in \{0, 1\}^* \mid w \text{ has at least three } 1\text{'s and has odd length} \}.$$

2 (15 PTS.) I have a question about NFAs.

2.A. (5 PTS.) Recall that an NFA N is specified as $(Q, \delta, \Sigma, s, F)$ where Q is a finite set of states, Σ is a finite alphabet, $s \in Q$ is the start state, $F \subseteq Q$ is the set of accepting states, and $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$ is the transition function. Recall that δ^* extends δ to strings: $\delta^*(q, w)$ is the set of states reachable in N from state q on input string w.

In the NFA shown in the figure below what is $\delta^*(S, 0)$?



2.B. (10 PTS.) Given an arbitrary NFA $N = (Q, \Sigma, \delta, s, F)$ and an arbitrary state $q \in Q$ and an arbitrary string $w \in \Sigma^*$ of length t, describe an efficient algorithm, as fast as possible, that computes $\delta^*(q, w)$. Express the running time of your algorithm in terms of n, m, t, where $n = |Q|$, $m = \sum_{p \in Q} \sum_{b \in \Sigma \cup \{\epsilon\}} |\delta(p, b)|$, and $t = |w|$. Note that faster solutions can earn more points. You can assume that $|\Sigma| = O(1)$.

You do not need to prove the correctness of your algorithm (no credit for incorrect algorithm).

(Hint: Construct the appropriate graph, and do the appropriate things to it.)

3 (15 PTS.) MST WHEN THERE ARE FEW WEIGHTS.

Let $G = (V, E)$ be a connected undirected graph with n vertices and m edges, and with positive edge weights. Here, the edge weights are taken from a small set of k possible weights

$\{w_1, w_2, \ldots, w_k\}$ (for simplicity, assume that $w_1 < w_2 < \ldots < w_k$). Describe a **linear time** algorithm to compute the MST of G for the case that k is a constant. (For partial credit, you can solve the case $k = 2$.)

Provide a short explanation of why your algorithm is correct (no need for a formal proof).

4 (15 PTS.) SHUFFLE IT.

Let $w \in \Sigma^*$ be a string. A sequence of strings $u_1, u_2, \ldots, u_h$, where each $u_i \in \Sigma^*$, is a valid *split* of $w \iff w = u_1 u_2 \ldots u_h$ (i.e., w is the concatenation of $u_1, u_2, \ldots, u_h$). Given a valid split $u_1, u_2, \ldots, u_h$ of w, its ***price*** is $p(w) = \sum_{i=1}^{h} |u_i|^2$. For example, for the string INTRODUCTION, the split INT · RODUC · TION has price $3^2 + 5^2 + 4^2 = 50$.

Given two languages $T_1, T_2 \subseteq \Sigma^*$ a string w is a ***shuffle*** iff there is a valid split $u_1, u_2, \ldots, u_h$ of w such that $u_{2i-1} \in T_1$ and $u_{2i} \in T_2$, for all i (for simplicity, assume that $\varepsilon \notin T_1$ and $\varepsilon \notin T_2$). You are given a subroutine `isInT`(x, i) which outputs whether the input string x is in T_i or not, for $i \in \{1, 2\}$. To evaluate the running time of your solution you can assume that each call to `isInT` takes constant time.

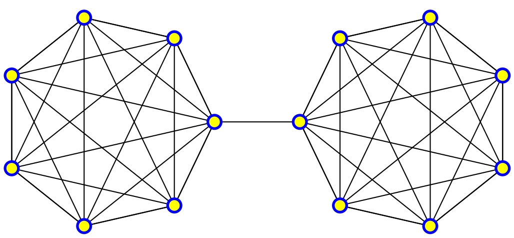
Describe an efficient algorithm, as fast as possible, that given a string $w = w_1 w_2 \ldots w_n$, of length n, and access to T_1 and T_2 via `isInT`, outputs the minimum ℓ_2 price of a shuffle if one exists. Your algorithm should output ∞ if there is no valid shuffle.

You will get partial credit for a correct, but slow (but still efficient), algorithm. An exponential time or incorrect algorithm would get no points at all.

What is the running time of your algorithm?

5 (15 PTS.) Dumb and dumbbell.

For $k > 1$, a (k, k)-*dumbbell* is a graph formed by two disjoint complete graphs (cliques), each one on k vertices, plus an edge connecting the two (i.e., its a graph with $2k$ vertices). See figure for a $(7, 7)$-dumbbell. The DUMB problem is the following: given an undirected graph $\mathsf{G} = (V, E)$ and an integer k, does G contain a (k, k)-dumbbell as a subgraph? Prove that DUMB is NP-COMPLETE.



6 (15 PTS.) YOU ARE THE DECIDER.

Prove (via reduction) that the following language is undecidable.

$$L = \{\langle M \rangle \mid M \text{ is a Turing machine that accepts at least 374 strings}\}.$$

(You can not use Rice's Theorem in solving this problem.)

7 (15 PTS.) YELLOW STREET NEEDS BITS.

There are n customers living on yellow street in Shampoo-banana. Yellow street is perfectly straight, going from south by southeast to north by northwest. The ith customers lives in distance

s_i meters from the beginning of the street (i.e., you are given n numbers: $0 \leq s_1 < s_2 < \cdots < s_n$). A new internet provider `Bits4You` is planning to connect all of these customers together using wireless network. A base station, which can be placed anywhere along yellow street, can serve all the customers in distance r from it.

The input is $s_1, s_2, \ldots, s_n, r$. Describe an efficient algorithm, as fast as possible, that computes the *smallest* number of base stations that can serve all the n customers. Namely, every one of the n customers must be in distance $\leq r$ from some base station that your algorithm decided to build. Incorrect algorithms will earn few, if any points. (Your algorithm output is just the number of base stations – there is no need to output their locations.)

Prove the correctness of your algorithm. What is the running time of your algorithm?