Final Exam Study Guide

CSE 30151 Spring 2022

Exam Date: May 4, 2022

Instructions

- The exam will take place on Wednesday, May 4, from 10:30 AM to 12:30 PM in our usual room (DeBartolo Hall 138).
- The exam will be worth a total of 120 points, or 20% of your final grade.
- You will have the whole period of two hours to write your solutions.
- You may consult the following materials during the exam: the Sipser textbook, your notes, and homework solutions. All of these materials must be printed on paper. If you have a digital copy of the textbook, you must print out any sections you want to use beforehand.
- The following materials may *not* be used during the exam: solutions to problems taken from the Internet or other outside sources; any electronic devices (including smart watches, phones, tablets, computers, and other Turing-equivalent machines).
- You may re-use any theorems or proofs provided in class, in the textbook, or in the homework assignments. If you re-use a theorem or proof from the textbook not used in class, please cite the page number. Example: (Sipser p. 42).
- If you think any problem contains typos or is unclear, please ask the instructor for clarification.

Topics

This exam is comprehensive, but with an emphasis on topics that we have covered since Midterm Exam 2 (including the pumping lemma for CFLs and Turing machines, up to and including NP-completeness and polynomial reductions). You should review HW5 problem 3, HW6, HW7, and HW8.

The exam may include any of the following topics or types of question:

- General knowledge about relationships among language classes, examples of languages in each class, and major theoretical results covered in the course such as the Church-Turing thesis, undecidability, and the P = NP question. What would it take to prove whether P = NP or P $\neq$ NP? What would be the implications of P = NP or P $\neq$ NP?
- Prove that a language is not context-free. You can use the pumping lemma for CFLs, known closure properties on CFLs (including reversal, intersection with a regular language, and homomorphisms), or any combination thereof. You may refer to any known non-context-free languages discussed in class, the textbook, or homework. When using the pumping lemma, take special care to pick a string s that is *in the language* and is *at least p symbols long.*

- Given a language, provide a state diagram of a single-tape Turing machine that decides it. You may use "stay" (S) moves, and transitions to the reject state may be implicit.
- Non-deterministic Turing machines.
- Prove that a Turing machine variant is equivalent to standard single-tape Turing machines.
- Turing-recognizability vs. decidability.
- Prove that a language is undecidable. You can use a reduction from any known undecidable language discussed in class, the textbook, or homework.
- Given a language, prove that it is in P. You can do this by giving a deterministic polynomial-time algorithm that decides it.
- Given a language, prove that is is in NP. You can do this by giving a nondeterministic polynomial-time algorithm that decides it, or a polynomial-time verifier that verifies it.
- Prove that a language is NP-complete. You can do this by proving that it is a member of NP and providing a polynomial-time reduction from any known NP-complete problem discussed in class, the textbook, or homework. You do not need to give an argument of correctness.
- To a lesser extent, topics from the first two thirds of the course (regular languages and context-free languages). To prove a language is regular, provide a DFA, NFA, or regular expression. To prove a language is context-free, provide a CFG or PDA.

The following types of problem will *not* be on the exam, although the theorems and algorithms involved may still be referenced:

- Converting among DFAs, NFAs, and regular expressions.
- DFA state minimization.
- Converting between CFGs and PDAs.
- Converting a CFG to Chomsky normal form.
- Removing ambiguity from CFGs.
- Proving that languages are undecidable using the diagonalization method (e.g. A_{TM}) or using computation histories (e.g. PCP).
- Polynomial reductions from *3SAT* using "gadgets", and questions about details of the polynomial reductions in section 7.5 of the textbook.

Practice Problems

Problem numbers for the international edition of the textbook are given in parentheses when they differ from the domestic edition.

1. General knowledge.
   (a) Draw a Venn diagram that shows the relationships among the following language classes: context-free languages, decidable languages, finite languages, P, regular languages, Turing-recognizable languages.
   (b) Problems 3.22 (intl. 3.9), 5.23 (intl. 5.11), and 7.18 (intl. 7.45) in the textbook.
2. Prove that a language is not context-free.
   (a) $\{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid 0 \leq i \leq j \leq k\}$

(b) $\{ww^R\texttt{a}^n \mid w \in \{\texttt{a},\texttt{b}\}^* \text{ and } n = |w|\}$

(c) $\{x\texttt{\#}y \mid x, y \in \{\texttt{a},\texttt{b}\}^* \text{ and } c_{\texttt{a}}(x) = c_{\texttt{a}}(y) \text{ and } c_{\texttt{b}}(x) = c_{\texttt{b}}(y)\}$

(d) Problems 2.30 (intl. 2.42), 2.42 (intl. 2.54), and 2.48b (intl. 2.59b) in the textbook.

3. Given a language, provide a state diagram of a Turing machine that decides it.

   (a) $\{\texttt{\#}w \mid w \in \{\texttt{a},\texttt{b},\texttt{c}\}^* \text{ and } c_{\texttt{a}}(w) \leq c_{\texttt{b}}(w) \leq c_{\texttt{c}}(w)\}$

   (b) $\{\texttt{\#1}^n\texttt{\#1}^m \mid n = \sqrt{m}\}$

   (c) $\{\texttt{\#1}^n \mid n \text{ is a perfect square}\}$

4. Prove that a Turing machine variant is equivalent to single-tape Turing machines.

   (a) A *two-stack pushdown automaton* is a pushdown automaton with two stacks instead of one. Each transition pops and pushes a symbol from each stack simultaneously and is of the form $(r, y_1, y_2) \in \delta(q, a, x_1, x_2)$, where $q, r \in Q$ and $x_1, x_2, y_1, y_2 \in \Gamma_\varepsilon$. Show that any Turing machine can be converted to a two-stack pushdown automaton.

   (b) Problems 3.12 (intl. 3.19) and 3.14 (intl. 3.21) in the textbook.

5. Prove that a language is undecidable.

   (a) $\mathit{REACHES}_{\mathrm{TM}} = \{\langle M, w, q\rangle \mid M \text{ is a TM that visits state } q \text{ when run on } w\}$

   (b) Problems 5.9-5.13 (intl. 5.25-5.29) in the textbook.

6. Prove that a language is NP-complete.

   (a) A *simple path* in a directed graph is a path that visits no node more than once. Prove that the following language is NP-complete.

$$\begin{aligned}\mathit{SIMPLE\text{-}PATH} = \{\langle G, k\rangle \mid\ & G \text{ is a directed graph and} \\ & \text{contains a simple path that visits } k \text{ nodes}\}\end{aligned}$$

   (b) Suppose you are signing up for courses next semester, and you want to sign up for exactly k courses. However, some courses cannot be taken together (perhaps due to time conflicts or prerequisites). Let us represent the courses as the nodes of an undirected graph G, and conflicts between classes as edges between the nodes. Show that the problem of determining whether it is possible to sign up for k classes given graph G, expressed as the following language, is NP-complete.

$$\begin{aligned}\mathit{NO\text{-}CONFLICTS} = \{\langle G, k\rangle \mid\ & G \text{ is an undirected graph with a} \\ & \text{subset of } k \text{ nodes with no conflicts}\}\end{aligned}$$

   (c) Problem 7.23 (intl. 7.50) in the textbook.

Changelog

- **Apr 26:** Added problem numbers for the international edition of the textbook. Added two more NP-complete problems.