CS 125 Section #10 **Midterm 2 Review** 11/5/14

1 Topics Covered

This midterm covers up through NP-completeness; countability, decidability, and recognizability will not appear on this midterm.

***Disclaimer**: This is not a comprehensive list of possible topics for the exam; in general, any material from lecture, homeworks, or sections is fair game.*

1.1 Models of Computation

- **DFA**: Accept if we end at an accept state
- **NFA**: Accept if any computation path ends at an accept state
- **Turing Machine (TM)**: DFA with a memory tape and a head that moves along the tape.
  - Also, **NTM**s, which are the nondeterministic variant.
- **Word-RAM**: Program with random memory access but a finite number of registers to work with.

1.2 Power of Computation

- **Regular Language**. A language accepted by a DFA.
  - This is the same as languages accepted using NFAs (proven using the subset construction). However, the smallest DFA may have exponentially many states compared to the smallest NFA.
  - Ways to show that a language is/isn't regular:
    * Myhill-Nerode: A language L is regular iff there are only finitely many equivalence classes under the equivalence relation $x \sim y$ iff for all strings z, $xz \in L \Leftrightarrow yz \in L$. The minimum number of states in a DFA for L is exactly the number of equivalence classes under $\sim$.
    * Constructions with NFAs and/or DFAs
    * Closure properties (union, intersection, complement, homomorphism, inverse homomorphism) - these were covered in section 8 / problem set 7.
- The standard TM is computationally equivalent to many variants (multi-tape TM, TM with bidirectional tape, etc.).
- A word-RAM can simulate a TM. A TMs can simulate a word-RAM with only quadratic slowdown.

1.3 Complexity Classes

- P: the set of languages that can be decided by a deterministic TM in polynomial time
- NP: the set of languages that can be decided by a deterministic TM in polynomial time

  - Alternate characterization: the set of languages for which there exists a polynomial time verifier V such that for any x, there exists a certificate y with $V(x, y) = 1$ if and only if x is in the language. The idea is that an 'answer' is easy to check once we have one.
  - coNP is the set of languages whose complements are in NP (problem set 8).
- While the languages above are formulated as decision problems, as we have seen, we can often use the decision problem to solve the search/optimization problem with only polynomial slowdown.

1.4 Reductions

- NP-complete problems are ones that are both in NP and are NP hard.
- Examples we've seen include Sat, 3-Sat, Clique, Vertex Cover, Integer Linear Programming.
- We usually show that a language L is NP-complete (or complete for some other class) by giving a reduction from some language L' which we already know to be NP-complete. That is, we give some polynomial-time computable function f such that $x \in L' \Leftrightarrow f(x) \in L$.
  - The intuition behind this is that if we know how to solve L, then we know can use that to solve L' – thus, L' must be an easier problem to solve than L.

1.5 Algorithms

You should be able to run any of these algorithms by hand on the exam; you should also be familiar with their runtimes and/or space requirements.

- **Pattern Recognition using DFAs**.
- **Depth First Search** and **Breadth First Search**.
- Shortest-path algorithms & negative cycle detection:
  - **Djikstra's Algorithm** (single-source). Modified BFS.
  - **Bellman-Ford** (single-source). Successive approximation: repeatedly look at through the edges and see if any edge gives a shorter path to its endpoint than what we have already found.
  - **Floyd-Warshall** (all pairs). Dynamic programming, with the subproblems looking at paths which can only use a subset of the vertices.
- **Linear Programming**. Maximize a linear function given a set of linear constraints.
- **Zero-sum Games**. Reduce to linear programming or solve by hand.
- Network flows:
  - Max flow = min cut
  - Can reduce this to linear programming
  - **Augmenting Paths**. Keep trying to find a path in the residual network where we can push more flow across. Stop when we are no longer able to do so; at this point, the set of vertices reachable in the residual network from the source defines a minimum cut.

2 Practice Problems

2.1 Shortest Paths and Linear Programming

Exercise. *In section, we already saw a way to formulate the shortest path problem as a linear program, by using a variable on each edge to denote whether or not it is included.*

Now, consider the LP

$$max d_t$$

subject to

$$d_v \leq d_u + w(u, v) \text{ for all edges } (u, v) \in E$$

$$d_s = 0$$

Argue that this LP finds the length of the shortest path from s to t.

Show how to extend the above LP to find the length of the shortest paths from s to every vertex in the graph. (Assume the shortest paths are finite for simplicity.)

Exercise. *You're playing a new mobile phone game, Very Happy Pigs, and are having trouble on the latest level. You decide to model the level as a directed graph, where each vertex represents a platform you can reach, and each edge represents a jump you can try to make. After extensive experimentation, youve labeled each edge with the probability (a number in $[0, 1]$) that you can successfully make the jump. Unfortunately, if you fail to make any jump, you instantly die, and have to start over. Describe an efficient algorithm to find a path from the start platform to the goal platform that maximizes the probability of getting through the level. You should assume that whether you make a certain jump of not is independent of all other jumps.*

2.2 Zero-Sum Games

Exercise. *Consider the following zero-sum game, where the entries denote the payoffs of the row player:*

$$\begin{pmatrix} 1 & -4 & 3 \\ 7 & 2 & -1 \\ -5 & 2 & 2 \end{pmatrix}$$

Write the linear programs for both players.

Exercise. *How would the answer to the previous problem change if the payoffs were to the* ***column*** *player?*

2.3 Network Flows

Exercise. *The edge connectivity of an undirected graph is the minimum number k of edges that must be removed to disconnect the graph. For example, the edge connectivity of a tree is 1, and the edge connectivity of a cyclic chain of vertices is 2. Show how the edge connectivity of an undirected graph $G = (V, E)$ can be determined by running a maximum-flow algorithm on at most $|V|$ flow networks, each having $O(V)$ vertices and $O(E)$ edges.*

2.4 Regular Languages

Exercise. *TRUE or FALSE? Justify your answers in one or two sentences.*

1. *If* 3-SAT $\in$ P, *then the* TRAVELLING SALESMAN PROBLEM *is in* P.
2. *There is an algorithm that decides* SAT *in exponential time.*
3. *If L is regular, then $L' = \{x \in L : |x|$ is not a multiple of 2014$\}$ is also regular.*
4. *If L is regular, then $L' = \{x \in L : |x| > 2009\}$ is also regular.*
5. *Every infinite subset of a non-regular language is non-regular.*
6. *The complement of any non-regular language is non-regular.*
7. *If every state of an NFA N is final, then $L(N) = \Sigma^*$.*

Exercise. *For each of the following languages, say whether it is regular or not. Briefly justify your answers.*

1. $L = \{a^n b^n a^n : n \geq 0\}$
2. $L = \{w : w$ *contains both obama and romney as substrings*$\}$, *over alphabet* $\Sigma = \{a, b, \ldots, z\}$
3. $L = \{wa^n : w \in \{a, b\}^*, n =$ *the number of a's in* $w\}$.
4. $L = \{w \in \{a, b\}^* : w$ *has at least 2 b's or at least 2 a's*$\}$
5. $\{a^{5n} b^{2m} c^{5p} : m, n, p \geq 0\}$
6. *The set of strings over alphabet $\{a, b\}$ in which every b is both preceded by an a and followed by an a.*
7. $L = \{a^{3i} b^{5j} : i, j \in \mathbb{N}\}$.

Exercise. *For any language L, let* $\textsc{Copy}(L) = \{w^n : w \in L, n \geq 0\}$.

1. *Prove that if L is finite, then* $\textsc{Copy}(L)$ *is regular.*
2. *Prove that if L is regular, then* $\textsc{Copy}(L)$ *need not be regular.*

Exercise. *For a string $w = w_1 \cdots w_n$, we consider a proper prefix to be a substring of the form $w_1 \cdots w_k$ for some $0 < k < n$. Prove that if L is regular then* $\mathsf{PrefixFree}(L) = \{w : w \in L$ *and no proper prefix of w is in* $L\}$ *is also regular.*

2.5 Polynomial Time Complexity

Exercise. *For an integer N, let let $\langle N\rangle_B$ denote the binary representation of N (over alphabet $\{0, 1\}$) and $\langle N\rangle_U$ denote the unary representation of N (i.e. a string of N ones). For each of the following two functions, state whether it is computable in polynomial time and justify your answer. You may quote without proof basic facts about the complexity of arithmetic operations (e.g. addition and multiplication).*

1. $\text{BinaryToUnary}(\langle N\rangle_B) = \langle N\rangle_U$.
2. $\text{UnaryToBinary}(\langle N\rangle_U) = \langle N\rangle_B$.

2.6 NP-completeness

Exercise. *TRUE or FALSE? Justify your answers in one or two sentences.*

1. *Every* NP*-hard language is* NP*-complete.*
2. *If there is a polynomial-time mapping reduction from A to B (i.e. $A \leq_p B$) and B is in NP, then A is in NP.*

Exercise. SetCover *is the following problem: given a finite universe U, a family of subsets $S_1, \dots, S_n \subseteq U$, and a number $\ell \in \mathbb{N}$, are there ℓ of the sets S_i whose union equals the entire universe U?*

Prove that SetCover *is* NP*-complete. (Hint: recall the* NP*-complete problem* VertexCover*: given a graph G and a number k, determine whether there is a set of k vertices that includes an endpoint of every edge in G.)*

Exercise. *Consider the following two languages, whose instances are of the form $\langle G, k\rangle$ where G is an undirected graph and k is a natural number:*

- Clique *consists of all $\langle G, k\rangle$ for which there is a set S of k vertices in such that every pair of vertices in S are connected by an edge in G.*
- Independent Set *consists of all $\langle G, k\rangle$ for which G has a set S of k vertices such that no pair of vertices in S is connected by an edge in G.*

In lecture, we proved that Clique *is* NP*-complete. Show that* Independent Set *is also* NP*-complete.*

Exercise. 1. *Prove that* Cubic Equations modulo 2, *that is, the problem of deciding, given a system of equations modulo 2 over several integer variables where each term has total degree at most 3, whether the system has a solution, is* NP-*complete. (Hint: reduce from 3-SAT.)*

2. *(Challenge) Prove that* Quadratic Equations modulo 2, *which is defined analogously, is* NP-*complete. (Hint: reduce the degree of a monomial $x_i x_j x_k$ by introducing another variable and another equation.)*