CSCI 1010 - Fall'25 Theory of Computation Lorenzo De Stefani

HW8

n/a

Reminder: Submit your assignment on Gradescope by the due date. Submissions must be typeset. Each page should include work for only one problem (i.e., make a new page/new pages for each problem). See the course syllabus for the late policy.

While collaboration is encouraged, please remember not to take away notes from any labs or collaboration sessions. Your work should be your own. Use of other third-party resources is strictly forbidden.

Please monitor Ed discussion, as we will post clarifications of questions there.

Problem 1

A language A is said to be P-complete if $A \in P$ and if $\forall L \in P, L \leq_p A$. Which one of the following statements is the most correct?

- No regular language is P-complete.
- Some regular languages are P-complete but not all of them.
- All regular languages are P-complete.
- There exists a regular language which is P-complete iff $P = NP$.

Choose a statement you believe to be correct and provide complete proof of why is that the case.

Solution:

The correct statement among the options provided is some but not all regular Languages are P-complete.

Let R be a non-trivial regular language in P. This means that there exists $x, y \in \Sigma^*$ (where Σ is the alphabet) such that $x \in R$ and $y \notin R$.

Let $A \in P$. This means that there exists a polynomial time decider D_A for A. We construct a polynomial time reduction f from A to R in the following way:

Let $w \in A$. Run D_A on w. If D_A accepts, $f(w) = x$,e else $f(w) = y$. $f(w) \in R$ if and only if $w \in A$ because D_A is a decider. Moreover, since D_A runs in polynomial time, the reduction takes polynomial time. (Note: for full credit this would need to be expanded on).

On the other hand, if R is a trivial language, then this reduction cannot take place. There simply does not exist a map from A to R that maps preserves membership since either all the strings are in R or no string is in R. For example, the language that does not accept any string is regular, but it is not possible to map a member from another language to a member in this language. (Note: for full credit this would need to be expanded on).

Problem 2

Given a Boolean formula ϕ, we say that an assignment of its Boolean variables is a *satisfactory assignment* if ϕ evaluates to TRUE using the truth values for its variable given in the assignment.

Further, we say that two assignments of *truth value* to its Boolean variable are *distinct* if there is at least a variable which is assigned value TRUE in one assignment and value FALSE in the other.

Let $QUADRUPLELESAT = \{< \phi > | \phi$ is a Boolean formula in CNF and it has at least four distinct satisfying assignments$\}$.

Show that $QUADRUPLESAT$ is $NP - complete$. Here is an outline of how to do so:

1. Propose a Deterministic Verifier requiring worst case polynomial time for $QUADRUPLESAT$. Given the encoding of a Boolean formula $\langle\phi\rangle$ we say that its *size* is the number of literals in ϕ (i.e., the sum of occurrences of variables in ϕ).
   (a) Provide a description of a deterministic polynomial-time verifier V for $QUADRUPLESAT$. What is the certificate c used?
   (b) Prove the *correctness* of your verifier: Two directions:
   - $< \phi > \in QUADRUPLESAT \implies \exists c$ s.t. the verifies accepts $< \phi >$ using c.
   - $< \phi > \notin QUADRUPLESAT \implies \nexists c$ s.t. the verifies accepts $< \phi >$ using c.

   (c) Prove that the proposed verifier required worst case polynomial time with respect to the size of the input and of the certificate.

2. Provide a description of a nondeterministic polynomial-time decider N for for $QUADRUPLESAT$.

   (a) Prove the *correctness* of your decider N that is $L(N) = QUADRUPLESAT$.

   (b) Prove that your decider requires worst case non-deterministic polynomial time with respect to the size of the input.

3. Describe a deterministic algorithm that given a string w computes $f(w)$ such that $f(w)$ is the encoding of a Boolean formula $\phi'' \in QUADRUPLESAT$ if and only if w is the encoding of a *satisfiable* Boolean formula ϕ'. Given a Boolean formula you can assume that its encoding can be produced (i.e., written) in time *linear with respect to its size*.

   - Argue the correctness of your algorithm, that is that it computes $f(\cdot)$ s.t. w is the encoding of a *satisfiable* Boolean formula $\iff$ $f(w) \in QUADRUPLESAT$.
   - Prove that your proposed algorithm runs in worst case polynomial time with respect to the size of the input (i.e., $|w|$)

Here is a solution sketch.

Deterministic polynomial time verifier Let $\phi(a_1, ..., a_n)$ be a Boolean formula in CNF (if the format is incorrect, reject immediately) and four truth assignments for $a_1, ..., a_n$ (if they aren't such assignments, reject immediately. Check that all the assignments are distinct in linear time by comparing each variable assignment for all four truth assignments at the same time. Then, evaluate $\phi(a_1, ..., a_n)$ (can be done in linear time) at each truth assignment. If all four yield true, accept. Else, reject.

If ϕ is in QUADRUPLESAT, then it has at least four distinct satisfying assignments A_1, A_2, A_3, A_4, so the verifier will accept $< \phi, (A_1, A_2, A_3, A_4) >$. On the other hand, if it is not in QUADRUPLESAT, then it has strictly less than four distinct satisfying assignments, so for any distinct A_1, A_2, A_3, A_4 assignments, the verifier will reject $< \phi, (A_1, A_2, A_3, A_4) >$.

Moreover, the process takes polynomial time, since checking distinctness takes $O(n)$, and evaluating also takes $O(n)$, for a total runtime in $O(n)$.

Nondeterministic polynomial time decider
Let $\phi(a_1, ..., a_n)$ be a Boolean formula in CNF (if the format is incorrect, reject immediately). Consider the following nondeterministic process: non-deterministically select four distinct truth assignments for $a_1, ..., a_n$, and evaluate them. If they all satisfy the formula, then accept ϕ. Else, reject.

If ϕ is in QUADRUPLESAT, then it has at least four satisfying assignments. There will be a branch of the non-deterministic process that considers this set of four assignments and accepts ϕ.

On the other hand, if ϕ is not in QUADRUPLESAT, then there is no set of four assignments that will satisfy it at the same time. Therefore, non of the non-deterministic branches of the process will end in acceptance, and ϕ will be rejected.

Therefore, this process decides whether *phi* is in QUADRUPLESAT or not. Moreover, it does so in nondeterministic linear time: each branch takes $O(n) + O(n) + O(n) + O(n) \approx O(n)$ time.

It remains to show that $QUADRUPLESAT$ is NP-hard. To do so we may reduce SAT, a known NP-hard problem to $QUADRUPLESAT$. Let F be an instance of SAT, namely a Boolean formula. Then let $z1$ and z_2 be variables which do not appear in F. Then the Boolean formula $E = F \wedge (z1 \vee \neg z1) \wedge (z2 \vee \neg z2)$ has a truth assignment iff F has a truth assignment (since the last clause is always true), and moreover E has at least four truth assignments iff F has a truth assignment, since we may assign the variables z1 and z2 arbitrarily. It takes linear time to add these two clauses to the end of the input. So $F \to E$ defines a poly-time reduction of $QUADRUPLESAT$ to SAT.

Problem 3

Given a graph $G = (V, E)$, a clique of size k is a subset $V' \subseteq V$ such that (i) $|V'| = k$, and (ii) each vertex in V' is connected to all other vertices in V' by an edge in E. We say that two cliques V' and V'' are distinct if they differ in at least one vertex

Let $TRIPLE - CLIQUE = \{\langle G, k\rangle | G$ is a graph with at least three distinct $k-$ cliques$\}$.

1. Show that $TRIPLE - CLIQUE \in NP$. You can either show a deterministic verifier for $TRIPLE - CLIQUE$ or a non-deterministic decider. You

should provide a proof of correctness and of polynomial running time (as outlined in Problem 2).

2. Given a graph $G = (V, E)$, it is possible construct a string encoding of G of size $O(|E|)$. Argue that $CLIQUE \leq_p TRIPLECLIQUE$

Solution:

First, we will show that $TRIPLE - CLIQUE \in NP$. To do so, consider the following algorithmic description of an NTM N. On input w, check that w is of the form $\langle G, k\rangle$, where G is a graph and k is a positive integer. If not, reject the input. Nondeterministically pick a triple (V_1, V_2, V_3), where V_1, V_2, V_3 are each a subset of k vertices of G. If V_1, V_2, V_3 do not differ in any vertex, reject the input. Otherwise for each $V_\ell \in \{V_1, V_2, V_3\}$, check if G contains an edge between the pair (v_i, v_j) for all $v_i, v_j \in V_\ell$ such that $i < j$. If this is true for all $V_\ell \in \{V_1, V_2, V_3\}$, accept the input, otherwise reject.

First, we will show that $w \in TRIPLE - CLIQUE$ if and only if $w \in L(N)$. Suppose $w \in TRIPLE - CLIQUE$. Then, w is of the form $\langle G, k\rangle$, so it is not rejected immediately. Then, since $w \in TRIPLE - CLIQUE$, there must exist at least three distinct k-cliques in G. Denote any such three distinct k-cliques in G as V_1, V_2, V_3. Since we go through all triples of subsets of k vertices of G, we will eventually pick the triple (V_1, V_2, V_3); consider this branch of execution. By our choice V_1, V_2, V_3 must differ in at least one vertex, so we do not reject immediately in this branch. Then, it must be that G contains the an edge between (v_i, v_j) for all $v_i, v_j \in V_\ell$ for all $\ell \in \{1, 2, 3\}$, because by our choice, V_1, V_2, V_3, are cliques of size k. So, we accept, as desired. Suppose $w \notin TRIPLE - CLIQUE$. If w is not of the form $\langle G, k\rangle$, it is rejected immediately. Otherwise, w is of the form $\langle G, k\rangle$, where G is a graph and k is a positive integer. Suppose, heading for a contradiction, that we accept the input. For this to happen, we must have found a triple (V_1, V_2, V_3) of distinct subsets of k vertices such that there is an edge between each pair of vertices for each of V_1, V_2, V_3. But, by our definition of a clique, this means that V_1, V_2, V_3 are k-cliques and are distinct. So, there exist three distinct k-cliques in G so $w \in TRIPLE - CLIQUE$. Therefore, we have a contradiction and it must be that we reject the input, as desired.

The initial step of verifying that the input is of the form $\langle G, k\rangle$ can be done in linear time with respect to the size of the input. Then, we can

construct the adjacency matrix of G in $O(n^2)$ time, where $n = |V|$; note that $|E| = O(n^2)$. Within each branch, checking that V_1, V_2, V_3 are distinct can be done in $O(k^3)$ by iterating over each triple of vertices across the three subsets. Then, for each $V_i \in \{V_1, V_2, V_3\}$, we can check that each pair of vertices in V_i has an edge in G in time $O(k^2)$ using the adjacency matrix representation of G, since it takes $O(1)$ time to check if there is an edge between a given pair of vertices. So, overall, N runs in polynomial time with respect to the size of the input ($O(n^2)+O(k^3)+O(3\cdot k^2) = O(n^2+k^3)$).

Since there exists an NTM N that decides $TRIPLE-CLIQUE$ in polynomial time, we have $TRIPLE - CLIQUE \in NP$.

Now, for the reduction. If G is not of the incorrect format, then return G. Otherwise, make two copies of the original graph G denoted by G' and G'' where the vertices in G' and G'' are denoted by e.g. v' and v'', respectively (i.e. use different labels for the vertices). Then, return the combined graph with G, G' and G'' as separate components. Suppose $w \notin CLIQUE$. If it is incorrect format, then the output is also in incorrect format so $f(w) \notin TRIPLECLIQUE$. Otherwise, it does not have a k-clique, so copying the graph three times certainly cannot result in 3 k-cliques. Suppose $w \in CLIQUE$. Then, the graph has at least one k-clique. Then, copying the graph twice results in at least three distinct k-cliques, so $f(w) \in TRIPLECLIQUE$. Copying the graph twice requires at most quadratic time since the maximum number of edges is n^2, where $n = |V|$. Thus, this reduction completes in polynomial time.