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

HW10

Due:

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

In the Maximal Clique Problem, given an undirected graph $G = (v, E)$ and a positive integer value k we want to decide if k is the *exact* size of the largest clique in G. We can define the corresponding language as:

$MAXCLIQUE = \{< G, k > | k \text{ is the exact size of the largest clique in } G\}$

1. Modify the reduction $f(\cdot)$ seen in class from $3SAT$ to $CLIQUE$, to obtain a computable function $f^*(\cdot)$ that given as input ϕ returns a pair (G, k) where G is an undirected graph and k is a non-negative integer value such that
   - if ϕ is not a Boolean formula in 3CNF, G includes a single edge and $k = 1$
   - if ϕ is a satisfiable Boolean formula in 3CNF with k clauses then the exact size of the clique of maximum size in G is k.
   - if ϕ is a non-satisfiable Boolean formula in 3CNF with k clauses then the exact size of the clique of maximum size in G is $k - 1$.
2. Argue that $MAXCLIQUE$ is NP-hard.
3. Argue that $MAXCLIQUE$ is coNP-hard.
4. Argue that $MAXCLIQUE \in NP \implies HAMPATH^c \in NP$

Solution Note: the following solutions would have to be written out more to receive full credit.

1. We define f as follows:

   - if the formula is not in 3CNF form, output $< K_2, 1 >$
   - if the formula is in 3CNF form, do the same construction as in the notes, but add a $k-1$ clique.

   Showing this construction is computable in polynomial time is similar to the notes in class. Further, according to the notes there is a k clique where k is the number of clauses if and only if the formula is satisfiable. It is impossible to have a larger clique: suppose for the sake of contradiction that there is a $k+i$ clique where $i > 0$. Then two variables in the same clause must be connected by the pigeonhole principle, which is a contradiction by construction. Therefore, the maximum clique is of size k. If the formula is unsatisfiable and in 3CNF form, then its largest clique is $k-1$ by construction. If it is in the incorrect format f outputs $< K_2, 1 >$.

2. 3SAT is NP-complete, and therefore NP hard. Since we have a polynomial time reduction from 3SAT to MAXCLIQUE, any problem in NP can be reduced to MAXCLIQUE in polynomial time by chaining, so MAXCLIQUE is NP hard.

3. We showed in class that a language L is NP-complete iff its complement is coNP-complete. Thus, $3SAT^C$ is coNP-complete. We do a similar reduction from $3SAT^C$, but passing $< G, k-1 >$ when the formula is in 3CNF form and $< K_2, 2 >$ when it is not instead. Since $3SAT^C$ is coNP-hard and there is a polynomial time reduction from $3SAT^C$ to $MAXCLIQUE$, any problem in coNP can be reduced to $MAXCLIQUE$ in polynomial time by chaining of reductions. Therefore, $MAXCLIQUE$ is coNP-hard

4. HAMPATH discussed in slide 16, it is a language in NP. Therefore, $HAMPATH^C$ is in coNP. Since MAXCLIQUE is coNP-hard, there is a polynomial time reduction from $HAMPATH^C$ to MAXCLIQUE. If MAXCLIQUE is in NP, $HAMPATH^C$ must also be in NP.

Problem 2

Prove that $MAXCLIQUE \in PSPACE$.

Solution

Rough idea: construct adjacency matrix (it takes $O(n^2)$ space). A list of edges (at most $O(n^2)$ space) also works. Then, for every subset of k distinct vertices check if they form a clique (for each pair of vertices in this subset, loop through all edges to make sure that there exists an edge between them). If no subset of size k is a clique, reject. Then, do the same procedure for subsets of size $k+1$. If one of those form a clique, reject. If the procedure ends, accept. The key to making this PSPACE is reusing the same space for each check of k and $k+1$ vertices and also using the same space to store each subset (e.g. enumerate through the binary strings x of length n from $000...000, 000...001, 000...010, 000...011$, to $111...111$, each of which corresponds to a unique subset with $x_i = 1$ meaning vertex i is in the current subset and $x_i = 0$ meaning vertex i is not in the current subset).

Problem 3

Please review with attention the definition of the TQBF language as given in the textbook on pages 338-341. Please review the proof of this language being a member of $PSPACE$ and $PSPACE$-hard.

Given a Quantified Boolean formula $\phi = Q_1 x_1, Q_2 x_2, \ldots Q_m x_m[\psi]$ such that ψ is a Boolean formula with m variables, we say that ϕ is a Quantified CNF Boolean formula if ψ is a Boolean formula in CNF format.

The language $TQBF - CNF = \{< \phi > | \phi$ is a true fully quantified CNF Boolean formula$\}$ is also $PSPACE$-complete.

Similarly, given a Quantified Boolean formula $phi = Q_1 x_1, Q_2 x_2, \ldots Q_m x_m[\psi]$ such that ψ is a Boolean formula with m variables, we say that ϕ is a Quantified 3CNF Boolean formula if ψ is a Boolean formula in 3CNF format.

Consider now the language $TQBF - 3CNF = \{< \phi > | \phi$ is a true fully quantified 3-CNF Boolean formula$\}$.

1. Prove $TQBF - 3CNF \in PSPACE$.
2. Prove $TQBF - 3CNF$ is $PSPACE$-hard.

Hint: Read with lots of care the results for TQBF. This should give you all the needed inspiration. For the proof of hardness, I would *strongly* suggest proving $TQBF - CNF \leq_p TQBF - 3CNF$. Perhaps we have

seen something similar before....careful how you manage (or quantify) the new variables.

Solution The language $TQBF - CNF$ is given to be in $PSPACE$. Thus, there exists a decider M' that decides $TQBF - CNF$ in polynomial space. We will construct a decider M to decide $TQBF - CNF$ as follows.

- First, check whether the input w is a quantified boolean formula $\langle\phi\rangle$, where $\phi = Q_1x_1, Q_2x_2, \ldots Q_mx_m[\psi]$ such that ψ is a Boolean formula in 3CNF. If not, reject.
- Provide $\langle\phi\rangle$ as input to M'. If M' accepts, accept. If M' rejects, reject.

We can check the format using a polynomial amount of space with respect to the size of the input. Then, we know that M' takes polynomial amount of space with respect to the size of the input. So, the total space used is polynomial with respect to the size of the input.

Now, we have the proof of correctness. Suppose $w \in TQBF - 3CNF$. Then, it is of the correct format, so we do not reject on step 1. Note that $TQBF - 3CNF \subseteq TQBF - CNF$, so since $w \in TQBF - 3CNF$, it must be that $w \in TQBF - CNF$. So, M' accepts, and we accept, as desired.

Suppose $w \notin TQBF - 3CNF$. If it is not of the correct format, we reject in step 1. So, if we don't reject in step 1, it must be that $w = \langle\phi\rangle$ is of the correct format. So, it must be that ψ is $3CNF$. So, if $w \notin TQBF - 3CNF$, it must be because the quantified boolean formula is not true. So, it must be that M' rejects, so we reject, as desired.

For part 2, here's the rough idea. We first use the transformation in Lecture 18, Slide 7 to convert ψ from CNF to 3CNF using polynomial space. This might introduce some new variables that did not appear in ψ originally, so we need to add quantifiers for these variables. We can make all of these quantifiers of the form $\exists$ ("there exists") to preserve whether the new formula we make is True or False as a whole.