Nathan Bickel
CSCE 551: Section 001
Professor: Dr. Fenner
April 23, 2024

CSCE 551 Homework 6

For the following, you can assume that all devices use the binary alphabet $\Sigma := \{0, 1\}$ for input/output/printing. For these problems, you may assume that every string can be interpreted as the encoding of a Turing machine.

Problem 7.5 Is the following forumla satisfiable?

$$(x \vee y) \wedge (x \vee \overline{y}) \wedge (\overline{x} \vee y) \wedge (\overline{x} \vee \overline{y})$$

A boolean formula ϕ (formed from constants $0, 1$ and boolean variables $x_1, \ldots, x_n$ and $\vee, \wedge, \neg$) is satisfiable if there is a setting of $x_1, \ldots, x_n$ that makes ϕ true. The formula given is not satisfiable: there are only four possible settings of x and y, and each of the four settings makes one of the clauses false.

Problem 7.6 Show that P is closed under union, concatenation, and complement.

We show each in turn.

Union: Let $L_1, L_2 \in \mathsf{P}$ (we will show that $L_1 \cup L_2 \in \mathsf{P}$). By definition, L_1 and L_2 are both decidable in polynomial time, and so there exist TMs M_1 and M_2 that decide L_1 and L_2 in ptime, respectively. We will construct a TM M' that decides $L_1 \cup L_2$ in ptime by

$M' :=$ "On input $\langle w \rangle$:
1. Run M_1 on w, and accept w if it accepts.
2. Run M_2 on w."

It is not hard to prove that M' decides $L_1 \cup L_2$, and since the sum of polynomials is a polynomial, M' runs in ptime. Therefore, $L_1 \cup L_2 \in \mathsf{P}$.

Concatenation: Let $L_1, L_2 \in \mathsf{P}$ (we will show that $L_1 L_2 \in \mathsf{P}$). By definition, L_1 and L_2 are both decidable in ptime, and so there exist TMs M_1 and M_2 that decide L_1 and L_2 in ptime, respectively. We will construct a TM M' that decides $L_1 L_2$ in ptime by

$M' :=$ "On input $\langle w \rangle$:
1. Run the following for each $i \in \{0, 1, \ldots, |w|\}$:
    1. Let w_1 be the substring composed of the first i symbols of w, and let w_2 be the substring such that $w = w_1 w_2$.
    2. Run M_1 on w_1, and continue to a new i if it rejects.
    3. Run M_2 on w_2, and accept w if it accepts.
2. Otherwise, reject w."

Then, for any $w \in L_1L_2$, we have $w = w_1w_2$ for some $w_1 \in L_1$, $w_2 \in L_2$. When step 1 choose $i = |w_1|$, M' will accept w in Step 1.3. Otherwise, M' will reject its input. Thus, M' decides L_1L_2. Also, M' only runs M_1 and M_2 $|w|$ times each, so since the sum and product of polynomials is a polynomial, M' runs in ptime. Therefore, $L_1L_2 \in \mathsf{P}$.

Complement: Let $L \in \mathsf{P}$ (we will show that $\overline{L_1} \in \mathsf{P}$). By definition, L is decidable in ptime, so there exists a TM M that decides L in ptime. We will construct a TM M' that decides $\overline{L}$ in ptime by

$M' :=$ "On input $\langle w \rangle$:
1. Run M on w.
2. If M accepts w, reject w.
3. If M rejects w, accept w."

It is not hard to show that M' decides $\overline{L}$, and the time to run M' on w differs from the time to run M on w by only a constant. Thus, M' runs in ptime, so $\overline{L} \in \mathsf{P}$. □

Problem 7.7 Show that NP is closed under union and concatenation.

NP ("nondeterministic ptime") is the class of all languages L such that there exists a TM V such that on input $w\#y$, V halts in time polynomial in $|w|$ and for all w, $w \in L$ if and only if there exists a y such that $V(w\#y)$ accepts. V is called a verifier, and y is called a candidate proof that $w \in L$. In a sense, $V(w\#y)$ checks whether y is a legitimate proof. We will now show each closure in turn.

Union: Let $L_1, L_2 \in \mathsf{NP}$ (we will show that $L_1 \cup L_2 \in \mathsf{NP}$). Then, there exist verifiers V_1, V_2 and candidate proofs y_1, y_2 for L_1, L_2, respectively. We will construct a verifier V' such that on input $w\#y$, V' halts in time polynomial in $|w|$ and for all w, $w \in L_1 \cup L_2$ if and only if there exists a y such that $V(w\#y)$ accepts by

$V' :=$ "On input $w\#y$:
1. Run V_1 on $w\#y$, and accept $w\#y$ if it accepts.
2. Run V_2 on $w\#y$."

Let $w \in L_1 \cup L_2$. Then, $w \in L_1$ or $w \in L_2$. If $w \in L_1$, then V' will accept $w\#y_1$, and if $w \in L_2$, then V' will accept $w\#y_2$. If any candidate proof exists, it must be from a string in L_1 or L_2, so the other direction holds as well. Thus, V' is a verifier for $L_1 \cup L_2$, and since the sum of polynomials is a polynomial, V' runs in ptime. Thus, $L_1 \cup L_2 \in \mathsf{NP}$.

Concatenation: Let $L_1, L_2 \in \mathsf{NP}$ (we will show that $L_1L_2 \in \mathsf{NP}$). Then, there exist verifiers V_1, V_2 and candidate proofs y_1, y_2 for L_1, L_2, respectively. Let & be a symbol not in the alphabet. We will construct a verifier V' such that on input $w\#y$, V' halts in time polynomial in $|w|$ and for all w, $w \in L_1L_2$ if and only if there exists a y such that $V(w\#y)$ accepts by

$V' :=$ "On input $w\#y$:
1. Let i, y_1, and y_2 be such that $y = n\&y_1\&y_2$ (if no such strings exist, then reject $w\#y$).
2. Let w_1 be the substring composed of the first n symbols of w, and let w_2 be the substring such that $w = w_1w_2$.
3. Run V_1 on $w_1\#y_1$, and reject $w\#y$ if it rejects.
4. Run V_2 on $w_2\#y_2$."

If $w \in L_1L_2$, then there exist $w_1 \in L_1$, $w_2 \in L_2$ with candidate proofs y_1, y_2, respectively, such that $w = w_1w_2$. In this case, the candidate proof $y = |w_1|\&y_1\&y_2$ will be accepted by V'. If any candidate proof exists, it must be from a string in L_1 or L_2, so the other direction holds as well. Also, V' only runs V_1 and V_2 once each, so since the sum of polynomials is a polynomial, V' halts in ptime for every input. Therefore, $L_1L_2 \in \mathsf{NP}$. □

Problem 7.22 Let

$$DOUBLE\text{-}SAT = \{\langle\phi\rangle \mid \phi \text{ has at least two satisfying assignments}\}.$$

Show that $DOUBLE\text{-}SAT$ is NP-complete.

We first write a number of definitions. A function $f : \Sigma^* \to \Sigma^*$ is ptime computable if there is a transducer M that computes f and for all $w \in \Sigma^*$, M runs in time $\text{Poly}(|w|)$. A polynomial reduction is an m-reduction that is ptime computable. For any $A, B \subseteq \Sigma^*$, we say that A is ptime m-reducible (p-reducible) to B ($A \leq_p B$) if $A \leq_m B$ via a polynomial reduction. We showed in class that if $A \leq_p B$, then $B \in \mathsf{P} \implies A \in \mathsf{P}$ and $B \in \mathsf{NP} \implies A \in \mathsf{NP}$. We say that a language A is NP-hard if $B \leq_p A$ for every $B \in \mathsf{NP}$. Finally, we say that A is NP-complete if $A \in \mathsf{NP}$ and A is NP-hard. We showed in class that SAT, and in particular the restrictions $CNF\text{-}SAT$ and 3-$CNF\text{-}SAT$, are all NP-complete.

We can now do the problem by showing that $DOUBLE\text{-}SAT \in \mathsf{NP}$ and $SAT \leq_p DOUBLE\text{-}SAT$. That $DOUBLE\text{-}SAT \in \mathsf{NP}$ is easy: a candidate proof would consist of two satisfying assignments of ϕ. We now show that $SAT \leq_p DOUBLE\text{-}SAT$. Given a SAT formula ϕ, we construct in ptime a $DOUBLE\text{-}SAT$ formula ϕ' such that

$$\langle\phi\rangle \in SAT \iff \langle\phi'\rangle \in DOUBLE\text{-}SAT.$$

Let n be the number of boolean variables of ϕ. Then, add a fresh boolean variable x_{n+1}, and set

$$\phi := \phi \wedge (x_{n+1} \vee \overline{x_{n+1}}). \qquad (*)$$

($\Rightarrow$) Suppose ϕ is satisfiable. Then there exists a setting $x_1, \ldots, x_n$ of the variables that satisfies ϕ. By our construction, then, this settings with x_{n+1} either true or false satisfies ϕ' since $(x_{n+1} \vee \overline{x_{n+1}})$ is true with either assignment to x_{n+1}, so ϕ' has at least two satisfiying assignments.

($\Leftarrow$) We will prove the contrapositive. Suppose ϕ is not satisfiable. Then there is no setting $x_1, \ldots, x_n$ that satisfies ϕ, so there is no way to satisfy ϕ' (we can satisfy the clause we added, but not the part of ϕ' that is ϕ).

Our construction is clearly in ptime, and since we have shown that $(*)$ holds, we have that $SAT \leq_p DOUBLE\text{-}SAT$. Thus, since SAT is NP-hard, $DOUBLE\text{-}SAT$ is also NP-hard. Therefore, $DOUBLE\text{-}SAT$ is NP-complete. □

Problem 7.35 A subset of notes of a graph G is a dominating set if every other node of G is adjacent to some node in the subset. Let

$$DOMINATING\text{-}SET = \{\langle G, k\rangle \mid G \text{ has a dominating set with } k \text{ nodes}\}.$$

Show that it is NP-complete by giving a reduction from $VERTEX\text{-}COVER$.

Let G be a graph. A set $C \subseteq V(G)$ is a vertex cover of G if every edge in G has at least one endpoint in C. Also, we say that a vertex is isolated if it has no neighbors. Then, we have that

$$VERTEX\text{-}COVER := \{\langle G, v\rangle \mid G \text{ is a graph, } k \in \mathbb{N}, \text{ and } G \text{ has a vertex cover } C \text{ with } |C| \leq k\}.$$

We showed in class that $VERTEX\text{-}COVER$ is NP-complete. Thus, we will show that

$$VERTEX\text{-}COVER \leq_p DOMINATING\text{-}SET, \tag{$*$}$$

which proves that $DOMINATING\text{-}SET$ is NP-complete (when combined with the somewhat obvious fact that $DOMINATING\text{-}SET \in \mathsf{NP}$, since it is easy to verify in polynomial time whether a set of vertices is a dominating set and the right size).

Let $\langle G, k\rangle \in VERTEX\text{-}COVER$. We will construct in ptime a graph G' and a natural number k' such that

$$\langle G, k\rangle \in VERTEX\text{-}COVER \iff \langle G', k'\rangle \in DOMINATING\text{-}SET. \tag{$**$}$$

Initially, set $G' := G$ where each $v \in V(G)$ is labelled v' in $V(G')$. Then,

1. If there are fewer than k vertices in G that are not isolated, add $m' := k - |G|$ isolated "dummy" vertices.
2. For all adjacent vertices $u', v' \in V(G')$, add a vertex $w_{u'v'}$.
3. Add an edge from $w_{u'v'}$ to both u' and v'.

Also, let m be the number of isolated vertices in G, and set $k' := k + m$. This is clearly doable in polynomial time. We now show that $(**)$ holds.

($\Rightarrow$) Suppose G has a vertex cover C with $|C| \leq k$. Then, we can choose $C' \subseteq V(G')$ by:

1. For all $v \in C$, add the corresponding vertex v' in G' to C'.
2. If $|C| < k$, arbitrarily add $m' := k - |C|$ non-isolated vertices to C' (step 1 on the construction of G' guarantees this is possible). This "pads" the size of C'.
3. For all isolated vertices v in G, add the corresponding v' in G' to C'.

Then, $C' = |C| + k - |C| + m = k + m = k'$, and we claim that C' is a dominating set of G'. To see this, let v' be a vertex in G'. We need to show that v' is either in C' or has a neighbor u' in C'. Suppose v' is not in C'. Then v is not in C in G (or we would have added it to C' in step 1), and v is not isolated in G (or we would have added it to C' in step 3). So v has a neighbor u in G, and thus $u \in C$ (since C must cover uv). So $u' \in C'$, and thus v' is adjacent to a vertex in C'. Thus, every vertex is in C' or adjacent to it, and so C' is a dominating set of size k' in G'.

($\Leftarrow$) Suppose G' has a dominating set S' with $|S'| = k'$. Then we can choose $S \subseteq V(G)$ by:

1. For all $v' \in S'$ that corresponds to some $v \in V(G)$, add v to S if v is not isolated in G.
2. For all $w_{u'v'} \in S'$, add the vertex u in G corresponding to u' to S.

Since S' is a dominating set, it must contain all isolated vertices in G'. Since no isolated vertices are added to S, we have that $|S| \leq |S'| - m = k + m - m = k$, and we claim that S is a vertex cover of G. To see this, let uv be an edge in G. We need to show that either u or v is in S. Suppose v is not in S. Since v is not isolated (it has neighbor u), we have that v' is not in S' (or we would have added it in step 1). Also, since S' is a dominating set, $w_{u'v'}$ must be adjacent to some node in S'. But since $w_{u'v'}$ has only u' and v' as neighbors, and $v' \notin S'$, we must have that $u' \in S'$. So we must have added u to S in step 1 (since it is not isolated), and thus $u \in S$. So S is a vertex cover with $|S| \leq K$. Therefore, $(**)$ and $(*)$ hold. $\square$

Problem 1 Suppose $\mathsf{P} = \mathsf{NP}$. Show that there exists a polynomial-time algorithm A that on input $\langle G \rangle$ outputs the maximum size of any clique in G. [Note that if $\mathsf{P} = \mathsf{NP}$, then there exists a polynomial-time decider for the $CLIQUE$ decision problem.]

We can define

$A :=$ "On input $\langle G \rangle$:
1. Set $k = 0$.
2. While $\langle G, k+1 \rangle \in CLIQUE$, set $k := k + 1$.
3. Output k."

Since there exists a polynomial time decider for $CLIQUE$, and the sum of polynomials is a polynomial, step 2 runs in polynomial time. We have that A will run until it cannot find a clique of a certain size. Then, it will not increment, and the largest k for which $\langle G, k \rangle \in CLIQUE$ will be output. This is the maximum size of any clique in G, so A is correct. □

Problem 2 Suppose $\mathsf{P} = \mathsf{NP}$. Show that there exists a polynomial-time algorithm B that on input $\langle G, k \rangle$ outputs a clique of G with k many vertices, if there is one; otherwise B outputs "none."

We can define

$B :=$ "On input $\langle G, k \rangle$:
1. If A on $\langle G \rangle$ outputs a number less than k, output "none".
2. Set $G' := G$. Let $v_1, v_2, \ldots, v_n$ be the vertices of G', and let $i := 1$.
3. If running A on $\langle G' \rangle$ outputs a number greater than or equal to k after removing v_i, remove v_i from G'. Then, increment i and repeat until $i = n$.
4. Output $V(G')$."

Since A runs in polynomial time, and the product of polynomials is a polynomial, B runs in polynomial time. It is relatively clear that it is correct, so B is a polynomial-time algorithm for outputting a clique of G with k many vertices. □

Problem 3 Suppose $\mathsf{P} = \mathsf{NP}$. Show that there exists a polynomial-time algorithm C that on input $\langle G \rangle$ outputs a maximum-size clique in G. [Hint: Combine the algorithms of the last problem of the last two problems.]

We can define

$C :=$ "On input $\langle G \rangle$:
1. Let k be the output of A on $\langle G \rangle$.
2. Output the clique output by B on $\langle G, k \rangle$."

This is clearly in polynomial time since A and B are, and it is clearly correct. □