CSC2401 - Introduction to Computational Complexity
Lecture #11, Thursday October 18 2001
Lecturer: Allan Borodin
Scribe: Denilson Barbosa

1 Why 3CNF SAT? (cont'd)

In previous lectures we saw that kCNF-SAT $\leq_{\mathsf{log-space}}$ 3CNF-SAT, for $k > 3$. Now we show that 2CNF-SAT$\in NL$.

Let F be an instance of 2CNF-SAT. F is a formula of the form $C_1 \wedge \cdots \wedge C_n$, where $C_i = (y_i \vee z_i)$. Define a graph $\hat{G}_F = (V, E)$ as follows. The vertices of $\hat{G}_F$ are the variables in F and their negations, i.e., $V = \{x_i, \bar{x}_i : x_i \text{ is a variable in } F\}$; and there are edges (y_i, z_i) and $(\bar{z}_i, \bar{v}_i)$ whenever there is a clause $(\bar{y}_i \vee z_i)$. Intuitively, the edges of $\hat{G}_F$ capture the logical implications of F.

Claim: *F is unsatisfiable **iff** there is a variable x in F such that there are paths from x to $\bar{x}$ and from $\bar{x}$ to x in $\hat{G}_F$.*

Proof: Suppose such paths exist and F can still be satisfied by some truth assignment T, and suppose $T(x) = \mathbf{true}$ (thus, $T(\bar{x}) = \mathbf{false}$). Since there is a path from x to $\bar{x}$ in $\hat{G}_F$, then there must exist an edge (α, β) in the graph such that $T(\alpha) = \mathbf{true}$ and $T(\beta) = \mathbf{false}$. However, (α, β) is an edge implies that $(\bar{\alpha} \vee \beta)$ is a clause in F; contradiction. A similar argument applies when $T(x) = \mathbf{false}$.

Now, suppose there is no variable x for which the paths described above exist. We construct a truth assignment T that satisfies F (i.e., a truth assignment such that no edge in $\hat{G}_F$ goes from **true** to **false**). Initially, all nodes in $\hat{G}_F$ have no truth value assigned to them. The following procedure is repeated until all variables are assigned a truth value: pick a node α whose truth value has not yet been assigned, such that there is no path from α to $\bar{\alpha}$. Such a node must exist by our assumption. Set $\alpha = \mathbf{true}$ and for every node β reachable by a directed path set $\beta = \mathbf{true}$. This is the satisfying truth assignment. The only problem is that it might seem that we could have already set some $\bar{\beta} = \mathbf{true}$ and hence cant set $\beta = \mathbf{true}$. But if $\bar{\beta} = \mathbf{true}$ then there was some path from some γ to $\bar{\beta}$ for which we previously set $\gamma = \mathbf{true}$. But since there is a path from α to β, there must also be a path from $\bar{\beta}$ to $\bar{\alpha}$ sinc ewhenever there is an edge (u, v) the contrapositive edge $(\bar{u}, \bar{v})$ also occurs. This there is a path from γ to $\bar{\alpha}$ and $\bar{\alpha}$ would have already been set. □

Corollary: *2CNF-SAT is in NL, and thus in P.*

Proof: We know that REACHABILITY $\in NL$, thus we can recognize *unsatisfiable* 2CNF expressions in NL. By the Immerman-Szelepscényi Theorem (NL is closed under complement), 2CNF-SAT $\in NL$. □

2 More NP-complete problems

Now we show more problems to be NP-complete by giving interesting and/or amusing reductions:

2.1 VC $\leq_{\mathsf{log-space}}$ IS $\leq_{\mathsf{log-space}}$ VC

Vertex Cover (VC): given a graph $G = (E, V)$ and an integer index $k \geq 0$, $(G, k) \in$ VC if G has a vertex cover of size at most k; i.e., there exists $V' \subseteq V$, $|V'| \leq k$ such that each $e \in E$ has at least one endpoint in V'.

Independent Set (IS): given a graph $G = (E, V)$ and an integer index $k \geq 0$, $(G, k) \in$ IS if G has an independent set of size k; i.e., there exists $V' \subset V$ of size k such that for any two vertices $u, v \in V', (u, v) \notin E$.

Claim: *$VC \leq_{\mathsf{log-space}} IS \leq_{\mathsf{log-space}} VC$.*

Proof: It is easy to see from the definitions of VC and IS that $(G, k) \in VC$ **iff** $(G, |V| - k) \in IS$. $\square$

2.2 IS $\leq_{\mathsf{log-space}}$ CLIQUE $\leq_{\mathsf{log-space}}$ IS

CLIQUE: given a graph $G = (E, V)$ and an integer index $k \geq 0$, $(G, k) \in$ IS if G has a clique of size at least k; i.e., there exists $V' \subseteq V$, $|V'| \geq k$ such that for every two vertices $u, v \in V', (u, v) \in E$.

Claim: *$IS \leq_{\mathsf{log-space}} CLIQUE \leq_{\mathsf{log-space}} IS$.*

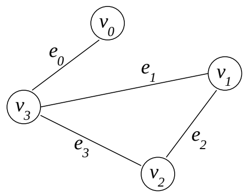
Proof: It is easy to see from the definitions of IS and CLIQUE that $(G, k) \in IS$ **iff** $(\bar{G}, k) \in CLIQUE$; where $\bar{G}$ is the complement of graph $G = (V, E)$: $\bar{G} = (V, \bar{E})$, where $\bar{E} = \{(u, v) : u, v \in V \text{ and } (u, v) \notin E\}$. $\square$

The three problems stated above are intricately connected as follows:

Lemma 1 *For any graph $G = (V, E)$ and subset $V' \subseteq V$, the following statements are equivalent:*

- *V' is a vertex cover for G;*
- *$V - V'$ is an independent set for G;*
- *$V - V'$ is a clique for the complement of G.*

It is easy to see that these three problems are in NP. Thus, by showing any of them to be NP-hard we show that all of them are NP-complete. The textbook shows that VC is NP-complete by reducing 3CNF-SAT to it. [1] shows that IS is NP-complete by reducing SAT to it.



(a)

	e_3	e_2	e_1	e_0
v_0	0	0	0	1
v_1	0	1	1	0
v_2	1	1	0	0
v_3	1	0	1	1

(b)

			e_3	e_2	e_1	e_0		decimal
x_0	=	1	0	0	0	1	=	257
x_1	=	1	0	1	1	0	=	276
x_2	=	1	1	1	0	0	=	336
x_3	=	1	1	0	1	1	=	325
y_0	=	0	0	0	0	1	=	1
y_1	=	0	0	0	1	0	=	4
y_2	=	0	0	1	0	0	=	16
y_3	=	0	1	0	0	0	=	64
t	=	2	2	2	2	2	=	682

(c)

Figure 1: The reduction of vertex cover problem to subset sum. (a) an undirected graph G; note that $V' = \{v_1, v_3\}$ is a vertex cover of size $k = 2$ for G. (b) the incidence matrix of G. (c) the corresponding subset sum instance for the vertex cover problem.

2.3 VC $\leq_{\mathsf{log-space}}$ SS

Subset Sum (SS): given a finite set S of natural numbers $\{w_1, \ldots w_n\}$ and a natural number t, $(S, t) \in$ SS if there exists $S' \subseteq S$ such that $t = \sum_{w_i \in S'} w_i$.

To show that SS $\in$ NP, note that S' can be used as a "certificate" for checking (in polynomial time) that $(S, t) \in$ SS. To show that SS is NP-hard, thus NP-complete, we reduce VC to it.

Claim: *VC $\leq_{\mathsf{log-space}}$ SS.*

Proof: We will give a reduction algorithm that constructs S, t such that $(G, k) \in$ VC **iff** $(S, t) \in$ SS. Note: this proof works for a slightly modified version of VC, in which we require the vertex cover to be exactly of length k.

Let $G = (V, E)$ be an undirected graph. The *incidence matrix* of G is a $|V| \times |E|$ matrix $B = (b_{ij})$ such that $b_{ij} = 1$, if edge e_j is incident on vertex

v_i, or 0, otherwise. Figure 1 shows how the instance of subset sum is derived. The set S contains two "kinds" of numbers, corresponding to edges and from vertices of G, computed as follows. For each vertex $v_i \in V$, we define

$$x_i = 4^{|E|} + \sum_{j=0}^{|E|-1} b_{ij} 4^j.$$

And for each edge e_j we define $y_j = 4^j$. Now, let $S = \{x_1, \ldots, x_{|V|}\} \cup \{y_1, \ldots, y_{|E|}\}$. Finally, define

$$t = k4^{|E|} + \sum_{j=0}^{|E|-1} 2 \cdot 4^j.$$

Suppose G has a vertex cover $V' \subseteq V$ of size k: $V' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$. Let $S' = \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\} \cup \{y_j : e_j \text{ is incident on precisely one vertex in } V'\}$. In our example $V' = \{v_1, v_3\}$ and $S' = \{x_1, x_3, y_0, y_2, y_3\}$.

To see why $t = \sum_{v \in S'} v$, note that summing the k leading 1's of $x_{i_j} \in S'$ gives the factor $k4^{|E|}$ in t. Now, for low order digits of t (which are all 2), consider each digit position j, which in turn correspond to an edge e_j. Since V' is a vertex cover, then e_j is incident in at least one vertex in V', thus, there exists at least one $x_{i_m} \in S'$ such that its jth bit is 1. Now, if edge e_j is incident to two vertices in V' then both "contribute" with a 1 in the jth position (i.e., we get a $2 \cdot 4^j$ factor in t). If e_j is incident in only one vertex in V' we add y_j to S', thus adding another 1 to the jth factor of t. Thus, S', t is a solution to the subset sum problem.

To prove the other direction of the implication, suppose there is a subset $S' \subseteq S$ that sums to t. Let $S' = \{x_{i_1}, x_{i_2}, \ldots, x_{i_m}\} \cup \{y_{j_1}, y_{j_2}, \ldots, y_{j_p}\}$. We now show that $m = k$ and $V' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$. To see this, note that for each edge e_j there are three 1's in the set S in the jth position: one from each of the vertices that e_j connects and another from y_j[1]. Since for each of the lower bits of t, at least one and at most two x_i contribute to the sum for each edge, we have that V' is a vertex cover. Now, observe also that the only way to account for the $k4^{|E|}$ factor in t is to have $m = k$. Thus, we have that V' is a vertex cover of size k. $\square$

2.4 In passing: HC $\leq_{\mathsf{log-space}}$ TSP

HAMILTONIAN CIRCUIT (HC): given a graph $G = (E, V)$, $G \in$ HC if G has a Hamiltonian circuit; i.e., there exists an ordering $\langle v_1, \ldots, v_n \rangle$ of the vertices of G, where $n = |V|$, such that $(v_n, v_1) \in E$ and $(v_i, v_{i+1}) \in E$.

TRAVELING SALESMAN (TSP): given a set C of m cities, distance $d(c_i, c_j) \in Z^+$ for each pair of cities $c_i, c_j \in C$ and a positive integer k, $(C, d, k) \in TSP$ if

[1]Note that since we are using a base-4 representation of the numbers, there is no carries from position j to position $j + 1$.

there exists a "tour" of C with length k or less; i.e., if there exists a permutation $\langle c_{\pi(1)}, \ldots, c_{\pi(m)} \rangle$ of C such that

$$\left(\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)}) \right) + d(c_{\pi(m)}, c_{\pi(1)}) \leq k.$$

Claim: *HC* $\leq_{\mathsf{log-space}}$ *TSP.*

Proof: To determine whether a graph $G = (V, E) \in$ HC, generate an instance of of TSP as follows. Define $C = V$, and define $d(c_i, c_j) = 1$ if $(c_i, c_j) \in E$ or 2 otherwise. It is easy to see that G has a Hamiltonian circuit **iff** there is a tour of C with cost $k = n + 1, n = |V|$. □

The textbook shows that HC is NP-complete, by reducing SAT to it. Hence, TSP is also NP-complete.

3 Pseudopolynomial Algorithms and Strong NP-completeness

The notion of NP-completeness in the strong and in the weak sense are somewhat easy to understand and hard to formalize. We will get back to this discussion in the future. The key point here is to observe what happens to a NP-complete problem when we change the encoding of the inputs to the problem. This issue is relevant only for problems involving potentially big numbers (such as SS, Knapsack, Bin Packing, Partition, etc.) encoded in unary.

L is NP-hard in the strong sense (strongly NP-hard) if L remains *NP*-hard even if any instance of length n is restricted to contain integers of size at most $p(n)$; p a polynomial. Alternatively, L remains *NP*-hard if all integers are encoded in unary.

Claim: *TSP is strongly NP-hard.*

Claim: *If P≠NP SS is not strongly NP-hard.*

We add the proviso that P≠NP since otherwise every NP problem is NP-complete (in any sense).

References

[1] Christos Papadimitriou. *Computational Complexity.* Addison Wesley, 1995.