CS180 Homework 9

Zhehao Wang 404380075 (Dis 1B)

May 31, 2016

1 2-SAT polynomial

The polynomial algorithm is shown by combinating the answers to the questions (a) to (f) below.

(a) $x_1 \vee x_2$ is true $\implies$ x_1 is true or x_2 is true. x_1 is true $\implies$ $(\neg x_1 \implies x_2)$ is true; x_2 is true $\implies$ $(\neg x_2 \implies x_1)$ is true. So $(x_1 \vee x_2) \implies (\neg x_1 \implies x_2) \vee (\neg x_2 \implies x_1)$.

$(\neg x_1 \implies x_2) \vee (\neg x_2 \implies x_1)$ is true when (x_1, x_2) takes the combinations (true, false), (true, true) or (false, true). Under all of these combinatinons $(x_1 \vee x_2)$ is true. So $(\neg x_1 \implies x_2) \vee (\neg x_2 \implies x_1)$ $\implies$ $(x_1 \vee x_2)$. And to summarize, $(x_1 \vee x_2)$ iff $(\neg x_1 \implies x_2) \vee (\neg x_2 \implies x_1)$

(b) If the formula is evaluated for true, then $(p \implies q)$ and $(q \implies r)$ are both true. Since p is true, q is true, and r is also true. The conclusion holds

(c) For each literal x_i in φ, two nodes in the implication graph G correspond each with x_i and $\neg x_i$. For each phrase of $(x_i \vee x_j)$, turn them into the form in (a): $(\neg x_i \implies x_j) \vee (\neg x_j \implies x_i)$. And for each $(x_m \implies x_n)$, add an edge from x_m to x_n, and we have the implication graph.

(d) ($\implies$) We want to prove that if φ is satisfiable, $\forall x_i$, x_i and $\neg x_i$ are not in the same SCC.

Proof by contradiction: assume in $G = (V, E)$ $\exists$ some m s.t. x_m and $\neg x_m$ are in one SCC (strongly connected component), then $\exists$ some $x_{k_1} ... x_{k_n}$ s.t. $(x_m \implies x_{k_1}) \wedge (x_{k_1} \implies x_{k_2}) \wedge ... (x_{k_n} \implies \neg x_m)$. According to the conclusion of (b), $x_m \implies \neg x_m$, and similarly $\neg x_m \implies x_m$. These two needs to hold at the same time for φ to be satisfiable, and we've a contradiction. Thus if φ is satisfiable, $\forall x_i$, x_i and $\neg x_i$ are not in the same SCC.

($\impliedby$) We want to prove that if $\forall x_i$, x_i and $\neg x_i$ are not in the same SCC, φ is satisfiable.

Proof by finding the assignment: we can find a set of trees $t_1 ... t_m$ by each time selecting a node x_i as the root of the tree, and add all nodes reachable from x_i to the tree as long as it's not $\neg x_i$, and if $\neg x_i$ is reachable, we build the tree from $\neg x_i$ instead of x_i. We remove all the nodes and their negation involved in the tree, and repeat this process until all nodes are covered by the set of trees. For each tree, assign true or false according to if x_i or $\neg x_i$'s in the tree, and φ can be satisfied. We want to show that (1) such a set of tree exists, and (2) such assignment is correct.

For (1), assume that the set of trees does not exist, meaning some node x_i and $\neg x_i$ cannot be covered. According to the description of building the tree, x_i cannot be covered because exists a path going from x_i that goes through $\neg x_i$, similarly for $\neg x_i$, there exists a path from $\neg x_i$ that goes through x_i. This contradicts with their not being in the same SCC.

For (2), the correctness is obvious as this set of trees covers the phrases of φ expressed in implication form. And according to (a), each phrase in the original φ in CNF form can hold. Thus φ is satisfiable.

With both directions we proved that the conclusion holds.

(e) Tarjan's algorithm in hw3.1 can be used to build all the SCCs in $O(|E|)$, and a scan through all the SCC would give if any x_i and $\neg x_i$ are in the same SCC.

(f) The assignment is given as part of proof ($\Longleftarrow$ direction) in (d). This algorithm is polynomial as the transformation to implication graph is $O(n)$, and finding the assignment takes no more time than doing a BFS from each node, which is $O(|E||V|)$, and $O(n^3)$ where n is the number of literals.

2 Longest and shortest simple path reduction from Hamiltonian Path

(a) Reduce Hamiltonian Path to Longest Simple Path (LSP). Given a graph $G = (V, E)$, we contruct the input for $LSP(G', k')$ by having $G' = G$ with all the weights in G' set to 1, and $k = |V - 1|$.

This reduction is obviously polynomial. The reduction holds as (1) if a Hamiltonian Path p exists in G, length of $p \geq |V - 1|$ and LSP returns true; and (2) if LSP returns true, a simple path p of length at least $|V - 1|$ exists, p is a Hamiltonian path as $|V|$ is the total number of nodes and p is a simple path.

(b) Reduce Hamiltonian Path to Shortest Simple Path with negative weights (SSP). Given a graph $G = (V, E)$, we contruct the input for $LSP(G', k')$ by having $G' = G$ with all the weights in G' set to -1, and $k = -|V - 1|$. This reduction is obviously polynomial.

The reduction is correct as (1) if a Hamiltonian Path p exists in G, length of $p \leq -|V - 1|$ and SSP returns true; and (2) if SSP returns true, a simple path p of length at most $-|V - 1|$ exists, p is a Hamiltonian path as $|V|$ is the total number of nodes and p is a simple path.

3 Polynomial reduction between set problems

(a) Reduce Hitting-Set (HS) to Dominating-Set (DS): given sets $s_1...s_n$, for each element $e \in E = s_1 \bigcup ... \bigcup s_n$, we create a node and connect it with all other nodes in E. For each set s_i, we add a node s_i and connects it with all the nodes e that satisfies $e \in s_i$. We have a reduction by giving this graph G' and $k' = k$ to DS. This reduction is obviously polynomial.

The reduction is correct as (1) if HS has a solution $S = e_1...e_m$ $m \leq k$, S has at least one element from all sets. Then the nodes $e_1...e_m$ would dominate all the s_i nodes as well as all the e_i nodes. Thus S forms a DS solution of size $\leq k'$. (2) if DS has a solution S and $|S| \leq k'$. If S contains some s_i nodes, we form S' by picking one of e_i nodes that each s_i in S connects to. Given how G' is constructed, S' is still a DS in G' satisfying $|S'| \leq |S| \leq k'$. Picking values corresponding with nodes in S' would obviously give a solution for HS whose size is $\leq k$.

(b) Reduce DS to HS: given a graph $G = (V, E)$, for each $v_i \in V$, we assign a value e_i to it and build a set $s_i = \{e_i\} \bigcup \{\text{all nodes } e_j \text{ connected to } e \text{ in } G\}$. We have a reduction by passing all the s_i sets and $k' = k$ to HS. This reduction is obviously polynomial in $|V|$ (at most $|V|^3$).

The reduction is correct as (1) if DS has a solution $S = v_1...v_m$ and $|S| \leq k$, then combine each e_i corresponding with each v_i would be a solution for HS, and its size $\leq k'$ (if some set in HS does not have a representative, its "center" node must not be dominated). (2) if HS has a solution $S = e_1...e_m$ $m \leq k'$, then combine each v_i corresponding with each e_i would be a solution for DS, and its size $\leq k$ (if some node e in G is not dominated, the set centered at e in HS must not have a representative).

(c) Reduce Set-Cover (SC) to DS: given sets $M = s_1...s_n$, for each element in $E = s_1 \bigcup ... \bigcup s_n$, we create a node. For each set s_i, we add a node s_i, connect it with all other $s \in M$, and connect it with all the nodes e that satisfies $e \in s_i$. We have a reduction by giving this graph G' and $k' = k$ to DS. This reduction is obviously polynomial.

The reduction is correct as (1) if SC has a solution $S = s_1...s_m$ $m \leq k$, S has all the elements in E. Then the nodes $s_1...s_m$ would dominate all the s_i nodes as well as all the e_i nodes. Thus S forms a

DS solution of size $\leq k'$. (2) if DS has a solution S and $|S| \leq k'$. If S contains some e_i nodes, we form S' by picking one of s_i nodes that each e_i in S connects to. Given how G' is constructed, S' is still a DS in G' satisfying $|S'| \leq |S| \leq k'$. Picking sets corresponding with nodes in S' would obviously give a solution for SC whose size is $\leq k$.

(d) Reduce DS to SC: given a graph $G = (V, E)$, for each $v_i \in V$, we assign a value e_i to it and build a set $s_i = \{e_i\} \bigcup \{\text{all nodes } e_j \text{ connected to } e \text{ in } G\}$. We have a reduction by passing all the s_i sets and $k' = k$ to SC. This reduction is obviously polynomial in $|V|$ (at most $|V|^3$).

The reduction is correct as (1) if DS has a solution $S = v_1...v_m$ and $|S| \leq k$, then combine each s_i set "centered at" each v_i would be a solution for SC, and its size $\leq k'$ (if some value e is not covered by SC, then all the sets centered at e's neighbors in G are not picked and those nodes are not in S, thus e is not dominated). (2) if SC has a solution $S = s_1...s_m$ $m \leq k'$, then combine each v_i which is the center of each s_i would be a solution for DS, and its size $\leq k$ (if some node e in G is not dominated, then all the sets centered at e's neighbors in G are not in S, thus e is not covered by S).

4 Degree bounded spanning tree NP-hardness

Reduce Hamilton path to degree bounded spanning tree (DST). Given a graph G, pass the same $G' = G$ and $k = 2$ to DST. This is a polynomial reduction that reduces Hamilton path problem to DST.

The reduction is correct since (1) if there is a Hamilton path p in G, p is obviously also a spanning tree that can be found by $DST(G', 2)$. (2) if there's a spanning tree t found by $DST(G', 2)$, t is obviously also a Hamilton path in G.

Thus DST is NP-hard.