CS 4820, Spring 2014 **Homework 7, Problem 1**

Name: Piyush Maheshwari
NetID: pm489
Collaborators: None

Hand in your solutions electronically using CMS. Each solution should be submitted as a separate file. For multi-part problems, all parts of the solution to that problem should be included in a single file.

Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time. The running time must be bounded by a polynomial function of the input size.

The following problems ask you to design "self-contained polynomial-time reductions." You should interpret that term as follows. If* a *and* b *are two decision problems, a self-contained polynomial-time reduction from* a *to* b *is a polynomial-time algorithm that takes an instance of* a *and outputs an instance of* b*. To show that the reduction is correct, you must prove two things. First, that it transforms* yes *instances of* a *to* yes *instances of* b*. Second, that it transforms* no *instances of* a *to* no *instances of* b*.

(1) *(10 points)* In class, we stated without proof the important fact that 3SAT is NP-hard, which means that every problem in NP reduces to 3SAT in polynomial time.

To gain some intuition about how these reductions work, this exercise asks you to give a self-contained polynomial-time reduction from INDEPENDENT SET to 3SAT.

Solution

To solve this first we will reduce INDEPENDENT SET to VERTEX COVER, then VERTEX COVER to SAT and then reduce SAT to 3SAT.

Reduction from Independent set to Vertex Cover This reduction was covered in class and we will assume its correctness from class.

Reduction from Vertex Cover to SAT

The input to vertex cover is the following - graph G = (V,E) and an integer k. First lets transform this into an input to a SAT instance.

Let the variables of the SAT instance be $x_{i,v} | \forall v \in V, 1 \leq i \leq k$ where $x_{i,v}$ denotes that node v is the i^{th} element in the vertex cover.

Let us define an intermediate operator :

at_least_one$\{l_1, l_2, ...l_x\} = (l_1 \vee l_2 ... \vee l_x)$

We can write the equivalent SAT expression as follows F =

$$\bigwedge_{(u,v)\in E} \text{at\_least\_one}\{\ x_{i,u} \vee x_{i,v} | 1 \leq i \leq k\}$$
$$\wedge \bigwedge_{1\leq i\leq k} \bigwedge_{u\in V, v\in V, u\neq v} (\neg x_{i,u} \vee \neg x_{iv})$$
$$\wedge \bigwedge_{1\leq i\leq k} \text{at\_least\_one}\{\ x_{i,u} | u \in V\}$$

The first part of the expression makes sure that we select at least one vertex for each edge for any value of k. The second clause makes sure that we don't end up having the 2 nodes for the same value of k. The third part of the clause makes sure that we choose a node for every value of k and so we end up with at most k nodes.

Now we prove that this is valid reduction

Proof. We claim that a YES instance of SAT implies a YES instance of vertex cover. We can see that if the SAT instance gives the answer Yes, then we know that for each value of edge u-v, we have atleast one of x_{iv} or x_{iu} set to one which means that we cover this edge. This is by construction of the SAT expression as explained above. Also by the second and the third clauses we are picking up atmost k elements since we don't allow multiple nodes are a single value of k and we have a valid assignment for each value of k. Hence if we have a satisfying assignment for SAT, then the vertex cover is the set of all v's for which x_{iv} is set to true for some value of k.

Now we show that a yes instance of vertex cover implies a yes instance of SAT. If we have a yes instance of the vertex cover, then we can just pick elements from the vertex cover and assign them subscripts from 1 to k to generate a set of variables of the form $x_{i,v}$. Now we know that these variable satisfy the first clause since nodes in vertex cover cover all edges. Since we have atmost k nodes in the vertex cover and have assigned every value of k a node, clauses two and third will also be satisfied. Hence if we have the set of vertices which cover the edges, we can have a satisfying assignment for this SAT instance.

Now we show that the algorithm for converting from an instance of vertex cover to SAT takes polynomial time. We can just show this by proving that our SAT clause have polynomial terms. We can see that the first set of clauses have terms equal to $2 * |E| * k$. The second set of clauses have terms equal to $(2 * |V|^2) * k$. The third set of clause have term equal to $k * |V|$. The combined sum of terms is hence polynomial in $|V|$, $|E|$ and k.

Hence the reduction is complete. □

Reduction from sat to 3sat

A general SAT expression can have expressions of any length. We will show how to convert the following length expressions into equivalent 3SAT expressions

1. Clauses of length one : (x_1) - Add two additional variables z_1 and z_2. Then add theses clauses $(x_1 \vee z_1 \vee x_2) \wedge (x_1 \vee \neg z_1 \vee x_2) \wedge (x_1 \vee z_1 \vee \neg x_2) \wedge (x_1 \vee \neg z_1 \vee \neg x_2)$
2. Clauses of length two : $(x_1 \vee x_2)$ - Additional variable z . These can be replaced by $(x_1 \vee x_2 \vee z) \wedge (x_1 \vee x_2 \vee \neg z)$
3. Clauses of length three : Do nothing
4. Clauses of length $n > 3$: $(x_1 \vee x_2 .. \vee x_n)$ - We will introduce extra variable z_1 to z_{n-3}

We can write such a clause of length n as the following clause :
$(x_1 \vee x_2 \vee z_1) \wedge (\neg z_1 \vee x_3 \vee z_2) \wedge (\neg z_2 \vee x_4 \vee z_3)$ .... $\wedge(\neg z_{n-3} \vee x_{n-1} \vee x_n)$

The conversion for $n > 3$ works because no matter whatever be the assignments of the variables z_1 to z_{n-3}, we can never have a satisfying clause without one of the x_i's to be true. This is because the there is a not operator on a single z variable in each consecutive clause.

Now we prove that this is valid reduction

Proof. First we show that a valid instance of 3SAT implies a valid instance of SAT. Note that if we assign the values of x_i in our SAT identical to the values of x_i in an yes instance of 3SAT, we would get a yes instance of SAT. This is by construction.

Now we show that a valid instance of the SAT will result in a valid instance of 3SAT that we have constructed. Note that for clauses of one a satisfying assignment in SAT would result in a satisfying assignment of 3SAT because no matter what values z_1 and z_2 take, the final value would depend upon the value of x_1. The same argument applies for n = 2. We have explained above by for n > 3, atleast one of the x's must be true to make the clause true. So a satisfying assignment for SAT would result in a satisfying assignment for 3SAT

Finally we show that this reduction takes polynomial time. We can show this by showing that the number of terms in the 3SAT expression that we have created is polynomial. For n = 1,2,3 the length of the 3SAT is constant number of terms more than the corresponding SAT clauses. For $n > 3$, notice that the number of terms in 3SAT is atmost twice more than the number of terms in SAT. This shows that the number of terms in 3SAT would be remain polynomial in terms of the number of clauses in SAT.

This completes the proof. □

Hence by the transitivity of reductions we have shown that the reduction of INDEPENDENT SET to 3SAT