CSCE 629-601 Homework 11
28th November, 2016

Name: Adil Hamid Malla **UIN:** 425008306

Problem 1. Textbook page 110, Exercise 34-5.1.

Question: Sub Graph Isomorphism Problem - Given two graphs G and H, check whether the H is isomorphic to subgraph of G. Show that this decision problem is NP Complete.
Solution:
Since the question is a decision problem asking whether the graph H is isomorphic to the subgraph of G. To prove that this problem, which we will use as sub graph isomorphic problem no one is NP complete.
First we must know that what the problem is. Lets us assume that we have two graphs, $G = (V, E)$ and $H = (V', E')$. Now we have to check whether there exists a sub graph of G as $G_0 = (V_0, E_0)$ such that

$$V_0 \in V$$
$$E_0 \in E \text{ and } E_0 \cap (V_0 \times V_0)$$

Such that $G_0 \cong H$, i.e there exists an $f : V_0 \to V'$ such that $(u, v) \in E_0 \Leftrightarrow (f(u), f(v)) \in E'$.
To show that it is NP complete we will follow three steps:

1. **We need to prove that sub graph isomorphism is NP**.
   To prove that we need a certificate, and we should be able to say whether the given certificate gets a yes or no as our problem is a decision problem. To see that the problem is in NP, we observe that a certificate is a mapping ϕ from the nodes of G_0 (a subset of G) to the nodes of H, describing which vertices of H correspond to vertices of G_0. The certifier then needs to make sure that for each edge e = (u, v) in G_0, the edge (ϕ(u), ϕ(v)) is also in H, and whenever (u, v) is not an edge of G_0, then (ϕ(u), ϕ(v)) is not an edge of H. This can be done with two simple nested loops, and takes at most $O(n^2)$ time, i.e., polynomial. So this proves that we can say whether a given certificate is solution of our problem or not.

2. **We need to have a mapping from this problem to any known NP Complete problem.** The easiest problem we can think of is Reduction from the Clique Problem. Since we have already proved in the class that the Clique problem is NP Complete, then if we prove that we can reduce Clique problem to our sub graph isomorphism problem in polynomial time then we can prove the second step of showing that the given problem is NP complete. We can show that, for instance, the Clique problem is a special case. Given an instance of Clique, consisting of a graph G_{new} and a number k, we generate an instance of Subgraph Isomorphism by setting $G = G_{new}$, and H a clique on k vertices. This reduction clearly takes polynomial time, since all we do is copy a graph, and write down a complete graph in time $O(k^2)$. So thus the reduction of the given problem takes place in polynomial time.

3. **We need to prove that the reduction/mapping to Clique problem preserves the Yes and No answer.** We can prove that by getting a yes from both the sides. To prove correctness of the reduction, first assume that G contains a clique of size k. Then, those k nodes of G = G_0 are isomorphic to H, because H is a clique of size k. So this is equivalent to having the same number of edges in H as well as G_0 which is k-subset of G and also the correspondence is maintained. Conversely, if G contains a subgraph G_0 isomorphic to H, because H is a k-clique, G must contain a k-clique. Thus, we have the **Yes** preservance from both the sides.

Hence the sub graph isomorphism is a NP Complete Problem.

Problem 2. Textbook page 110, Exercise 34-5.2.

Question/Output: Need to prove the **NP-Completeness** of
0-1 Integer-Programming Problem and prove the existence of a n-vector x, such that A$x \leq$ b.
To Prove the **NP-Completeness** of ***Integer-Programming Prob.***, we need to prove that this problem is a **NP** problem and a "well-known" **NP-Complete** can be reduced to this problem in Polynomial time, such that they preserve the "YES" and "NO" answer for every instance of the "well-known" **NP-Complete** problem.

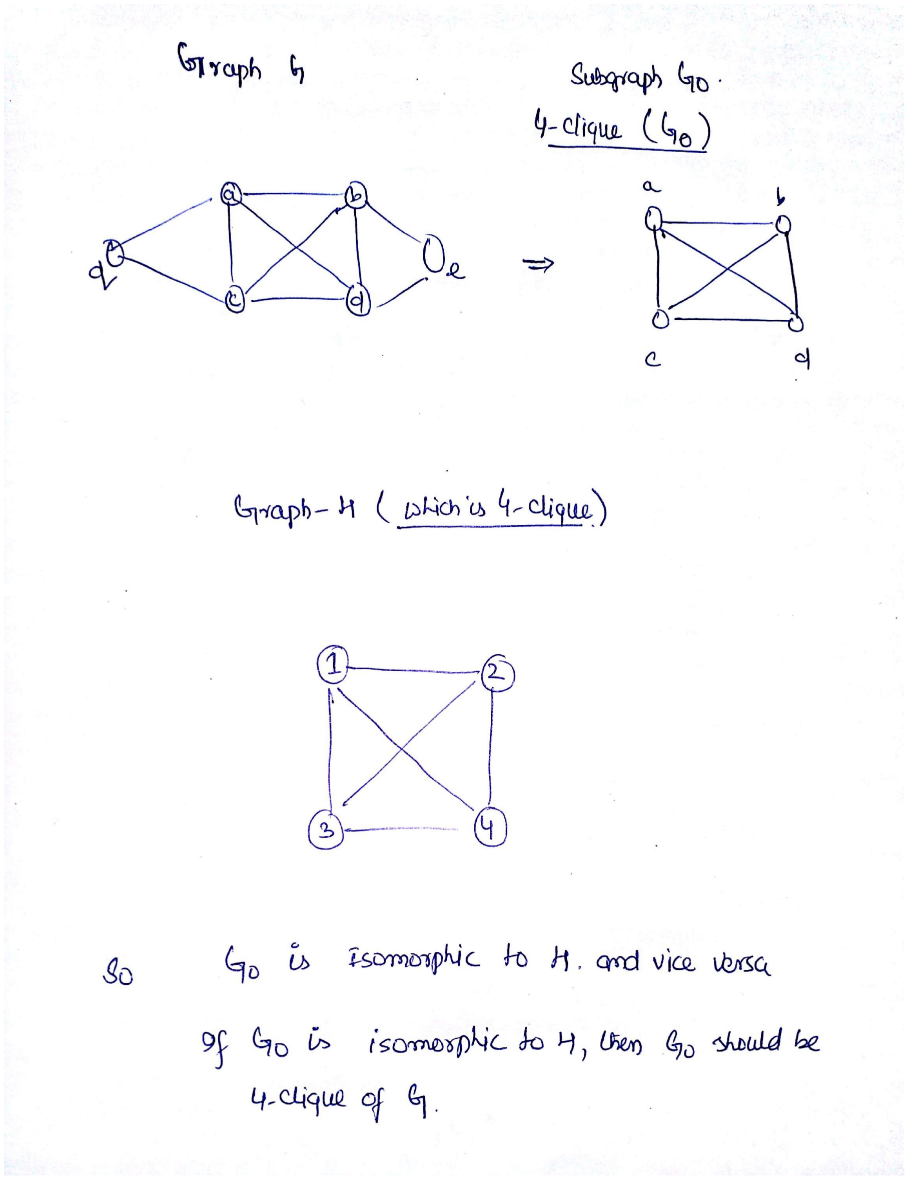
Graph G
Subgraph G0.
4-clique (G0)
a
b
c
d
e
⇒
Graph-H (which is 4-clique)
1
2
3
4
So G0 is Isomorphic to H. and vice versa
If G0 is isomorphic to H, then G0 should be 4-clique of G.

Figure 1: Example of the Sub Graph Isomorphism

Solution: Proving the **NP**-ness of the Problem:
Now Suppose that we are given with a n-vector x, to prove the relation: A$x \leq$ b, YES Certificate, it would just take Polynomial time to prove the correctness of the Solution as we have very few equations to check after the matrix multiplication. Now, For proving the Completeness, We would consider(As per Hint), the well-known 3-SAT problem and a reduction from 3-SAT problem to Integer-Programming Prob. would finish the proof.
Lets Consider a construction as such this: Let the number of clause be 'm' and the number of literals be 'n'. Now Placing the K-Clauses in the row and all the N literals on the column, We will get the matrix A_{m*n}.
Now this Problem can be modeled in the Line of a **Linear Programming Problem**, where we would have x_1,x_2,x_3,x_4,.... x_n in the 3-SAT problem, where as we would have z_1,z_2,z_3,z_4,.... z_n , a set of integers lying in the set {0,1}.
Assigning $z_i = 1$ in the integer program represents setting x_i=true in the formula, and $z_i = 0$ represents x_i = false.so for each Clause present in the 3-SAT problem we can frame a Linear Programming equation such as:

(x_1 or not(x_2) or x_3), will have constraint like this:
$$z_1 + (1\text{-}z_2) + z_3 > 0$$

Now this can be rephrased and written as:

$$z_1 - z_2 + z_3 \geq -1$$
which is same as:
$$-z_1 + z_2 - z_3 \leq 1$$

Now we can form a matrix in this manner, which will comprise of coefficient of z_i's(Using the LHS) and the RHS would give us the m-vector b.
The points to remember now for each expression row are:
a) Each has exactly one column of minimum value.
b) This column corresponds to a nonsatisfying truth assignment.
c) Every other column satisfies the expression.
d) All other columnms have higher values.
Here is how we build a matrix from a set of clauses. First let the columns of the matrix correspond to the variables from the clauses. The rows of the matrix represent the clauses - one row for each one. For each clause, put a 1 under each variable which is not complemented and a -1 under those that are. Fill in the rest of the row with zeros. Or we could say:

$$a_{ij} = \begin{cases} 1 \text{ if } v_j \in \mathit{clause} \text{ i} \\ -1 \text{ if} v'_j \in \mathit{clause} \text{ i} \\ 0 \text{ otherwise} \end{cases}$$

The vector b is merely made up of the appropriate minimum values plus one from the above chart. In other words:
$b_i = 1$ - (the number of complemented variables in clause i).
The above chart provides the needed ammunition for the proof that our construction is correct. The proper vector x is merely the truth assignment to the variables which satisfies all of the clauses. If there is such a truth assignment then each value in the vector Ax will indeed be greater than the minimum value in the appropriate chart column.
If a 0-1 valued vector x does exist such that $Ax \leq b$, then it from the chart we can easily see that it is a truth assignment for the variables which satisfies each and every clause. If not, then one of the values of the Ax vector will always be less than the corresponding value in b. This means that the that at least one clause is not satisfied for any truth assignment.
Here is a quick example. If we have the three clauses:

$$(x_1, x'_3, , x_4), (x'_2, x'_3, x_4), (x_1, x_2, x_3)$$

then according to the above algorithm we build A and b as in figure 1.

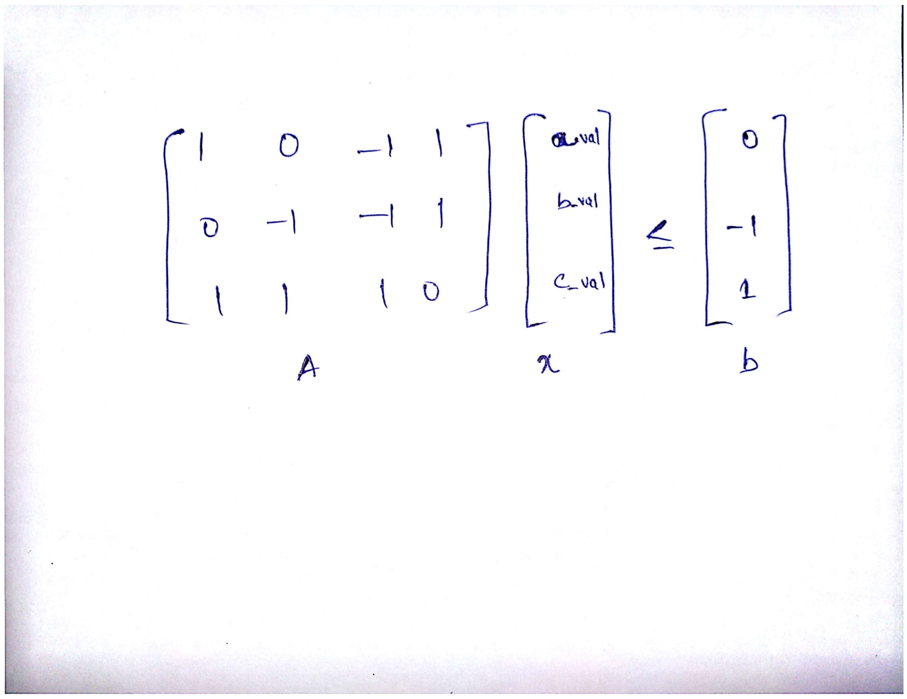
$$\underbrace{\begin{bmatrix} 1 & 0 & -1 & 1 \\ 0 & -1 & -1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a\_val \\ b\_val \\ c\_val \end{bmatrix}}_{x} \leq \underbrace{\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}}_{b}$$

Figure 2: Equivalent Matrix Form

Note that everything comes out fine if the proper values for the xi are put in place. If x3 is 0 then the first entry of Ax cannot come out less than 0 nor can the second ever be below -1. And if either x2 or x1 is 1 then the third entry will be at least 1.
Now we see that we only require to find a m-vector x such that:

$$\sum a_i * x_i \leq b_i$$

where a_i's are coefficient z_i's and are values present in the matrix A.So Using this example we are able to demonstrate the mapping between the 3-SAT problem and the 1-0 Integer Problem.

Correctness: Now For every such clause in the 3-SAT problem we can reduce it to a 1-0 Integer problem such that Ax $\leq$ b.Since a valid solution in 3-SAT problem, which means a YES, would always give us a value of x in set $\{0,1\}$, which inherently means YES in 1-0 Integer Problem and Vice-versa.
Since the mapping demonstrated above can be achieved in Polynomial Time, and since the mapping above is if and only if applicable, we can reaffirm 1-0 Integer Problem as NPC problem such that Ax $\leq$ b.