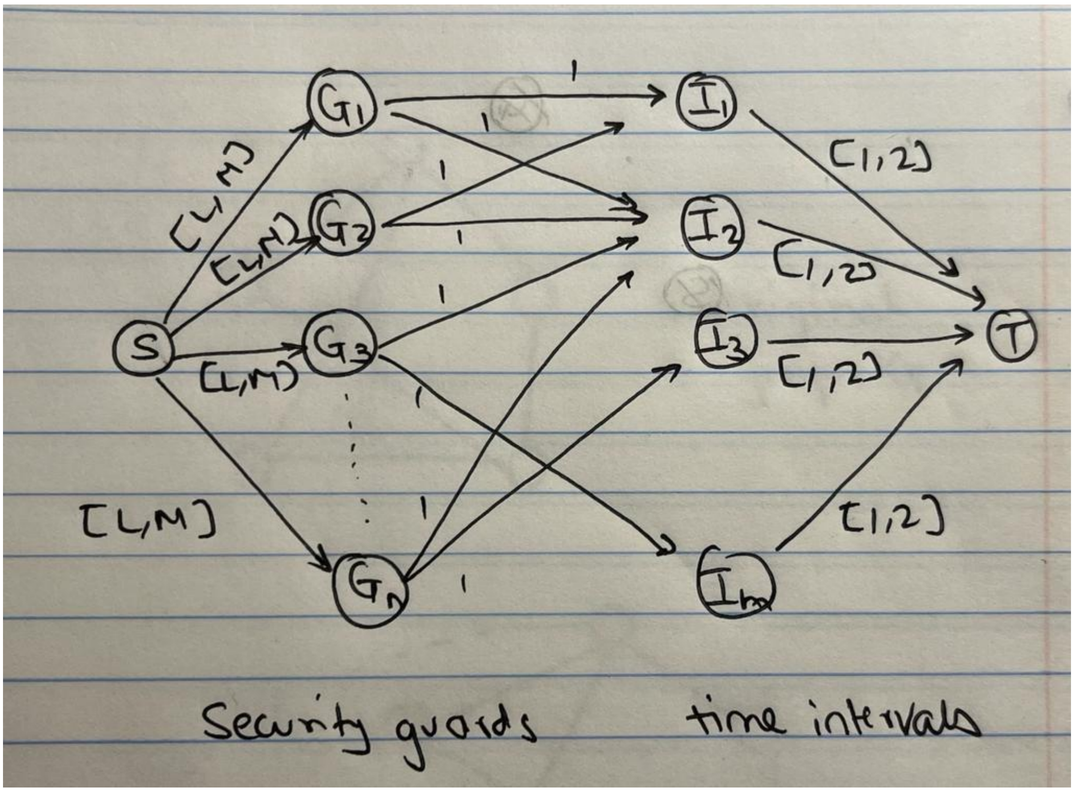
dnabar@usc.edu
Deven Parag Nabar
USC ID: 7229446568

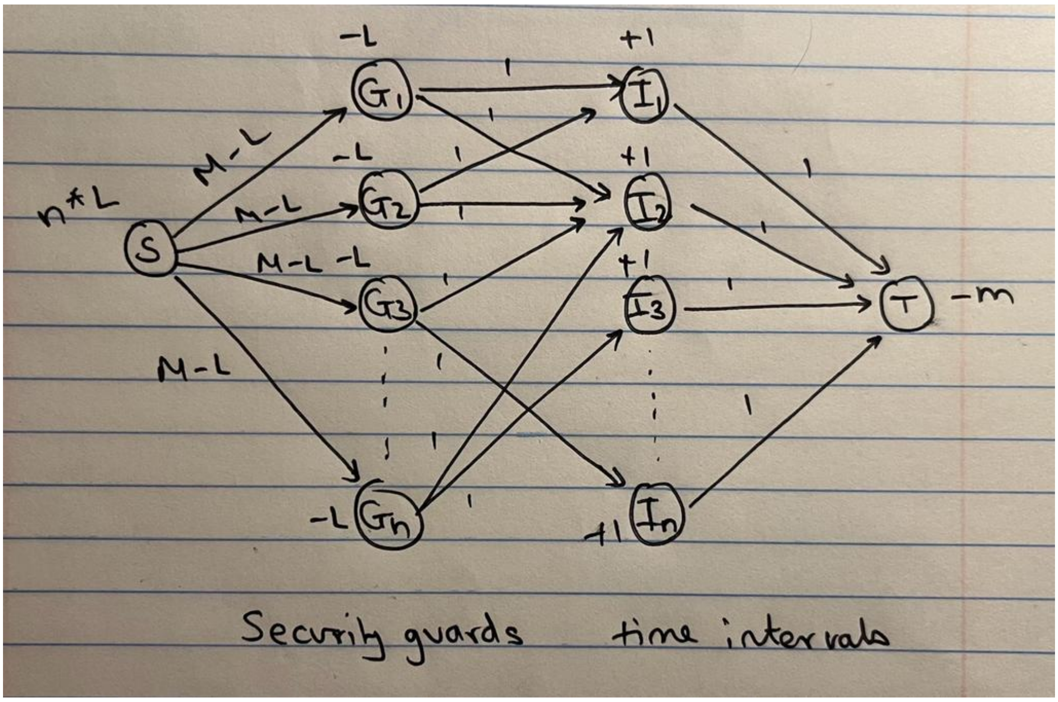
CSCI570 Homework 5

1. **SOLUTION:**
   - The problem can be solved by modelling it as a circulation problem with lower bounds.
   - We have a set of n nodes $G_1, G_2, \ldots, G_n$ representing the guards.
   - We have a set of m nodes representing time intervals $I_1, I_2, \ldots, I_n$.
   - We connect each guard to their preferred time intervals with a directed edge of capacity 1
   - We connect the source to every single guard with an edge of capacity M and lower bound L
   - We connect the time intervals to the target with edges of capacity of 2 and lower bound value 1

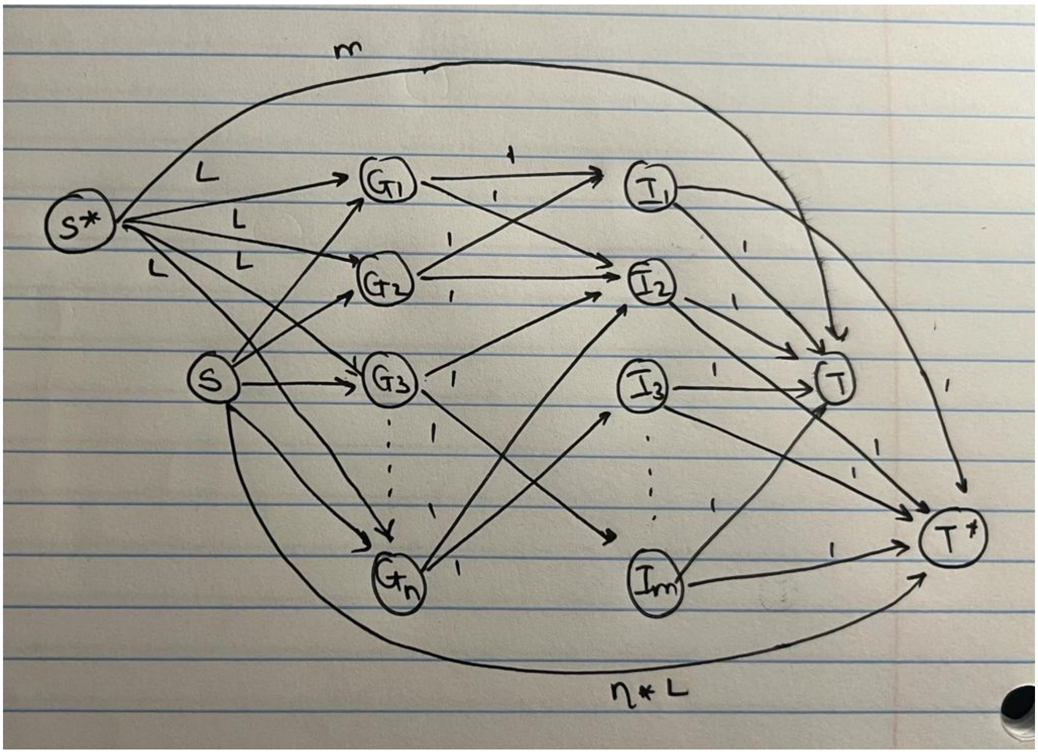


   - We now remove the lower bounds and recalculate the demands
   - The new graph for the same is as follow:

dnabar@usc.edu
Deven Parag Nabar
USC ID: 7229446568



- We remove the demands by creating a super source S* and a super sink T*
- We remove the demands in the graph and connect all the negative demands to S* and positive demands to the T*.
- We have thus reduced the original problem into a network flow problem.



- We can run Ford Fulkerson Algorithm on the above graph to find the max flow.

Claim:

- The original problem has a solution (a valid assignment is indeed possible) if and only if the constructed network has a max-flow = n*L +m

Proof:

==> If there is a feasible assignment, then the flow in the network is n*L + m. It implies that each guard is assigned to at least L time intervals which is the lower bound. Along with this, each interval has 1 or 2 guards. In this case the floe to the super sink T* has to be n * L + m.

<==) Conversely,

If the max flow is n * L + m, an assignment exists. If we traverse the network from each guard node to time interval having a non-zero flow capacities, the condition is satisfied. Because of this, each guard is assigned to at least L Time intervals and each interval has 1 or 2 guards.

dnabar@usc.edu
Deven Parag Nabar
USC ID: 7229446568

2. SOLUTION:

- Let x_{ij} be the units of produced of Product i at Manufacturing Plant j
- Now the goal is to maximize the profit

- The objective function for this problem becomes

  $$\text{Max } (10x_{11} + 8x_{12} + 6x_{13} + 9x_{14} + 18x_{21} + 20x_{22} + 15x_{23} + 17x_{24} + 15x_{31} + 16x_{32} + 13x_{33} + 17x_{34})$$

- Subject to:

- The constraints on time
  $5x_{11} + 6x_{21} + 13x_{31} <= 2100$
  $7x_{12} + 12x_{22} + 14x_{32} <= 2100$
  $4x_{13} + 8x_{23} + 9x_{33} <= 2100$
  $10x_{14} + 15x_{24} + 17x_{34} <= 2100$

- The constraints on number of products
  $x_{11} + x_{12} + x_{13} + x_{14} >= 100$
  $x_{21} + x_{22} + x_{23} + x_{24} >= 150$
  $x_{31} + x_{32} + x_{33} + x_{34} >= 100$

- The constraints on number of products at each plant individually
  $x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{23}, x_{24}, x_{31}, x_{32}, x_{33}, x_{34} >= 0$

dnabar@usc.edu

Deven Parag Nabar
USC ID: 7229446568

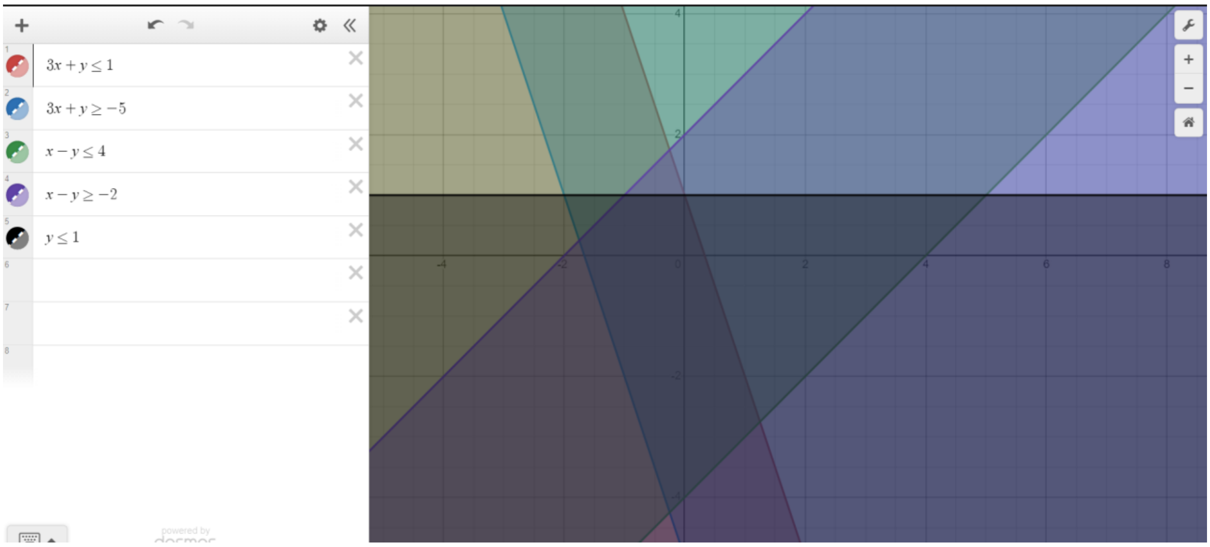
3. SOLUTION:

The given linear program is

$$\max(-x_1 + 4x_2)$$

subject to

$$\begin{aligned} 3x_1 + x_2 &\leq 1 \\ 3x_1 + x_2 &\geq -5 \\ x_1 - x_2 &\leq 4 \\ x_1 - x_2 &\geq -2 \\ x_2 &\leq 1 \end{aligned}$$



- The optimal solution is found at one of the vertices of the bounded polygon
- The max values are as follows:

x_1	x_2	Max (x_1, x_2)
-1	1	5
0	1	4
-1.75	0.25	2.75
-0.25	-4.25	-16.75
1.25	-2.75	-12.25

- **Hence the max value is 5 at (-1,1)**

dnabar@usc.edu
Deven Parag Nabar
USC ID: 7229446568

4. **SOLUTION**:
   - Let x_j be the units of j^{th} food. We have a total of n available foods.
   - We know that there are m ingredients and each ingredient i requires minimum of b_i units.
   - Each food item has a_{ij} units of i^{th} nutrient
   - Every j^{th} food item costs c_j per unit
   - The constraints to satisfy the nutrient requirement of the day are:
     $\mathbf{a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n >= b_1}$
     $\mathbf{a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n >= b_2}$
     .
     .
     .
     $\mathbf{a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n >= b_m}$

   - The constraints to satisfy the quantities of each food are
     $\mathbf{x_1 >= 0}$
     $\mathbf{x_2 >= 0}$
     …
     …
     …
     $\mathbf{x_n >= 0}$

   - The objective function is

     Min $\left(\sum_{j=1}^{n} c_j x_j\right)$

     Which in **standard form** is
     Max $\left(-\sum_{j=1}^{n} c_j x_j\right)$

5. **SOLUTION:**
    - In an undirected graph G = (V, E),
    - Let x_i be the binary variable for vertex i that indicates if that vertex will be included in the vertex cover.
    - If $x_i = 0$ then the vertex cover is not included in the vertex cover and if $x_i = 1$ then the vertex is included in the vertex cover.
    - We need to find the vertex cover of the smallest possible size.
    - Hence, the objective function is **min** $(\sum_i x_i)$
    - Subject to

      $$x_i + x_j >= 1 \text{ for every edge } (i, j) \in E$$

      and

      $$x_i \in \{0, 1\} \text{ for all } i \in V$$
    - For every edge in E at least one vertex of the edge is included and the constraint above ensures that it happens
    - The second constraint ensures that x_i is either 1 or 0.

dnabar@usc.edu

Deven Parag Nabar
USC ID: 7229446568

6. SOLUTION:

The given Linear Program is as follows:

$$\max(x_1 - 3x_2 + 4x_3 - x_4)$$

subject to

$$x_1 - x_2 - 3x_3 \leq -1$$
$$x_2 + 3x_3 \leq 5$$
$$x_3 \leq 1$$
$$x_1, x_2, x_3, x_4 \geq 0$$

The dual program of the given LP is as follows

$$\min (-y_1+5y_2+y_3)$$

Subject to

$$y_1 >= 1$$

$$-y_1 + y_2 >= -3$$

$$-3y_1 + 3y_2 + y_3 >= 4$$

$$0y_1 + 0y_2 + 0y_3 >= -1$$

.

.

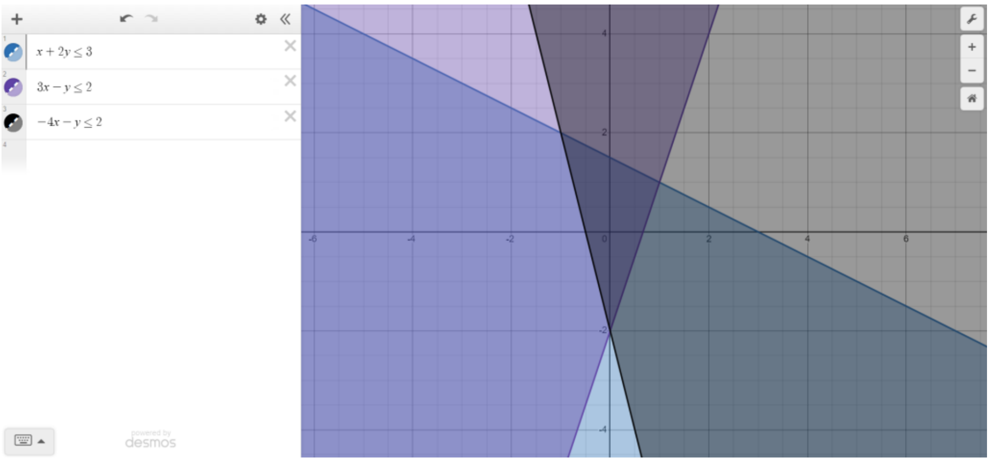
7. SOLUTION:

(a)

$$\max(x_1 + x_2)$$

subject to

$$x_1 + 2x_2 \leq 3$$
$$3x_1 - x_2 \leq 2$$
$$-4x_1 - x_2 \leq 2$$



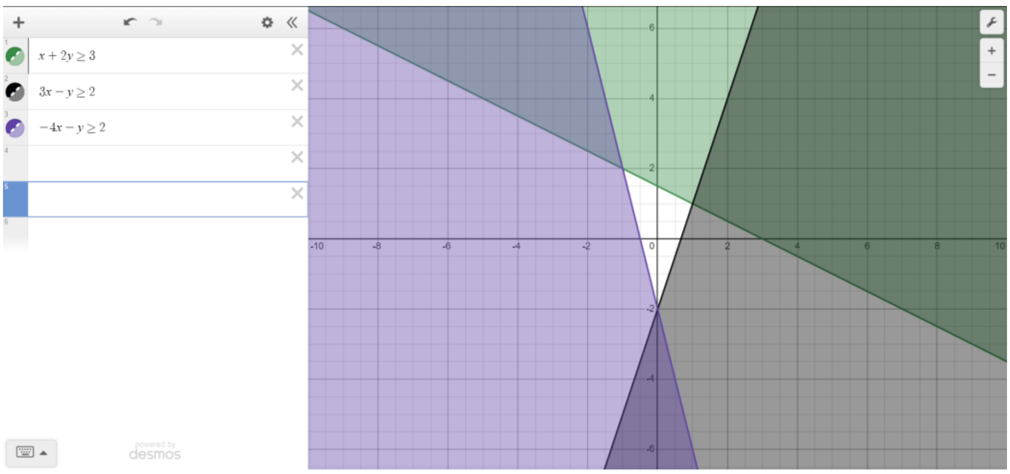
The linear program is **feasible bounded** with the max value = 2 at (1,1)

(b)

$$\max(x_1 + x_2)$$

subject to

$$x_1 + 2x_2 \geq 3$$
$$3x_1 - x_2 \geq 2$$
$$-4x_1 - x_2 \geq 2$$



The linear program is **infeasible**

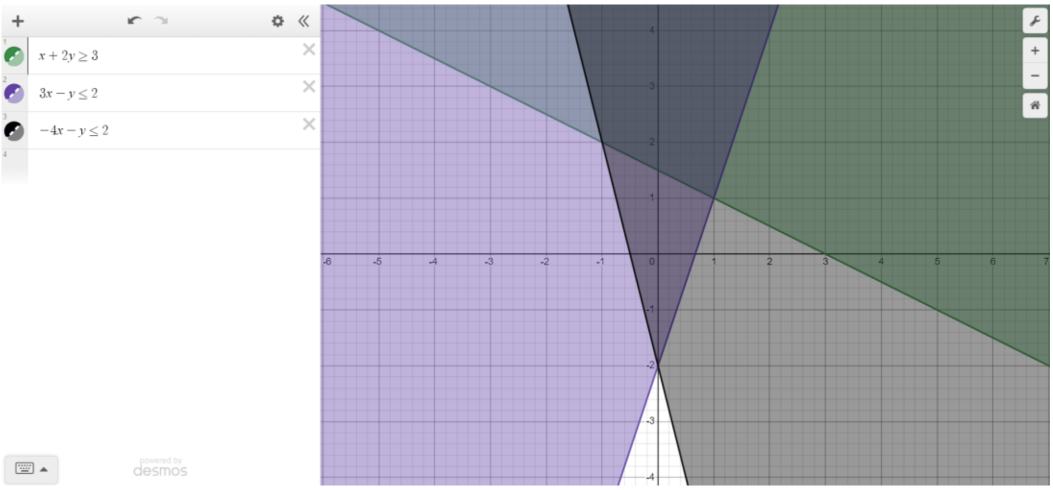
dnabar@usc.edu
Deven Parag Nabar
USC ID: 7229446568

(c)

$$\max(x_1 + x_2)$$

subject to

$$x_1 + 2x_2 \geq 3$$
$$3x_1 - x_2 \leq 2$$
$$-4x_1 - x_2 \leq 2$$

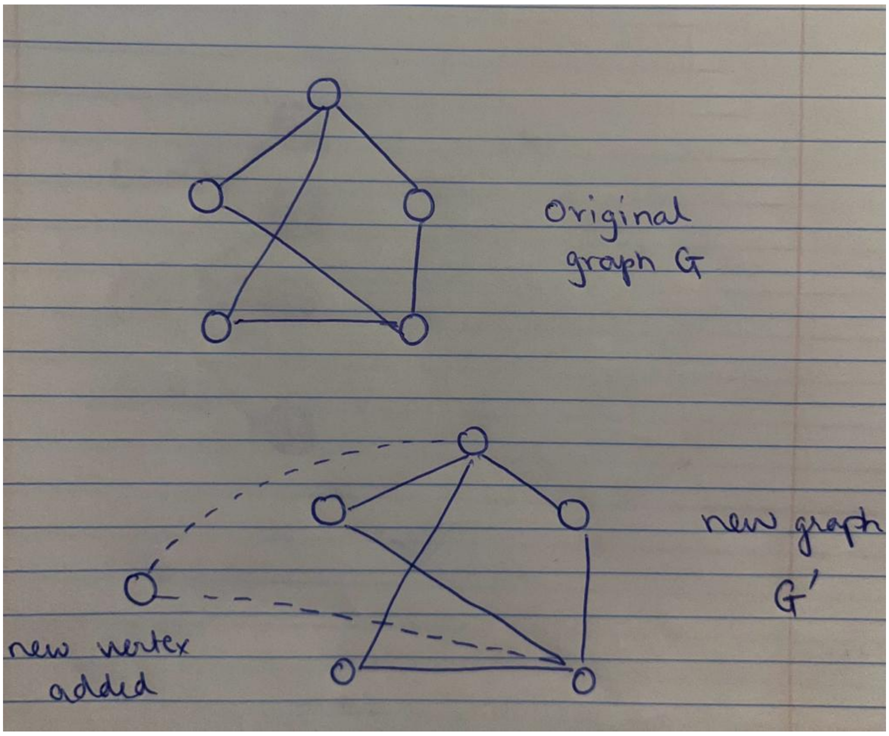


The linear program is **feasible unbounded**

dnabar@usc.edu
Deven Parag Nabar
USC ID: 7229446568

8. SOLUTION:

- Vertex cover (VC) problem ∈ NP
- As per this fact, it follows $(VC)_E \in NP$
- $(VC)_E$ is the same problem as (VC) problem with more restrictions placed on the input.
- We have to show that $(VC)_E$ is NP-hard. To do this, we have to use reduction from VC
  - $VC \leq_p (VC)_E$
- We now need to convert any graph G into a Graph G' with all even degree vertices.
- We also know that any undirected graph has an even number of odd degree vertices.
- Hence, we construct G' by adding an extra vertex to G.
- We then connect this vertex to all vertices of odd degrees.



- In the above example we added a new vertex and connected it to the odd vertices. This gave us a new graph G' with all vertices of even degree.

Claim:

G has a vertex cover of size k if and only if G' has a vertex cover of size k+1

Proof:

==>) Assume G has vertex cover of size k. Vertex cover of G' is created by adding a new vertex and hence the vertex cover size is k+1

<==) Conversely, assume G' has a vertex cover of size k+1. We need to remove a vertex to get the vertex cover of G. However, this is not always possible. In the above example if we remove the a vertex we get a vertex cover of smaller size but it is for a different graph.

And, hence the reduction above is inaccurate. We revise our construction and add 3 new vertices of odd degrees.

New Claim:

G has a vertex cover of size k iff G' has a vertex cover of size k+2

Proof:

==>) Assume G has a vertex cover of size k. Vertex cover of G' is created by adding 2 extra vertices and hence vertex cover is of size k+2

<==) Conversely, assume G' has a vertex cover of size k+2. In order to get vertex cover of G, we have to remove 2 vertices to get vertex cover of size k. These two are easily identified as they must be from a set of extra vertices. Removing any two of these three vertices will result in the right formation of vertex cover for graph G.

Hence, we have shown that vertex cover remains NP-Complete even if the instances are restricted to graphs with only even degree vertices.

dnabar@usc.edu
Deven Parag Nabar
USC ID: 7229446568

9. SOLUTION:

- We are given a polynomial time algorithm that given a 3 SAT instance decides in polynomial time if it has a satisfying assignment.
- The polynomial time algorithm that finds a satisfying assignment to a given 3-SAT instance is as follows:
    - If satisfying assignment exists:
        a. Create an empty set of assigned variables
        b. For each clause:
            1. Pick one literal (eg x_1) and set it to true in the assignment
            2. Remove all clauses that are satisfied by this assignment as they all will become true
            3. If all clauses are satisfied then return the assignment
        c. If there are clauses that cannot be satisfied, go back to the last assignment and remove it. Repeat b and c until all possible assignments are explored.
    - Else return 'cannot be satisfied'
- The main reason we do this is to iteratively pick an unsatisfied clause and assign a truth value to one of its unassigned variables.
- By doing this we increase the number of satisfied clauses.
- By repeating this process till all clauses are satisfied, we make sure that we find a satisfying assignment.