$\boxed{1.}$ Prove by induction that any graph with maximum degree 3 can be colored with at most 4 colors.

<u>Base Case:</u> 1 node, no edges, so no more than 4 colors required.
<u>Inductive Hypothesis:</u> Any graph with max degree 3, with # of nodes $j < n$, requires at most 4 colors to color.
<u>Inductive Step:</u>

Let G be some graph with degree 3 or less and n nodes.

Let T be some node in G. Let G' be the graph obtained by the removal of T and all its edges. By the Inductive Hypothesis, G' is colorable by at most 4 colors, since it has $n-1 < n$ nodes.

Let K be the set of nodes in G' that were connected to T in G. Since G is of maximum degree 3, there are no more than 3 nodes in K, and the nodes in K are therefore colored by at most 3 colors, and G' is colored by at least as many colors in K. That is, let $c(x)$ be the number of colors in item x. We know, then, that:

$$c(K) \leq 3 \leq c(G') \leq 4$$

Therefore, if we add a node to G', connected to each node in K, we can allow it to be some color in G' and not in K, if it exists. If it does not, then $c(G') \leq 3$, and we can color this new node some new color, at which point $c(G') \leq 4$, and G' has n nodes. Therefore any graph with max degree 3 requires at most 4 colors to color. Q.E.D.

$\boxed{2.}$ Subgraph Isomorphism — takes two undirected graphs G_1 and G_2, and asks whether G_1 is isomorphic to a subgraph of G_2. Show that the problem is NP-Complete.

If we have some isomorphism between G_1 and G_2, then we have a set of equivalences of nodes in G_1 to nodes in G_2. We can check the correctness of this isomorphism in polynomial time:

Let V_j and E_j be the # of nodes and edges in G_j, respectively.

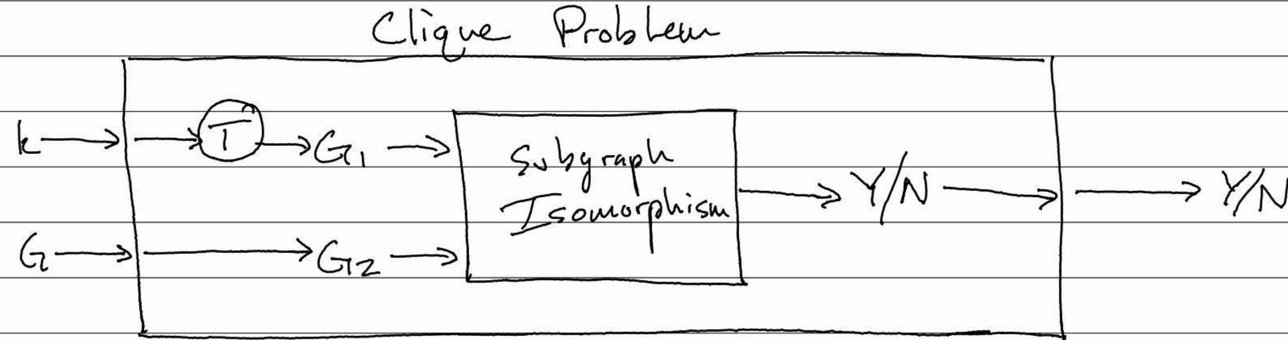
For each equivalence in $E = \{e_1, e_2, \ldots e_k\}$, where $k = V_1$ the number of nodes in G_1, let T_1 and T_2 be the nodes in the equivalence corresponding to G_1 and G_2, respectively. For each equivalence, each edge attached to T_1 must lead to a node for which some other equivalence equates a node in G_1 to a node in G_2, to which T_2 must have an edge. Stepping through these, we can see that checking the isomorphism requires

$O(V_1 \cdot E_1 \cdot V_1 \cdot E_2)$. Since this is a polynomial runtime, the subgraph isomorphism problem must be NP. Therefore, in order to show that it is NP-Complete, it remains only to show that it is NP-Hard.

Continued on next page...

2., continued...

Fact: The subgraph isomorphism problem is NP-Hard.
Proof: Let us assume we have an algorithm which can solve the S.I. problem in polynomial time. We will use a reduction from the Clique problem to show how powerful this would be:



T is a transformation of information k into G_1. Let T take k, and output a clique of size k. G is passed directly in as G_2. This set of transformations takes only $O(k^2)$ (k nodes. at least k-1 edges for each) time, so creating G_1 and G_2 from G and k takes only polynomial time.

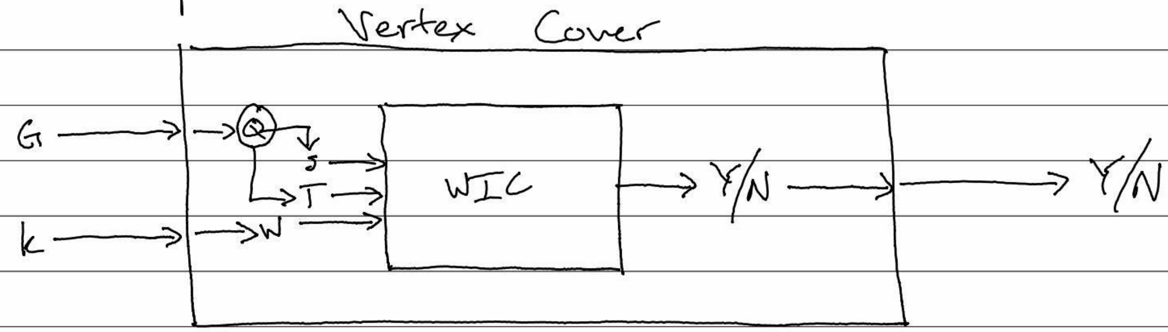
Since the Clique problem asks if there is some Clique of size k in G, if we have a solution which can solve SI in polynomial time, then we could give it that clique, and the original graph G, and it would tell us in polynomial time if that clique existed in G. But the clique problem is NP-Hard, and therefore by reduction SI is NP-Hard, and a polynomial solution is unlikely to exist.

We have shown that SI is both NP and NP-Hard, and it is therefore NP-Complete. Q.E.D.

3. Weighted Item Cover. (WIC)

Fact: WIC is NP-Hard.

Proof: Let us assume that we have some algorithm which will solve WIC in polynomial time. We will use a reduction from the vertex cover problem to show how powerful this would be:



Q in this diagram is a transformation of the graph G into S and T. Q is performed as follows:

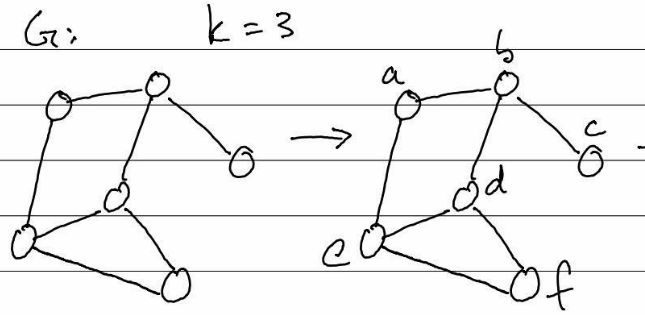
Give each node in G some unique identifier, each with weight = 1. Populate S with this list of weighted nodes.

For each edge in G, add a subset to T which contains the nodes which that edge connects.

Additionally, set W = k.

Like so:

G: $k = 3$



$S = \{a, b, c, d, e, f\}$,

$a = b = c = d = e = f = 1$

$\rightarrow T = \{\{a,b\}, \{b,c\}, \{b,d\}, \{a,e\}, \{e,d\}, \{d,f\}, \{e,f\}\}$

$W = k = 3$

This transformation requires iteration over the nodes once to name them, and populate S, and then over the edges to populate T. If V = # of nodes and E = # of edges in G, then Q runs in $O(V+E)$ time, which is polynomial.

$\longrightarrow$

But does this transformation ensure that WIC will yield a correct result for the transformed Vertex Cover problem?

We should note that each subset of T represents an undirected edge in G. It is clear then that in order for a node to touch an edge, it must be one of the two nodes listed in that edge's subset. Therefore a set of nodes that is a vertex cover will contain at least one node from every subset in T, which WIC will provide, by definition.

The WIC problem provides a solution to uneven weighting, but we have used even weighting for our nodes, since the size of a vertex cover increases by one as one node is added. For this weighting, k can be given directly as W, since WIC will therefore solve for a # of nodes equal to their weight, and less than or equal to W, which Vertex Cover seeks.

Therefore, WIC provides a method for solving Vertex Cover in polynomial time. But Vertex Cover is NP-Hard, and so this is very unlikely, and WIC must be NP-Hard as well.

Q.E.D.