Pranav Prabhu

1. We know that X & Y are two decision problems and X reduces to Y in polynomial time. This means that Y is at least as hard as X or X is no harder than Y.

a. If Y is NP-complete, then so is X: **False**

Since Y is at least as hard as X and if Y is NP-complete, we can't infer X as NP-complete. To prove X is NP-complete, it needs that X is in NP and all problems in NP can be reduced to it. Neither can be inferred here. Hence, for NP-complete, X has to be a subset of NP-complete.

b. If X is NP-complete, then so is Y: **False**

We know that Y is at least as hard as X. So, Y is at least as hard as NP-complete, but we don't know if Y belongs to NP or not. Since X is NP-complete and reducible to Y, it does show that any problem in NP can be reduced to Y. So, we can't infer Y is NP-complete because X could be a lot easier than Y and still reduce to Y.

c. If Y is NP-complete and X is in NP, then X is NP-complete: **False**

Although X is reducible to Y and Y is NP-complete, it doesn't mean X is NP-complete because X could be relatively easier than Y. For X to be NP-complete, it should be at least as hard as the NP-complete problem and belong to NP.

d. If X is NP-complete and Y is in NP, then Y is NP-complete: **True**

Both conditions of NP-complete are met:

I. Y is at least as hard as X and X is NP-complete. So, Y is at least as hard as an NP-complete problem.

II. Y belongs to NP, so we can infer Y is NP-complete.

e. If X is in P, then Y is in P: **False**

If X is polynomial-time reducible to Y and since Y is at least as hard as X, this means that Y can be a lot harder than X. So, it can't be inferred that Y is also in P.

f. If Y is in P, then X is in P: **True**

Since X is polynomial-time reducible to Y, X is no harder than Y. So, if Y is in P, it can be inferred that X is also in P.

g. X and Y can't both be in NP: **False**

X and Y can both be in NP because both can be equally hard ot Y can be a lot harder than X. So, there are possibilities that both X and Y can be in NP.

2. We know that the 2-SAT problem is a SAT variant in which each clause contains at most two literals, and it is known to have a polynomial-time algorithm.

a. 3-SAT ≤p TSP: **True**

If one NP problem could be solved in polynomial time, then all NP problems can be solved in polynomial time. Since 3-SAT is an NP-complete problem and TSP is also an NP problem, if TSP is polynomial-time reducible, 3-SAT could also be reduced.

b. If P ≠ NP, then 3-SAT ≤p 2-SAT: **False**

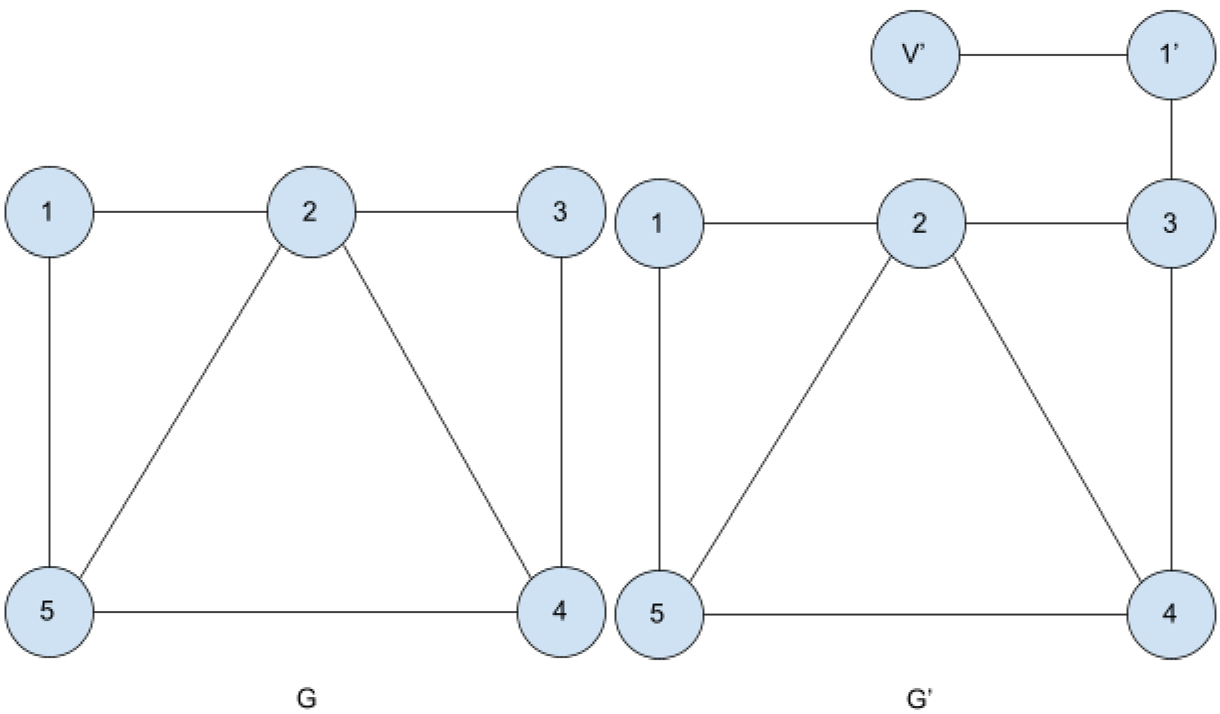
Here, the fact that 3-SAT is NP-complete implies that all problems in NP are reducible to 2-SAT. If there is polynomial-time reduction from 3-SAT to 2-SAT, then 2-SAT is NP-complete, but 2-SAT is also solvable in polynomial time. If 2-SAT has polynomial time algorithm, this implies that all problems in NP have a polynomial-time algorithm, which in turn implies that P = NP. However, this contradicts the assumption that P ≠ NP. Hence, the statement 3-SAT ≤p 2-SAT is false.

c. If TSP ≤p 2-SAT, then P = NP: **True**

If P ≠ NP, then no NP-complete problem can be solved in polynomial time. If one NP-complete problem can be solved in polynomial time, then all NP-complete problems can be solved in polynomial time. If 2-SAT has polynomial-time algorithm, this implies that all problems in NP have polynomial-time. Hence, P = NP.

3. We know that a Hamiltonian path in a graph is a simple path that visits every vertex exactly once and HAM-CYCLE is NP-complete. To show that this problem is NP-complete, we first need to show that it belongs to the NP class and then find a known NP-complete problem that can be reduced to Hamiltonian Path. For a given graph G, we can solve Hamiltonian Path by non-deterministically choosing edges from G that can be included in the path. Then, we traverse the path and make sure that we visit each vertex exactly once. This obviously can be done in polynomial time, and hence, the problem belongs to NP (P ⊆ NP).

Next, we know that a Hamiltonian Path begins and ends in the same vertex. Moreover, we know that Hamiltoninan Cycle is NP-complete, so we may try to reduce this problem to Hamiltonian Path. In graph G = <V, E>, we construct G' such that G contains a Hamiltonian cycle if and only if G' contains a Hamiltonian path. By definition, Hamiltonian Path in an undirected graph is a path that visits each vertex exactly once. A Hamiltonian cycle is a Hamiltonian Path such that there is an edge in the graph from the last vertex to the first vertex of the Hamiltonian Path. Hence, we choose an arbitrary vertex u in G and adding all edges as we traverse throught the graph. See the figure below:



Suppose that the above graph G has a Hamiltonian cyle of 1, 2, 3, 4, 5, 1. In G', this corresponds to 1, 1, 2, 3, 4, 5, 1', v'. Conversely, suppose G' contains a Hamiltonian path. In that case, the path must necessarily have endpoints in v and v'. This path can be transformed to a cycle in G. So, the path in graph G matches the cycle in graph G' and vice versa. Therefore, by definition, graph G contains a Hamiltonian Path in this undirected graph which visits each vertex exactly once and returns back to the starting vertex. Also, by definition, there exists a Hamiltonian Cycle such that there exists an edge between the last vertex 5 and first vertex 1. This construction won't work when G is a single edge, which is a special case. Hence, we have shown that G contains a Hamiltonian cycle if and only if G contains a Hamiltonian Path. This concludes the proof that Hamiltonian path is NP-complete.

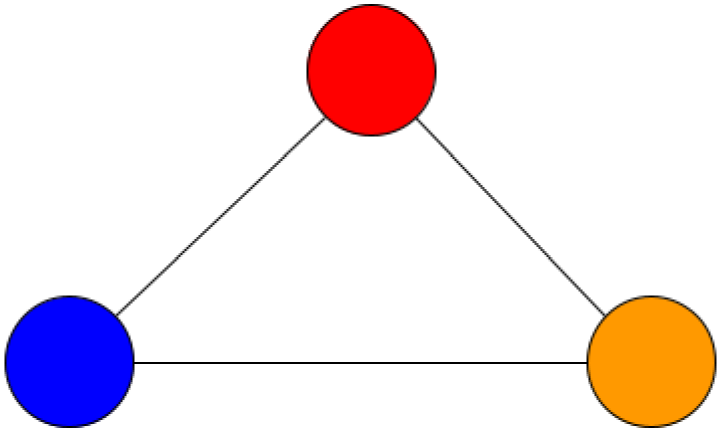
4. Given a graph G = (V,E), a k-coloring is a function c: V -> {1, 2, ... , k} such that c(u) ≠ c(v) for every edge (u,v) ∈ E. In other words the number 1, 2, …, k represent the k colors and adjacent vertices must have different colors. The 3-COLOR decision problem is NP-complete by using a reduction from SAT. We need to prove 4-COLOR is also NP-complete using the fact that 3-COLOR is NP-complete. Let two colors be red and blue. A graph can be said as bipartite if it can be colored with 2 colors with no adjacent vertex having the same color. Following are the steps to find out whether a given graph is Bipartite or not using Breadth First Search (BFS):

a. Assign color RED to source vertex (RED ∈ U).
b. Color all the neighbors with the color BLUE (BLUE ∈ V)
c. Color all of the source vertex's neighbor's neighbors with RED color (RED ∈ U).
d. Repeat steps 2 and 3 such that it satisfies the k-way coloring problem where k = 2.
e. While assigning colors, if a neighbor is colored with the same color as the current neighbor, then the 2 vertices cannot be colored the same (or graph is not Bipartite).

We know that 3-COLOR is NP-Complete. So, we have to prove it can be "reduced to" or transformed into 4-COLOR. A problem should be both NP and NP-Hard to be NP-Complete. So, we need to now prove that the problem is first NP and then NP-Hard to be NP-Complete.

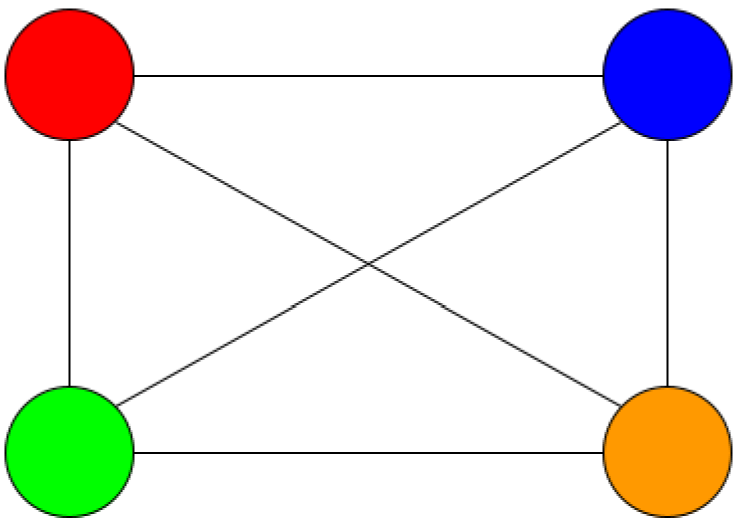
I. **4-COLOR is in NP:** The coloring is the certificate (i.e., list of nodes and colors). The following is a verifier for 4-COLOR:
   A. Color each node of G using specified colors.
   B. For each node, check that it has a unique color from each of it's neighbors.
   C. Graph is accepted if all checks pass; otherwise, we reject.

II. **4-COLOR is NP-Hard:** We give a polynomial-time reduction from 3-COLOR to 4-COLOR. The reduction maps a graph G into a new graph G' such that G' ∈ 3-COLOR iff G'' = 4-COLOR. We do so by setting G'' to G and then adding a new node x and connecting x to each node in G''. If G is 3-colorable, then G'' can be 4-colored exactly as G. Similarly, if G'' is 4-colorable, then we know that node x must be the only node of its color because it is connected to every other node in G''. Hence, G is 3-colorable.



G

The above 3-COLOR graph G can be transformed into a 4-COLOR graph like so:



G'

Since 4-COLOR is in NP and NP-Hard, we have proved that 4-COLOR is NP-Complete.