Problem 1 (DPV 8.2)

In the graph G(V, E), there are |E| number of edges. Let's call the Rudrata Path, which is still a graph of nodes and connected edges, be called $R(V_R, E_R)$. Then the relationship is given as following:

1. $R \subseteq G$
2. $V_R = V$
3. $E_R \subset E$ if $R \neq G$

1. means that G is either already a Rudrata Path R, or if not, R is a proper subset of G

2. means that the Rudrata Path R should include all the vertices of G

3. means that if indeed G wasn't already a Rudrata path, then E_R will not contain all the edges of E

So now, for the actual algorithm, we take each edge in E and then subtract it from the graph G.

- ⇨ **$G(V, E-\{e_i\})$** where e_i represents each edge in e

Then we run the given polynomial-time RUDRATA PATH on **$G(V, E-\{e_i\})$**. For each edge, we assess whether subtracting that edge would still yield a Rudrata Path. We repeat this process for every edge. Along the process, if **$G(V, E-\{e_i\})$** returns TRUE, which means G without edge e_i still yields a Rudrath Path, we remove that edge from the graph and continue the process.

Once all the unnecessary edges are removed, G would have become R by the time the algorithm terminates.

Problem 2 (DPV 8.4) – a,b only

(a) We go through every vertex in the graph and calculate the degree, i.e., assess if each vertex has at most two other vertices connected to it. Given n vertices in a graph, it takes $O(n^2)$ for adjacency matrix and $O(n^3)$ for adjacency list. Regardless, the algorithm runs in polynomial time $\Theta(n^k)$.

(b) In the proof, it states that 'so it is enough to present a reduction from CLIQUE-3 to CLIQUE. However, this is certainly not true because you cannot reduce a problem into a bigger set of problems. The correct way should be to reduce from CLIQUE to CLIQUE-3. From CLIQUE, we show a polynomial-time reduction to CLIQUE-3, and that should be the proof of showing CLIQUE-3 is in the NP-complete set.

Problem 3 (DPV 8.13) – a,b,c only

(a) This can be solved in poly time. Imagine we had the same graph G but only without L, so we call this new graph G'=(V-L, E) and this graph has to be connected. We find a spanning tree of G', which will include all the nodes V-L. Now what we do is just bring back all the nodes in L, and each one of these L nodes should have at least one neighboring node that was included in the earlier spanning tree.

(b) If we take G'=(V-L, E) and use it in this context, then there are a couple of interesting things going on with the graph. First, V-L still has to form a spanning tree, meaning all the remaining vertices are used. Second, now that all the vertices of L are gone, V-L cannot have any more leaf nodes. Lastly, V-L must form a spanning tree, so the number of edges in that spanning tree must be |(V-L)|-1. This is because a spanning tree contains n-1 number of edges, n being the number of total vertices.

A kind of graph (or path) that can satisfy all the three criteria above is a Hamiltonian path, i.e., a Rudrata Path.

So, if RUDRATA of G'=(V-L, E) returns true, then the problem can be solved. Once we know that V-L can form a Rudrata path, then we do what we did earlier in DPV 8.2 and return the actual path. With the given (Rudrata) path, we just add L and connect arbitrary edges between each pair of nodes <V-L, L>.

Since we run RUDRATA multiple times, the runtime remains a constant multiple of polynomial, or $C * n^k = O(n^k)$.

(c) This problem can be solved in the same way as above. And the same method of reduction yields polynomial time. However, because of how the question is defined, there is an edge case. The question states 'find a spanning tree such that its set of leaves is included in the set L.'

Either a smallest possible tree (two nodes with one edge between them) or a linear line of path, both are the minimal forms of a tree that have two leaf nodes. In fact, a tree requires to have two or more leaf nodes. Because a tree needs at least two leaves, when we run RUDRATA to find a Rudrata path from u to v, an edge case occurs if both u and v are in L. When L = {u, v} and $u \neq v$, the set of leaves must be equal to L. This is because two of the leaves will be given by u and v as they form a Rudrata path, and the rest of the leaves must be exactly L – {u, v}. This proves that the problem is still in polynomial.