Problem 1: DPV 8.1 TSP Optimization vs Search
Recall the traveling salesman problem:

TSP
Input : A matrix of distances; a budget b
Outpus : A tour which passes through all the cities and has length $\leq b$, if such a tour exists

The optimization version of this problem directly asks for the shortest tour.

TSP-OPT
Input : A matrix of distances
Outpus : The shortest tour which passes through all the cities.

Show that if TSP can be solved in polynomial time, then so can TSP-OPT.

Solution.

In order to show this, we will show that TSP-OPT reduces to TSP in polynomial time. The idea of this reduction is to use binary search with the b input for TSP to find the length of the shortest tour (then TSP will find the shortest tour itself). Let the distance matrix be denoted D. First, we need to find an upper bound B for the length of the shortest path. We can set B to be the sum of all distances in the distance matrix (so $B = \sum_{i,j} D_{i,j}$). Then, we run TSP with inputs D and $B/2$. If this finds a tour with length at most $B/2$, then we know the shortest tour has length in the range $[0, B/2]$. Otherwise, the shortest tour has length in the range $[B/2, B]$. Here is where we can apply binary search; for each successive iteration, run TSP on the midpoint of the remaining range for the length of the shortest path, then eliminate half of the range based on whether TSP finds a tour. Continue this recursive process until the range is narrowed to a single integer number, then return the path found by TSP on this integer. TSP will be run $O(\log B)$ times (since this is a binary search from 0 to B). The input size of TSP-OPT must be of length $\Omega(\log B)$, since B was the sum of all the distances in D. Therefore, this reduction is polynomial, and if TSP can be solved in polynomial time, then so can TSP-OPT.

Problem 2: DPV 8.3 Stringy SAT

STINGY SAT is the following problem: given a set of clauses (each a disjunction of literals) and an integer k, find a satisfying assignment in which at most k variables are `true`, if such an assignment exists. Prove that STINGY SAT is NP-complete.

Solution.

First, we show that STINGY SAT is in NP. Let the input formula be denoted by f, and let n be the number of variables and m be the number of clauses. Given a satisfying assignment σ of F, we can just check clause by clause that σ satisfies every one; this takes $O(n)$ time per clause and thus $O(nm)$ total time. Then in $O(n)$ time, we count how many variables are set to be TRUE in σ and check that the number is larger than k or not.

Now, we show: SAT $\rightarrow$ STINGY SAT. Suppose you have an input formula f for SAT with n variables and m clauses. Now, run STINGY SAT on f with $k = n$. Clearly, every assignment over n variables will have at most n variables set to true, so this extra restriction will not constrain the set of solutions. Then, if STINGY SAT returns a satisfying assignment, that assignment will also satisfy the original SAT problem. Likewise, if STINGY SAT returns that there is no such assignment, then there can be no solution to the SAT problem. This completes the reduction, which is clearly polynomial time (the only change was setting $k = n$). Then, since SAT is NP-complete, STINGY SAT must also be NP-complete.

Problem 3: DPV 8.4 (a),(b),(c) Clique-3
Consider the CLIQUE problem restricted to graphs in which every vertex has degree at most 3. Call this problem CLIQUE-3.

(a) Prove that CLIQUE-3 is in NP.

Solution.

Given an input graph $G = (V, E)$, a goal g, and a potential solution set $S \subseteq V$, it is easy to verify whether S is a clique by checking that all pairs of vertices in S are connected in $O(n^2)$ time, and to check that $|S| \geq g$ in $O(n)$ time. Since a solution can be verified in polynomial time, CLIQUE-3 is in NP.

(b) What is wrong with the following proof of NP-completeness for CLIQUE-3?

We know that the CLIQUE problem in general graphs is NP-complete, so it is enough to present a reduction from CLIQUE-3 to CLIQUE. Given a graph G with vertices of degree ≤ 3, and a parameter g, the reduction leaves the graph and the parameter unchanged: clearly the output of the reduction is a possible input for the CLIQUE problem. Furthermore, the answer to both problems is identical. This proves the correctness of the reduction and, therefore, the NP-completeness of CLIQUE-3.

Solution.

The direction of this reduction is wrong. In order to show that CLIQUE-3 is NP-complete, we would instead need to show that some known NP-complete problem (such as CLIQUE) reduces to CLIQUE-3.

We want to prove that CLIQUE-3 is a computational difficult problem. To do that we want to show that if we somehow solve CLIQUE-3 efficiently then we can efficiently solve every problem in NP. We know that CLIQUE is NP-complete and hence if we can efficiently solve CLIQUE then we can efficiently solve every problem in NP. Therefore, we need to show that CLIQUE $\rightarrow$ CLIQUE-3, but the above proof does the reverse reduction.

(c) It is true that the VERTEX COVER problem remains NP-complete even when restricted to graphs in which every vertex has degree at most 3. Call this problem VC-3. What is wrong with the following proof of NP-completeness for CLIQUE-3?

We present a reduction from VC-3 to CLIQUE-3. Given a graph $G = (V, E)$ with node degrees bounded by 3, and a parameter b, we create an instance of CLIQUE-3 by leaving the graph unchanged and switching the parameter to $|V| - b$. Now, a subset $C \subseteq V$ is a vertex cover in G if and only if the complementary set $V - C$ is a clique in G. Therefore G has a vertex cover of size $\leq b$ if and only if it has a clique of size $\geq |V| - b$. This proves the correctness of the reduction and, consequently, the NP-completeness of CLIQUE-3.

Solution.

The reduction is incorrect. Specifically, the statement "a subset $C \subseteq V$ is a vertex cover in G if and only if the complementary set $V - C$ is a clique in the same graph G" is incorrect. A correct statement would be "a subset $C \subseteq V$ is a vertex cover in G if and only if the complementary set $V - C$ is a clique in the complementary graph $\bar{G} = (V, V^2 - E)$". Hence, one should change the input graph G for VC-3 to $\bar{G}$ for CLIQUE-3

Problem 4: DPV 8.10 (a) Subgraph Isomorphism
For the problem below, prove that it is NP-complete by showing that it is a *generalization* of some NP-complete problem we have seen.

SUBGRAPH ISOMORPHISM: Given as input two undirected graphs G and H, determine whether G is a subgraph of H (that is, whether by deleting certain vertices and edges of H we obtain a graph that is, up to renaming of vertices, identical to G), and if so, return the corresponding mapping of $V(G)$ into $V(H)$.

Solution.

First, we show that SUBGRAPH ISOMORPHISM is in NP. For a pair of graphs $G = (V, E)$ and $H = (V', E')$, given a mapping π from $V(G)$ to $V(H)$, we can verify the solution in $O(|V| + |E|)$ time by checking that each edge $(u, v) \in E$ maps to an edge $(\pi(u), \pi(v)) \in E'$.

Now, we show: INDEPENDENT SET $\rightarrow$ SUBGRAPH ISOMORPHISM. Let the inputs to INDEPENDENT SET be $H = (V, E)$ and an integer k. To construct the input to SUBGRAPH ISOMORPHISM, let $G = (V, E)$ where $|V'| = k$ and $E' = \emptyset$. Run SUBGRAPH ISOMORPHISM on G and H. H has an independent set of size k if and only if there is a solution to SUBGRAPH ISOMORPHISM, with the independent set being the set of vertices that map to the subgraph G with no edges. The solution will also provide a mapping from V' to V, allowing us to find the independent set in the original graph H. This reduction takes polynomial time, since it creates a graph with k vertices and no edge ($O(k)$ time). Then, since INDEPENDENT SET is NP-complete, SUBGRAPH ISOMORPHISM must be as well.