Solutions to Homework Practice Problems

Practice problems:

[DPV] Problem 8.1 (TSP optimization versus search)

Recall the search version of the traveling salesman problem:

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

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

- `TSP-OPT`
- Input: A matrix of distances
- Output: 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 $O(\log B)$, 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.

[DPV] Problem 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. While it would be correct to say "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 $\bar{G} = (V, \bar{E})$", the complement of G is not guaranteed to retain the property that every vertex has at degree at most 3, so this alternate statement is not useful for the desired reduction.

[DPV] Problem 8.8 (Exact 4SAT)

Solution:
We'll denote this `EXACT 4SAT` problem as `E4SAT`. We assume there are n variables, m clauses, and each clause has exactly 4 literals.

First we need to prove that `E4SAT` is in the class NP: Given a candidate solution of truth value assignments for each variable, in O(1) time per clause we can verify that at least one literal is satisfied, and hence in $O(m)$ total time we can verify that the given assignment satisfies the input formula f.

We now prove that `E4SAT` is at least as hard as a known NP-Complete problem by reducing `3SAT` $\rightarrow$ `E4SAT`.

Input Transformation: Consider an input formula f to `3SAT`, where f has n variables and m clauses. We will define an input formula f' for the `E4SAT` problem. Suppose C is a clause in f with three literals, $C = (x \vee y \vee z)$. We will create a new variable w and replace C by a pair of clauses: $C' = (x \vee y \vee z \vee w) \wedge (x \vee y \vee z \vee \overline{w})$. Notice that the extra variable w can only satisfy one of the two clauses, so this pair of clauses C' is satisfiable iff a truth assignment exists to satisfy C.

If C has two literals, say $C = (x \vee y)$, then we add two new variables w_1, w_2 and we replace C by 4 clauses: $C' = (x \vee y \vee w_1 \vee w_2) \wedge (x \vee y \vee w_1 \vee \overline{w_2}) \wedge (x \vee y \vee \overline{w_1} \vee w_2) \wedge (x \vee y \vee \overline{w_1} \vee \overline{w_2})$. Once again, as any combination of the truth assignments for the two new variables can satisfy at most 3 of these 4 clauses, C' is satisfiable iff a truth assignment exists to satisfy C. Similarly, if C has a single literal, we add 3 new variables and we replace C by 8 clauses. Note that f' will have at most $O(8m) = O(m)$ clauses and $O(n + 3) = O(n)$ variables, and that this input transformation takes $O(m)$ time.

Output Transformation: If `E4SAT` reports NO, we return NO for `3SAT`. If `E4SAT` returns a satisfying truth assignment, we simply remove the added variables and return the truth assignment for the original variables in `3SAT`. There are at most three added variables to remove, so this takes $O(n)$ linear time.

Correctness Justification: As we have argued on a clause-by-clause basis, f is satisfiable if and only if f' is satisfiable. Moreover, given a satisfying assignment to f', we immediately get a satisfying assignment to f by returning the truth assignment for the original n variables (in $O(n)$ time). Thus, as `E4SAT` is in the class NP, and at least as hard as `3SAT`, we concluded that `E4SAT` is NP-Complete.

[DPV] Problem 8.10 (a) (Subgraph isomorphism)

Proving NP-completeness by generalization. For each of the problems below, prove that it is NP-complete by showing that it is a generalization of some NP-complete problem we have seen in this chapter.

(a) `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 to V', we can verify the solution as follows: Traverse graph G. At each vertex v, find it's compliment v' in the adjacency list of H. (note this is a pair of O(1) lookups). For each edge (v, u) adjacent to v, check all the edges adjacent to v' to see if one is (v', u'). Traversing the original graph takes $O(|V| + |E|)$; and at each edge we have $|E'|$ edges to check, yielding $O((|V| + |E|)|E'|)$ which is polynomial.

Now, we show: Clique $\rightarrow$ Subgraph isomorphism.

Input Transformation: Let the inputs to Clique be $H = (V', E')$ and an integer k. To construct the input to Subgraph isomorphism, let $G = (V, E)$ be a complete graph (every pair of vertices is connected by an edge) where $|V| = k$. This input reduction takes polynomial time, since it creates a complete graph with k vertices in $O(k^2) = O(|V'|^2)$ time (k being bounded by $|V'|$, the number of vertices in H).

Output Transformation: Run Subgraph isomorphism on G and H. If NO mapping is reported, report NO solution for Clique If the output is a mapping π from V to V', we use this output to identify the vertices which define the clique in H. Iterating through the K vertices of G in the mapping to identify the vertices of the clique in H takes $O(|V|)$ time.

Correctness Justification: Now, H has an clique of size k if and only if there is a solution to Subgraph isomorphism, with the clique being the set of vertices that map to the subgraph G. Why do we know this is true? Finding a clique in H is a special case of Subgraph Isomorphism where G is a clique - in this case, the one of size k we created. If a mapping between G and H is returned, that must define a clique of size K in H; and if no mapping is returned that means that no clique of size k was present in H.

Then, since Clique is NP-complete, Subgraph isomorphism must be as well. And as Clique is a special case of Subgraph isomorphism, we conclude that Subgraph isomorphism is a generalization of Clique.