HW7

Problem 1

Consider the vertex cover problem that restricts the input graphs to be those where all vertices have even degree. Call this problem VC-EDG. Show that VC-EDG is NP-complete. (15pts)

Hint: To shown NP-hardness, reduce from VC, i.e., think of a construction where given the input of VC, i.e., a graph G and a number K, you can construct G′,K′, (where G′ has even degrees for all its vertices), such that G has a vertex cover of size K if and only if G′ has a vertex cover of size K′.

Solution: To prove that VC-EDG is NP-complete, we need to prove two things:

1. VC-EDG is in NP.
2. VC-EDG is NP-hard.

1.VC-EDG is in NP:
Given a graph G' and a set of vertices S with size K' , we can easily verify in polynomial time if S is a vertex cover of G' by checking if every edge has at least one endpoint in S .

2. VC-EDG is NP-hard:
To show this, we'll reduce the Vertex Cover problem (VC) to VC-EDG . Given an instance of VC with a graph G = (V, E) and a number K , we will construct a new graph G' and number K' such that G' has even degrees for all its vertices and G has a vertex cover of size K if and only if G' has a vertex cover of size K' .

Reduction:

For every vertex v in G that has odd degree, add a new vertex v' and connect v' to v . This will increase the degree of v by 1, making it even. v' will also have even degree (specifically 2). Repeat this for every vertex with odd degree.

Let G' be the resulting graph after these additions. Set K' = K + (number of added vertices).

Proof:
(a) If G has a vertex cover S of size K , then G' has a vertex cover S' of size K' .

S covers all the edges in G . In G' , all the edges that were added have one endpoint as the newly added vertex and the other as the vertex with odd degree. By adding these new vertices to S , we can cover all the edges of G' .

(b) If G' has a vertex cover S' of size K' , then G has a vertex cover S of size K .

Any vertex cover of G' that includes the new vertices (added for odd degree vertices in G) can be transformed into a vertex cover for G by including the original odd-degree vertex and excluding the new vertex. Since every new vertex is only connected to its corresponding original vertex, including the original vertex ensures that all edges are still covered.

Given this, we have reduced the VC problem to VC-EDG . Therefore, VC-EDG is NP-hard.

Since VC-EDG is both in NP and NP-hard, it is NP-complete.

Problem 2

Consider the partial satisfiability problem, denoted as 3-Sat(α) defined with a fixed parameter α where 0 ≤ α ≤ 1. As input, we are given a collection of k clauses, each of which contains exactly three literals (i.e. the same input as the 3-SAT problem from lecture). The goal is to determine whether there is an assignment of true/false values to the literals such that at least αk clauses will be true. Note that for α = 1, we require all k clauses true, thus 3-Sat(1) is exactly the regular 3-SAT problem.

Prove that 3-Sat(15/16)is NP-complete. (20 points)

Hint: If x, y, and z are variables, note that there are eight possible clauses containing them: (x ∨ y ∨ z), (!x ∨ y ∨ z), (x∨!y ∨ z), (x ∨ y∨!z), (!x∨!y ∨ z), (!x ∨ y∨!z), (x∨!y∨!z), (!x∨!y∨!z) Think about how many of these are true for a given assignment of x, y, and z.

Solution: To prove that 3-Sat(\alpha) is NP-complete, where \alpha = 15/16, we need to establish the following:

1. 3-Sat(15/16) is in NP.
2. 3-Sat(15/16) is NP-hard.

1. 3-Sat(15/16) is in NP:

For a given assignment, we can verify in polynomial time if at least \alpha k = 15k/16 clauses are satisfied. Therefore, 3-Sat(15/16) is in NP.

2. 3-Sat(15/16) is NP-hard:

We'll use a reduction from the classic 3-SAT problem to 3-Sat(15/16). Given an instance of 3-SAT with a collection of k clauses, we will construct an instance of 3-Sat(15/16).

Reduction:

Let's consider a clause from the 3-SAT instance: (a \vee b \vee c) .

For the three variables $a, b,$ and c, there are 8 possible 3-literal clauses, as hinted. For any assignment of $a, b,$ and c, 7 out of these 8 clauses will always be true.

Now, for each clause in the 3-SAT instance, replace it with these 8 clauses. So, if the 3-SAT instance had k clauses, the new instance has $8k$ clauses.

Proof:
(a) If the 3-SAT instance is satisfiable, then there exists an assignment such that all k original clauses are true. For each of these k clauses, 7 out of 8 of the new clauses are true. Therefore, $7k$ out of $8k$ new clauses are true, which is equivalent to 7/8 or 14/16. But we need 15k/16 clauses to be true. However, note that the one true assignment for the original clause in 3-SAT ensures one more clause to be true in the new set. So, in total, we have $7k + k$ or $8k$ clauses that are true, which satisfies the requirement for 3-Sat(15/16).

(b) Conversely, if there exists an assignment for the 3-Sat(15/16) instance such that $15k/16$ clauses are true, then at least one of the 8 clauses generated for each original clause in the 3-SAT instance must be true. This is because the maximum number of true clauses we can get without the original 3-SAT clause being satisfied is $7k$, but we need $15k/16$ clauses to be true. Hence, the original 3-SAT instance must also be satisfiable.

This reduction is clearly polynomial since we're just expanding each clause in the 3-SAT instance to a set of 8 clauses. So, we have shown that 3-Sat(15/16) is NP-hard.

Combining our results, we conclude that 3-Sat(15/16) is NP-complete.

Problem 3

There are N cities, and there are some undirected roads connecting them, so they form an undirected graph G(V,E). You want to know, given K and M, if there exists a subset of cities of size K, and the total number of roads between these cities is larger or equal to M. Prove that the problem is NP-Complete.

Solution: a) The Clique problem is NP-complete:
1. The Clique problem is in NP:
Given a subset of vertices, it is easy to verify in polynomial time if they form a clique by checking if every pair of distinct vertices in the subset are adjacent.

2. Reduction from the Independent Set problem to Clique:
Given an instance of the Independent Set problem, which asks if there's a subset of vertices such that no two vertices in the subset are adjacent, we can transform it into a Clique problem instance as follows:

For every instance (G, k) of the Independent Set problem, create an instance (G', k) for the Clique problem, where G' is the complement graph of G (i.e., (u, v) is an edge in G' if and only if (u, v) is not an edge in G).

Claim: G has an independent set of size k if and only if G' has a clique of size k.

Proof: Suppose S is an independent set in G of size k. This means no two vertices in S are adjacent in G. Thus, every pair of distinct vertices in S are adjacent in G'. So, S forms a clique in G'. Conversely, if S is a clique in G' of size k, then S is an independent set in G.

Since the Independent Set problem is known to be NP-complete, the Clique problem is also NP-complete because we have shown a polynomial-time reduction from Independent Set to Clique.

b) Dense Subgraph Problem is NP-complete:
1. The Dense Subgraph problem is in NP:
Given a subgraph G' = (V', E'), it's easy to verify in polynomial time whether $|V'| \leq k$ and $|E'| \geq m$.

2. Reduction from the Clique problem to the Dense Subgraph problem:

Given an instance (G, k) of the Clique problem, we create an instance (G, k, k*(k-1)/2) for the Dense Subgraph problem.

Claim: G has a clique of size k if and only if G has a dense subgraph with at most k vertices and at least k*(k-1)/2 edges.

Proof:
- If G has a clique of size k, then this clique is a subgraph with k vertices and k*(k-1)/2 edges (because each vertex in the clique is connected to every other vertex in the clique).
- Conversely, if G has a dense subgraph with at most k vertices and at least k*(k-1)/2 edges, then this subgraph is a clique. The reasoning is that the maximum number of edges a graph with k vertices can have (without forming a clique) is less than k*(k-1)/2. Therefore, to have at least k*(k-1)/2 edges, the subgraph must be a clique.

Since we've shown a polynomial-time reduction from the Clique problem (which is known to be NP-complete) to the Dense Subgraph problem, the Dense Subgraph problem is also NP-complete.

<u>Problem 4</u>
Suppose we have a variation on the 3-SAT problem called Min-3-SAT, where the literals are never negated. Of course, in this case it is possible to satisfy all clauses by simply setting all literals to true. But, we are additionally given a number k, and are asked to determine whether we can satisfy all clauses while setting at most k literals to be true. Prove that Min-3-SAT is NP-Complete.

Solution: Min-3-SAT Problem: Given a 3-SAT formula where no literals are negated and a number k , determine if all clauses can be satisfied by setting at most k literals to true.

To show that Min-3-SAT is NP-Complete, we will show that:
1. Min-3-SAT is in NP.
2. Min-3-SAT is NP-hard.

1. Min-3-SAT is in NP:
Given a set of k literals that are set to true, we can easily check in polynomial time whether the entire formula is satisfied or not.

2. Min-3-SAT is NP-hard:
We will reduce from the Vertex Cover problem, which is a known NP-complete problem.

Vertex Cover Problem: Given a graph $G = (V, E)$ and an integer k, is there a subset $V' \subseteq V$ such that $|V'| \leq k$ and for every edge $(u, v) \in E$, at least one of u or v is in V'?

Reduction:
Given an instance (G, k) of the Vertex Cover problem, construct an instance of Min-3-SAT as follows:

For each edge $(u, v) \in E$, introduce a clause $(u \vee v)$. This clause requires at least one of the literals u or v to be set to true to satisfy the clause.

Claim: Graph G has a vertex cover of size k if and only if the Min-3-SAT formula can be satisfied by setting at most k literals to true.

Proof:
- If G has a vertex cover V' of size k, then set all literals corresponding to vertices in V' to true. Since V' is a vertex cover, for every edge $(u, v) \in E$, at least one of u or v is in V', thus satisfying the corresponding clause in the Min-3-SAT formula.

- Conversely, if the Min-3-SAT formula can be satisfied by setting at most k literals to true, then the set of vertices corresponding to these literals forms a vertex cover in G. This is because each edge $(u, v) \in E$ corresponds to a clause in the formula, and if the formula is satisfied, then for each clause at least one of the literals must be true. This ensures that for every edge, at least one of its endpoints is in the vertex cover.

The reduction is polynomial since for each edge in G, we simply create one clause in the Min-3-SAT instance.

Since Vertex Cover is NP-complete and we've demonstrated a polynomial-time reduction from Vertex Cover to Min-3-SAT, Min-3-SAT is NP-hard.

Considering both the points, we conclude that Min-3-SAT is NP-complete.