1. Consider the following three definitions in a connected undirected graph $G = (V, E)$:

   - **Minimum Vertex Cover** (MVC) is a subset V' of V with minimum size such that every edge (u, v) in E has at least one end in V'.
   - **Maximum Clique** (MaxCliq) is a complete subgraph of G that has a maximum size.
   - **Maximum Independent Set** (MIS) is a subset V' of V with maximum size such that for all u and v in V', edge (u, v) is not in E, called the *independence* property.

   - Explain how these problems relate with each other.

   - Write the decision problem of each one of the problems.

   Solution. If V' is a min vertex cover in G, then $V - V'$ is a max independent set in G and $V - V'$ is a max clique in the complement of G, denoted $\bar{G}$, where $\bar{G} = (V, \bar{E})$ and $\bar{E}$ contains all edges that do not exist in G.

   Decision problem of Minimum Vertex Cover (MVC):

   - INPUT: An undirected graph $G = (V, E)$ and an integer $k \leq |V|$.
   - DECISION: Does G contain a vertex cover with size at most k?

   Decision problem of Maximum Clique (MaxCliq):

   - INPUT: An undirected graph $G = (V, E)$ and an integer $k \leq |V|$.
   - DECISION: Does G contain a clique with size at least k?

   Decision problem of Maximum Independent Set (MIS):

   - INPUT: An undirected graph $G = (V, E)$ and an integer $k \leq |V|$.
   - DECISION: Does G contain an independent set with size at least k?

2. Prove that MIS is NP-complete.

   Solution. *NP membership*: A certificate would be a subset of vertices V' in V. To verify whether V' is an MIS, we check if there is an edge between any pair of vertices in V'. This can be done in polynomial time deterministically. Therefore, MIS is in NP.

NP hardness: We reduce MVC to MIS. Let the graph $G = (V, E)$ be an instance of MVC with a vertex cover V' of size k. We consider G to be the instance of MIS too. As a result, G has an MIS of size $|V| - k$ that contains all vertices in $V - V'$. We know we cannot improve this MIS because any other vertex added to it would break the independence property; i.e., would result in at least two vertices that are connected by an edge. Therefore, the instance of MVC has a minimum vertex cover of size k if and only if the instance of MIS has a maximum cover of size $|V| - k$.

3. Prove the NP-completeness of MaxCliq by a reduction from MVC.

   Solution: *NP membership*: Given an instance $G = (V, E)$ and k of MaxCliq and a certificate of V' of size k, we can check if there is an edge between every pair of vertices in V'. This verification can be done in polynomial time deterministically. Therefore, MaxCliq is in NP.

   NP hardness: We reduce MVC to MaxCliq. Let $G = (V, E)$ be an instance of MVC with a cover $V' \subseteq V$ of size k. Observe that, $V - V'$ cannot contain an edge (w, z), where both w and z are in $V - V'$ because then (w, z) would not be covered by V'. That is, V' is not a valid vertex cover. Thus, *there are no pairs of vertices in $V - V'$ that are connected by an edge.*

   Now, we create the complement of G, denoted $\bar{G}$, where $\bar{G}$ has all vertices in V. Moreover, $\bar{G}$ has an edge (u, v) if and only if $(u, v) \notin E$. That is, $\bar{E} = \{(u, v) | (u, v) \notin E\}$. Obviously, the construction of $\bar{G}$ can be done in polynomial time in $|V|$. Based on the above observation, every pair of vertices in $V - V'$ has an edge that connects them in $\bar{G}$. Thus, $V - V'$ is a clique in $\bar{G}$. Moreover, $V - V'$ is a max clique because $V - V'$ is an MIS in G (i.e., V' is a minimum vertex cover in G. Therefore, G has a minimum vertex cover of size k if and only if $\bar{G}$ has a max clique of size $|V| - k$.

4. The SAT problem is NP-complete, in general. A 2-SAT formula is a conjunction of a set of disjunctions (i.e., clauses) such that each clause has at most two literals. The decision problem of 2-SAT asks whether or not a given 2-SAT formula is satisfiable.

   - Prove that the 2-SAT problem is in P.

   - Now, consider the following decision problem:

   MAX-2-SAT: Given a 2-SAT formula with n propositional

variables and m clauses, does there exist a value assignment that satisfies *at least* k clauses, where $k \leq m$?

Prove that MAX-2-SAT is NP-complete.

Solution. To show 2-SAT is in P, let Φ be a 2-SAT formula with n Boolean variables $x_1, \cdots, x_n$ and m clauses with at most two literals in each clause. Create a directed graph $H = (V, E)$, where H has $2n$ vertices corresponding to $x_1, \cdots, x_n$ and $\neg x_1, \cdots, \neg x_n$. Construct the edges of E as follows: for each clause $(l_j \vee l_k)$ in Φ include an arc from $\neg l_j$ to l_k, and an arc from $\neg l_k$ to l_j. Then, prove the following:

Φ is satisfiable if and only if there is no strongly connected component in graph H that contains a variable and its negation.

Proof left for you.

NP membership of MAX-2-SAT: Verifying that a given assignment satisfies at least k clauses can be done in polynomial time with a deterministic algorithm. Therefore, MAX-2-SAT is in NP.

NP-hardness of MAX-2-SAT: We reduce 3-SAT to MAX-2-SAT. Consider an instance of 3-SAT, denoted ϕ, with n Boolean variables and m clauses each with exactly 3 literals. For each clause $C_j = (l_1 \vee l_2 \vee l_3)$ in 3-SAT, introduce a new variable y_j, $1 \leq j \leq m$, and create the following 10 clauses:

(l_1)
(l_2)
(l_3)
(y_j)
$(\neg l_1 \vee \neg l_2)$
$(\neg l_2 \vee \neg l_3)$
$(\neg l_1 \vee \neg l_3)$
$(l_1 \vee \neg y_j)$
$(l_2 \vee \neg y_j)$
$(l_3 \vee \neg y_j)$

To construct the instance of MAX-2-SAT, denoted ψ, we replace each clause C_j in ϕ with the conjunction of 10 clauses as above, which means ψ has $10m$ clauses and $n + m$ variables. This can obviously be done in polynomial time in the size of 3-SAT instance. Is there a truth value assignment for ψ such that at least $7m$ clauses in ψ evaluate to true?

Lemma: If C_j is satisfiable, then there is a value assignment to y_j, that satisfies 7 out of 10 clauses. Otherwise, for all value assignments to y_j, at most 6 clauses are satisfied. Proof left for you. (*Hint*: enumerate all possible 8 cases of value assignments to l_1, l_2 and l_3).

Correctness: If the 3-SAT instance, ϕ, is satisfiable, then the MAX-2-SAT instance will have a satisfying assignment that maximizes the number of true clauses. The reason behind it is that all clauses in ϕ evaluate to true, and based on the above lemma, we showed that each true clause will result in maximum number of clauses that can be true in its corresponding 10 clauses created in the mapping. If ϕ is not satisfied, then there is some clause C_k that evaluates to false. As a result, we can have at most 6 clauses being true in the corresponding set of 10 clauses we create corresponding to C_k (in the instance of MAX-2-SAT ψ).

5. Consider the k-cluster problem defined as follows:

   - **Input**: A set of points $P = \{p_1, \cdots, p_n\}$ in the Cartesian space (where $n > 1$) and a positive integer $k \leq n$.
   - **Output**: A partition of the points into k clusters $C_1, \cdots, C_k$ such that the diameter of the clusters is **minimized**. The *diameter* of the clusters is defined as follows:

     $\Delta = \text{Max}_j\ (\text{Max}_{x,y \in C_j}\ d(x, y))$ where $d(x, y)$ is the Euclidean distance of x and y.

   The k-cluster is known to be NP-hard. An approximation algorithm for solving this problem is as follows. Let $d[x]$ denote the distance of any point $x \in P$ to its closest center.

   Step 1: For any point $x \in P$, set $d[x]$ to infinity. Also, let S be the set of the k center. Initially, $S = \emptyset$.

   Step 2: For $i := 1$ to k do
   - Let $x \in P$ be the point with maximum distance;
     i.e., $d[x]$ is max amongst all n points.
   - Add x to S.
   - For $i := 1$ to n do
     Set $d[p_i]$ to the minimum of $d[p_i]$ and distance(x, p_i).

- Set the diameter of the cluster, denoted Δ, to the max of all $d[x]$, for any $x \in P$.

Step 3: return S and Δ.

Prove that the above algorithm gives a 2-approximation solution; i.e., the resulting diameter is at most twice as the optimal diameter.

Solution: Let Δ_{OPT} be the optimal diameter and Δ_i be the diameter calculated in each iteration i of the outter for-loop in the algorithm. First, we make some observations.

Observation 1. For $1 \leq i \leq k+1$, we have $\Delta_{i+1} \leq \Delta_i$.

Notice that when we include a new center, the distance of each point to its closest center will either be the same or be decreased. Therefore, the max of distances cannot increase either.

Observation 2. In each iteration i, every pair of centers c_1 and c_2 have a distance of at least Δ_{i-1}.

Let Δ_{i-2} be diameter of the first $i-1$ centers in iteration i. By Observation 1, we have $\Delta_{i-2} \geq \Delta_{i-1}$. The newly added center c_i has a distance Δ_{i-1} from its closest center. Thus, the distance of c_i from all other $i-1$ centers is at least Δ_{i-1}.

Now, we claim that, for any set C of k cluster centers, we have $\Delta_C \geq \Delta_S/2$, where Δ_S is the diameter returned by the algorithm.

We know that every point in P is within distance Δ_C of some center of C. This also holds for the solution returned by the algorithm; i.e., $S \subseteq P$. Now, imagine that we continued the outter for-loop for one more iteration, and computed a new center c_{k+1}. This way, we have $k+1$ centers $\{c_1, \ldots, c_k, c_{k+1}\}$. At least two of the centers c', c'' in $\{c_1, \ldots, c_k, c_{k+1}\}$ must be in the same cluster of C by the pigeon hole principle. Let the center of the cluster in C that contains c', c'' be c_e. Thus, we have $max(d[c'], d[c'']) \leq \Delta_{c_e}$. Observation 2 implies that the distance between c' and c'' is greater than or equal to Δ_S, which is the Δ computed in iteration k. Considering the three points c', c'', c_e and the triangle inequality, we have

$$\Delta_S \leq dist(c', c'') \leq d[c'] + d[c''] \leq \Delta_{c_e} + \Delta_{c_e} \leq \Delta_C + \Delta_C = 2\Delta_C$$

Since we have $\Delta_S \leq 2\Delta_C$ for any arbitray clsuter C for k centers, the inequality should also hold for the optimal cluster too. Therefore, $\Delta_S \leq 2\Delta_{OPT}$.