1. Show that the following TRIANGLE decision problem belongs to P.

Input: An undirected graph $G = (V, E)$.

Question: Does $G$ contain a “triangle”, i.e., a subset of three vertices with all edges between them present in the graph?

**Solution:** The following algorithm decides TRIANGLE.

On input $G$:

For each triplet of vertices $(u, v, w)$ in $G$:

Return True if $G$ contains all edges $(u, v), (v, w), (w, u)$.

Return False.

By definition of TRIANGLE, the algorithm will return True iff $G$ contains a triangle.

Let $n = |V|$ (number of vertices) and $m = |E|$ (number of edges) in $G$. There are $\binom{n}{3} = \Theta(n^3)$ many triplets of vertices in $G$, and it is possible to enumerate them one by one in time $O(n^3)$. For each triplet, it takes time $O(m)$ to verify the presence of the three edges (depending on how $G$ is encoded, this could be reduced). So the algorithm runs in time $O(mn^3)$.

2. Show that the following CLIQUE decision problem belongs to NP.

Input: An undirected graph $G = (V, E)$ and a positive integer $k$.

Question: Does $G$ contain a $k$-clique, i.e., a subset of $k$ vertices with all edges between them present in the graph?

For example, the shown graph contains a 3-clique (there are sets of 3 vertices with all edges between them, e.g., $\{a, b, c\}$), but it does not contain a 4-clique (every set of 4 vertices is missing at least one edge, e.g., $\{a, b, c, d\}$ is missing $(b, d)$).

**Solution:**

Verifier for CLIQUE:

On input $< G, k, c >$, where $c$ is a subset of vertices:

Return True if $c$ contains $k$ vertices and $G$ contains edges between all pairs of vertices in $c$; return False otherwise.
Verifier runs in polytime (where $n = |V|, m = |E|$): checking all pairs of vertices in $c$ takes time $O(k^2m)$ ($O(k^2)$ pairs in $c$, times $O(m)$ for each one).
If $<G, k> \in \text{CLIQUE}$, then verifier returns True when $c$ is a $k$-clique of $G$;
if verifier returns True for some $c$, then $<G, k> \in \text{CLIQUE}$ ($c$ is a $k$-clique).

CLIQUE $\in P$? Unknown (checking all possible subsets not polytime because $k$ not fixed, part of input).

Contrast CLIQUE with TRIANGLE: TRIANGLE $\in \text{NP}$ (on input $<G, c>$, check $c$ encodes a triangle in $G$), but TRIANGLE $\in \text{P}$ as well.

What’s the difference? Same algorithm to decide CLIQUE takes time $O(n^{k+1})$, except that $k$ is part of the input (instead of being fixed) so this could be as bad as, e.g., $O(n^{n/2})$ – not polytime.

3. Show that the following IndependentSet (IS) decision problem belongs to NP.
Input: An undirected graph $G = (V, E)$ and a positive integer $k$.
Question: Does $G$ contain an independent set of size at least $k$, i.e., a subset of vertices $I \subseteq V$ such that $|I| \geq k$ and $G$ contains no edge between any two vertices in $I$?

Solution: Verifier for IS:
On input $(G, k, c)$, where $c$ is a subset of $k$ vertices of $G$:

Return True if $G$ does not contain any one of the edges between vertices in $c$; return False otherwise.

This takes time $O(k^2m)$: there are $O(k^2)$ pairs of vertices in $c$ and $O(m)$ edges to check for each one.
Also, if there is some value of $c$ such that the verifier returns True for $(G, k, c)$, then $G$ contains an independent set of size $k$ or more ($c$ is such an independent set), and if $G$ contains an independent set of size $k$ or more, then there is some value of $c$ such that the verifier returns True for $(G, k, c)$ (let $c$ be the independent set).
It does not appear likely that IS $\in \text{P}$, because checking every subset of $k$ vertices takes more than polynomial time (time $\Omega(n^k)$ where $k$ can depend on $n$), and there is no obvious way to speed this up.

4. Show that the following UNARY-PRIMES decision problem belongs to P.
Input: $1^n$ (i.e., a string of 1’s of length $n$).
Question: Is $n$ prime?

Solution: The following algorithm decides UNARY-PRIMES:
On input $1^n$:

For $k = 2, 3, ..., n - 1$:

If $k$ divides $n$, return False

Return True if no value of $k$ worked.

The algorithm returns True iff $n$ is prime, by definition. The division can be carried out by repeated subtraction, which takes time $O(n^2)$ for each value of $k$, so the entire algorithm runs in time $O(n^3)$.

**NOTE:** This works because $n$ is the size of the input at the same time as the value of the input. For any other base, this would NOT work because the value $m$ would be represented using $n = \log m$ many digits so the size would be proportional to $n = \log m$ and the running time would become exponential (as a function of $n$).