1. Professor Amongus is researcher that keeps trying to prove $P = NP$.

   (a) First he showed that a decision problem L is polynomial-time reducible to an NP-complete problem M. Moreover, after 80 pages of dense mathematics, he has also just proven that L can be solved in polynomial time. Has he just proven that P = NP ? Why, or why not?

   Solution: Professor Amongus reduced a problem in P to a problem in NP. To show $P = NP$, he would have to reduce a problem in NP to a problem in P. We already know that every problem in P is already in NP. ■

   (b) Next, he designed an algorithm that can take any graph G with n vertices and determine in O(nk) time whether G contains a clique of size k. Does Professor Amongus deserve the Turing Award for having just shown that P = NP? Why or why not?

   Solution: This is not a polynomial-time algorithm, since k is a number in the input, not the input size. Moreover, k is not a constant, as would be required of a general polynomial-time algorithm for the CLIQUE problem. ■

2. **Cyclic paths of hell.** A *Hamiltonian cycle* in a graph is a cycle that visits every vertex exactly once. A *Hamiltonian cycle* in a graph is a path that visits every vertex exactly once, but it need not be a cycle (the last vertex in the path may not be adjacent to the first vertex in the path.)

   Consider the following three problems:

   - *Directed Hamiltonian Cycle* problem: checks whether a Hamiltonian cycle exists in a *directed* graph,
   - *Undirected Hamiltonian Cycle* problem: checks whether a Hamiltonian cycle exists in an *undirected* graph.
   - *Undirected Hamiltonian Path* problem: checks whether a Hamiltonian path exists in an *undirected* graph.

a. Give a polynomial time reduction from the *directed* Hamiltonian cycle problem to the *undirected* Hamiltonian cycle problem.

Solution: For any arbitrary directed graph $G_d := \{V_d, E_d\}$, construct the following undirected graph $G_u := \{V_u, E_u\}$:

- $V_u := \left\{ v_{in}, v_{mid}, v_{out} \middle| v \in V_d \right\}$. For each of the vertices in the directed graph, we split them into a triplet of in, mid, and out.
- $E_u := \left\{ \left(u_{out}, v_{in}\right) \middle| (u,v) \in E_d \right\} \cup \left\{ \left(v_{in}, v_{mid}\right), \left(v_{mid}, v_{out}\right) \middle| v \in V_d \right\}$. For each of the triplets that comes from the same vertex, we connect them in the order of in-mid-out. The directed edges in the V_d become the undirected ones that connect out and in between corresponding triplets.

Notice that $\left|V_u\right| = 3\left|V_d\right|$ and $\left|E_u\right| = \left|E_d\right| + 2\left|V_d\right|$, so this reduction is linear.

$\Rightarrow$: Suppose that in G_d there exists a Hamiltonian cycle $C_d := \left(c_1, c_2, \ldots, c_{|V_d|}\right)$, where $c_i \in V_d$. Then in G_u there should also exist

$$C_u := \left(c_{1in}, c_{1mid}, c_{1out}, c_{2in}, c_{2mid}, c_{2out}, \ldots, c_{|V_d|_{in}}, c_{|V_d|_{mid}}, c_{|V_d|_{out}}\right),$$

which is a Hamiltonian cycle in G_u.

$\Leftarrow$: Suppose that in G_u there exists a Hamiltonian cycle C'_u. By definition, within each of the triplets there should only be a path of order in-mid-out, and between two triplets there should only be an edge of out-in. Thus, C'_u should always be of the following form

$$C'_u := \left(c'_{1in}, c'_{1mid}, c'_{1out}, c'_{2in}, c'_{2mid}, c'_{2out}, \ldots, c'_{|V_d|_{in}}, c'_{|V_d|_{mid}}, c'_{|V_d|_{out}}\right),$$

which corresponds to a Hamiltonian cycle $C'_d := \left(c'_1, c'_2, \ldots, c'_{|V_d|}\right)$ in G_d.

■

b. Give a polynomial time reduction from the *undirected* Hamiltonian Cycle to *directed* Hamiltonian cycle.

Solution: This reduction is simpler than the previous one. Given an instance G of undirected Hamiltonian cycle, Let G' be the directed graph with the same vertices as G and containing edges $u \to v$ and $v \to u$ for every edge $uv \in G$.

($\Rightarrow$) If C is a cycle in G, then C is also a cycle in the directed graph G'. For every $u \to v \in G'$, the edge uv is in G.

($\Leftarrow$) If C is a cycle in G', then C is also a cycle in the original graph G. For every $uv \in G$, the edge $u \to v \in G'$ by the construction.

■

c. Give a polynomial-time reduction from an undirected Hamiltonian *path* to an undirected Hamiltonian *cycle*.

Solution: Let the input to this problem be an undirected graph G. The goal is to produce G' such that G has a Hamiltonian path if G' has a Hamiltonian cycle.

This can be done by adding a vertex v with edges to every vertex in the original graph G, this will be G'.

$\Rightarrow$: If there exists a Hamiltonian path P in G, starting with vertex s and ending with vertex t. Then $[v, s, P, t, v]$ is a Hamiltonian cycle in G'.

$\Leftarrow$: In the other case, if C is the Hamiltonian cycle in G', then removing v from C will return a Hamiltonian path in G.

■

3. We have the following two problems:

   - LONGEST-PATH-LENGTH - you are given a undirected graph ($G = (V, E)$) and returns the number of edges in the longest simple path between any two vertices in G.
   - LONGEST-PATH - you are given a graph $G = (V, E)$, two vertices $u, v \in V$, an integer $k \geq 0$, and the problem outputs whether or not there is a simple path from u to v in G containing at least k edges.

   Show that the optimization problem, LONGEST-PATH-LENGTH can be solved in polynomial time if and only if LONGEST-PATH $\in P$

Solution: First let's show LONGEST-PATH-LENGTH $\implies$ LONGEST-PATH:

For every pair of vertices and value of k from 1 to n, run the polynomial time algorithm for LONGEST-PATH. Whenever you receive a "no" results (for vertices s and t and $k = x$), that's when you know the longest path from $s - t$ is $x - 1$.

This algorithm takes $O(n^3 \cdot A_{LP})$ time where A_{LP} is the running time of the LONGEST-PATH algorithm. If it is polynomial, then the total running time would be $O(n^{3+k})$ which is also polynomial.

So this proves that if LONGEST-PATH is in P, then LONGEST-PATH-LENGTH is in P. But what if LONGEST-PATH is not in P? What we did doesn't necessarily prove LONGEST-PATH-LENGTH won't be in P.

To do that we need to show LONGEST-PATH $\implies$ LONGEST-PATH-LENGTH. But this reduction is trivial right? If LONGEST-PATH-LENGTH is in P then we can easily just have a check that determines if its output $\leq k$.

This way we show that the two problems are polynomial time equivalent (LONGEST-PATH-LENGTH $\iff$ LONGEST-PATH).

■

4. For each of the following problems, pick true or false and explain why.

   (a) **True/False:** If L is an NP-complete language and $L \in P$, then $P = NP$.

   Solution: True. If L is an NP-complete language and $L \in P$, then $P = NP$. This is because NP-complete problems are the hardest problems in NP, and if any NP-complete problem is in P, then all problems in NP are in P. ■

   (b) **True/False:** There exists a polynomial-time reduction from every problem in NP to every problem in P.

   Solution: False. There does not necessarily exist a polynomial-time reduction from every problem in NP to every problem in P. A reduction must map a problem in NP to another problem in NP or a problem in P to another problem in P. ■

   (c) **True/False:** If a problem is both NP-hard and co-NP-hard, then it must be in NP.

   Solution: False. If a problem is both NP-hard and co-NP-hard, it does not necessarily imply it is in NP. Being NP-hard means every problem in NP can be reduced to it in polynomial time, and co-NP-hard means every problem in co-NP can be reduced to it in polynomial time. ■

   (d) **True/False:** If there is a polynomial-time reduction from problem A to problem B and B is in NP, then A must also be in NP.

   Solution: True. If there is a polynomial-time reduction from problem A to problem B and B is in NP, then A must also be in NP. This is because the reduction allows us to transform A into B, and since B is in NP, the transformed instance can be verified in polynomial time, implying that A is also in NP. ■

   (e) **True/False:** If a problem is solvable in polynomial space, then it is also solvable in polynomial time.

   Solution: False. If a problem is solvable in polynomial space, it does not necessarily mean it is solvable in polynomial time. Polynomial space includes problems that might take exponential time but use polynomial memory, such as PSPACE-complete problems. ■