1. ***No fast automata checkers.*** Consider the following problem about NFAs.

> NFA-REJECT
> - ***Input:*** *An n-state NFA N.*
> - ***Output:*** *Does NFA N reject at least one string? In other words, is $L(N) \neq \Sigma^*$?*

Prove that the NFA-REJECT problem is NP-complete.

Solution: It is sufficient to demonstrate that NFA-REJECT is both NP-hard and in NP. To certify that NFA-REJECT is in NP, the prover can provide a string w rejected by N, and the verifier can run the NFA N on w in polynomial time and check if w is in fact rejected.

We prove that NFA-REJECT is NP-hard by reduction from the well-known NP-hard problem 3SAT. Let ϕ be an arbitrary 3-CNF formula for 3SAT. We will construct an NFA N from ϕ, such that ϕ is a satisfiable formula if and only if N rejects at least one string.

For each clause C in ϕ, we construct the following gadget in N: Take 8 distinct directed paths, each of $n+1$ edges, where each of the first n vertices corresponds to a variable, and the last vertex has a self-loop.

- If the k-th variable is not in the clause C, we add the label `0`, `1` to the k-th edge of each path.
- Label the three edges corresponding to the three variables in clause C with all the 8 possible `0`/`1` labels, one for each path.
- Add the label `0`, `1` to the self-loop on the last vertex.

We add an auxiliary starting state, with ε-transitions to the first vertex of all paths in every clause gadget. Set all the vertices to be accepting, except for the following 7 vertices in each clause gadget: exactly one of the 8 paths corresponds to a non-satisfying assignment for the clause; make the n-th/second-to-last vertex in the other 7 paths to be rejecting.

To see why we construct NFA N this way, let's first prove that a satisfiable formula ϕ turns into an NFA N rejecting at least one string. Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ be a satisfying assignment for ϕ. Let string w be $\alpha_1 \ldots \alpha_n$. We claim that N rejects w:

- Using the ε-transitions we put multiple (in fact, $8 \cdot$ #clauses many) fingers on the first vertex of each path in all the clause gadgets.
- On reading symbol α_i, some fingers might be dropped if (1) the finger lies in a clause gadget with variable x_i, and (2) the path where the finger is does not correspond to assigning x_i to α_i.
- After reading the whole w, exactly one finger is left in each clause gadget, and must be on that path that corresponds to the assignment α.
- Since α is satisfying, the state where the finger is at must be rejecting by construction.

Therefore NFA N rejects w.

For the opposite direction, we prove that any string w rejected by N can be interpreted as a satisfying assignment. First we argue that w must have length n: any string shorter than n will left some fingers in the middle of some path which must be accepting; and any string longer than n must successfully travel through at lease one path from each clause and stays at the last vertex of the path with a self-loop, which is also accepting. Now if w has length n, from the above analysis there will be exactly one finger left per gadget, and the only way that all the fingers are on rejecting states is that the corresponding assignment α is satisfying each clause.

To summarize, the clause gadget on reading a string w of `0`/`1`-symbols will end up at a rejecting state if and only if the string $w = \alpha_1 \ldots \alpha_n$ has exactly n symbols and the corresponding assignment α satisfies clause C. All the construction can be carried out in polynomial time as the size of N is roughly n times the number of clauses in ϕ. ■

Rubric: Proving NP-hardness through mapping reduction comes in five steps:

- state the direction of reduction *from* a known NP-hard problem A *to* your problem B;
- describe how to construct a *specific* instance of B from an *arbitrary* instance of A;
- argue that yes-instance of A turns into yes-instance of B by your construction;
- argue that no-instance of A turns into a no-instance of B by your construction; or equivalently, yes-instance of B *that can be generated from the construction* comes from yes-instance of A (this tends to be the difficult part of the proof);
- explain why the construction can be done in polynomial time.

It is important to remember that the string w is *not* part of the input in NFA-Reject, and therefore you don't get to restrict the length of w when making the argument; you must prove that the NFA can handle *all inputs*, in particular when the length of w is not n.

Standard 5-point grading scale (plus deadly-sins and sudden-death rules) for the problem. Maximum 3 points if the case when w has length other than n is not handled correctly. Maximum 3 points if the clause gadget does not have the property that on reading w ends up at a rejecting state if and only if w has length n and corresponds to an assignment satisfying the clause (in particular, single-path constructions are most likely incorrect). Maximum 1 point if the direction of reduction is wrong.

2. ***A friend of a friend is my friend.*** Modern social networks model the relationship between people as *graphs*. We want to find a social subgroup within the network such that every person in the subgroup knows each other.

> k-CLIQUE
> - ***Input:*** *An undirected graph G.*
> - ***Output:*** *Is there a clique (complete subgraph) of size k in G?*

(a) Prove that the k-CLIQUE problem cannot be solved efficiently under the exponential-time hypothesis. What is the best lower bound you can get?

(For full credit, your lower bound has to imply that the $(\log n)$-clique problem cannot be solved in polynomial time.)

Solution: We reduce the 3SAT problem to k-CLIQUE.

For an arbitrary 3CNF formula ϕ, construct a graph H as an instance to the k-CLIQUE:

- Partition the m clauses of ϕ into k groups, each of size m/k.
- For each group containing m/k literals, there are at most $3m/k$ many variables relevant to these clauses; construct up to $2^{3m/k}$ many vertices in H, one for each possible assignment that also satisfies all the m/k clauses in the group simultaneously.
- Between any two vertices in H from different groups, add an edge between them if the variable-choices are *consistent*; in other words, if in one group variable x_i is assigned with a particular value (say `0`), then in the other group x_i is also assigned with the same value `0`.

The constructed graph H has $2^{3m/k} \cdot k$ vertices and thus size at most $(2^{3m/k} \cdot k)^2$. Each vertex can be constructed in $O(n+m)$ time (checking satisfiability), and each edge in $O(n)$ time (checking consistency). Thus the construction itself can be carried out in $O(n+m) \cdot 2^{O(m/k)}$ time (when $k = \log m$), which is near-linear in the size of H.

To prove that the reduction work, one observe that a k-clique exists in H if and only if the corresponding variable assignments are consistent and satisfies all the clauses in ϕ.

Now any algorithm solving k-CLIQUE in $|H|^{o(k)}$ time will solve 3SAT in time

$$O(n+m) \cdot 2^{O(m/k)} + 2^{O(m/k) \cdot o(k)} = 2^{o(m)},$$

which violates the string exponential-time hypothesis. This shows that k-CLIQUE cannot be solved in $|H|^{o(k)}$ time. ■

> **Rubric:** Standard 5-point grading scale (plus deadly-sins and sudden-death rules) for the problem. Maximum 3 points if the lower bound is of the form $2^{o(k)}$.