1. (Adapted from Sipser 7-44) We have seen (but not proved) in class that 2SAT = \{ \phi \mid \phi \text{ is a satisfiable 2-CNF formula}\} \in \mathcal{P}$. We’ll prove it here. Suppose we have 2-CNF for-

mula $\phi$ on $n$ variables $x_1, \ldots, x_n$ with $m$ clauses, where WLOG each clause is nontrivial (i.e. there are no clauses of the form $(x \lor \neg x)$; note that this assumption is WLOG because removing trivial clauses does not affect the truth of any assignment). We’ll use $\phi$ to create a directed graph as follows: create two nodes for each variable, corresponding to $x$ and $\bar{x}$. Next, for each clause $(x \lor y) \in \phi$ add edges $\bar{x} \rightarrow y$ and $\bar{y} \rightarrow x$. Prove that this process yields a polytime algorithm for 2SAT, i.e. prove that $G_{\phi}$ can be constructed and used to check the satisfiability of $\phi$ in poly$(n,m)$ time.

2. Given a set of (finite) binary strings $S = \{s_1, \ldots, s_k\}$ we say that a string $s$ is a concatenation over $S$ if $s = s_{i_1} \ldots s_{i_t}$ for some $i_1, \ldots, i_t \in \{1, \ldots, k\}$. Now consider two sets of binary strings $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$, and suppose we want to know if there exists a string $s$ that is both a concatenation over $A$ and a concatenation over $B$. Note that it’s not even obvious that this problem is in NP - what if this magical string $s$ is exponentially long? Deal with this by proving the following claim: if there exists a string $s$ that is a concatenation over both $A$ and $B$ (i.e. is a concatenation over $A$ and is also a concatenation over $B$) then there exists a string that is a concatenation over $A$ and $B$ whose length is at most $n^2 x^2$ where $x$ = the length of the longest string in $A \cup B$.

3. Define the Component Union Problem (CUP) as follows: given a disconnected graph $G$ and some number $k$, does there exist a subset of its components whose union has exactly $k$ nodes? Now consider the following “proof” that CUP is NP-hard:

We’ll reduce from Subset Sum. Given an instance of Subset Sum with binary numbers $n_1, \ldots, n_m$ and target $N$, we construct a CUP instance as follows: for each $i$ construct a path $P_i$ of length $n_i$, and let $G$ be the union of paths $P_1, \ldots, P_m$, each of which is a separate connected component. Finally, set $k = N$. Since $G$ has a set of connected components whose union has size $k \iff$ some subset of $n_1, \ldots, n_m$ add up to $N$, it follows that CUP is NP-hard.

This proof has an error. What is it?

4. You are planning a wedding reception, and one of your jobs is to decide which guest will sit at which table. You want to avoid creating an awkward table (a table at which no one has any shared interests) but are not even sure if such a table exists. The Awkward Table Problem (ATP) is therefore as follows: given a set of interests $I = \{i_1, \ldots, i_n\}$ and a set of $m$ guests such that each guest $G$ has their own subset of interests $I_g \subset I$, if each table has $k$
people, is it possible to have an awkward table (i.e. does there exist $S \subset \{1, \ldots, n\}$ such that $|S| \geq k$ and for any two $i, j \in S$, $I_i \cap I_j = \emptyset$)?

Prove that ATP is NP-complete.

5. While planning the wedding reception from #1, you learn that $n$ additional guests will be attending. You have already assigned existing guests to $m$ tables, and each table can accommodate $k$ new guests. This yields a Conflicting Table Problem (CTP): each of the $n$ additional guests has an associated subset of the other $n - 1$ additional guests with which they cannot share a table. (Note that there are no conflicts between additional guests and existing guests.) Is there a way to fit all $n$ additional guests into the $m$ tables so that no table is overloaded and each additional guest sits at a conflict-free table?

Prove that CTP is NP-complete.

6. The Hamiltonian Cycle Problem (HCP) is as follows: given graph $G$, does $G$ contain a Hamiltonian cycle, i.e. a cycle that visits each node in $G$ exactly once (except for its starting node, which it visits twice)?

Prove that HCP is NP-complete.

7. You are preparing Thanksgiving dinner for your friends. You have a set of ingredients and a group of friends, each of whom has their own set of favorite ingredients from your set of ingredients. You want everyone to have something they like, but you also want to do as little cooking as possible. The Good Thanksgiving Problem (GTP) therefore asks: given set of $n$ ingredients $I$ and friends’ favorite ingredients $F_1, \ldots, F_m \subset I$, can you cook a dinner in which everyone has an ingredient they like, using at most $k$ ingredients (i.e. does there exist $H \subset I$ such that $|H| \leq k$ and $H \cap F_i \neq \emptyset$ for all $1 \leq i \leq m$)?

Prove that GTP is NP-complete.

8. While preparing for Thanksgiving in #4, as stress relief you decide to cook something that tastes as strange as possible. Specifically, you are given $n$ ingredients and for each pair of ingredients $i, i'$ know either $i$ tastes good with $i'$ or $i$ does not taste good with $i'$, and you want to know if you can combine at least $k$ ingredients such that not only do no two of the ingredients taste good together, no two of the ingredients have a shared third ingredient that they both taste good with. Mathematically, the Strange Dish Problem (SDP) is as follows: given $n$ ingredients arranged as vertices $v_1, \ldots, v_n$ in a graph, where an edge between ingredients indicates that they taste good together, determine whether or not there exists a set of $k$ ingredients $v'_1, \ldots, v'_k$ such that no two $v'_i, v'_j$ are neighbors and no two $v'_i, v'_j$ have a common neighbor.

Prove that SDP is NP-complete.

9. Suppose you have an oracle for solving MAX-CLIQUE, i.e. a black-box subroutine that takes as input a graph $G = (V, E)$ and outputs the size of the largest clique in $G$. Prove that you can find a max-size clique in $G$ using poly($|V|, |E|$) many calls to this oracle.