Problem 1

Recall that a clique of size \( n \) in a graph \( G = (V, E) \) is a subset of \( V \) of size \( n \) such that every pair of vertices in the subset is connected by an edge in \( E \).

In this problem, you will show that the language \( \text{Clique} \) is NP-complete:

\[
\text{Clique} = \{\langle G, n \rangle \mid G = (V, E) \text{ is an undirected graph and } G \text{ contains a clique of size at least } n\}
\]

To prove that \( \text{Clique} \) is NP-hard, you can use the following reduction from 3SAT to \( \text{Clique} \):

*Given a 3CNF Boolean formula, convert it to an undirected graph as follows: Create a vertex for each literal in each clause. Draw an edge from every vertex to every other vertex unless the two vertices correspond to literals from the same clause, or the two vertices correspond to literals that are negations of each other.*

Let \( k \) be the number of clauses in the 3CNF Boolean formula. Your job is to show that the constructed graph will have a clique of size \( k \) if and only if the Boolean formula is satisfiable.

a. Show that you understand the reduction by converting the following 3CNF Boolean formula to the associated graph:
CSCI 1010

Due: November 7, 2019

Literals: \(X = \{x_1, \overline{x}_1, x_2, \overline{x}_2, x_3, \overline{x}_3\}\)
Clauses: \(C = \{(x_1, x_2, x_3), (\overline{x}_2), (x_2, \overline{x}_3)\}\)

Note: Recall that 3CNF means there are at most three literals in each clause.

b. Explain why any satisfiable 3CNF Boolean formula is translated into a graph with a clique of size at least \(k\) using the above reduction.

c. Explain why having a clique of size at least \(k\) in a graph resulting from this reduction implies that the original 3CNF Boolean formula was satisfiable.

d. Describe an NTM that determines whether a graph has a clique of size at least \(k\). Use this NTM to explain why Clique is in the class NP.

e. How do parts (b), (c), and (d) help show that Clique is NP-complete?
   This can be a brief sentence or two.

Hint: Is there a connection between this reduction and the textbook’s reduction from 3SAT to IndependentSet?

Problem 2

Ascii is a cook tasked with the immensely important task of constructing tiny digital pancakes so that internet users from all around the world can flip pancakes to their heart’s content. However, since he has to cook so many pancakes, Ascii has created a machine that takes in stacks of uncooked pancakes and outputs completed pancakes. Now, he only has to build and stack the pancakes!

However, Ascii is running low on digital flour. In order to conserve materials, Ascii designs each pancake to have holes that align with a two-column, \(N\)-row grid. So that the machine recognizes these stacks as valid stacks, Ascii needs to stack the pancakes such that for every position in the grid, at least one pancake covers the position. (A pancake covers a position in the grid if it does NOT have a hole in that position.)

Ascii is allowed to flip a pancake along its vertical axis, but cannot rotate it or manipulate it in any other way, because otherwise the pancakes wouldn’t line up correctly. See Figure 1 for an example of a valid way to manipulate a pancake.
Define PancakeFlip to be the language consisting of sets of pancakes in which an arrangement exists such that every position in the grid is covered. Prove that PancakeFlip is NP-complete. (Recall from Homework 8, Lab Problem 2 the things that need to be shown in order to prove that a language is NP-complete.)

**Hint:** Think of the orientation of each pancake as an assignment to a Boolean variable.

![Figure 1: A pancakes’ two possible configurations.](image)

**Problem 3**

Prove that the following problem is NP-complete.
You’re given the language $2SATASSIGN = \{ \phi \text{ s.t.: there exist } \geq 2 \text{ satisfying assignments for } \phi \}$
($\phi$ is a Boolean formula)