Problem 1

Consider the following language:

\[ L_{01} = \{ \langle M \rangle \mid M \text{ is a TM such that } \forall x \in \{0,1\}^*, \]
\[ x \circ 0 \in L(M) \text{ if and only if } x \circ 1 \in L(M) \} \]

a. Can we use Rice’s Theorem to prove that \( L_{01} \) is undecidable? Justify your answer.

b. Prove by constructing a reduction from your favorite undecidable language that \( L_{01} \) is undecidable.

Problem 2

Consider the language

\[ L = \{ \langle M \rangle \mid M \text{ is a Turing machine such that, whenever } \]
\[ M \text{ halts (on any input), the tape is blank } \} \]

Prove that \( L \) is undecidable.
Problem 3

A Turing machine $M'$ is an extension of a Turing machine $M$ if $L(M) \subseteq L(M')$. A Turing machine $A$ is an extender if, on input a description of a TM $M$, it outputs the description of some extension, $M'$, of $M$. Consider the following language:

$$L_{\text{EXT}} = \{\langle A \rangle \mid A \text{ is an extender} \}$$

a. Is $L_{\text{EXT}}$ Turing-recognizable? Prove your answer.

b. Is $L_{\text{EXT}}^c$ Turing-recognizable? Prove your answer.

Problem 4

In a major battle of the Earth-Romulan war, the United Earth forces wanted to defeat the Romulans by disconnecting their network of space stations through strategic attacks on their supply lines. The network of Romulan space stations can be thought of as a connected, undirected graph $G = (V, E)$ where each vertex represents a space station and each edge represents a supply line between two stations.

Some supply lines are more well-defended than others, and the better defended a supply line is, the more it costs to attack it. Thus, every edge $e \in E$ has an attack cost $c(e)$ which is a positive integer. Moreover, some pairs of space stations are more important to disconnect than others, and the more important the pair is, the higher the payoff for disconnecting them. Thus each pair of vertices $u, v$ has a positive integer, $p(u, v)$ indicating what the payoff is for disconnecting them.

With limited resources, Starfleet wants to partition $V$ into two components, $S$ and $S^c = V - S$, to simultaneously do two things:

1. minimize the cost of attacking edges between $S$ and $S^c$
2. maximize the payoff they get by disconnecting the vertices in $S$ from the vertices in $S^c$.

To achieve both goals at once, they decide to partition $V$ in a way that minimizes the ratio of cost to payoff.
More precisely, for any set of vertices $S \subset V$, let $\delta(S)$ denote the set of edges that have exactly one endpoint in $S$. Then Starfleet’s ratio function is:

$$r(S) = \frac{\sum_{e \in \delta(S)} c(e)}{\sum_{u \in S} \sum_{v \in S^c} p(u, v)}$$

(You may worry about what happens when the denominator in the fraction above is 0. Do not worry! This never happens: here $S$ is a proper subset of the vertices, and the payoff is always positive.)

They then wish to create a decider for the language

$$\text{BattleStrategy} = \{ \langle G, c, p, x \rangle \mid G \text{ is a connected undirected graph } (V, E),$$

$$c \text{ is a cost function on edges,}$$

$$p \text{ is a payoff function on pairs of vertices,}$$

$$\text{and } x \text{ is a rational number s.t. } \exists S \subset V \text{ with } r(S) \leq x \}$$

You might have a good hunch that the decision problem $\text{BattleStrategy}$ is NP-hard. And you would be right! However, you won’t have to prove that for this problem. Luckily for Starfleet, the Romulan network of supply lines is a tree. We will show that this new language:

$$\text{TreeBattleStrategy} = \{ \langle G, c, p, x \rangle \mid G \text{ is a tree } (V, E),$$

$$c \text{ is a cost function on edges,}$$

$$p \text{ is an importance function on pairs of vertices,}$$

$$\text{and } x \text{ is a rational number s.t. } \exists S \subset V \text{ with } r(S) \leq x \}$$

is in fact in $P$.

a. Recall that for any two vertices in a tree, there is a unique path connecting the two vertices. For any tree $T = (V, E)$, and any edge $e \in E$, let

$$D(e) = \{ \text{pairs of vertices } u, v \mid \text{the unique path between } u \text{ and } v \text{ contains } e \}$$

Prove that for any $S \subset V$:

$$\sum_{u \in S} \sum_{v \in S^c} p(u, v) \leq \sum_{e \in \delta(S)} \sum_{(u, v) \in D(e)} p(u, v)$$
Hint: Think about which terms are in both the sum on the left and the sum on the right.

b. Prove that, given a tree $T = (V, E)$, and any $S \subset V$ such that $|\delta(S)| > 1$ (i.e. $\delta(S)$ contains more than one edge), there exists another subset $S' \subset V$ such that $|\delta(S')| = 1$ and $r(S') \leq r(S)$. This implies that there is some set $S$ that minimizes $r(S)$, such that $\delta(S)$ contains just one edge.

You may find the following arithmetic fact useful: for any positive numbers $a_1, \ldots, a_k$ and $b_1, \ldots, b_k$ we have:

$$\min_{1 \leq i \leq k} \frac{a_i}{b_i} \leq \frac{\sum_{i=1}^{k} a_i}{\sum_{i=1}^{k} b_i}$$

Hint: You can use part a.

c. Give an algorithm that decides TreeBattleStrategy in polynomial time.

Problem 5

After centuries of military action on the planet Funfoo, its fuzzy residents have all joined various mutually exclusive factions, all of which are mortal enemies. However, nobody particularly wants to fight, so a young civil engineer has come up with a brilliant idea for how to keep everyone alive: civic planning!

Given a graph of houses $(V, E)$, where $(v_1, v_2) \in E$ if $v_1$ is adjacent to $v_2$, the engineer wants to assign each house to a faction so that the Funfans do not break into fighting. Funfans don’t fight without major support from their allies, so an optimal assignment avoids placing many Funfans of the same faction together. However, Funfans refuse to live anywhere with no adjacent allies.

Given there are $k$ different factions to be housed, the engineer wants to find out whether an assignment exists such that $c$ or fewer houses have more than one adjacent member of the same faction. The engineer will use an unlimited supply of colored flags to assign each house to a particular faction - each faction has its own unique color, and each house will be populated

\footnote{Fun fact about Funfans: they can access higher dimensions, and so the graph $G$ need not be planar!}
by a single Funfan from the corresponding faction. Note that the engineer is not required to house a member of every faction (e.g. the city may house only two out of three factions, if that is a valid assignment).

Consider the following language: \textsc{FunfanCivicPlanning} = \{\langle G, k, c \rangle \text{ such that there exists an assignment of vertices in } G \text{ to } k \text{ factions such that (1) each vertex has a neighbor from the same faction; (2) no more than } c \text{ vertices have more than one neighbor of the same faction} \}.

a. Show that \textsc{FunfanCivicPlanning} \in \text{NP}.
b. Show that \textsc{FunfanCivicPlanning} is NP-hard.

**Problem 6**

An oracle for a language \(L\) is a magic subroutine that, on input \(x\), outputs \(\text{YES}\) if \(x \in L\), and \(\text{NO}\) if \(x \notin L\). You have seen oracles in HW 7 (the oracle for \(H_{TM}\)).

a. Give an algorithm that meets all three of the following conditions: (1) your algorithm has access to an oracle for a language \(L \in \text{NP}\) of your own choice; (2) on input a graph \(G\), your algorithm outputs a three-coloring of \(G\) if it exists, and rejects otherwise; (3) your algorithm runs in deterministic polynomial time.

In describing your algorithms, first define the language \(L\) for which your algorithm has an oracle, and show that \(L \in \text{NP}\). Then, give your algorithm, and analyze its running time and correctness. You may assume that it takes the oracle just one step to decide its language \(L\).

**Hint:** consider a language that consists of partially colored graphs; membership in this language should tell you something about how to color the remaining vertices.

b. Give a deterministic polynomial-time algorithm that, as in part (a), makes use of an oracle for a language \(L \in \text{NP}\) of your own choice, such that, on input a graph \(G\), it outputs a \(k\)-coloring for \(G\) such that \(k\) is optimal (i.e. no \((k - 1)\)-coloring exists).

Consider the following language:

\(\text{MINCOLORABILITY} = \{\langle G, k \rangle \mid G \text{ is } k\text{-colorable but not } (k - 1)\text{-colorable}\}\)
In part (b), you just showed that having an oracle for $L \in NP$ would allow deciding MinColorability in polynomial time. Does it follow that MinColorability $\in NP$? Actually it does not: if $NP \neq coNP$, then MinColorability $\not\in NP$, because now, in parts (c) and (d), you will see that it is both NP- and coNP-hard.

c. Show that MinColorability is NP-hard.

d. Show that MinColorability is coNP-hard. (It may be helpful to recall the proof we did in class that 3-colorability in NP-hard. Was the graph output by that reduction always 4-colorable?)