CMPS 6610 Problem Set 6

In this assignment we'll revisit some of the topics from our discussion of computational complexity and network flows.

To make grading easier, please place all written solutions directly in `answers.md`, rather than scanning in handwritten work or editing this file.

All coding portions should go in `main.py` as usual.

Part 1: coNP

Recall that the class "NP" was defined as the set of decision problems for which, given a particular solution, we could efficiently check if it is a YES instance to the decision problem (e.g., "Is F a satisfiable Boolean formula?"). The complexity class "coNP" is the "complement" of NP. That is, a problem X is in coNP if we can efficiently check that a candidate solution is a NO instance to X (e.g. "Is F an unsatisfiable Boolean formula?").

1a) Given a Boolean formula F that is a 3-CNF (an AND of clauses with three literals in an OR), consider the Tautology (TAUT) problem of idenfying whether *every* assigment produces a value of TRUE. Show that TAUT is coNP-complete.

1b) Prove that $P \subseteq coNP$.

Part 2: Hardness of Approximation

Given an undirected, unweighted graph $G = (V, E)$, the *Hamiltonian Path* (HP) problem asks us to find a path that visits every vertex in G exactly once. This problem is very similar to the Traveling Salesperson Problem (TSP), except that we don't require a weighted graph, or a cycle as the solution. And as we might expect, HP is NP-complete.Interestingly, we can use this fact to prove that it is NP-hard to approximate TSP to within a factor of 2. Here, you must show that if it is possible to find a 2-approximation to TSP in polynomial time/work, then it is possible to solve HP in polynomial time/work as well. As a hint, the reduction will require you to construct a weighted graph (with cleverly chosen weights) given an input graph to HP.

Part 3: Network Flow for Bipartite Matching

A *bipartite graph* $G = (A, B, E)$ with has the property that for every edge $(u, v) \in E$, $u \in A$ and $v \in B$. The *bipartite matching* problem asks us to find a maximal set of edges with no redundant endpoints.

3a) Solve the bipartite matching problem using a reduction to network flow.

3b) A *perfect matching* in a bipartite graph is possible when A and B are the same size. Use your reduction above to characterize when the network flow solution returns a perfect matching.

3c) A perfect matching is also possible in a standard unweighted graph $G = (V, E)$. Is it possible to identify a perfect matching in a standard graph using a reduction to network flow?