CS 6363.003 Homework 5

Due Saturday May 8th on eLearning

Please answer the following **4** questions, some of which have multiple parts.

1. A given flow network $G = (V, E)$ with source s, sink t, and capacity function $c : E \to \mathbb{R}_{\geq 0}$ may have more than one minimum (s, t)-cut.

   (a) Let (S, T) and (S', T') be two minimum (s, t)-cuts in G. Prove that $(S \cap S', T \cup T')$ and $(S \cup S', T \cap T')$ are also minimum (s, t)-cuts in G. *[Hint: Let f^* be a maximum (s, t)-flow in G. What have we learned about f^* and the edges crossing any minimum (s, t)-cut?]*

   (b) Describe and analyze an efficient algorithm to determine whether G contains a unique minimum (s, t)-cut. *[Hint: Use the claim from part (a). Modify the process for finding a minimum (s, t)-cut given a maximum (s, t)-flow.]*

2. The new Department of Computing Stuff at UTD has a flexible curriculum with a complex set of graduation requirements. The department offers n different courses, and there are m different requirements. Each requirement specifies a subset of the n courses and the number of courses that must be taken from that subset. The subsets for different requirements may overlap, but each course can be used to satisfy *at most one* requirement.

   For example, suppose there are $n = 5$ courses A, B, C, D, and E and $m = 2$ graduation requirements:

   - You must take at least 2 courses from the subset $\{A, B, C\}$.
   - You must take at least 2 courses from the subset $\{C, D, E\}$.

   Then a student who has only taken courses B, C, and D cannot graduate, but a student who has taken either A, B, C, and D *or* B, C, D, and E can graduate.

   Describe and analyze an algorithm to determine whether a given student can graduate. The input to your algorithm is the list of m requirements (each specifying a subset of the n courses and the number of courses that must be taken from that subset) and the list of courses the student has taken.

3. A boolean formula is in ***disjunctive normal form*** (DNF) if it consists of a ***disjunction*** (OR) of several ***terms***, each of which is the conjunction (AND) of one or more literals. For example, the formula

   $$(\bar{x} \wedge y \wedge \bar{z}) \vee (y \wedge z) \vee (x \wedge \bar{y} \wedge \bar{z})$$

   is in disjunctive normal form. DNF-SAT asks, given a boolean formula in disjunctive normal form, whether that formula is satisfiable.

(a) Describe a polynomial-time algorithm to solve DNF-SAT.

(b) What is the error in the following argument that P = NP?

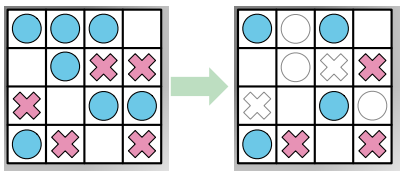
> Suppose we are given a boolean formula in conjunctive normal form with exactly three literals per clause, and we want to know if it is satisfiable. We can use the distributive law to construct an equivalent formula in disjunctive normal form. For example,
>
> $$(x \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z) \Leftrightarrow (x \wedge \bar{y}) \vee (x \wedge z) \vee (y \wedge \bar{x}) \vee (y \wedge z) \vee (\bar{z} \wedge \bar{x}) \vee (\bar{z} \wedge \bar{y})$$
>
> Now we can use the algorithm from part (a) to determine, in polynomial time, whether the resulting DNF formula is satisfiable. We have just solved 3SAT in polynomial time. Since 3SAT is NP-hard, we must conclude that P = NP!

4. Let's practice some NP-hardness proofs! For both parts, you may reduce from any NP-hard problem we discussed in class or that is listed in Erickson's textbook.

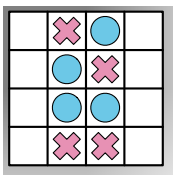
(a) ***Pebbling*** is a solitaire game played on an undirected graph G, where each vertex has zero or more ***pebbles***. A single ***pebbling move*** consists of removing two pebbles from any vertex v and adding one pebble to an arbitrary neighbor of v. (The vertex v must have at least two pebbles before the move.) The PEBBLEDESTRUCTION problem asks, given a graph $G = (V, E)$ and an initial pebble count $p(v)$ for each vertex v, whether there is a sequence of pebbling moves that removes all but one pebble from G.

Prove that PEBBLEDESTRUCTION is NP-hard.

(b) Consider the following solitaire game. The puzzle consists of an $n \times m$ grid of squares where each square may be empty, occupied by a red stone, or occupied by a blue stone. The goal of the puzzle is to remove some of the given stones so that the remaining stones satisfy two conditions: (1) every row contains at least one stone, and (2) no column contains stones of both colors. For some initial configurations of stones, reaching this goal is impossible.



A solvable puzzle and one of its many solutions.



An unsolvable puzzle.

The STONESOLITAIRE problem asks, given an initial configuration of red and blue stones in an $n \times m$ grid, whether the puzzle can be solved. Prove that STONESOLITAIRE is NP-***complete***.