CS/ECE 374 A ✧ Fall 2026

Homework 10

Due Tuesday, November 17, 2026 at 9pm Central Time

---

1. Let $G = (V, E)$. A ***matching*** in G is a subset $M \subseteq E$ of edges such that no two members of M share a vertex. The matching is called ***perfect*** if $2|M| = |V|$; in other words, every vertex is incident to exactly one member of M.

   In the PERFECTMATCHING problem, we are given an undirected graph $G = (V, E)$ and asked if G has a perfect matching.

   (a) Give a short argument that PERFECTMATCHING $\in$ NP.

   (b) Recall that the input to the *satisfiability* problem SAT is a boolean formula in *conjunctive normal form*: a conjunction (AND) of one or more *clauses*, each of which is the disjunction (OR) of one or more *literals*, each of which is either a variable or its negation. Individual clauses do *not* have to have any specific number of literals. The goal of SAT is to determine whether a given CNF formula Φ has a *satisfying assignment*, that is, an assignment of TRUE or FALSE to each variable so that Φ evaluates to TRUE.

   Describe a polynomial time many-one reduction *from* PERFECTMATCHING *to* SAT (yes, you have to describe it yourself; do not merely defer to the Cook-Levin theorem). In other words, describe a polynomial time algorithm that takes an instance $G = (V, E)$ of PERFECTMATCHING and transforms it into an instance Φ of SAT such that G has a perfect matching if and only if Φ is satisfiable. Don't forget to prove your reduction is correct!

   (c) There exists a polynomial time algorithm for PERFECTMATCHING (it's beyond the scope of this course). Why doesn't that algorithm together with the result from part (b) imply P = NP?

2. ***Prove*** that the following problem (which we call STRINGMATCH) is NP-hard. The input is a finite set S of strings, all of the same length n, over the alphabet $\{0, 1, 2\}$. The problem is to determine whether there is a string $w \in \{0, 1\}^n$ such that for every string $s \in S$, the strings s and w have the same symbol in at least one position.

   For example, given the set $S = \{01220, 21110, 21120, 00211, 11101\}$, the correct output is TRUE, because the string $w = 01001$ matches the first three strings of S in the second position, and matches the last two strings of S in the last position. On the other hand, given the set $S = \{2002, 2112, 2012, 2102\}$, the correct output is FALSE.

Solved Problem

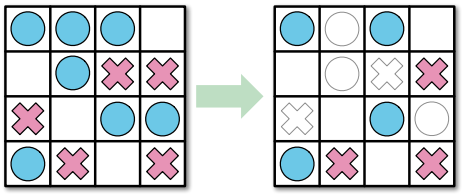
3. ***RedBlue*** is a puzzle that consists of an $n \times m$ grid of squares, where each square may be empty, occupied by a red stone, or occupied by a blue stone. The goal of the puzzle is to remove some of the given stones so that the remaining stones satisfy two conditions:

   (1) Every row contains at least one stone.

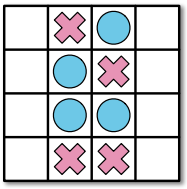
   (2) No column contains stones of both colors.

   For some RedBlue puzzles, reaching this goal is impossible; see the example below.

   Prove that it is NP-hard to determine whether a given RedBlue puzzle has a solution.



A solvable RedBlue puzzle and one of its many solutions.



An unsolvable RedBlue puzzle.

Solution: We show that RedBlue is NP-hard by describing a reduction from 3SAT.

Let Φ be a 3CNF boolean formula with m variables and n clauses. We transform this formula into a RedBlue instance X in polynomial time as follows. The size of the board is $n \times m$. The stones are placed as follows, for all indices i and j:

- If the variable x_j appears in the ith clause of Φ, we place a blue stone at (i, j).
- If the negated variable $\overline{x_j}$ appears in the ith clause of Φ, we place a red stone at (i, j).
- Otherwise, we leave cell (i, j) blank.

To prove that RedBlue is NP-hard, it suffices to prove the following claim:

Φ is satisfiable if and only if RedBlue puzzle X is solvable.

$\Longrightarrow$ First, suppose Φ is satisfiable; consider an arbitrary satisfying assignment. For each index j, remove stones from column j according to the value assigned to x_j:

- If $x_j = \text{TRUE}$, remove all red stones from column j.
- If $x_j = \text{FALSE}$, remove all blue stones from column j.

In other words, remove precisely the stones that correspond to FALSE literals. Because every variable appears in at least one clause, each column now contains stones of only one color (if any). On the other hand, each clause

of Φ must contain at least one TRUE literal, and thus each row still contains at least one stone. We conclude that RedBlue puzzle X is solvable.

$\Longleftarrow$ On the other hand, suppose RedBlue puzzle X is solvable; consider an arbitrary solution. For each index j, assign a value to x_j depending on the colors of stones left in column j:

- If column j contains blue stones, set $x_j = \text{TRUE}$.
- If column j contains red stones, set $x_j = \text{FALSE}$.
- If column j is empty, set x_j arbitrarily.

In other words, assign values to the variables so that the literals corresponding to the remaining stones are all TRUE. Each row still has at least one stone, so each clause of Φ contains at least one TRUE literal, so this assignment makes $\Phi = \text{TRUE}$. We conclude that Φ is satisfiable.

This reduction clearly requires only polynomial time. ■