CS/ECE 374 A ✧ Fall 2023

Homework 10

Due Tuesday, November 14, 2019 at 9pm

---

1. This problem asks you to describe polynomial-time reductions between two closely related problems:

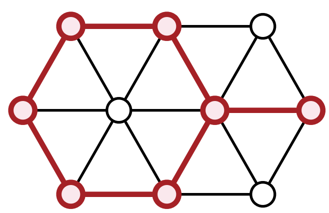
   - SubsetSum: Given a set S of positive integers and a target integer T, is there a subset of S whose sum is T?
   - Partition: Given a set S of positive integers, is there a way to partition S into two subsets S_1 and S_2 that have the same sum?

   (a) Describe a polynomial-time reduction from SubsetSum to Partition.

   (b) Describe a polynomial-time reduction from Partition to SubsetSum.

   Don't forget to to prove that your reductions are correct.

2. A subset S of vertices in an undirected graph G is called *triangle-free* if, for every triple of vertices $u, v, w \in S$, at least one of the three edges uv, uw, vw is *absent* from G. Prove that finding the size of the largest triangle-free subset of vertices in a given undirected graph is NP-hard.



A triangle-free subset of 7 vertices and its induced edges.
This is ***not*** the largest triangle-free subset in this graph.

Solved Problem

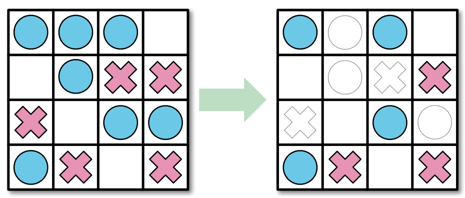
4. ***RedBlue*** is a puzzle that consists of an $n \times m$ grid of squares, where each square may be empty, occupied by a red stone, or occupied by a blue stone. The goal of the puzzle is to remove some of the given stones so that the remaining stones satisfy two conditions:

   (1) Every row contains at least one stone.

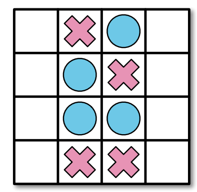
   (2) No column contains stones of both colors.

   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. ■

Standard NP-hardness rubric. 10 points =

+ 1 point for choosing a reasonable NP-hard problem X to reduce from.
  - The Cook-Levin theorem implies that *in principle* one can prove NP-hardness by reduction from *any* NP-complete problem. What we're looking for here is a problem where a simple and direct NP-hardness proof seems likely.
  - You can use any of the NP-hard problems listed on the next page or in the textbook (except the one you are trying to prove NP-hard, of course).
+ 2 points for a *structurally sound* polynomial-time reduction. Specifically, the reduction must:
  - take an *arbitrary* instance of the declared problem X ***and nothing else*** as input,
  - transform that input into a corresponding instance of Y (the problem we're trying to prove NP-hard),
  - transform the output of the magic algorithm for Y into a reasonable output for X, and
  - run in polynomial time.

  (The output transformation is usually trivial.) This is strictly about the structure of the reduction algorithm, not about its correctness. **No credit for the rest of the problem if this is wrong.**
+ 2 points for a *correct* polynomial-time reduction. That is, assuming a black-box algorithm that solves Y in polynomial time, the proposed reduction actually solves problem X in polynomial time.
+ 2 points for the "if" proof of correctness. (Every good instance of X is transformed into a good instance of Y.)
+ 2 points for the "only if" proof of correctness. (Every bad instance of X is transformed into a bad instance of Y.)
+ 1 point for writing "polynomial time"

- An incorrect but structurally sound polynomial-time reduction that still satisfies half of the correctness proof is worth at most 6/10.
- A reduction in the wrong direction is worth at most 1/10.

Some useful NP-hard problems. You are welcome to use any of these in your own NP-hardness proofs, except of course for the specific problem you are trying to prove NP-hard.

CircuitSat: Given a boolean circuit, are there any input values that make the circuit output True?

3Sat: Given a boolean formula in conjunctive normal form, with exactly three distinct literals per clause, does the formula have a satisfying assignment?

MaxIndependentSet: Given an undirected graph G, what is the size of the largest subset of vertices in G that have no edges among them?

MaxClique: Given an undirected graph G, what is the size of the largest complete subgraph of G?

MinVertexCover: Given an undirected graph G, what is the size of the smallest subset of vertices that touch every edge in G?

MinSetCover: Given a collection of subsets $S_1, S_2, \ldots, S_m$ of a set S, what is the size of the smallest subcollection whose union is S?

MinHittingSet: Given a collection of subsets $S_1, S_2, \ldots, S_m$ of a set S, what is the size of the smallest subset of S that intersects every subset S_i?

3Color: Given an undirected graph G, can its vertices be colored with three colors, so that every edge touches vertices with two different colors?

ChromaticNumber: Given an undirected graph G, what is the minimum number of colors required to color its vertices, so that every edge touches vertices with two different colors?

HamiltonianPath: Given graph G (either directed or undirected), is there a path in G that visits every vertex exactly once?

HamiltonianCycle: Given a graph G (either directed or undirected), is there a cycle in G that visits every vertex exactly once?

TravelingSalesman: Given a graph G (either directed or undirected) with weighted edges, what is the minimum total weight of any Hamiltonian path/cycle in G?

LongestPath: Given a graph G (either directed or undirected, possibly with weighted edges), what is the length of the longest simple path in G?

SteinerTree: Given an undirected graph G with some of the vertices marked, what is the minimum number of edges in a subtree of G that contains every marked vertex?

SubsetSum: Given a set X of positive integers and an integer k, does X have a subset whose elements sum to k?

Partition: Given a set X of positive integers, can X be partitioned into two subsets with the same sum?

3Partition: Given a set X of $3n$ positive integers, can X be partitioned into n three-element subsets, all with the same sum?

IntegerLinearProgramming: Given a matrix $A \in \mathbb{Z}^{n \times d}$ and two vectors $b \in \mathbb{Z}^n$ and $c \in Z^d$, compute $\max\{c \cdot x \mid Ax \leq b, x \geq 0, x \in \mathbb{Z}^d\}$.

FeasibleILP: Given a matrix $A \in \mathbb{Z}^{n \times d}$ and a vector $b \in \mathbb{Z}^n$, determine whether the set of feasible integer points $\max\{x \in \mathbb{Z}^d \mid Ax \leq b, x \geq 0\}$ is empty.

Draughts: Given an $n \times n$ international draughts configuration, what is the largest number of pieces that can (and therefore must) be captured in a single move?

SuperMarioBrothers: Given an $n \times n$ Super Mario Brothers level, can Mario reach the castle?