Homework 8: Hardness

Be sure to read the grading guidelines and style guidelines. Especially to see the suggested format for describing algorithms.

We sometimes describe how long are justifications or proofs are. These lengths are intended to help you estimate how much detail we're expecting, you should not take those estimates as hard length-limitations.

Our solutions for any individual problem will fit in approximately one page or less.

You are allowed (and encouraged!!) to collaborate with each other. Brainstorming is much easier to do in a group than alone! But you must follow the collaboration policy (which includes needing to write your submission on your own).

You will submit to Gradescope; we will have a different box for each problem, so please give yourself extra time to submit.

1. Tell the truth [10 points]

Answer this question directly on Gradescope. For each of the following questions answer True or False or Open.

(a) There exists a problem in **P** that is not in **NP**.

(b) There is an algorithm for 3SAT that runs in polynomial time.

(c) If there is an algorithm for 3Color that runs in polynomial time then there is an algorithm for 2Color that runs in polynomial time.

(d) 3SAT polynomial time reduces to 2Color.

(e) If **P** $\neq$ **NP** there does not exist a polynomial-time algorithm for Vertex-Cover.

(f) If $A \leq_P B$, **P** $\neq$ **NP**, and B is **NP**-hard then A is **NP**-hard.

(g) If $A \leq_P B$ and **A** is **NP**-hard then B is **NP**-hard.

2. A game of darts [25 points]

At an amusement park there is a dart game that has overlapping regions, each enclosed by its own colored line marked on the wall. Contestants win at the dart game if they can toss the darts at the wall and land at least one dart inside every marked region. A single dart may land inside multiple regions, but they only get to throw k darts.

We consider an abstract version of the dart game, which is a discrete variant with only n places, numbered 1 through n, where a dart could land, and each of the m regions, $R_1, \ldots, R_m$, is just an arbitrary subset of $\{1, \ldots, n\}$. The *Abstract-Dart-Game* problem asks whether there is a set of at most k spots among $\{1, \ldots, n\}$ so that if darts land in all those spots then every region would contain a dart.

For this problem we will prove that the *Abstract-Dart-Game* problem is NP-complete.

(a) First show that Abstract-Dart-Game is in NP by describing a polynomial-size certificate and a polynomial-time verifier.

(b) Next show that Abstract-Dart-Game is NP-Hard by providing a reduction involving an NP-Hard problem from class (select one of VertexCover, IndpendentSet, Clique, 3Color, or 3SAT for your reduction).

3. Breaking up those intervals [25 points]

This problem is a variant of the Interval Scheduling problem that we saw how to solve with a greedy algorithm. In this version of the problem, there is a single resource available for scheduling, but instead of requiring the resource for the whole time between the start time s_i and finish time f_i, the request has scheduled breaks during which time other jobs may use the resource. That is, though job i has an overall start time of s_i and finish time f_i, it may have a break that starts at some time b_{i1} and ends a e_{i1} and, after returning for a while, might have a second break beginning at some b_{i2} and ending at e_{i2}, etc. In general we would have $s_i < b_{i1} < e_{i1} < b_{i2} < e_{i2} ... < f_i$ and no bound on the number of breaks.

For example, if we had two requests, one beginning on hour 0 and one beginning on hour 1, each of which repeatedly runs for an hour and then takes an hour break, then the two requests would be compatible with each other and could both be scheduled.

Assume that all start, finish, and break times for all requests are integers and the total time range is between 0 and T. The *Interval-Scheduling-with-Breaks* Problem is to determine. given a collection of descriptions of n requests with breaks and an integer k, whether or not it is possible to schedule at least k of the requests on the single resource.

For this problem we will prove that the *Interval-Scheduling-with-Breaks* problem is NP-complete.

(a) First show that *Interval-Scheduling-with-Breaks* is in NP by describing a polynomial-size certificate and a polynomial-time verifier.

(b) Next show that Interval-Scheduling-with-Breaks is NP-Hard by providing a reduction involving an NP-Hard problem from class (select one of VertexCover, IndpendentSet, Clique, 3Color, or 3SAT for your reduction).

4. Test and you can find [25 points]

Recall that for an undirected graph $G = (V, E)$ a 3-coloring is an assignment of labels $R,\ G,\ B$ to the vertices of G such that no two vertices sharing an edge have the same color.

Suppose that you had a black-box algorithm C that on any input G can correctly answer the 3-COLOR decision problem. That is, on input G, for G an undirected graph, it correctly answers YES if G has a 3-coloring and NO if there is no such coloring. Show that on any input G with a polynomial amount of work and a polynomial number of calls to C, you can actually *return* a 3-coloring of G if one exists, and answer FAIL otherwise.

Hint: Consider how adding edges between vertices can let you test whether they can be the same color and can help you narrow down the set of possible 3-colorings.

5. Round numbers [Extra Credit]

In this problem, as input you are given an $m \times n$ array A of real numbers for which you are promised that the sum of each row and the sum of each column is an integer. Your goal is to find a way of rounding each matrix entry up to the closest integer or down to the closest integer while maintaining all the row and column sums. More precisely, give a polynomial-time algorithm that will produce a new $m \times n$ integer array B such that $B[i, j]$ is $\lceil A[i, j] \rceil$ or $\lfloor A[i, j] \rfloor$, and each row sum or column sum in B is equal to the corresponding sum in A.

For example, if the input is the array on the left, your algorithm could output any of the arrays on the right.

0.4	0.1	1.5
0.6	1.9	0.5

$\Rightarrow$

1	0	1
0	2	1

or

0	0	2
1	2	0

or

0	1	1
1	1	1