CPSC 320 2019S2: Assignment 6

Please submit this assignment via GradeScope at `https://gradescope.com`. Be sure to identify everyone in your group if you're making a group submission (which we encourage!).

Submit by the deadline **Sunday August 11 at 11PM**. For credit, your group must make a **single** submission via one group member's account, marking all other group members in that submission **using GradeScope's interface**. Your group's submission must:

- Be on time.
- Consist of a single, clearly legible file uploadable to GradeScope with clearly indicated solutions to the problems. (PDFs produced via LaTeX, Word, Google Docs, or other editing software work well. Scanned documents will likely work well. **High-quality** photographs are OK if we agree they're legible.)
- Include prominent numbering that corresponds to the numbering used in this assignment handout. Put these **in order** starting each problem on a new page, ideally. If not, **very clearly** and prominently indicate which problem is answered where!
- Include at the start of the document the **ugrad.cs.ubc.ca e-mail addresses** of each member of your team. Please **do not** include names on the assignment. If you don't mind private information being stored outside of Canada and want an extra double-check on your identity, include your student number rather than your name.
- Include at the start of the document the statement: "All group members have read and followed the guidelines for academic conduct in CPSC 320. As part of those rules, when collaborating with anyone outside my group, (1) I and my collaborators took no record but names (and GradeScope information) away, and (2) after a suitable break, my group created the assignment I am submitting without help from anyone other than the course staff." (Go read those guidelines!)
- Include at the start of the document your outside-group collaborators' ugrad.cs.ubc.ca IDs, but **not** their names. (Be sure to get those IDs when you collaborate!)

1 The $3n$ Musketeers

To maximize their success on the semester's final assignment, the CPSC 320 class has decided to work in teams! They want to work on their assignment in groups of two. Additionally, they want to make sure that each pair has all the skills necessary to succeed on the assignment.

The class has $2n$ students, of whom n are good at understanding large blocks of text (in the form of problem statements and proposed algorithms), and the other n are good at writing pseudocode. They want each of the n teams to have one Text Decipherer (a student who is good at understanding text) and one Pseudocode Writer (a student who is good at writing pseudocode).

The problem is that not everyone in the class likes each other, and some of them don't want to be partners. Each Decipherer is willing to work with some (non-empty) subset of the Writers, and each Writer is willing to work with some (non-empty) subset of the Decipherers. Your goal is to form a set of n teams so that everybody is paired with someone they're willing to work with.

We refer to this problem as the **TEAM2 Problem**. We frame this as a decision problem: we want to return YES if it's possible to generate such a matching (i.e., n distinct teams where everyone is willing to work with their partner), and NO otherwise.

1. Give and briefly explain a small example of the TEAM2 problem in which it is not possible to generate a matching in which everyone is paired with someone they're willing to work with (i.e., provide a NO instance).

2. Give a reduction from TEAM2 to Maximum Bipartite Matching, and briefly justify the correctness of your reduction. Note: in Assignment 1 and the midterm exam, you gave reductions to maximum bipartite matching on a **weighted** graph. For this problem, it will be better to reduce to the **unweighted** version. (If you're unsure about how this would work on an unweighted graph: look it up!)

Oh, no! Your CPSC 320 instructor, the fiendish and diabolical Milady Susanne,[1] has demanded you do *formal proofs* as part of your assignment![2] Fortunately, the brave and valiant 320 students are fighting the dark forces of professorial tyranny by enlisting n Theorem Provers, who are (as their name suggests) good at writing proofs.

Unsurprisingly, each Prover is only willing to work with some of the Decipherers and some of the Writers, and vice versa. We want to divide the class into n disjoint teams of three – each consisting of a Decipherer, Writer, and Prover – such that everyone is willing to work with both other people on their team. Specifically, we want the decision variant of this problem: so we return YES if such a matching is possible, and NO otherwise. We call this the **TEAM3 Problem**.

3. Here is a proposed reduction from TEAM3 to TEAM2 (where D, W, and P denote Decipherers, Writers, and Provers, respectively):

   ```
   return YES to TEAM3(D, W, P) iff:
   TEAM2(D, W) returns YES and
   TEAM2(W, P) returns YES and
   TEAM2(D, P) returns YES.
   ```

   Give and briefly explain a small counterexample (with n no greater than 2) to prove that this reduction is not correct.

Suppose now that, instead of having lists of people each person is willing to work with, you have a 3-dimensional array `happy`, which has value TRUE at `happy[i,j,k]` if and only if the Decipherer d_i, Writer w_j, and Prover p_k are all willing to be on a team together. There are no logical constraints on the values in the `happy` array, other than that each value is TRUE or FALSE. For example: if `happy[1,1,k]`, `happy[1,j,1]`, and `happy[i,1,1]` are all TRUE for some values of i, j, and k, it could still be the case that `happy[1,1,1]` is FALSE. Now our goal is to determine whether there's a set of n disjoint teams (d_i, w_j, p_k) such that `happy[i,j,k]` is TRUE for every team.

4. The **3D Matching Problem** is defined as follows: you're given three disjoint sets X, Y, and Z, and $|X| = |Y| = |Z|$. Also given is a list T consisting of triples (x, y, z) for $x \in X, y \in Y$, and $z \in Z$. The 3D Matching Problem asks: given sets X, Y, Z and list of triples T, can we find a subset of T such that every element in X, Y and Z appears **exactly once** in one of the triples? The 3D Matching Problem is known to be NP-complete.

[1] "Milady" (name derived from "my lady") is one of the central antagonists in Alexandre Dumas's *The Three Musketeers*, the novel from which this question draws its title.

[2] Mwahaha!

Prove[3] that TEAM3 is NP-hard using a reduction between TEAM3 and 3D Matching. You do not need to prove the correctness of your reduction, but you should clearly explain the key elements of your reduction and why they are there.

5. Assume that P $\neq$ NP. Then is TEAM2 also NP-hard? Justify your answer. (Note: answering this question may also require you to look things up!)

6. Suppose I claim to have generated a polynomial-time reduction from TEAM3 to TEAM2, using a polynomial number of calls to TEAM2. Again assuming that P $\neq$ NP, explain why the new reduction cannot possibly be correct.

7. **[FOR PRACTICE ONLY, BUT HIGHLY RECOMMENDED AS EXAM PREPARATION]** Prove the correctness of your reduction from question 4. You will want to define the certificates for both the TEAM3 and 3D Matching problems first!

2 War and P

RECALL the **TUG-O-WAR problem**: given n people with weights $W = [w_1, \ldots, w_n]$ (which we assume to be integers), we want to generate two teams that are well-balanced.

Specifically, we want to partition all n people into two teams T_1 and T_2 such that the absolute difference in total weight between team 1 and team 2 is minimized. Note that we do not require the same number of people on both teams. TUG-O-WAR is an optimization problem, but we can give a decision version by asking: given n people and their weights, *and a number* k, can we generate two teams whose weight difference is no more than k?

1. Give a good certificate for this problem and prove that TUG-O-WAR is in NP.

2. SUBSET-SUM takes a list of numbers A and a value w, and returns YES if and only if there exists a subset of A with sum w. SUBSET-SUM is NP-complete. Your friend gives the following reduction to show that TUG-O-WAR is NP-hard:

```
Given a TUG-O-WAR instance with list of weights W and value k,
we can reduce TUG-O-WAR to SUBSET-SUM as follows:

   Let m be the sum of all elements in W.

   // Run SUBSET-SUM multiple times in a loop. We define the loop
   // differently, depending on whether m is even or odd.
   If m is even:
      For i=0 to k:
         if SUBSET-SUM(W, m/2 + i) returns YES:
            // Found two teams with weight difference = i
            return YES to TUG-O-WAR
   Else: // m is odd
      For i=0 to k-1:
         if SUBSET-SUM(W, floor(m/2) + i) returns YES:
            // Found two teams with weight difference = i+1
            return YES to TUG-O-WAR
return NO to TUG-O-WAR

SUBSET-SUM is NP-complete. Therefore, TUG-O-WAR is NP-hard.
```

[3] MwahahaHAHAHAHAHA!

Explain why this does **not** prove that TUG-O-WAR is NP-hard.

3. Duly chastened, your friend tries again with the following proof. Note that the 320 instructor has told your friend he may assume that all elements in the SUBSET-SUM input list A are positive, and that the target value w is such that $\frac{m}{2} \leq w \leq m$, where m is the sum of all elements in A.

   ```
   Given a SUBSET-SUM instance with list A and target w, we can reduce to TUG-O-WAR as
   follows:

      Let m be the sum of all elements in A.
      If a subset of A sums to w, the remaining elements in A sum to m-w; i.e.,
      they form two ‘‘teams’’ with absolute weight difference w-(m-w)=2w-m.
      Therefore, we call TUG-O-WAR(A, 2w-m) and return YES to SUBSET-SUM iff YES to
      TUG-O-WAR.

   SUBSET-SUM is NP-complete. Therefore, TUG-O-WAR is NP-hard.
   ```

   Explain to your friend (by giving and explaining a counterexample) why this reduction is incorrect.

4. Prove that TUG-O-WAR is NP-hard with a reduction from SUBSET-SUM to TUG-O-WAR. This, together with your earlier proof that TUG-O-WAR is in NP, proves that TUG-O-WAR is NP-complete. As before, assume that all elements in the SUBSET-SUM input list A are positive and that $\frac{m}{2} \leq w \leq m$. You do not need to prove the correctness of your reduction, but you should clearly explain the key elements of your reduction and why they are there.

5. **[FOR PRACTICE ONLY, BUT HIGHLY RECOMMENDED AS EXAM PREPARATION]** Because you've twice pointed out the flaws in your friend's proposed reduction, he's probably going to try do the same to you. Make sure he can't succeed by proving the correctness of your reduction.