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. **If you submit an answer substantially longer than 2 pages, the TAs are allowed to stop reading at the end of page 2.**

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 have a different box for each problem, please give yourself time to submit.

NP-Complete Problems

For this assignment, you may use that the following problems are NP-complete (do not use any other problems, unless you also prove them NP-complete in your proof, even if we covered them in class).

- k-COLOR: Given a graph $G = (V, E)$ and an integer k (where $k \geq 3$), return `true` if there is a function $f : V \to \{1, .., k\}$ such that if $(u, v) \in E$ then $f(u) \neq f(v)$.
- VERTEX-COVER: Given a graph $G = (V, E)$ and an integer k, return `true` if there is a set of vertices S, such that $|S| \leq k$ and $\forall (u, v) \in E : u \in S \vee v \in S$.
- CLIQUE:Given a graph $G = (V, E)$ and an integer k, return `true` if there is a set of vertices S, such that $|S| \geq k$ and $\forall u, v : [u \neq v \wedge u, v \in S] \to (u, v) \in E$.
- 3-SAT: Given an expression in CNF form, where each clause contains exactly three literals, return `true` if there is a setting of the variables that causes the expression to evaluate to true.

1. $\wedge$, $\vee$, $\neg$[10 points]

Given two decision problems A and B, let $A \wedge B$ denote the decision problem where on input x the answer is "yes" if and only if the answer for x on both A and B is yes. Similarly define $A \vee B$ as the decision problem for which on input x the correct answer is yes if and only if A or B (or both) answer yes on x.

If A, B are both in NP, can we say that the following problems are also in NP? Provide a few sentences of justification for your answer.

(a) $A \wedge B$

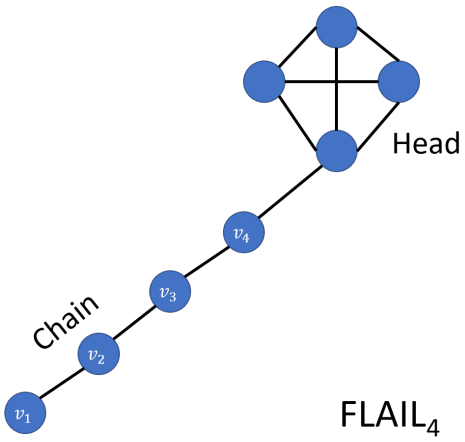
(b) $A \vee B$

2. The chain isn't the hard part [25 points]

A FLAIL$_n$ is a graph that looks kinda like a flail. Formally, it is a graph on $2n$ vertices with the following pieces:

- The chain: n vertices connected in a path (i.e., $v_1, v_2, ...v_n$, such that $(v_i, v_{i+1}) \in E$ for $1 \leq i < n$).
- The head: the other n vertices, forming a clique. (A "*clique*" also called a "complete subgraph" is a set of vertices with every possible edge between them, i.e. $S \subseteq V$: $\forall$u$\neq$v $\in$S, (u,v)$\in$E).
- An edge between one end of the chain (v_1 or v_n) and a vertex in the head.

For example, here is a FLAIL$_4$



Consider the problem MAX-FLAIL

MAX-FLAIL
Input: A graph G
Output: the largest integer k such that there is a FLAIL$_k$ subgraph of G

Recall that $G_1 = (V_1, E_1)$ is a subgraph of $G_2 = (V_2, E_2)$ if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.

Prove that MAX-FLAIL is NP-hard.
Note that you're not proving it's NP-complete. It isn't! It's not a decision problem.

3. LPs are making a comeback [25 points]

When we talked about linear programs, we studied **optimization** LPs. Another common type of LP is a **feasibility** LP. A feasibility LP is an LP without an objective function. For a feasibility LP, we aren't interested in the maximum (or minimum) value the objective function can take on, we just want to know is there a feasible point at all (i.e., is there a point which simultaneously satisfies all the constraints)?

More formally, a feasibility LP has:

- A set of (real-valued) variables $x_1, ..., x_n$
- A set of linear inequality constraints (i.e., each is of the form $\sum a_i x_i \leq b$ for a_i, b constants)

Recall that a common problem when using LPs is that we might want our variables to be integers, but a standard LP could produce non-integer real numbers. Define INT-PROG as the following problem:

INT-PROG
Input: A feasibility linear program
Output: `true` if there is a feasible point **where every variable is set to an integer**, `false` otherwise.

Note that a linear program could be feasible, but INT-PROG might still return `false` (if all feasible points contain at least one non-integer).

Show that INT-PROG is NP-complete.
Hint: Don't worry too much about which NP-complete problem you pick, it's hard to go wrong on this one. You can make all of the ones on the list work. Try to break down your problem into decision variables (booleans) and then find a set of constraints that will force the variables to represent what you are looking for.

4. A Taste of Complexity Theory [25 points]

We defined **polynomial time** reductions in class. We said $A \leq_P B$ if there is an algorithm to solve problem A that

- Makes a polynomial number of calls to a library for B.
- Takes a polynomial amount of time **ignoring** the library calls.

A key benefit of this definition is that it is transitive: If $A \leq_P B$ and $B \leq_P C$ then $A \leq_P C$. The transitivity fact is key in using reductions to show new problems are NP-hard.

Define a **cubic time** reduction, which we'll denote as $A \leq_{n^3} B$, as:
an algorithm to solve problem A using a library function for problem B that

- Calls the library at most $\mathcal{O}\left(n^3\right)$ times.
- Takes at most $\mathcal{O}\left(n^3\right)$ time other than the library calls.

And define a **linear time** reduction, which we'll denote as $A \leq_n B$, as:
an algorithm to solve problem A using a library function for problem B that

- Calls the library at most $\mathcal{O}\left(n\right)$ times.
- Takes at most $\mathcal{O}\left(n\right)$ time other than the library calls.

(a) Suppose $A \leq_P B$ and $B \leq_P C$. Describe in words the reduction that shows $A \leq_P C$, and explain why it's still a polynomial reduction. We did this reduction in class in visual form, you'll be doing the same proof! We just want you to practice here. [5 points]

(b) Now suppose $A \leq_{n^3} B$ and $B \leq_{n^3} C$ is it true that, $A \leq_{n^3} C$? State true or false. Then: if it is true, give the reduction and explain why the reduction meets the condition (pay particular attention to the $\mathcal{O}\left(n^3\right)$ requirements). If it is false, briefly explain why you can't just combine the reductions like you did in (a). [10 points]

(c) Now suppose $A \leq_n B$ and $B \leq_n C$ is it true that, $A \leq_n C$? State true or false. Then: if it is true, give the reduction and explain why the reduction meets the condition (pay particular attention to the $\mathcal{O}\left(n\right)$ requirements). If it is false, briefly explain why you can't just combine the reductions like you did in (a). [10 points]

(d) Reflect on the prior parts, and think about why we use polynomial-time as our definition of efficiency. You don't have to write anything for this part. [0 points]