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 $\forall u, v : [u \neq v \wedge u, v \in S] \to (u, v) \in E$.
- IND-SET: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 $\forall u, v : [u \neq v \wedge u, v \in S] \to (u, v) \notin E$.
- 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. Multiple Choice [10 points]

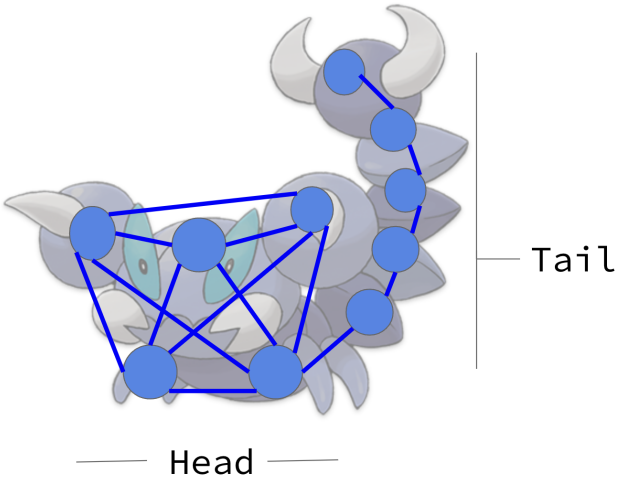
We have multiple choice questions to answer on gradescope. Unlike previous autograded submissions, we will **not** release scores on this problem until after the late deadline.

2. Skorupi, I choose you [25 points]

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

- The tail: 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. $S \subseteq V$: ∀u≠v ∈S, (u,v)∈E).
- An edge between one end of the tail (v_1 or v_n) and a vertex in the head.

For example, here is a SKORUPI$_5$



Consider the problem MAX-SKORUPI

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

Recall that a graph $G_1 = (V_1, E_1)$ 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-SKORUPI is NP-hard.
Note that you're not proving it's NP-complete. It isn't! It's not a decision problem.

3. Birds of a feather flock together [25 points]

Given a graph G such that the edge represents some positive relationship between the entities, we might want to find a way to group the entities such that everyone in the same group has an edge going between each other.

We therefore define a problem called k-GROUPING, which aims to assign each vertex in G one of the k labels, such that if two vertices share the same label, then there must be an edge going between them.

Formally:

k-GROUPING
Input: A graph $G = (V, E)$, an integer k
Output: `true` if there is a function $f : V \to \{1, ..., k\}$ such that if $f(u) = f(v)$ then $(u, v) \in E$, `false` otherwise.

Prove that k-GROUPING is NP-complete.

4. A Familiar Interval Scheduling [25 points]

In lecture, we learned how to solve the interval scheduling problem. This problem is a modification of that.

As before, you have a single processor, and a set of n jobs with fixed start and end times. The goal is to determine if it's possible to complete at least k jobs.

Unlike before, each job does not run continuously between the start and end time. Instead, each job is scheduled to sleep at some fixed intervals in the middle. While a job is sleeping, another job is allowed to run. Specifically, job j is described by start and end times a_j and b_j, as well as m_j sleeping times, described with x_{ji}, y_{ji}, indicating that the job will sleep during the interval (x_{ji}, y_{ji}). It's guaranteed that the sleeping times do not overlap, are contained in the start and end times, and are in sorted order (i.e. $a_j < x_{j1} < y_{j1} < x_{j2} < y_{j2} < ... < x_{jm_j} < y_{jm_j} < b_j$).

All times in this problem are integers.

Note that jobs may sleep for a different number of times, the number of times a job might sleep is unbounded (but finite), and a job might not sleep at all.

Define SLEEPY-JOBS as the following problem:

SLEEPY-JOBS
Input: An integer k and a list of n jobs, as described above
Output: `true` if at least k jobs can be run on the processor without overlapping non-sleeping times, `false` otherwise.

Show that SLEEPY-JOBS is NP-complete.

Hint: You're allowed to use any problem from the list, but a graph problem will be the easiest.