Due date: **Monday, June 5**, 11:59 PM. You will need to submit this via GradeScope.

While I reserve the right to extend this due date at my sole discretion. I will not move it backwards (i.e., I will not make it due before Monday June 5 at 11:59 PM) for any reason.

Formatting: please start each *numbered* question on a separate piece of paper, and limit your response to one page. If you wish to do some subsets of question 2, on separate pages, you may do so.

Upload a **PDF** consisting of these 2-4 pages to GradeScope and tag each page as appropriate. Write in dark ink on light paper, whether you are producing your work digitally or on physical paper. I realize this seems strict, but it really does help with grading, especially with a large class.

Please remember you have a professor and four TAs who want to help you. If you're lost on this, let us know!

1. In lecture, we saw that Independent Set is in $\mathcal{NP}$. Suppose you had a deterministic polynomial-time algorithm (or a Turing Machine, however you prefer to think about this) for the *decision* version of the problem. Note that this means the algorithm reads $\langle G, k \rangle$ and only outputs "yes" or "no" (or accept/reject, or true/false) – and you can't inspect the source code or schematic to see how it works! Just the same, show how you can use this to create a deterministic, polynomial-time algorithm that determines which subset, if any, of the vertices of an input graph G constitute an independent set of size k. If multiple subsets do, you need only provide one such subset.

   How many calls to the decider does your function make?

   Note: solutions that do not follow directions will get zero credit. Do not try to create your own deterministic, polynomial time algorithm for Independent Set here. Although, ngl, I will be very impressed if you are successful in doing so.

2. In the Clustering problem, we are given a weighted graph $G = (V, E)$, an integer k, and a target T. We want to divide the vertices into k sets such that any pair of nodes in the same set have a shortest path of length $\leq T$ to every other vertex in that set.

   (a) Express this as a language recognition problem.

   (b) Prove that your language from the previous part is in $\mathcal{NP}$.

   (c) Prove that your language is $\mathcal{NP}$-complete.