Write your solution using \LaTeX. Submit this homework using handin41. You must work with exactly one partner. It is still OK to discuss approaches with others at a high level, but most of your discussions (and all of your detailed discussions!) should be just with your lab partner.

Note: Make sure your homework includes the names of both lab partners. Only one partner should submit. Make sure the files you submit are in the cs41/hw/xx directory before calling handin41.

Note: Make sure your homework gives credit when you and your partner did not completely solve the problem on your own. If you worked with other(s) during lab, say so. If you discussed problems at a high level with others, say so. If you accidentally stumbled on the solution or significant parts of a solution while surfing the web, say so. Not including attributions is a violation of the academic integrity policy.

1. (Kleinberg and Tardos, 8.14) Throughout the semester, we’ve seen several kinds of \textit{interval scheduling} problems (often dressed up as Harry/Hoagie problems). In this problem, we’ll see a much harder version called \textsc{Multiple-Interval-Scheduling}. As before, there is a machine that is available to run jobs over some period of time, say 9AM to 5PM.

People submit jobs to run on the processor; the processor can only work on one job at any single point in time. In this problem however, each job requires a \textbf{set of intervals} of time during which it needs to use the machine. Thus, for example, one job could require the processor from 10AM to 11AM and again from 2PM to 3PM. If you accept this job, it ties up your machine during those two hours, but you could still accept jobs that need any other time periods (including the hours from 11AM to 2PM).

Now, you’re given an integer \( k \) and a set of \( n \) jobs, each specified by a set of time intervals, and you want to answer the following question: is it possible to accept at least \( k \) of the jobs so that no two of the accepted jobs have any overlap in time?

In this problem, you are to show that \textsc{Multiple-Interval-Scheduling} \( \in \text{NP-Complete} \). To assist you, I’ve broken down this problem into smaller parts:

(a) First, show that \textsc{Multiple-Interval-Scheduling} \( \in \text{NP} \).

(b) In the remaining two parts, you will reduce \textsc{Independent-Set} \( \leq_p \textsc{Multiple-Interval-Scheduling} \).

Given input \((G = (V,E), k)\) for \textsc{Independent-Set}, create a valid input for \textsc{Multiple-Interval-Scheduling}. First, divide the processor time window into \( m \) distinct and disjoint intervals \( i_1, \ldots, i_m \). Associate each interval \( i_j \) with an edge \( e_j \). Next, create a different job \( J_v \) for each vertex \( v \in V \). What set of time intervals should you pick for job \( J_v \)?

(c) Finally, run the \textsc{Multiple-Interval-Scheduling} algorithm on the input you create, and output \textsc{YES} iff the \textsc{Multiple-Interval-Scheduling} algorithm outputs \textsc{YES}. Argue that the answer to \textsc{Multiple-Interval-Scheduling} gives you a correct answer to \textsc{Independent-Set}.
2. (Kleinberg and Tardos, 8.16) Consider the problem of reasoning about the identity of a set from the size of its intersections with other sets. You are given a finite set $U$ of size $n$, and a collection $A_1, \ldots, A_m \subseteq U$ of subsets of $U$. You are also given integers $c_1, \ldots, c_m$. The question is: does there exist $X \subseteq U$ such that for each $i = 1, 2, \ldots, m$, the cardinality of $X \cap A_i$ equals $c_i$. We will call this an instance of the INTERSECTION-INFERENCE problem, with input $U, \{A_i\}, \{c_i\}$.

Prove that INTERSECTION-INFERENCE is NP-COMPLETE. **Hint:** reduce from the following problem, which you may assume is NP-COMPLETE:

**Problem One-In-Three-Sat:**

**Inputs:** $n$ variables $x_1, \ldots, x_n$ and $m$ clauses $c_1, \ldots, c_m$ where each clause is the OR of three literals e.g. $c_i = (x_1 \lor \bar{x}_2 \lor x_3)$.

**Output:** YES iff there is a truth assignment to the variables such that for each clause there is exactly one satisfied variable.

**Hint:** Let $U$ be the set of literals. You’ll have to work to ensure that a variable and it’s negation cannot both end up in $X$.

3. (Kleinberg and Tardos, 8.9) Consider the following problem. You are managing a communication network, modeled by a directed graph $G = (V, E)$. There are $c$ users who are interested in making use of this network. User $i$ (for each $i = 1, \ldots, c$) issues a request to reserve a specific path $P_i$ in $G$ on which to transmit data.

You are interested in accepting as many of these path requests as possible, subject to the following restriction: if you accept both $P_i$ and $P_j$, then $P_i$ and $P_j$ cannot share any nodes.

Thus, the Path-Selection problem asks: Given a directed graph $G = (V, E)$, set of requests $P_1, \ldots, P_c$—each of which is a path in $G$, and a number $k$, output YES iff it is possible to select at least $k$ paths so that no two of the selected paths share any nodes.

Prove that Path-Selection is NP-COMPLETE.

4. **(Extra Credit).** Show that One-In-Three-Sat is NP-COMPLETE.