CS 332: *Theory of Computation* — Prof. Mark Bun
Boston University — May 1, 2024

Homework 11 – Due Wednesday, May 1 at 11:59 PM

Reminder Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to the course staff if asked. You must also identify your collaborators and write "Collaborators: none" if you worked by yourself. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden. Collaboration is not allowed on problems marked "INDIVIDUAL."

Note You may use various generalizations of the Turing machine model we have seen in class, such as TMs with two-way infinite tapes, stay-put, or multiple tapes. If you choose to use such a generalization, state clearly and precisely what model you are using. **You may describe Turing machines at a high-level on this assignment.**

Problems There are 4 required problems.

1. (**Poly-time Reductions**) Assume $\mathsf{P} \neq \mathsf{NP}$. For each of the following, give a language (if it exists) with the stated property. Explain why your language satisfies the given property, or explain why no such language can exist.
   (a) $A \leq_{\mathrm{p}} SAT$ and A is NP-complete.
   (b) $SAT \leq_{\mathrm{p}} B$ and B is not NP-complete.
   (c) $SAT \leq_{\mathrm{p}} C$ and C is not NP-hard.
   (d) D is both regular and NP-complete.

2. (**NP-Completeness Mad-Libs**) Given m BU Hub areas and a course catalog consisting of k classes fulfilling those areas, you wish to determine whether there is a small set of courses that will supply you with all of your Hub requirements. Specifically, each course $i = 1, \ldots, k$ supplies you with a set $S_i \subseteq [m]$ of Hub requirements. A *valid course plan* is a collection T of courses that, taken together, supply you with all m Hub requirements: $\cup_{i \in T} S_i = [m]$. Define the language $HUB = \{\langle S_1, \ldots, S_k, r\rangle \mid \text{ there exists a valid course plan } T \subseteq [k] \text{ of size } |T| \leq r\}$.

   This problem will walk you through a proof that HUB is NP-complete.

   (a) We'll first argue that $HUB \in \mathsf{NP}$ by describing a poly-time verifier. A certificate is $\underline{\quad\text{(i)}\quad}$. On input $\langle S_1, \ldots, S_k, r\rangle$, the verifier checks that $|T| \leq r$ and that $\cup_{i \in T} S_i = [m]$ and accepts if and only if this is the case. (For brevity, we're omitting the proof of correctness and runtime analysis that should go here.)

   Fill in the blank labeled (i) with a description of what a certificate for this problem should look like.

   (b) Now we will argue that HUB is NP-hard by giving a reduction showing $VERTEX{-}COVER \leq_{\mathrm{p}} HUB$. (See page 312 of Sipser for discussion of this problem.) A vertex cover of a graph G is a set of vertices T such that every edge in the graph is incident to at least one vertex in T. The language $VERTEX - COVER = \{\langle G, r\rangle \mid G \text{ has a vertex cover of size at most } r\}$.

   In the reduction described below, fill in the blank labeled (ii) with a description of what the algorithm computing the reduction should output.

Algorithm: VERTEX-COVER to HUB Reduction
Input : $\langle G, r\rangle$ where $G = (V, E)$ is a graph and $r \in \mathbb{N}$ 1. Relabel the vertices and edges of the graph so that $V = [k]$ and $E = [m]$.   2. For each $i = 1, \ldots, k$:   Let $S_i = \{j \in [m] \mid$ edge j is incident to vertex $i\}$ 3. Output $\underline{\quad\text{(ii)}\quad}$.

Your job is now done, but here are explanations of correctness and runtime for this reduction.

Correctness: If $\langle G, r\rangle \in VERTEX-COVER$, then there exists a set T of at most r vertices such that every edge in the graph is incident to a member of T. After relabeling, that means T is a set of courses such that every requirement in $[m]$ appears in at least one of the sets S_i, so T is a valid course plan of size at most r. Conversely, if there is a valid course plan T of size at most r in the instance of HUB produced, then T corresponds to a set of vertices such that every edge in G is incident to a member of T, and hence $\langle G, r\rangle \in VERTEX - COVER$.

Runtime: Suppose for concreteness that we are working with the adjacency list representation of G on a multi-tape. Inside the main loop of step 2, constructing each set S_i takes time linear in m, the number of edges of the graph. So overall, the algorithm runs in time $O(km + \log r)$ which is polynomial in the description length of the input.

3. (**Popular Cliques**) A *popular clique* in an undirected graph $G = (V, E)$ is a set of vertices S such that a) S is a clique, i.e., all vertices in S are adjacent to each other and b) for every vertex $v \in V$, there exists a vertex $w \in S$ such that v is adjacent to w. In the parlance of high school social dynamics, a popular clique is a group of students who are all friends with each other, and for which every student in the school is friends with at least one member of the clique.

   Define the language $PC = \{\langle G, k\rangle \mid$ undirected graph $G = (V, E)$ contains a popular clique with at least k vertices$\}$.

   (a) Show that $PC \in \mathsf{NP}$. For brevity, you can omit the proof of correctness and runtime of your NTM or verifier, as long as those are reasonably clear.

   (b) Show that $CLIQUE \leq_{\mathrm{p}} PC$ and use this to conclude that PC is NP-complete. Describe your reduction, explain why it is correct, and analyze its runtime.

4. (**Systems of linear inequalities**) A *linear inequality* I over variables $x_1, \ldots, x_k$ is an inequality of the form $c_1x_1 + \ldots c_kx_k \leq b$, where $c_1, \ldots, c_k$ and b are integers. For example, $5x_1 - 3x_2 + x_3 \leq -1$ is a linear inequality. A *system of linear inequalities* is a set $\{I_1, \ldots, I_m\}$ of inequalities over the same variables. Such a system *has an boolean solution* if one can assign boolean values (either 0 or 1) to all variables in such a way that all inequalities are satisfied.

   Define the language $BI = \{\langle I_1, \ldots, I_m\rangle \mid$ the system $\{I_1, \ldots, I_m\}$ has a boolean solution$\}$.

   (a) Show that BI is in NP. For brevity, you can omit the proof of correctness and runtime of your NTM or verifier, as long as those are reasonably clear.

   (b) Show that $3SAT \leq_{\mathrm{p}} BI$ and use this to conclude that BI is NP-complete. Describe your reduction, explain why it is correct, and analyze its runtime.

5. (**Bonus**) In a directed graph, the ***indegree*** of a node is the number of incoming edges and the ***outdegree*** of a node is the number of outgoing edges. Show that the following problem is NP-complete. Given an undirected graph G and a subset S of the nodes in G, determine whether it

possible to convert G to a directed graph by assigning directions to each of its edges so that every node in S has indegree 0 or outdegree 0, and all remaining nodes in G have indegree at least 1.