CS 332: *Theory of Computation* — Prof. Mark Bun
Boston University — April 22, 2021

Homework 9 – Due Friday, April 30, 2021 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.

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.

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) satisfying the stated property. Explain why your language satisfies the given property, or explain why no such language can exist. (32 points)
   (a) $A \leq_{\mathrm{p}} SAT$ and A is not in NP.
   (b) $SAT \leq_{\mathrm{p}} B$ and B is not NP-hard.
   (c) $SAT \leq_{\mathrm{p}} C$ and C is not NP-complete. Hint: What if C is undecidable?
   (d) D is both regular and NP-complete.

2. (**NP-Completeness Mad-Libs**) Given m nutrients and a menu of n food items supplying those nutrients, you wish to determine whether there is a small set of food items that will supply you with all of the nutrition you need. Specifically, each food item $i = 1, \dots, n$ supplies you with a set $S_i \subseteq [m]$ of nutrients. A *valid diet* is a collection T of foods that, taken together, supply you with all m nutrients: $\cup_{i \in T} S_i = [m]$. Define the language $DIET = \{\langle S_1, \dots, S_n, k\rangle \mid \text{there exists a valid diet } T \subseteq [n] \text{ of size } |T| \leq k\}$.

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

   (a) We'll first argue that $DIET \in \mathsf{NP}$ by describing a poly-time verifier. A certificate is $\underline{\quad\text{(i)}\quad}$. On input $\langle S_1, \dots, S_n, k\rangle$, the verifier checks that $|T| \leq k$ 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. (6 points)

   (b) Now we will argue that $DIET$ is NP-hard by giving a reduction showing $VERTEX-COVER \leq_{\mathrm{p}} DIET$. Recall that 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 $VERTEX-COVER = \{\langle G, k\rangle \mid G \text{ has a vertex cover of size at most } k\}$.

   In the reduction described below, fill in the blank labeled (ii) with a description of what the algorithm computing the reduction should output. (7 points)

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

```
Algorithm: Vertex Cover to Diet Reduction
Input  : $\langle G, k\rangle$ where $G = (V, E)$ is a graph and $k \in \mathbb{N}$
  1. Relabel the vertices and edges of the graph so that $V = [n]$ and $E = [m]$.
  2. For each $i = 1, \ldots, n$:
         Let $S_i = \{j \in [m] \mid$ edge $j$ is incident to vertex $i\}$
  3. Output ___(ii)___.
```

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

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

3. (**Satisfiability**) Let $XSAT = \{\langle \varphi_1, \varphi_2\rangle \mid$ there exists an assignment x satisfying exactly one of $\varphi_1, \varphi_2\}$. On Homework 8, Problem 5 you showed that $XSAT \in \mathsf{NP}$. Show that $SAT \leq_{\mathrm{p}} XSAT$. Your reduction should include an explanation of correctness and an explanation of why it runs in **deterministc** polynomial time. Finally, explain how you can conclude from this that $XSAT$ is NP-complete. (20 points)

4. (**Boolean Roots**) Let $p(x_1, \ldots, x_n)$ be an n-variate polynomial with integer coefficients. A *boolean root* of p is point $(b_1, \ldots, b_n) \in \{0,1\}^n$ such that $p(b_1, \ldots, b_n) = 0$. For example, $(0, 1, 1)$ is a boolean root of the polynomial $7x_1^2 + 2x_2x_3 - 2x_3^8$. Define the language $BROOT = \{\langle p\rangle \mid p$ is an integer polynomial that has at least one boolean root$\}$.

   (a) Explain why $BROOT \in \mathsf{NP}$ by either describing a poly-time NTM or a deterministic verifier. In an effort to keep this assignment short, you do not need to analyze correctness or runtime of your algorithm as long as those are reasonably clear. (10 points)

   (b) Show that $3SAT \leq_{\mathrm{p}} BROOT$ and conclude that $BROOT$ is NP-complete. (25 points)
   Hint: First determine how to transform a single clause $u \vee v \vee w$ into a polynomial $p(u, v, w)$ such that $p(u, v, w) = 0$ iff at least one of u, v, w is set to 1. Then create a polynomial q that evaluates to 0 if and only if all of its inputs are 0. Finally, use q to combine the individual polynomials that correspond to the clauses of your $3SAT$ instance.

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.