Homework 10 – Due Tuesday, April 22, 2025 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. (**Protein Folding**) In the translation stage of protein biosynthesis, a ribosome decodes mRNA to produce a chain of amino acids (a polypeptide). This polypeptide then folds into an active protein in order to perform a biological function in the cell.

   Here we describe a massive simplification of the problem of determining whether a polypeptide can fold into a stable protein. For us, a polypeptide is a string $s \in \{0,1\}^k$ for some natural number k. (The 1's correspond to hydrophobic amino acids and the 0's correspond to hydrophilic ones.) A *folding* is an embedding of the indices $1, \dots, k$ into a two-dimensional $k \times k$ grid. Formally, a folding is a function $f : [k] \to [k] \times [k]$ with the following two properties:

   1) Consecutive indices always map to adjacent grid cells. Formally, for every $i = 1, 2, \dots, k-1$, if $f(i) = (x, y)$ we have $f(i+1) \in \{(x-1, y), (x+1, y), (x, y-1), (x, y+1)\}$, and

   2) The function f is one-to-one (injective), i.e., it never maps two different indices to the same grid cell. Formally, for every $i \neq j$, we have $f(i) \neq f(j)$.

   If you've ever played the game "snake", all this is saying is that the sequence of cells $f(1), f(2), \dots, f(k)$ form a snake that does not intersect itself.

   (a) Is the function $f : [4] \to [4] \times [4]$ defined by $f(1) = (2,2), f(2) = (2,3), f(3) = (3,3), f(4) = (3,4)$ a valid folding? It may help to draw the picture and see if it forms a snake.

   (b) Let $s \in \{0,1\}^k$ be a polypeptide, $f : [k] \to [k] \times [k]$ be a folding, and d be a natural number. We say f is a d-stable folding if there are at least d "hydrophobic bonds," which are distinct pairs of indices $i < j$ such that $s_i = s_j = 1$ and $f(i)$ and $f(j)$ are adjacent cells in the grid.

   For example, let $s = 10110$ and let $f(1) = (1,2), f(2) = (1,3), f(3) = (2,3), f(4) = (2,2), f(5) = (2,1)$. Explain why f is a 2-stable folding for s.

   (c) Define the language $SF = \{\langle s, d\rangle \mid$ there exists a d-stable folding of $s\}$. Prove that SF is in NP by showing that it can be decided by nondeterministic TM in polynomial time. Be sure to describe your NTM, explain why it is correct, and explain why it halts in poly-time on every computational branch.

2. (**NFA Acceptance**) Let $A_{\text{NFA}} = \{\langle N, w\rangle \mid N \text{ is an NFA that accepts string } w\}$. Prove that A_{NFA} is in NP by giving a polynomial-time verifier. Be sure to describe the structure of a valid certificate, describe your (deterministic) verifier, explain why the verifier is correct, and explain why the verifier runs in time polynomial in the input length.

   Hint: You can use without proof the fact that if an NFA N accepts w, it can do so via a computational path consisting of $O(|w|^2)$ transitions. You're encouraged to think about how to prove this by the pigeonhole principle.

3. (**Satisfiability**)

   (a) Let $\varphi(x, y, z) = (x \wedge y) \vee (x \wedge \overline{z})$. Is φ satisfiable? If so, exhibit a satisfying assignment. Otherwise, explain why it is not satisfiable.

   (b) Let $\psi(x, y, z) = (x \vee y) \wedge (x \vee \overline{y}) \wedge (\overline{x} \vee z) \wedge (\overline{x} \vee \overline{z})$. Is ψ satisfiable? If so, exhibit a satisfying assignment. Otherwise, explain why it is not satisfiable.

   (c) Define the language $XSAT = \{\langle \varphi_1, \varphi_2\rangle \mid \text{there exists an assignment } x \text{ satisfying exactly one of } \varphi_1, \varphi_2\}$. Show that $XSAT$ is in NP. You may either give a poly-time NTM or describe a poly-time verifier; it's your choice. In either case, give an explanation of correctness and of runtime.

4. (**Search vs. decision**) 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 $HUB = \{\langle S_1, \ldots, S_k, r\rangle \mid$ there exists a valid course plan $T \subseteq [k]$ of size $|T| \leq r\}$.

   In this problem, you will prove that if P = NP, you can *find* a smallest plan of classes that will fulfill all of your Hub requirements in polynomial time.

   (a) Prove that $HUB \in$ NP. You may either give a poly-time NTM or describe a poly-time verifier; it's your choice. In either case, give an explanation of correctness and of runtime.

   (b) **If** P = NP, your solution to part (a) implies that $HUB \in$ P as well. That is, in polynomial time you can decide whether r classes is enough. Assume you have a subroutine that solves this problem in polynomial time. Show how to use it repeatedly to *find* a smallest valid course plan in polynomial time. (Note that your algorithm should output the actual *plan*, i.e., set of classes.)

   Hint: Similar to Sipser Problem 7.40, solved in the book.