

Due May 1, 2:45pm

Instructions: You may work in groups of up to 4 people, but make sure to list your study partners on your homework. As always, you are required to write up your homeworks yourself, and you must never share your write-ups with others.

1. (25 pts.) 4-coloring

The 3-COLORING problem is as follows:

Given: an undirected graph $G = (V, E)$

Find: a valid 3-coloring $c : V \rightarrow \{\text{Red, Green, Blue}\}$, or report that none exists

The 4-COLORING problem is as follows:

Given: an undirected graph $G = (V, E)$

Find: a valid 4-coloring $c : V \rightarrow \{\text{Red, Green, Blue, Yellow}\}$, or report that none exists

These are both search problems. (Recall that c is a valid coloring if $c(u) \neq c(v)$, for every edge $(u, v) \in E$.)

Find a reduction from 3-COLORING to 4-COLORING. Use this reduction to prove that if there is a polynomial-time algorithm for the 4-COLORING problem, then there is a polynomial-time algorithm for 3-COLORING, too.

2. (25 pts.) Hamiltonian path

The HAMILTONIAN PATH problem is as follows:

Given: a directed graph $G = (V, E)$

Find: a Hamiltonian path, or report that none exists

(A Hamiltonian path is a path that visits every vertex in V exactly once: no more, no less.) The HAMILTONIAN CYCLE problem is as follows:

Given: a directed graph $G = (V, E)$

Find: a Hamiltonian cycle, or report that none exists

A Hamiltonian cycle is a cycle that visits every vertex in V exactly once. (Recall that a cycle is a path that starts and ends at the same vertex.) These are both search problems. (These are basically what the textbook calls the RUDRATA PATH and RUDRATA CYCLE problems, except that here we focus on directed graphs.)

- (a) Prove that if you can solve HAMILTONIAN CYCLE in polynomial time, you can solve HAMILTONIAN PATH in polynomial time, by finding a reduction from HAMILTONIAN PATH to HAMILTONIAN CYCLE.
- (b) Prove that if you can solve HAMILTONIAN PATH in polynomial time, you can solve HAMILTONIAN CYCLE in polynomial time.

3. (25 pts.) Longest Path

Here is the LONGEST PATH problem:

Given: a directed graph $G = (V, E)$, where all edges have length 1
Find: the longest simple path in G

(Recall that a simple path is one where no vertex is repeated.) Also, here is the LONG PATH problem:

Given: a directed graph $G = (V, E)$, where all edges have length 1; a minimum $m \in \mathbb{N}$
Find: a simple path whose length is at least m , or report that none exists

LONG PATH is a search problem. A caution: LONGEST PATH does not seem to be a search problem: given a candidate path, it's not clear how to verify in polynomial time whether it is the longest possible simple path or not.

- (a) Suppose that Alice comes up with a polynomial-time algorithm A for LONGEST PATH. Explain how to solve the LONG PATH problem in polynomial time, using A as a subroutine.
- (b) Suppose that Bob comes up with a polynomial-time algorithm B for LONG PATH. Explain how to solve the LONGEST PATH problem in polynomial time, using B as a subroutine.
- (c) Find a reduction from HAMILTONIAN PATH to LONG PATH. In other words, prove that if LONG PATH can be solved in polynomial time, then so can HAMILTONIAN PATH.

4. (25 pts.) Bio-computing

One of the most remarkable developments in recent years has been development of powerful new methods for DNA sequencing. These have partly built upon algorithmic advances in bio-computing and sequence assembly. In modern approaches to DNA sequencing, the genome to be sequenced consists of a very long string made up of the four letters A, C, G, and T (these strings might be 10^8 – 10^9 letters long); we are given many short fragments (substrings) from the string, but with no clue about where in the string they came from, and possibly with some errors in these fragments; and the goal of a sequence assembly algorithm is to reconstruct the full string. Work on this area has led to many impressive algorithmic techniques that have helped make DNA sequencing feasible at large scale. This problem is intended to give you some taste of the computational challenges in this area, in a greatly simplified form.

Imagine that the genome to be sequenced consists of $n\ell$ letters, and it has been split into n fragments (substrings), each of length $\ell = 2k$. The first fragment is obtained from the letters at positions $0..k-1$; the next from the letters at $k..2k-1$; and so on, with no overlap between the fragments. We are given the set $F = \{f_1, f_2, \dots, f_n\}$ of fragments obtained in this way, but the order of them has been scrambled so we have no idea which f_i came from which position in the genome.

Also, we are given a set $C = \{c_1, \dots, c_m\}$ of additional strings, each of length ℓ . Suppose that we have a pair of fragments f_i, f_j , and we want to know whether f_j might have been taken from immediately after f_i in the original genome. If there is a string $c \in C$ such that the first k letters of c match the last k letters of f_i , and such that the last k letters of c match the first k letters of f_j , then we say the pair (f_i, f_j) is *corroborated* by c . The intuition is that maybe c represents a fragment of DNA that straddles the region where f_i and f_j appear in the genome, which would be consistent with the hypothesis that f_j could have been taken from right after f_i .

We'd like to recover the original genome G , so we are going to look for a string of $n\ell$ letters that is obtained by concatenating all the fragments of F in some order, with each fragment from F used exactly once, where the fragments are arranged in an order so that each consecutive pair of fragments is corroborated by some element of C . In other words, we want to find a string $G = f_{i_1} f_{i_2} \dots f_{i_n}$ such that i_1, i_2, \dots, i_n is a permutation of $1, 2, \dots, n$, and such that for each $j \in \{1, 2, \dots, n-1\}$, the pair $(f_{i_j}, f_{i_{j+1}})$ is corroborated by some element of C . Call any such string G a *corroborated assembly*.

The TOY SEQUENCING problem is then defined as:

Given: sets $F, C \subseteq \{A, C, G, T\}^\ell$

Find: a corroborated assembly $G \in \{A, C, G, T\}^{n\ell}$, or report that none exists

Prove that, if none of the search problems defined earlier in this homework can be solved in polynomial time, then there is no polynomial-time algorithm for the TOY SEQUENCING problem.

Hint: Reduce one of the problems defined earlier in this homework (3-COLORING, 4-COLORING, HAMILTONIAN PATH, HAMILTONIAN CYCLE, or LONG PATH) to TOY SEQUENCING. Make sure to state which problem you are reducing from; then describe the reduction; then explain why this reduction proves the desired result. To help the readers, please label each of these 3 parts clearly.