6.890: Algorithmic Lower Bounds — Prof. Erik Demaine
Fall 2014

Problem Set 3

Due: Wednesday, October 22nd, 2014

Problem 1. A Tour of Hamiltonicity Variants
For each of the following variations on the Hamiltonian Cycle problem, either prove it is in P by giving a polynomial time algorithm, or prove it is NP-hard.

(a) Given a simple graph, is there a cycle that visits every node at least once and no more than twice?

(b) Given a simple directed graph, is there a path that visits every node at least once, but does not cross an edge more than once?

(c) Given a simple graph, is there a path that visits at least half the nodes, but never visits a node more than once?

(d) Given a simple dense graph ($|E| = \Theta(|V|^2)$, is there a Hamiltonian Cycle?

Problem 2. Quartet Ordering
Suppose that you are given a set of elements $U = \{u_1, \ldots, u_n\}$ and a set of ordered quartets $Q = \{\langle a_1, b_1, c_1, d_1\rangle, \ldots, \langle a_k, b_k, c_k, d_k\rangle\}$ containing elements of U. For each quartet $\langle a_i, b_i, c_i, d_i\rangle$, the four elements must be distinct. Prove that it is NP-complete to find an ordering for U such that for each $\langle a_i, b_i, c_i, d_i\rangle$, either a_i and b_i are the extremes of the quartet (minimum and maximum, not necessarily in that order), or c_i and d_i are the extremes of the quartet.

Problem 3. *n* Days Later
Suppose that you are given a grid graph, representing a network of cities connected by roads. Several of the cities are marked as infected. One city is marked as the player's starting position. The player has two possible actions: place the city that they're in under quarantine (which keeps that city from becoming infected in the future), or move to an adjacent city. Each day, two things happen in sequence: the player performs a single action, and then the infection status of each city is updated according to the following rules:

R1. If the city was infected at the start of the day, it remains infected.

R2. If the city is not under quarantine and is adjacent to another city that was infected at the start of the day, it becomes infected.

R3. All other cities remain uninfected.

The scenario is over when the infection cannot spread any further. Prove that it is NP-hard for the player to maximize the number of cities left uninfected.

Problem 4. Cheating on Puzzles
Sometimes puzzles are too hard to solve exactly. What if we're satisfied with a solution that's "almost" correct?

Recall that a *Light Up* puzzle[1] consists of a board with black squares and white squares. Some of the black squares are labled with a number. For this problem, we also specify the number of lights to be placed. Two white squares on this board can *see* each other if they lie in the same row or column, and they do not have a black square between them. In a solution, lights are placed in some of the white squares, subject to the following constraints:

C1. Every white square can see a light.

C2. No two lights can see each other.

C3. Every black square labeled with value k has exactly k lights adjacent to it.

Here we consider what happens when a solution is allowed to violate these constraints. Define the *cost* of a solution to be the number of violated constraints, i.e., the number of white squares that cannot see a light, plus the number of pairs of lights that can see each other, plus the number of black squares with incorrect labels.

(a) Given a *Light Up* puzzle, prove that it is NP-hard to decide whether there is a solution that violates at most an ε fraction of the constraints, for any constant $\varepsilon > 0$.

(b) Given a *Light Up* puzzle, prove that it is APX-hard to minimize the cost of a solution. (Hint: first convert the Circuit SAT reduction from class to a reduction from 3SAT.)

[1] Refer to Lecture 6 and http://www.mountainvistasoft.com/docs/lightup-is-np-complete.pdf