CS/ECE 374 ✧ Fall 2018

Homework 10

Due Wednesday, December 5, 2018 at 10am

Groups of up to three people can submit joint solutions. Each problem should be submitted by exactly one person, and the beginning of the homework should clearly state the Gradescope names and email addresses of each group member. In addition, whoever submits the homework must tell Gradescope who their other group members are.

The following unnumbered problems are not for submission or grading. No solutions for them will be provided but you can discuss them on Piazza.

- Consider an instance of the Satisfiability Problem, specified by clauses $C_1, \ldots, C_k$ over a set of Boolean variables $x_1, \ldots, x_n$. We say that the instance is *monotone* if each term in each clause consists of a nonnegated variable; that is each term is equal to x_i, for some i, rather than $\bar{x}_i$. Monotone instance of Satisfiability are very easy to solve: They are always satisfiable, by setting each variable equal to 1.

  For example, suppose we have the three clauses

  $$(x_1 \vee x_2), (x_1 \vee x_3), (x_2 \vee x_3)$$

  This is monotone, and indeed the assignment that sets all three variables to 1 satisfies all the clauses. But we can observe that this is not the only satisfying assignment; we could also have set x_1 and x_2 to 1 and x_3 to 0. Indeed, for any monotone instance, it is natural to ask how few variables we need to set to 1 in order to satisfy it.

  Given a monotone instance of Satisfiability, together with a number k, the problem of *Monotone Satisfiability with Few True Variables* asks: Is there a satisfying assignment for the instance in which at most k variables are set to 1? Prove that this problem is NP-Complete. *Hint:* Reduce from Vertex Cover.

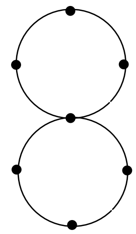
- Given an undirected graph $G = (V, E)$ a *matching* in G is a set of edges $M \subseteq E$ such that no two edges in M share a node. A matching M is *perfect* if $2|M| = |V|$, in other words if every node is incident to some edge of M. PerfectMatching is the following decision problem: does a given graph G have a perfect matching? Describe a polynomial-time reduction from PerfectMatching to SAT. Does this prove that PerfectMatching is a difficult problem?

1. Consider the language

   $$L_{374\mathrm{H}} = \{\langle M \rangle \mid M \text{ halts on at least 374 distinct input strings}\}.$$

   Note that for $\langle M \rangle \in L_{374\mathrm{H}}$, it is not necessary for M to *accept* any string; it is sufficient for it to *halt* on (and possibly reject) 374 different strings. Prove that $L_{374\mathrm{H}}$ is undecidable.

2. We call an undirected graph an *eight-graph* if it has an odd number of nodes, say $2n-1$, and consists of two cycles C_1 and C_2 on n nodes each and C_1 and C_2 share exactly one node. See figure below for an eight-graph on 7 nodes.



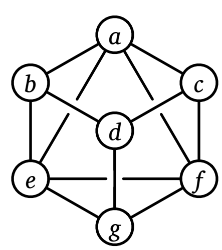
Given an undirected graph G and an integer k, the EIGHT problem asks whether or not there exists a subgraph which is an eight-graph on $2k-1$ nodes. Prove that EIGHT is NP-Complete.

3. Given an undirected graph $G=(V,E)$, a partition of V into $V_1, V_2, \dots, V_k$ is said to be a clique cover of size k if each V_i is a clique in G. CLIQUE-COVER is the following decision problem: given G and integer k, does G have a clique cover of size at most k?

   - Describe a polynomial-time reduction from CLIQUE-COVER to SAT. Does this prove that CLIQUE-COVER is NP-Complete? For this part you just need to describe the reduction clearly, no proof of correctness is necessary. *Hint:* Use variablex $x(u,i)$ to indicate that node u is in partition i.
   - Prove that CLIQUE-COVER is NP-Complete.

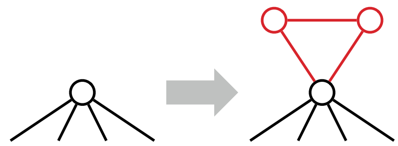
Solved Problem

4. A *double-Hamiltonian tour* in an undirected graph G is a closed walk that visits every vertex in G exactly twice. Prove that it is NP-hard to decide whether a given graph G has a double-Hamiltonian tour.



This graph contains the double-Hamiltonian tour $a \to b \to d \to g \to e \to b \to d \to c \to f \to a \to c \to f \to g \to e \to a$.

Solution: We prove the problem is NP-hard with a reduction from the standard Hamiltonian cycle problem. Let G be an arbitrary undirected graph. We construct a new graph H by attaching a



A vertex in G, and the corresponding vertex gadget in H.

small gadget to every vertex of G. Specifically, for each vertex v, we add two vertices $v^\sharp$ and $v^\flat$, along with three edges $vv^\flat$, $vv^\sharp$, and $v^\flat v^\sharp$.

I claim that G has a Hamiltonian cycle if and only if H has a double-Hamiltonian tour.

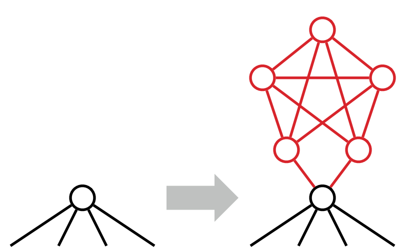
$\Longrightarrow$ Suppose G has a Hamiltonian cycle $v_1 \to v_2 \to \cdots \to v_n \to v_1$. We can construct a double-Hamiltonian tour of H by replacing each vertex v_i with the following walk:

$$\cdots \to v_i \to v_i^\flat \to v_i^\sharp \to v_i^\flat \to v_i^\sharp \to v_i \to \cdots$$

$\Longleftarrow$ Conversely, suppose H has a double-Hamiltonian tour D. Consider any vertex v in the original graph G; the tour D must visit v exactly twice. Those two visits split D into two closed walks, each of which visits v exactly once. Any walk from $v^\flat$ or $v^\sharp$ to any other vertex in H must pass through v. Thus, one of the two closed walks visits only the vertices v, $v^\flat$, and $v^\sharp$. Thus, if we simply remove the vertices in $H \setminus G$ from D, we obtain a closed walk in G that visits every vertex in G once.

Given any graph G, we can clearly construct the corresponding graph H in polynomial time.

With more effort, we can construct a graph H that contains a double-Hamiltonian tour ***that traverses each edge of H at most once*** if and only if G contains a Hamiltonian cycle. For each vertex v in G we attach a more complex gadget containing five vertices and eleven edges, as shown on the next page. ■

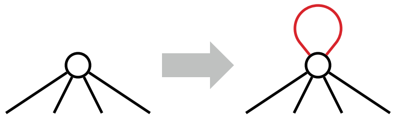


A vertex in G, and the corresponding modified vertex gadget in H.

Common incorrect solution (self-loops): We attempt to prove the problem is NP-hard with a reduction from the Hamiltonian cycle problem. Let G be an arbitrary undirected graph. We construct a new graph H by attaching a self-loop every vertex of G. Given any graph G, we can clearly construct the corresponding graph H in polynomial time.

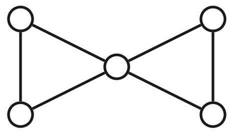
Suppose G has a Hamiltonian cycle $v_1 \to v_2 \to \cdots \to v_n \to v_1$. We can construct a double-Hamiltonian tour of H by alternating between edges of the Hamiltonian cycle and self-loops:

$$v_1 \to v_1 \to v_2 \to v_2 \to v_3 \to \cdots \to v_n \to v_n \to v_1.$$



An incorrect vertex gadget.

On the other hand, if H has a double-Hamiltonian tour, we *cannot* conclude that G has a Hamiltonian cycle, because we cannot guarantee that a double-Hamiltonian tour in H uses *any* self-loops. The graph G shown below is a counterexample; it has a double-Hamiltonian tour (even before adding self-loops) but no Hamiltonian cycle.



This graph has a double-Hamiltonian tour.

◇

Rubric (for all polynomial-time reductions): 10 points =

+ 3 points for the reduction itself
  – For an NP-hardness proof, the reduction must be from a known NP-hard problem. You can use any of the NP-hard problems listed in the lecture notes (except the one you are trying to prove NP-hard, of course).
+ 3 points for the "if" proof of correctness
+ 3 points for the "only if" proof of correctness
+ 1 point for writing "polynomial time"

- An incorrect polynomial-time reduction that still satisfies half of the correctness proof is worth at most 4/10.
- A reduction in the wrong direction is worth 0/10.